INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
313/761-4700 800/521-0600

THE REPRESENTATION OF COMBINATORIAL GAMES AND THE ALGORITHMS USED TO PLAY THEM

by

Arthur Allan Leff

A dissertation submitted to the Graduate Faculty in Computer Science in partial fulfillment of the requirements for the degree of Doctor of Philosophy, The City University of New York

1995

This manuscript has been read and accepted for the Graduate Faculty in Computer Science in satisfaction of the dissertation requirement for the degree of Doctor of Philosophy.

8/21/95 — Michael Anshel

Date — Chair of the Examining Committee

Aug 24, 1995 — Stanley Selkow

Date — Executive Officer

Professor Stephan Burr

Dr. Raqui Kane

Professor Stephen Lucci

Professor Valentin Turchin

Supervisory Committee

THE CITY UNIVERSITY OF NEW YORK

Table of Contents

LIST OF ILLUSTRATIONS

CHAPTER 1

Introduction

This is a study of the representation of combinatorial games and the algorithms used to play them.

There exists a very large set of combinatorial games, called Wonderland by Fraenkel [CG p.147], for which the complexities are hard to determine. Most problems in classical computer science can be classified as tractable, conditionally intractable or intractable; two notable exceptions are primality testing and the graph isomorphism problem. Thus, Wonderland represents an important source of experimental material both for theoretical computer scientists and for the development of algorithmic techniques. Wythoff's game, a member of this set, was the focus of this dissertation.

Wythoff's game is played with two or more heaps of stones. Two players alternately remove stones from the heaps according to the following rules:

Type-one: Remove one or more stones from exactly one of the heaps;

Type-two: Remove K stones from one heap and L stones from a second heap provided that $|K-L| < A \in \mathbf{Z}^+$.

In the classical version of the game, A is equal to 1, that is, you remove the same number of stones from each of the two heaps.

Wythoff's game is, additionally, the hardest game of a class of games which Fraenkel calls Nimhoff games. The easiest game in this class is Nim. Everything is known about the game of Nim. For Wythoff's game the only known polynomial time algorithms are for determining which of the two players has a forced win in the two-heap game; there are no known polynomial time algorithms for finding the Grundy values for Wythoff's game with two or more heaps. "No polynomial algorithm is known for computing the sum of Wythoff games, and this seems to contribute to the fact that no natural way to generalize Wythoff's game to more than two piles is known." [ASF2 p.2]

For Wythoff's game, the following are open questions [ASF3 p.169]:

1. Decide whether a finite disjoint sum of games has a polynomial strategy;
2. Generalize the game to more than two heaps.

The results presented in this dissertation were obtained while attempting to generalize Wythoff's game to three heaps. My methodology was to use various representations for Wythoff's game in an attempt to obtain insights into the game.

Wyhoff's game for $n \geq 2$ heaps has a representation as a queen on an n-dimensional chessboard. Following the example

of Fraenkel in [ASF2], new chesspieces were developed representing features or extensions of the queen. These games were then analyzed in the hope that they might reveal some structure that could be related to Wythoff's game. Working with the chesspiece representation led to two new algorithms for the three-heap Wythoff's game; there are no published algorithms for the three-heap Wythoff's game. An additional result was the chesspiece Latin Rook, the Grundy values of which form latin squares of size $2^n - 1$, $n \in \mathbb{Z}^+$.

A representation using string rewriting systems was also attempted. This representation resulted in the development of a schema, Literary Wythoff, which generates a myriad of Wythoff variations which would be at best difficult to formulate in the standard representation. Literary Wythoff for $n \geq 2$ stones is a superset of Wythoff's game under both the normal and the misère ending conditions.

Four new games based on string rewriting systems are introduced. One of these games, Abel, is analyzed. These games were developed as models of what a string rewriting game would look like.

The representation of Wythoff's game as a vector addition system was also considered. This did not lead to any new results. The interest in vector addition systems is witnessed by the section "Generalized knights on a multidimensional chessboard" in a recently published book by Matiyasevich [MAT, pp.184-196]. His knight's games provide

a game theoretic interpretation of vector addition systems.

This work has been experimental and four new variations of Wythoff's game were developed. In one variation, Linear Wythoff, the Type-two move was weakened. In another, Designer Wythoff, the game space is a set, that is, reducing a heap to the size of a second heap effectively removes that heap from play. A third variation, WW2 and WW3, respectively for the two-heap and three-heap versions, adds a Type-two move to Welter's game. The fourth variation, Blocked Wythoff, removes squares from a chessboard, A queen cannot start from, pass through, or land on a blocked square.

In addition, a new variation of Nim, Antenim, was discovered by me during the work on representations and is introduced in this dissertation.

Many of the algorithms used to analyze the games were implemented as programs using IBM VS FORTRAN 77.

CHAPTER 2

What is a Combinatorial Game?

A combinatorial game satisfies the following eight conditions.

1. There are exactly two players, say, Left and Right. By convention, Left is the first to move from the initial position.

2. There are, usually, finitely many positions.

3. There are clearly-defined rules that specify the moves each player is able to make from any particular position. A move is from the current position to a position different from the current one. If both of the players have the same moves available from each of the positions, then the game is *impartial*; otherwise, the game is *partizan*.

4. Left and Right move alternately, in the game as a whole.

5. In the *Normal Ending Condition*, the first player unable to move loses and his or her opponent wins. In the *Misère Ending Condition*, the first player unable to move wins and his or her opponent loses.

6. The rules are such that the play always comes to an end under the assumption of best play by both players.

7. There is complete (perfect) information; that is, each of the players has the information needed to effect the best play.

8. There are no chance moves.

Wythoff's game is an impartial combinatorial game.

Northcott's game is a partizan combinatorial game. In Northcott's game the notion of best play is extended to play that terminates, otherwise, the game might never end.

Backgammon is not a combinatorial game in that the move a player can make at his or her turn is determined in part by a roll of the dice.

Gin Rummy is not a combinatorial game since the initial position (the two hands and the stack of cards from which the players will draw) are determined by the shuffle (a chance move); and, neither player knows what the other player holds, or in what order the cards appear on the stack (imperfect information). Moreover, the last player to move (say, pick up a card) does not necessarily win since his or her opponent can win by underknocking and going over the required score (violating Condition 5).

Chess is not a combinatorial game since the game can end in a draw. All three types of draws: (i) by stalemate; (ii) three repetitions of the same position; or, (iii) by having made x number of moves without the capture or promotion of material violate condition 5. The play of a novice with a king, bishop, and knight against a master having only a king may end in a draw or barring (ii) and (iii) above may never end.

The following game, G, is the disjunctive sum of chess and checkers. Neither of these games are combinatorial, in that both chess and checkers both have draws. Both of these games are partizan, since in both games, each of the players move different pieces. What is interesting about this game is that the moves in chess are worth more than the moves in checkers, so that, a player would, usually, want to move in the chess game and not in the checker game.

Let $G = G_1 + G_2$, where G_1 is the game of chess and G_2 is the game of checkers and the + stands for the disjunctive union of the two games. [Figure 1] There are two players, Left and Right. Left moves first. The first player unable to move loses, his opponent wins. The two players move alternately. To move a player must either make a move in G_1 or make a move in G_2. Passes are not allowed. The moves in chess and checkers are the normal ones for those games with the exception that a king left in check may be captured and is equivalent to a checkmate. The first player to move in the chess game gets the white pieces but this is not necessary.

Given that the checkmated king (along with his army) vacates the board in G_1 leaving the other pieces on the board it seems obvious that all of the moves would initially be played in G_1 and none of the moves in G_2. E.g., Left moves in the chess game and never moves in the checker game. If Left wins the chess game he will have unlimited moves in

that game. Right continues to move in the checker game, G_2, but since none of his pieces have been promoted to a king (Since Left has not moved any checker, his checkers occupy Right's eighth row and cannot be captured; Right cannot promote a checker to a king unless the checker reaches the eighth row; a checker can only move forward or capture (jump) forward) eventually he must run out of moves. Right loses and Left wins. Moreover, chess is a game in which a player wants to move so any move that, say, Right makes in the checker game will result in the loss of a tempo in the chess game. The play would switch to checkers if it is to neither player's advantage to move in the chess game.

All the games considered in this dissertation are played under the Normal Ending Condition; and, with the exception of Northcott's game, always come to an end.

Combinatorial games have a representation as finite directed rooted trees; the root is the initial position of the game. The tree may contain only the root. In this case, the root is an ending position.

In Figure 2, two directed trees are displayed, for the position (1,2) in Wythoff's game with A = 1. The first is a directed tree of all the descendants of the position with the directed edges (arcs) labelled with the moves (1 = Type-one, 2 = Type-two) used to reach that descendant. The second is the graph of the game. Note, as an example, that (0,1) is equivalent to (1,0) since we are only interested in

the number of stones appearing in a heap and not the order in which the heaps appear.

A Grundy value is the value of a position in a game stated in terms of a single Nim-heap and calculated by use of a Sprague-Grundy function. The following three definitions are from [Blass]:

Definition 1.1. For any digraph (directed graph) $G(V,E)$, the set of followers of a vertex $u \in V$ is $F(u) = \{ v \in V : (u,v) \in E \}$.

Let $S \subset \mathbb{N}$ and let $\bar{S} = \mathbb{N} - S$.

Definition 1.2. The Minimum Excluded value (mex) of a set is

$$\text{mex } S = \min \bar{S} =$$

the least nonnegative integer not in S.

Definition 1.3. A Sprague-Grundy function (or Grundy function) is defined recursively by

$$g(u) = \text{mex } g(F(u))$$

where, for any set T,

$$g(T) = \{ g(t) : t \in T \}.$$

The set of all positions in a combinatorial game are partitioned into two equivalence classes: P-positions; and, N-positions. Under the Normal Ending Condition: P-positions are the set of partitions that are equivalent to the ending positions and have a Grundy value of zero; N-positions are all those positions that are not P-positions and have a Grundy value greater than zero. That is to say,

P-positions $\cup$ N-positions = The set of all positions

P-positions $\cap$ N-positions = ϕ.

In Figure 2, if the game is played under the Normal Ending Convention then all the leaves of the tree have a Grundy value of zero and are equivalently P-positions.

Under the Normal Ending Condition: There are no moves from a P-position to another P-position; only to another N-position or no move at all. From an N-position there is at least one move to a P-position and zero or more moves to another N-position. From which follows a strategy for playing any combinatorial game under the Normal Ending Condition:

1. Determine if the position you are playing from is a P-position or an N-position.
2. If the position that you are playing from is an N-position, then

    move to a P-position

    { In the Normal Ending Condition, there is always one move from the N-position to a P-position }

    Else

    make a random move and hope that your opponent makes a mistake

    Endif.

The usual method of determining the winning moves in a disjunctive sum of games is to find the Grundy values associated with each of the games and to use Bolton's algorithm to determine the value of the disjunctive sum. In a disjunctive sum of games, a player, in his or her turn,

1. Selects one of the games; and,
2. Makes a legal move in the game selected.

Nim, played with two or more heaps, is a disjunctive sum of games in which each of the Nim-heaps is a game; and, the player, at his or her turn,

1. Selects one of the Nim-heaps; and,
2. Removes one or more stones from that heap.

CHAPTER 3

The Representation of Games

We begin with the simplest of combinatorial games. There are two players, L and R. The board, B, consists of a heap of n coins, n is a natural number. L moves first and the two players alternate moves. A move consists of removing one coin from the heap. The first player unable to move loses.

We define a game as an ordered triple:

G1 = (R, C, B)

Where R is a set of rules

C is a control unit

and B is the state of the board.

A move consists of choosing a rule from R and applying it one or more times as allowed by the control unit. The control unit is the referee and keeps track of whose turn it is. The use of a control unit allows us to simplify the description of a move. The board state together with player whose turn it is gives the state of the game. An initial state is not introduced as this is only of historical value and not a description of the state of the game. If necessary we can introduce the initial state as part of the game description, i.e., G = (R, C, B, I).

It is clear from the description of the game that both players must play a perfect game. Moreover, it is equally clear that the player having to move loses when he is faced with an empty heap. By an inductive argument, we can assert that the player having an even number of coins in the heap when he moves loses and a player having an odd number of coins in the heap when he moves wins. We can say that the set of all winning positions is given by the set of all nonnegative even integers. We will call this set P for **P**revious player wins and define it to be the set of **P**-positions (again, for **P**revious player wins). The set P is disjoint from the set **N** for **N**ext player wins and $P \cup N = \mathbb{N}$, the set of all natural numbers.

Lemma 3.1.

$N (G1) = \{ n \text{ s.t. } n \geq 0 \text{ and } \mod (n, 2) = 0 \}$

Proof.

When n is equal to zero, there is no move, therefore, the Next player loses. The lemma is true for $k = 0$.

Assume that this is true for $n = 2 * k$, $k \in \mathbb{N}$ and $k > 0$.

Let $n = 2 * (k + 1) = 2 * k + 2$. It is Next player's move and Next player removes one coin leaving $2 * k + 1$ coins. It is now Previous player's move and Previous player removes one coin leaving $2 * k$ coins and it is Next players move. By induction hypothesis, this is a loss for Next player■

Lemma 3.2.

G1 has no draws.

Proof.

The board consists of a nonnegative integer, n. Each of the players, in turn, reduce this integer by one. There are a finite number of nonnegative integers between n and zero. Since each move decreases this integer by one, the number of moves is finite the game must eventual end. The control structure keeps track of whose turn it is. The first player unable to move loses and the other player wins■

Lemma 3.3.

P (G1) = $\mathbb{N}$ - N (G1).

Proof.

Let Previous player be on move with 2 * k + 1 coins on the heap, k an arbitrary natural number. Previous player moves and leaves 2 * k coins on the heap. By Lemma 3.2, Previous player wins. Since k is arbitrary, P (G1) is the set of all odd numbers greater than or equal to zero. { Natural Numbers } = { Odd nonnegative numbers } + { Even nonnegative numbers }, hence, the lemma is proved■

In this game, the set of winning positions is the set of all positions (nonnegative integers) which are equivalent to 0 mod 2. This equivalency to zero (a position in which

no move is possible) is an inductive definition of P. In other games, this equivalency is harder to find.

This game can be transformed into a vector addition system:

[PETERSON p.232]

Definition A *vector addition system V* is a pair $V = (B,s)$, where $B = \{b_1, b_2, \ldots, b_m\}$ is a set of m vectors, called the *basis* or *displacement* vectors. The vector s is the *start* vector. All vectors are composed of n integer values. The elements of s are nonnegative.

Let V (G1) = (V, C, B) be a vector addition game
where V = { (-1) } is the set of rules
C is the control unit
and B = (n), $n \in \mathbb{N}$, is the board state.

The control unit is given as follows:
Two players, L and R, move alternately. L moves first. The first player unable to move loses. A move consists of choosing a vector from V and applying it one time to the vector B with the constraint that at any point in the move no component of B may be less than zero.

That the two games are isomorphic [Jantzen] is obvious;

or, transformed into a rewriting game:

Let W (G1) = (Σ, R, C, B)

where Sigma = { a }

R = { (a, λ) } is the set of production Rules

C is the control unit

and B = { a^n, n ϵ $\mathbb{N}$ } is the board state.

The control system is given as follows:
Two players, L and R, move alternately. L moves first. The first player unable to move loses. A move consists of choosing a vector from V and applying it one time to the string.

The game can, in addition, be written as the following automatic game:

A (G1) = { Σ, R, B }

where Σ = { L, R, a }

R = { (Laa*, Ra*), (Raa*, La*) }

B = La*

The second game is less trivial in that either player can remove one or two coins from the pile. The description will be as a vector addition system.

G2 = (V, C, B)

where V = { (-1), (-2) }

B = (n), n ϵ $\mathbb{N}$

C = the control unit

The control unit is given as follows:

Two players, L and R, move alternately. L moves first. The first player unable to move loses. A move consists of choosing a vector from V and applying it one time to the vector B with the constraint that at no time during a move can any component of the vector be negative.

The question: What is the set P (G2)?, is harder to answer.

Zero, 0, is in the set P (G2).

One, 1, and Two, 2, are not. In either instance there is a move to zero which is in P. Hence, One and Two are in N. Three is in P since the only moves from Three are to One and to Two, both of which are in N. Four and Five are in N since in either case there exist a move to Three which is in P. From Six the only moves are to Four and Five, both of which are in N.

After some further examples we may find that the rule for winning this game is:

1) given a position n > 0, if mod (n, 3) $\neq$ 0 then
move to n - k, k = 1 or k = 2,
such that mod (n, k) = 0.

2) if mod (n, 3) = 0 then

make a random move and hope that your opponent makes a mistake.

The set of board states (the set of all possible states of the game) is divided up into three equivalence classes modulo three.

We modify G2 to produce G2′ by changing the board space from $n \in \mathbb{N}$ to $n \in \mathbf{Z}^+$.

G2′ = (V′, C′, B′)

where V′ = { (-1), (-2) }

B′ = (n), $n \in \mathbf{Z}^+$

C′ = the control unit

The control unit is given as follows:
Two players, L and R, move alternately. L moves first. The first player unable to move loses. A move consists of choosing a vector from V and applying it one time to the vector B with the constraint that at no time during a move can any component of the vector be less then or equal to zero.

The rule for winning G2′ is:

1) given a position n > 1, if mod (n, 3) $\neq$ 1 then

move to n - k, k = 1 or k = 2,

such that mod (n, k) = 1.

2) if mod (n, 3) = 1 then make a random move and hope that your opponent makes a mistake.

The next game is an example of a partitioning game.

CUTCAKE [BCG pp.26-28]

Mother has just made the oatmeal cookies shown in Figure 3a. She hasn't yet broken them up into little squares, although she has scored them along the lines indicated. Rita and her brother Lefty decide to play a game breaking them up. Lefty will cut any rectangle into two smaller ones along one of the North-South lines, and Rita will cut some rectangle along and East-West line. When one of the children are unable to move, the game ends, and that child is the loser.

Figure 3b gives a table of the Sprague-Grundy values for Cutcake.

Cutcake is a partizan game in that the two players, L for Lefty and R for Rita, do not necessarily have the same options. In this game the options are mutually disjoint. The game will be presented as a Markov System.

M (R, B, C)

Where R is an ordered list of rules (Productions)

B is the Board State (word)

and C is the Control Unit.

Definition of Symbols (used in B and R):

L - Lefty's turn. The control unit halts until a valid move (external input from a player) is made.

F, T - Denote internal states.

R - Rita's turn. The control unit halts until a valid move (external input from a player) is made.

G, H - Denote internal states.

A - row (horizontal) dimension.

B - column (vertical) dimension.

C - boarder.

D - divider (cut).

m, n, x, y in Z^+.

a - a string of the form (CA*B*)*

b - a string of the form (A*B*C)*

The **initial** Board State is of the form [L|R]{aC}

where [L|R] means either L or R

and {aC} means zero or one occurrences of the string aC

The Control Unit:

The Control Unit starts at the top of the list of rules and attempts to apply each rule on the list in turn. If a rule can be applied the control unit starts at the top of the list of rules and tries to apply the rules in turn. If no rule can be applied the machine outputs the final string and halts. Note that the machine must have a way of starting the game and entering and checking both the initial state

and each of the players moves as well as designating the I/O units.

The following two uses of square brackets are Meta-Notation which indicate actions taken by the Control Unit

[str] --> str1

[str] denotes that the player is to specify the move to be made (e.g., how to cut what piece of cake).

as opposed to

str --> [opt1|opt2]

which indicates that the player is to select one of the rewriting rules listed.

Rules:

1. CABC ---> C
2. FaCAnBxDByCb ---> RaCAnBxCAnByCb
3. GaCAxDAyBmCb ---> LaCAxBmCAyBmCb
4. [LaCAnBmCb] ---> FaCAnBxDByCb x + y = n
5. [RaCAnBmCb] ---> GaCAxDAyBmCb x + y = m

The game starts with the input of the initial board state with Left as the player. This input is external to the set of rewriting rules and contains an interface and unit to check the validity of the string input. The control unit then attempts to apply rule 1.; if rule 1. does not apply then the control unit attempts to apply rule 2.; if rule 2. does not apply the control unit attempts to apply

rule 3., and so on . If any rule is applicable the rule is applied and the control unit starts again trying to apply rule 1., then rule 2. and so on. If none of the rules apply then the game ends. Rules 4. and 5. require input from the two players as to what to cut; and, again, it requires an interface and a unit for checking the validity of the string input. Rules 4. and 5. may ask the player (respectively, Left and Right) to specify how to cut what piece of cake and make the substitution if the cut can be made then go on to attempt to apply additional rules starting from the top of the list.

A game of Cutcake is one of many games of cutting a single piece of cake. When Cutcake is played as one game within a game with two or more disjoint games in which each of the two players select a game to play in and make a move in that game. Fraenkel states [CG] that this is known in the literature as the disjoint sum of games. This version of the Markov game is presented below.

Cutcake as one game in the sum of one or more disjoint games

M2 = (B, R, G)

Additional symbols not presented in Cutcake:

S - denotes a separate game

p - zero or more games

Rules:

1. LpSFaCAnBxDByCbp --> RpSaCAnBxCAnByCbp
2. RpSGaCAxDAyBmCbp --> LpSaCAxBmCAyBmCbp
3. CABC --> C
4. SCS --> S
5. [LpSaCAnBmCbp] --> LpSFaCAnBxDByCbp x + y = m
6. [RpSaCAnBmCbp] --> RpSGaCAxDAyBmCbp x + y = n

The initial board position is of the form Lp.

Cutcake is a partitioning game. Lefty cuts the pieces of cake in a North-South direction, Rita cuts the pieces in an East-West direction. The question is: What is a piece of cake worth?

The value of a piece of cake is the value of that piece to the player cutting the cake. The game is partizan in that the moves available to both players are not the same. In this case, the game is strictly partizan in that the set of moves available to Lefty and the set of moves available to Rita do not intersect.

We say that the number of moves a piece of cake is worth to Lefty is positive and that the number of moves a piece of cake is worth to Rita is negative. If the number of moves available to both players is the same then the

value of the game is zero. That is, the first player to move loses under the normal ending condition.

A player with the greatest number of moves available wins the game. The number of moves available to each player is based on both players making the best move possible. In a game of complete information this is considered to be the case. It is descriptive to call the number of moves available free moves as it is disadvantageous to move in this game. A player may win the game by a much larger number of moves than the difference in the number of free moves available to the players. The other player having no free moves is forced to make moves that give the other player additional free moves. A separate question might be: Given a game of Cutcake, with best play on the part of both players, who wins the game and how many free moves does the winning player have when the game ends?

Given a piece of cake B squares wide and D squares deep the following FORTRAN 77 function will compute its value.

```
      Integer Function CC ( B, D )
*  a function to compute the integer associated with a B by
D piece of cake for the     *  game Cutcake.  This integer
represents the number of moves the piece is worth to  *
Lefty if it is positive and the number of moves the piece is
worth to Rita if it is
```

```
*  negative.  A zero indicates that the piece is worth the
same number of moves to
*  both players.

      Integer B, D
      Integer Block
      External Block

      If ( B .GE. D ) Then
*     The piece is worth zero or more moves to Lefty
           CC = B / Block ( D ) - 1
      Else
*     The piece is worth one or more moves to Rita
           CC = ( -1 ) * ( D / Block ( B ) - 1 )
      Endif

      End

      Integer Function Block ( Num )
*     Finds the largest power of two which is less than or
equal to Num
      Integer Num
      Integer P2

*     Two to the zero power equals one
      P2 = 1
```

```
*     Find the block that Num belongs in
100   If ( Num .GE. ( 2 * P2 ) ) Then
           P2 = P2 + P2
           Goto 100
      Endif
      Block = P2

      End
```

CHAPTER 4

Nim and Other Games

The classic impartial game is Nim. Here we present the two-heap version.

There are two heaps with $n \geq 0$ stones. Two players, Left and Right, move alternately. Left moves first. A move consists of removing one or more stones from one of the heaps. The first player unable to move loses.

If Left removes all of the stones from one heap then Right can remove all the stones from the other and Left is left without a move, a loss.

A strategy for Left:

1. If the two heaps do not have the same number of stones then remove enough stones from one heap so that both heaps have the same number of stones.
2. If both heaps have the same number of stones then make a random move and hope that your opponent makes a mistake.

Two things are important:

1. the value of a Nim-heap

and 2. the Tweedledum and Tweedledee Argument.

The value of a Nim-heap is denoted by $*n$, for $n \in \mathbb{Z}^+$. In the case of $*1$ the 1 is omitted. What the value $*n$ means is that the Nim-heap can be reduced to the values $*(n-1)$, $*(n-2)$, ..., $*3$, $*2$, $*$.

The Tweedledum and Tweedledee Argument is that if two games have identical values and you are allowed to move in either but not both of them then if your opponent makes a move in one of these games make an equivalent move in the other. Since your opponent moves first he will be the first one unable to make a move, hence, lose.

For Nim played with three or more heaps the strategy becomes more complex. The value of each heap of stones is converted to a binary number and the strategy is to leave an even number of ones in each binary position (given that the position is not already that way).

An algorithm for playing Nim

1. Convert the number of stones in each heap to its binary representation.
2. Count the number of ones in each binary position of the representations created in 1. above.
3. If the number of ones in each binary position is an even number then make a random move and hope that your opponent makes an error.
4. If they are not all even then choose a heap that
    a. has a one in a binary position such that the number of ones in that position for all the heaps is odd;
    b. is the largest binary position satisfying a;
    c. is the largest heap satisfying b;
    d. in case of ties choose the first pile converted.

5. Reduce the heap chosen in 4. above so that the sum of the ones for any binary position taken over the binary representations of the size of the heaps is even.

Lemma 4.1. For a game with Nim-heaps such that the XOR sum of the nim-values of the heaps represented as binary numbers is not zero in every position it is always possible to reduce one heap so that the XOR sum of the nim-values of the heaps represented as binary numbers is zero in every position.

Proof.

If there is only one heap then that heap cannot be empty. Reduce the size of that heap to zero. There are no ones in the binary representation of the size of the heap and the condition is satisfied.

Now suppose that the condition does not hold and there are two heaps. By the conditions of the algorithm, both of these heaps cannot be equal, hence, one must be larger. Choose the largest bit that is not matched. Let this bit represent 2^k, $k \in \mathbb{N}$. By reducing this heap to $2^k - 1$, we reduce the number of ones in the kth position by one. By hypothesis, we had an odd number of ones in that position, so, by reducing the sum by one, we now have an even number of ones in that position. By our algorithm k was the largest position with an odd number of ones so we only have

to show that we can produce an even number of ones in the lower positions.

We can produce every bit pattern from 0 to 2^{k-1}, that is, we can match any bit pattern occurring is the 0th to (k-1)st position of the other binary representation and so produce an even number of ones in each position.

For three or more heaps, choose the heap to be reduced as above and consider the bit by bit sum of the ones in the other heaps for the 0th through (k-1)st binary positions. This is the bit pattern we have to match in the 0th to (k-1)st bit positions of the number chosen; and, as before, we have available all the bit patterns representing zero through $2^k - 1$, that is, all bit positions■

We note here that a Nim-heap is a fuzzy game, i.e., one in which the first player to move wins. [BCG pp.30-31] defines four outcome classes of games:

1. zero - In this class the first person to move loses. A game of Nim with two heaps with one bean in each heap and a game of Nim with a single heap with two beans in it is an example of a zero game. The first player to move must remove one bean from a heap and his opponent then removes the other leaving the first player with no beans to remove (another zero game), i.e., the first player loses.

2. positive - These are games in which the first player to move in the original game, say, Blue, moves but wins the game. An example of such a game would be Blue-Red Hackenbush with a pile of blue counters. Blue removes one blue counter from the pile and his opponent, Red, is unable to answer since Blue may only remove blue counters and Red may only remove red counters.

3. negative - This is the opposite situation. Red moves first and wins. There is a single pile of counters with value -1/8, that is, a red counter on the bottom on top of which are three blue counters. Red removes the red counter (together with the three blue counters on top of the red one). Blue is faced with no move and loses. The value of this game is negative since, if we want to find the value of a game composed of several different games we have to know the value of each individual game so as to get the sum of the games. The method of BCG is to reduce the game to a number and determine who wins by looking at that number. We should note that since this game is negative, Blue cannot win by moving first. Blue moves first and removes a Blue counter leaving a pile with a red counter on the bottom with two blue counters on top of it (value -1/4), red removes his counter leaving, once again, the empty game. Blue has no move and loses.

4. fuzzy - The first player to move wins. An example is a game with one heap with two beans in it. The first player

moves by removing both beans leaving the empty game. The second player has no move and loses.

GREEN HACKENBUSH, THE GAME OF NIM, AND NIMBERS [BCG p.42]

In Chapter 7 we shall give a complete theory for Hackenbush pictures that are entirely green, containing neither blue nor red edges. Of course the game represented by a green Hackenbush picture is an **impartial** one, in the sense that from any position exactly the same moves are legal for each player. There are several of our chapters (4,12-17) devoted to impartial games, which make it clear that the game of **Nim** plays a central role in the theory of such games. We shall introduce this game by analyzing some particularly simple green Hackenbush positions.

A very simple kind of green Hackenbush picture is the green snake, which consists of a chain of green edges with just one end touching the ground. It will not affect the play to bend some of the topmost edges into loops, so allowing our snakes to have heads. Figure 4a illustrates a number of snakes, those of length 1 being perhaps better be called blades of grass. How shall we play such a game?

Plainly any move will affect just one snake, and will replace that snake by a strictly shorter one. This means that if we write $*n$ for the value of a snake with n edges (counting the head loop, if present), then we have

$$*0 = \{\ |\ \},$$
$$*1 = \{*0|*0\} = \{0|0\},\ \text{the game we call } *,$$
$$*2 = \{*0,*1|*0,*1\} = \{\ 0,*|0,*\},\ldots,$$
$$*n = \{*0,*1,*2,\ldots,*(n-1)|*0,*1,*2,\ldots,*(n-1)\}.$$

These special values are called **nimbers** and you'll hear about them incessantly from now on. The fact that the same options appear on both sides of the | emphasizes the impartiality of the game.

It might be safer to play the game with heaps of counters instead of snakes. In this form, the general position has a number of heaps, and the move is to remove any positive number of counters from any one heap. In the normal play version, the winner is the person who takes the last counter. So this is the same

as the snake game, with a *n*-edge snake replaced by a heap of *n* counters, and Figure 4a becomes Figure 4b.

The game is the celebrated game of Nim, analyzed by C.L. Bouton, and we shall meet it again and again, for R.P. Sprague and P.M. Grundy showe (independently) that it implicitly contains the additive theory of all impartial games. ...

GET NIMBLE WITH NIMBERS! [BCG p.44]

Firstly, note that a single non-empty heap is fuzzy, for the first player to move can take the whole heap. In the Hackenbush form, he chops the bottom edge of the snake. Next, two heaps of equal size add up to zero, for the impartiality ensures that a position is its *own* negative. So any pair of equal heaps in a position may be neglected—this allows us to neglect all four blades of game in Figure 4a. On the other hand, the sum of two unequal heaps is a fuzzy game, for the first player can equalize them by reducing the larger one.

These remarks show that in a three-heap game, the player who first (fatally) equalizes two of the heaps or empties any heap is the loser, for in the first case his opponent can remove the third heap, and in the second equalize the two non-empty heaps. But in the position *1 + *2 + *3, every move of the first player loses for one of these reasons, and so *1 + *2 + *3 = 0. Since nimbers are their own negatives this can also be written in the forms

$$*1 + *2 = *3 \qquad *1 + *3 = *2 \qquad *2 + *3 = *1,$$

which is very useful in simplifying positions. For example, any situation in which there is one heap of size 2 and another of size 3 may be simplified by regarding these as a single heap of size 1.

From the position *1 + *4 + *5 if either player reduces one of the larger heaps to 2 or 3, the other player can reduce the other to 3 or 2 respectively. Since all of the other moves are fatal for one of our two reasons, this shows that *1 + *4 + *5 = 0, enabling us in general to replace two heaps iff and distinct size from 1,4,5 by one heap of the third size.

The equality *2 + *4 = *6 = 0 can be checked in a similar way. If either player reduces one of the larger heaps to 1 or 3, his opponent can reduce the other to the other, getting *2 + *1 + *3. The only other moves

not obviously fatal are to reduce 2 to 1 or 6 to 5, and these counter each other since *1 + *4 + *5 = 0.

We can now do some rather clever nimber arithmetic:

*3 + *5 = *2 + *1 + *5 = *2 + *4 = *6,

so we have another equality, representable in any of the ways

*3+*5 = *6 *3+*6 = *5 *5+*6 = *3 *3+*5+*6 = 0.

Later on we shall show that the sum of any two nimbers is another nimber, and give rules for working out which one it will be. But we have already more than to work out who wins the game of Figures 4a and 4b, and how. Since the four blades of grass can be neglected, the value of this is *5 + *6 + *4 = *3 + *4 which, being fuzzy, is a first-player win by reducing 4 to 3. So one winning move is to chop the head off the third snake, reducing his value from *4 to *3.

GAMES WITH HEAPS [BCG p.82]

Consider any game played with a number of heaps in which each move affects *just one of the heaps* on the table, and in which exactly the same moves are available to each player. Any position in such a game is therefore the sum of its single heap positions, so the game is solved when we know the value of a heap of n beans for every n. Moreover, since the games are *impartial*, each such value is a Nim-heap $*m$. In this chapter we'll usually omit the stars, so if heaps of sizes 0,1,2,3,... have values $*a,*b,*c,*d,...$ we shall say that the game has the **nim-sequence**

$$a.bcd...$$

(sometimes we omit the decimal point) and refer to

$$\mathcal{G}(0) = a, \quad \mathcal{G}(1) = b, \quad \mathcal{G}(2) = c, \quad \mathcal{G}(3) = d, \quad ...$$

as the **(nim-)values**. Using the information contained in the nim-sequence, we can analyze any position.

The nim-value of the sum of heaps of sizes

$$i, \quad j, \quad k$$

is the nim-sum

$$\mathcal{G}(i) \overset{*}{+} \mathcal{G}(j) \overset{*}{+} \mathcal{G}(k) \overset{*}{+} \ldots$$

and each nim-value, g(n), is computed as the least one of 0,1,2,3,... that is *not* the nim-value of any option from the heap of size n. ... we shall call the least number (from 0,1,2,3,...) which is missing from a set {x,y,z,...} the mex (**min**imum **ex**cluded number) of the set. Thus

$$\text{mex}(0,1,3,7) = 2, \quad \text{mex}(2,4,5) = 0$$

and the mex of the empty set is 0.

P-POSITIONS AND *N*-POSITIONS [BCG p.83]

Impartial games can only have *two* outcome classes which we call

P-positions (Previous player winning), and
N-positions (Next player winning).

In this chapter we'll frequently be working with nimbers, so you'll need to know that

a value of 0 indicates a *P*-positions,

while

values *,*2,*3, ... indicate *N*-positions.

No other value is possible for an impartial game. But remember that if you're going to add up your games, you'll have to know the exact value, and not just the outcome class, of each component.

SUBTRACTION GAMES [BCG p.83]

We might modify the game of Nim by requiring that in any move the number of beans taken is at most three. This will mean that for the nim-values we have

$$g(n) = \text{mex}(g(n-1), g(n-2), g(n-3)),$$

so that the nim-sequence is

n =	0	1	2	3	4	5	6	7	8	9 ...
g(n) =	0.	1	2	3	0	1	2	3	0	1 ...

and a single heap is a *P*-position (previous player winning) just if its size is a multiple of 4. We could instead allow a heap to be reduced by any number up to k, when the nim-sequence would be

n =	0	1	2	...	k-1	k	k+1	k+2	...	2k	2k+1	2k+2	2k+3	...
g(n) =	0.	1	2	...	k-1	k	0	1	...	k-1	k	0	1	...

and a single heap would be a *P*-position just if its size were a multiple of k+1.

These results are known and easily discovered so that it might be wise to play a game in which a heap may be reduced only by taking 2, 5 or 6 beans from it. In this case

$$g(n) = \text{mex}(g(n-2), g(n-5), g(n-6))$$

and the nim-sequence is found to be

n =	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	...
$g(n)$ =	0.	1	1	0	2	1	3	0	2	1	0	0	1	1	0	2	1	3	...	

where the underlines indicate that the first eleven values repeat indefinitely, so a single heap is a P-position only if its size is congruent to 0,1,4 or 8 modulo 11. But of course we can also analyze positions with arbitrarily many heaps. Let us find all the winning moves from the position with three heaps of sizes

5, 7, 9 values 2, 3, 2.

In Nim, the winning moves from the position

2, 3, 2

are to

1, 3, 2 *or* 2, 0, 2 *or* 2, 3, 1.

So by subtracting 2, 5 or 6 we must achieve a change from

the 5-heap to one of value 1,
that is a 3-heap (5 - 2 = 3),

or the 7-heap to one of value 0,
that is a 1-heap (7 - 6 = 1),

or the 9-heap to one of value 1,
that is a 3-heap (9 - 6 = 3).

More generally, for any **subtraction set**

{$s1$, $s2$, $s3$,...}

we can define the corresponding **subtraction game** $S(s1,s2,s3,...)$ in which a heap may be reduced only by one of the numbers $s1,s2,s3,...$

Given the game G3 = (V, C, B)

where V = { (-1), (-2), (-4) }

B = (n), n a nonnegative integer

we compute its nim-sequence as follows:

g(0) = 0 since mex ϕ = 0, there is no move

g(1) = 1 since mex { 0 } = 1, we can move to 0

g(2) = 2 since mex { 0, 1 } = 2, we can move to 0 or 1

g(3) = 0 we can move to 3 - 1 = 2 or 3 - 2 = 1, so,
mex { 1, 2 } = 0

g(4) = 1 mex { 4-4, 4-2, 4-1 } = mex { 0, 2, 3 } =
mex { 0, 2, 0 } = mex { 0, 2 } = 1

g(5) = 2 mex { 5-4, 5-2, 5-1 } = mex { 1, 3, 4 } =
mex { 1, 0, 1 } = mex { 0, 1 }

n =	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	...
g(n) =	0.	1	2	0	1	2	0	1	2	0	1	2	0	1	2	0	1	2	...

after analysis we would find that the set of all positions are broken up into three equivalence classes modulo three and that the set P of P-positions (Previous player wins) is equal to the set of board states equivalent to zero mod three.

$P\ (\ G3\ ) = \{\ n = 0\ (\ \mathrm{mod}\ 3\ ),\ n \in \mathbb{N}\ \}$

Kayles is an example of a taking and breaking game. A move consists of removing one or two tokens from a Nim-heap with the option of dividing the remaining tokens into two Nim-heaps. The representation of Kayles as a string rewriting system is shown below.

Kayles (B, R, C)

B is the initial position having the form

[L|R] (AA*B)*

R is a set of rules or production system

L(AA*B)*AA*B(AA*B)* --> R(AA*B)*A*BA*B(AA*B)*

L(AA*B)*AAA*B(AA*B)* --> R(AA*B)*A*BA*B(AA*B)*

R(AA*B)*AA*B(AA*B)* --> L(AA*B)*A*BA*B(AA*B)*

R(AA*B)*AAA*B(AA*B)* --> L(AA*B)*A*BA*B(AA*B)*

C is the control system or set of metarules which acts as the referee. Under the normal ending condition if L is the prefix of a string that cannot be rewritten then L is the loser and R is the winner. Under the misère ending condition L is the winner and R the loser.

A classification system for taking and breaking games [attributed to Guy and Smith (1956), CG p.56] is given on page 92 of Winning Ways.

A taking and breaking game is designated by a string

.d1 d2 d3 d4 ...

where dk is the encoding of taking k beans from a heap, and the value of *dk* is the conditions associated with the removal of k beans from the heap.

Value of d_k	Conditions for the removal of *k* beans from a single heap.
0	Not permitted.
1	If the beans removed are the whole heap.
2	Only if some beans remain and are left as a single heap.
3	Only if the remaining beans, if any, are left in one heap.
4	Only if some beans remain and are left as exactly two nonempty heaps.
5	Provided the remaining beans if any are left as two nonempty heaps.
6	Only if some beans remain and are left in one or two heaps.
7	Provided the remaining beans if any are left in at most two heaps.
8	Only if some beans remain and are left in just three nonempty heaps.
etc.	

Table 4. Interpretation of Code Digits in Take and Break Games [BCG P.92]

Kayles would be classified as .77 or .77000...

Two additional examples of take and break games are .07, Doublecross, and, .007, Treblecross. A move in .07 is the removal of exactly two tokens from a Nim-heap with the option of splitting the remaining tokens in that heap into two Nim-heaps. A move in .007 is the removal of exactly three tokens with the option of splitting the tokens in that heap into two Nim-heaps. The representation of both .07 and .007 as string rewriting systems are shown below.

.07 (B, R, C)

B is a string of the form [L|R](A*B*)*

R is a set of productions or rewriting rules

L(A*B*)*AA(A*B*)* --> R(A*B*)*BB(A*B*)*

R(A*B*)*AA(A*B*)* --> L(A*B*)*BB(A*B*)*

C is a control structure which acts as referee. Under the normal ending condition, if L is the prefix of a string that cannot be rewritten then L is the loser and R the winner.

.007 (B, R, C)

B is a string of the form [L|R](A*B*)*

R is a set of productions or rewriting rules

L(A*B*)*AA(A*B*)* --> R(A*B*)*BB(A*B*)*

R(A*B*)*AA(A*B*)* --> L(A*B*)*BB(A*B*)*

A program for determining the nim-value of a single Nim-heap in .07 is shown below. By changing the value of tile to 3 and adjusting the initial values, a program for analyzing .007 is produced. Output for values for both games are given.

```
* FINDS THE GRUNDY NUMBERS OF POSITIONS IN .007
      INTEGER LMT
      PARAMETER ( LMT = 15 )

      INTEGER GV (0:LMT)
      LOGICAL SET(0:LMT)

      INTEGER NUM
      INTEGER TILE
      PARAMETER ( TILE = 2 )

      INTEGER I, J, K, L

      INTEGER  EVAL, MEX
      EXTERNAL EVAL, MEX

      DATA (GV(I), I = 0, 3) / 0, 0, 1, 1 /

      DO 5 NUM = 4, LMT
```

```
            CALL CLEAR ( SET, LMT )

            DO 15 I = NUM - TILE, ( NUM - TILE + 1 )/2, -1

               J = NUM - TILE - I
               K = EVAL ( GV, N, I, J )
               SET(K) = .TRUE.

15          CONTINUE

            L = MEX ( SET, LMT )
            GV(NUM) = L

5       CONTINUE

        DO 105 I = 0, LMT

           WRITE  ( 6, 1000 ) I, GV(I)
1000       FORMAT ( 1X, T5, I3, T10, I5 )

105     CONTINUE

        END


        SUBROUTINE CLEAR ( SET, LMT )

        INTEGER LMT
```

```
      LOGICAL SET (0:LMT)

      INTEGER I

      DO 5 I = 0, LMT
         SET(I) = .FALSE.
5     CONTINUE
      END

      INTEGER FUNCTION EVAL ( GV, N, A, B )
      INTEGER N
      INTEGER GV(0:N)
      INTEGER A, B

      INTEGER GVA, GVB, PARA, PARB
      INTEGER SUM, RES, VAL

      GVA = GV(A)
      GVB = GV(B)

      SUM = 0
      VAL = 1

5     IF ( GVA.GT.0 .AND. GVB.GT.0 ) THEN
```

```
      PARA = MOD ( GVA, 2 )
      PARB = MOD ( GVB, 2 )

      IF ( PARA.NE.PARB ) THEN
         SUM = SUM + VAL
      ENDIF

      GVA = GVA/2
      GVB = GVB/2

      VAL = VAL + VAL
      GO TO 5

   ENDIF

   RES = MAX ( GVA, GVB )

15 IF ( RES.GT.0 ) THEN

      IF ( MOD(RES,2).NE.0 ) THEN
         SUM = SUM + VAL
      ENDIF

      RES = RES / 2
      VAL = VAL + VAL
```

```
      GOTO 15

   ENDIF

   EVAL = SUM

   END

   INTEGER FUNCTION MEX ( SET, N )

   INTEGER N
   LOGICAL SET (0:N)

   INTEGER I

   DO 5 I = 0, N

      IF ( .NOT. SET(I) ) THEN

         MEX = I
         RETURN

      ENDIF

5     CONTINUE
```

```
END
```

A TABLE OF GRUNDY VALUES FOR .07

Block Size	Grundy Value
0	0
1	0
2	1
3	1
4	2
5	0
6	3
7	1
8	1
9	0
10	3
11	3
12	2
13	2
14	4
15	0

A TABLE OF GRUNDY VALUES FOR .007

Block Size	Grundy Value
0	0
1	0
2	1
3	1
4	1
5	2
6	0
7	2
8	3
9	1
10	1
11	4
12	0
13	4
14	5
15	1

The program for .07 follows the classic algorithm for determining the nim-value of a position. The value of the position is the MEX of the nim-values of all the possible moves from that position. Each of the separate nim-values is computed by applying the Sprague-Grundy Function to the two (one or more possibly empty) heaps remaining after removing two tokens from a heap of size at least two. The values are computed iteratively from the initial values given.

We introduce the Rook Game and use it to illustrate the calculation of the Sprague-Grundy Function, g(n), for each of the board positions.

THE ROOK GAME

A rook is placed on an arbitrary square of an n by n chessboard, $n \in \mathbb{Z}^+$, with the rows and columns indexed 0, 1, 2, ... starting at the upper left-hand corner [Figure 5]. Two players, Left and Right, alternately move the rook towards the upper left-hand corner. Left moves first. The first player unable to move loses, his opponent wins.

The square with row-major coordinates (0,0) has nim-value 0 since the set of options (options available to both players) is the empty set. From either (0,n) or (n,0), $n > 0$, a rook can only move respectively in a column or in a row. For an arbitrary n, $n > 0$, the set of options available are:

{ *0, *1, *2, *3, ..., *(n-1) | *0, *1, *2, *3, ..., *(n-1) }

which is to say that the rook can move to any position with a smaller coordinate along the column (respectively, row). Thus the Minimum EXcluded number in this set is n, which is the value of the Sprague-Grundy Function, g(x,y), for the given positions. The analysis of the second shell begins with an analysis of the board position (1,1):

g(1,1) = mex { *1 | *1 } = 0.

For position (2,1) we have:

g(2,1) = g(1,2) = mex { *0, *1, *2 | *0, *1, *2 } = 3;

and for position (3,1) we have:

g(3,1) = g(1,3) = mex { *0, *1, *3 | *0, *1, *3 } = 2.

The game is revealed to be equivalent to Nim with two heaps and the winning strategy to move the rook to positions with equal coordinates. These positions are sometimes referred to as safe positions and are characterized as those positions that have a nim-value of 0.

A TABLE OF GRUNDY VALUES FOR THE GAME OF ROOK PLAYED ON A STANDARD 8x8 CHESSBOARD UNDER THE NORMAL ENDING CONVENTION.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	0	1	2	3	4	5	6	7
1	1	0	3	2	5	4	7	6
2	2	3	0	1	6	7	4	5
3	3	2	1	0	7	6	5	4
4	4	5	6	7	0	1	2	3
5	5	4	7	6	1	0	3	2
6	6	7	4	5	2	3	0	1
7	7	6	5	4	3	2	1	0

Theorem 4.1. Let X, Y ϵ $\mathbb{N}$ and (X,Y) be a position in a two-heap Rook's Game then the Grundy number associated with the pair (X,Y) $\leq$ X + Y.

Proof.

Denote by {x} the set of Grundy values of the positions (x_i,Y), i = 0, X-1. Denote by {y} the set of Grundy values of the positions (X,y_i), i = 0, Y-1. The Grundy value of (X,Y), g(X,Y), is given by

$$g(X,Y) = \text{mex } \{x\} \cup \{y\}$$

If X equals zero then $g(X,Y) = \text{mex } \phi \cup \{y\} = \text{mex } \{y\}$; if both X and Y are both zero then $g(X,Y) = \text{mex } \phi = 0$.

Since the Grundy values are symmetric, WLOG, let X $\leq$ Y.

By definition of the game and the normal ending rule we have $g(0,0) = 0$. By simple induction it can shown that $g(X,0) = g(0,X) = X$, where $X \in \mathbb{N}$.

Suppose the theorem is false. Since the theorem is false there must be a counter example. Let (A,B) be a minimal pair such that the theorem is false, i.e., $g(A,B) > A + B$ and $A + B$ is minimal. So, $g(A,B) > A + B \Rightarrow mex[\{a\} \cup \{b\}] > A + B \Rightarrow \{a\} \cup \{b\}$ contains pair (C,D) such that $g(C,D) = A + B$. Now, $A > 0$ and $B > 0$; so, $C + D < A + B$ and $g(C,D) = A + B$ which contradicts the minimality of (A,B) ■

The following theorem gives us a tighter bound.

Theorem 4.2.

Let $X, Y \in \mathbb{N}$ and (X,Y) be a position in a two-heap Rook's Game then $g(X,Y) = MAX(X + Y - 1, 0)$.

Proof.

Denote by $\{x\}$ the set of Grundy values of the positions (x_i,Y), $i = 0, X-1$. Denote by $\{y\}$ the set of Grundy values of the positions (X,y_i), $i = 0, Y-1$. The Grundy value of (X,Y), $g(X<Y)$, is given by

$$g(X,Y) = mex\ \{x\} \cup \{y\}$$

If X equals zero then $g(X,Y) = mex\ \phi \cup \{y\} = mex\ \{y\}$; if both X and Y are both zero then $g(X,Y) = mex\ \phi = 0$. The last equality proves the lower bound.

Since the Grundy values are symmetric, WLOG, let $X \leq Y$.

By definition of the game and the normal ending rule we have $g(0,0) = 0$. By simple induction it can shown that $g(X,0) = g(0,X) = X$, where $X \in \mathbb{N}$.

(1) $\text{mex } \phi = 0 \leq g(X,Y) = \text{mex } \{x\} \cup \{y\}$.

Now,

(2) $\text{mex } \{x\} \cup \{y\} \leq X + Y - 1$.

Since, $|\{x\} \cup \{y\}| = |\{x\}| + |\{y\}| - |\{x\} \cap \{y\}|$,
and, there are X integers in {x} and Y integers in {y}, and one of these integers must be 0 by definition of the Mex Rule■

KNIGHT

A knight is placed on a chessboard [Figure 6]. There are two players, Left and Right, who alternately move the knight towards the Northwest corner of the chessboard. Left moves first. The first player unable to move loses and his or her opponent wins.

A knight moves two squares in one direction and one square in the other. Label the rows and columns of the chessboard 0, 1, 2, 3, ... starting from the Northwest corner of the board. The position of the knight can be thought of as two heaps of stones and the moves consist of removing two stones from one heap and one stone from the other or removing two stones from one heap and adding one

stone to the other. The total number of stones is decreased at each move.

The Grundy values for Knight played on a standard 8x8 chessboard are displayed below.

TABLE OF GRUNDY VALUES FOR THE GAME OF KNIGHT
PLAYED ON AN 8x8 CHESSBOARD WITH BORDERS : UNDER THE STANDARD ENDING CONDITION

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	0	0	1	1	0	0	1	1
1	0	0	2	1	0	0	1	1
2	1	2	2	2	3	2	2	2
3	1	1	2	1	4	3	2	3
4	0	0	3	4	0	0	1	1
5	0	0	2	3	0	0	2	1
6	1	1	2	2	1	2	2	2
7	1	1	2	3	1	1	2	0

A grid of Grundy values was produced using the Mex Rule. The table is symmetric and composed of three 4 by 4 matrices and two inverses. A FORTRAN function computing the Grundy value of a position of Knight appears below.

A GRID OF GRUNDY VALUES FOR THE KNIGHT'S GAME ON AN UNRESTRICTED BOARD

	0123	4567	11 8901	1111 2345	1111 6789	2222 0123	2222 4567	2233 8901	3333 2345
0	0011	0011	0011	0011	0011	0011	0011	0011	0011
1	0021	0011	0011	0011	0011	0011	0011	0011	0011
2	1222	2222	3222	3222	3222	3222	3222	3222	3222
3	1121	4323	3323	3323	3323	3323	3323	3323	3323
4	0034	0011	0011	0011	0011	0011	0011	0011	0011
5	0023	0021	0011	0011	0011	0011	0011	0011	0011
6	1122	1222	3222	3222	3222	3222	3222	3222	3222
7	1123	1121	4323	3323	3323	3323	3323	3323	3323
8	0033	0034	0011	0011	0011	0011	0011	0011	0011
9	0023	0023	0021	0011	0011	0011	0011	0011	0011
10	1122	1122	1222	3222	3222	3222	3222	3222	3222
11	1123	1123	1121	4323	3323	3323	3323	3323	3323
12	0033	0033	0034	0011	0011	0011	0011	0011	0011
13	0023	0023	0023	0021					
14	1122	1122	1122	3222	3223				
15	1123	1123	1123	1121					

WHICH FORMS THE PATTERN

	0:3	4:7	8:11	12:15	16:19	20:23	24:27	28:31	32:35
0: 3	A	B	C	C	C	C	C	C	C
4: 7	Bt	A	B	C	C	C	C	C	C
8:11	Ct	Bt	A	B	C	C	C	C	C
12:15	Ct	Ct	Bt	A	B	C	C	C	C
16:19	Ct	Ct	Ct	Bt	A	B	C	C	C
20:23	Ct	Ct	Ct	Ct	Bt	A	B	C	C
24:27	Ct	Ct	Ct	Ct	Ct	Bt	A	B	C
28:31	Ct	Ct	Ct	Ct	Ct	Ct	Bt	A	B
32:35	Ct	Ct	Ct	Ct	Ct	Ct	Ct	Bt	A
36:39	Ct	Ct	Ct	Ct	Ct	Ct	Ct	Ct	Bt

A Fortran function that takes advantage of the symmetry of the unrestricted board appears below

```
      INTEGER FUNCTION KNIGHT ( R, C )

      INTEGER R, C
      INTEGER RS, CS, RM, CM, D

      INTEGER I, J

      INTEGER TABLE1( 0:3, 0:3 )
      INTEGER TABLE2( 0:3, 0:3 )
      INTEGER TABLE3( 0:3, 0:3 )

      SAVE TABLE1, TABLE2, TABLE3

      DATA ( ( TABLE1( I, J ), J = 0, 3 ), I = 0, 3 )
     +     / 0, 0, 1, 1, 0, 0, 2, 1, 1, 2, 2, 2, 1, 1, 2, 1 /

      DATA ( ( TABLE2( I, J ), J = 0, 3 ), I = 0, 3 )
     +     / 0, 0, 1, 1, 0, 0, 1, 1, 3, 2, 2, 2, 4, 3, 2, 3 /

      DATA ( ( TABLE3( I, J ), J = 0, 3 ), I = 0, 3 )
     +     / 0, 0, 1, 1, 0, 0, 1, 1, 3, 2, 2, 2, 3, 3, 2, 3 /

      RS = R/4
      CS = C/4

      RM = MOD ( R,4 )
      CM = MOD ( C,4 )

      IF ( RS.GT.CS ) THEN

           CALL SWAP ( RS, CS )
           CALL SWAP ( RM, CM )

      ENDIF

      D = CS - RS

      IF ( D.EQ.0 ) THEN

           KNIGHT = TABLE1( RM, CM )

      ELSEIF ( D.EQ.1 ) THEN

           KNIGHT = TABLE2( RM, CM )

      ELSE
```

```
        KNIGHT = TABLE3( RM, CM )
    ENDIF
    END
```

Knight can be presented as a string rewriting system

Let Knight1 = (Σ, R, C, B)

where Σ = { a, b }

R = { (aab, λ), (abb, λ), (aab, bb),
(abb, aa) } is the set of production rules

C is the control unit

and B = { $a^m \| b^n$, m, n ϵ $\mathbb{N}$ } is the board state.

The control system is given as follows:

Two players, L and R, move alternately. L moves first. The first player unable to move loses. A move consists of choosing a rule from R and applying it to the string B.

or, as a vector addition system

Knight2 = (V, C, B)

where V = { (1, -2), (-2, 1), (-1, -2),
(-2, -1) }

B = (m, n), m, n ϵ $\mathbb{N}$

C = the control unit

The control unit is given as follows:

Two players, L and R, move alternately. L moves first. The first player unable to move loses. A move consists of choosing a vector from V and applying it one time to the vector B with the constraint that at no time during a move can any component of the vector be negative.

In White Knight, the idea of a vector addition game is generalized.

The White Knight (Lewis Carroll's) rides towards home carrying a pile of packages [Figure 7] losing packages along the way. Two players, Left, L, and Right, R, alternately either move the knight or drop (lose) some packages. Left moves first. The first player unable to move loses. This occurs when the knight has reached home and has lost all his packages.

$$G4 = (V, C, B)$$

$$B = (x, y, z) \text{ with } x, y, z \in \mathbb{N}$$

$$V = \{ (-1, -2, 0), (-2, -1, 0), (1, -2, 0), (-2, 1, 0), (0, 0, -1) \}$$

and C is as follows:

Two players Left and Right move alternately. Left moves first. The first player unable to move loses.

Moves consist of type one and type two moves.

A type one move is: $V - \{ (0, 0, -1) \}$

A type two move is: $\{ (0, 0, -1) \}$

To move a player

1. Chooses a vector from V.

2. If the vector is a type one move, the player applies it exactly one time.

3. If the vector is a type two move, the player applies it one or more times.

Application of the chosen vector is restricted by the requirement that after the application of a rule (vector) no component of B, i.e., x, y, or z, is less than zero.

The result of these rules is that the sum of the components of the vector B is reduced at each step, i.e.,

$$(x(n+1) + y(n+1) + z(n+1)) < (x(n) + y(n) + z(n))$$

which leads to confluent rewriting systems.

Note that the components x and y are interrelated in that they appear in type one moves while z appears only in type two moves. A move is either a type one move or a type two move. This suggests breaking White Knight into two separate games, Knight and Packages, and taking their sum.

The x and y components of the game White Knight becomes the game Knight in which a knight is placed on an arbitrary square of an n x m chessboard, $m,n \in Z^+$, and moves towards, say, the Northwest corner of the board with the restriction that the sum of the two components of the vector B are reduced by the knight's move; and, a game in which a player

may remove one or more packages from a heap of $z \in \mathbf{Z}^+$ packages. The players, L and R, move alternately. A player, at his turn, must make a move in one game but may not make a move in both games. L moves first. The first player unable to make a move in either game loses.

White Knight is the sum of the games we call Knight and Packages. Thus, the safe positions in White Knight are exactly those positions in which the Grundy value of the sum of these two games is zero.

The sum of the games which we will call Knight and Packages is found by taking the nim-sum of the two games. To get the nim- sum of the two games we must first compute the nim-value of each of these games separately and then take the sum of the two nim-values using vector addition modulo 2.

The value of the game Packages is the nim-value of a single nonnegative integer, n, under the rule that n can be reduced by any positive integer with the restriction that n is always greater than or equal to zero. This can be seen by applying the mex (minimal excluded integer) rule:

$$\text{nim-value of } n = \text{mex } \{ 0, 1, 2, \ldots , n - 2, n -1 \}$$

The nim-value of the game Knight with the knight on the square (x, y) is the nim-value of the position (x, y), x,y, nonnegative integers, is calculated by determining the value of a chessboard square by square (using the Mex Rule

to build a grid of Grundy values) or by using a recursive formula as in the last section on Knight.

It is the problem of determining the value of the present position by determining the value of all future positions arising from the current position, literally calculating all the values of the positions at the bottom of the game tree that are descendants of the current position and combining these values to get the value of the current position, that suggested looking at alternate ways to represent games.

Given the nim-value of Knight and the nim-value of Packages what we do is convert both numbers to their binary representation and Logical OR the two numbers bit by bit. If the result is zero, make a random move and hope that your opponent makes a mistake; otherwise, subtract and add bits appropriately to the larger of the two numbers so that the bit by bit logical Oring of the new numbers (vector addition over GF(2)) is equal to zero.

Example:

The nim-value of Knight is 4

The nim-value of Packages is 10

4	00100
10	01010
6	00110

The winning move in this case is to discard 6 packages. This leaves two games with the nim-value 4 with the other player on move. Whatever the other player does in one game, you do in the other (Tweedledum and Tweedledee Argument). Since each move reduces a nim-value the game ends with each game having the nim-value 0 with your opponent on move, i.e., you win.

White Knight has the following presentation as a string rewriting game:

G5 = (Σ, R, C, B)

where Σ = { a, b, c }

B is a member of the set a+b+c*

R is the union of the following sets of transitions:

S1 = { (aab, a), (abb, b),
(aab, b), (abb, b) }

S2 = { (c, λ) }

and C is

There are two players, Left and Right.
The two players move alternately. Left moves first.
A move consists of choosing a production from either S1 or S2 and applying the production (rule) exactly one time if it is from set S1 or applying the production one or more times if it is from set S2. The first player unable to move loses. The other player wins.

Between Nim for which a polynomial time algorithm to determine a winning strategy is known and Wythoff's nim for which a polynomial time algorithm is known for the version with two heaps but not known for heaps of size three or greater there are games that Fraenkel [ASF2] calls Nimhoff Games. The King-Rook Game is described below.

THE KING-ROOK GAME

A king-rook (a fairy chesspiece) is placed on an arbitrary square of an n X n ($n \in \mathbb{Z}+$) chessboard. The king-rook has the power of both a king and a rook and moves towards the Northwest corner of the board [Figure 8]. That is, the king-rook may move along its row towards the west or along its column towards the north or one square towards the northwest along its diagonal. Two players, Left and Right, take turns moving the king-rook. The players move alternately. Left moves first. The first player unable to move loses; his opponent wins.

Fraenkel notes [ASF2, p.3] that the King-Rook Game is a disguised form of Nim which can be analyzed from its grid of Grundy values. b_1, b_2 denote binary representations of row and column.

"We can see that the grid can be partitioned into 3x3 blocks. Every block in the cyclic permutation of $3m$, $3m + 1$, $3m + 2$ ($m \geq 0$). Moreover, if we replace every block with the corresponding m, we get the grid of Nim, which is

depicted in [THE ROOK'S GAME]. Hence the formula for the g-function is $g(b_1,b_2) = 3(\lfloor b_1/3 \rfloor \oplus \lfloor b_2/3 \rfloor) + (b_1 + b_2)(3)$."

A program for obtaining the Grundy values for the positions of a king-rook on a two dimensional chessboard along with the output is given below.

```
 * FINDS THE GRUNDY VALUES FOR THE TWO-HEAP KING-ROOK GAME

      INTEGER LMT
      PARAMETER ( LMT = 40 )

      INTEGER VALUES( 0:LMT, 0:LMT )
      LOGICAL INSET ( 0:LMT )

      INTEGER I, J, K, N, R, C
      INTEGER MEX

* THE VALUES FOR A SINGLE HEAP ARE KNOWN

      DO 5 I = 0, LMT

        VALUES( 0, I ) = I
        VALUES( I, 0 ) = I

5     CONTINUE
```

```
* FIND THE VALUES FOR EACH SIZE OF THE GAME N > 0

      DO 100 N = 2, LMT

        DO 10 J = 1, N/2

          K = N - J

*         FOR EACH PARTITION OF N INTO TWO NONEMPTY PARTS
*         FIND THE GRUNDY VALUE ASSOCIATED WITH THAT PAIR

*         INITIALLY THE SET OF GRUNDY VALUES ASSOCIATED WITH
THE
*         OPTIONS AVAILABLE FOR THAT PAIR ARE EMPTY

          DO 15 I = 0, LMT

             INSET(I) = .FALSE.

15        CONTINUE

*         FIND THE GRUNDY VALUES ASSOCIATED WITH THE ROOK

          DO 20 R = 0, J - 1

             INSET( VALUES( R, K ) ) = .TRUE.
```

```
20          CONTINUE

            DO 25 C = 0, K - 1

               INSET( VALUES( J, C ) ) = .TRUE.

25          CONTINUE

*           THEN FIND THE GRUNDY VALUE ASSOCIATED WITH KING'S
*           MOVE

            INSET ( VALUES( J-1, K-1 ) ) = .TRUE.

*           THE VALUE OF THE POSITION IS THE MEX OF INSET

            MEX = 0

30          IF ( INSET( MEX ) ) THEN

               MEX = MEX + 1
               GOTO 30

            ENDIF

            VALUES( J, K ) = MEX
            VALUES( K, J ) = MEX
```

```
10       CONTINUE

100    CONTINUE

       WRITE ( 6, 1000 ) ( I, I = 0, 30 )
1000   FORMAT( 1X, T8, 31I4 / )

       DO 50 R = 0, 30

          WRITE ( 6, 2000 ) R, ( VALUES( R, C ), C = 0, 30 )
2000      FORMAT( 1X, T3, I4, T8, 31I4 )

50     CONTINUE

       END
```

TABLE OF GRUNDY VALUES FOR THE KING-ROOK GAME

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	1	2	3	4	5	6	7	8	9	10	11	12
1	1	2	0	4	5	3	7	8	6	10	11	9	13
2	2	0	1	5	3	4	8	6	7	11	9	10	14
3	3	4	5	0	1	2	9	10	11	6	7	8	15
4	4	5	3	1	2	0	10	11	9	7	8	6	16
5	5	3	4	2	0	1	11	9	10	8	6	7	17
6	6	7	8	9	10	11	0	1	2	3	4	5	18
7	7	8	6	10	11	9	1	2	0	4	5	3	19
8	8	6	7	11	9	10	2	0	1	5	3	4	20
9	9	10	11	6	7	8	3	4	5	0	1	2	21
10	10	11	9	7	8	6	4	5	3	1	2	0	22
11	11	9	10	8	6	7	5	3	4	2	0	1	23
12	12	13	14	15	16	17	18	19	20	21	22	23	0
13	13	14	12	16	17	15	19	20	18	22	23	21	1
14	14	12	13	17	15	16	20	18	19	23	21	22	2

15	15	16	17	12	13	14	21	22	23	18	19	20	3
16	16	17	15	13	14	12	22	23	21	19	20	18	4
17	17	15	16	14	12	13	23	21	22	20	18	19	5
18	18	19	20	21	22	23	12	13	14	15	16	17	6
19	19	20	18	22	23	21	13	14	12	16	17	15	7
20	20	18	19	23	21	22	14	12	13	17	15	16	8
21	21	22	23	18	19	20	15	16	17	12	13	14	9
22	22	23	21	19	20	18	16	17	15	13	14	12	10
23	23	21	22	20	18	19	17	15	16	14	12	13	11
24	24	25	26	27	28	29	30	31	32	33	34	35	36
25	25	26	24	28	29	27	31	32	30	34	35	33	37
26	26	24	25	29	27	28	32	30	31	35	33	34	38
27	27	28	29	24	25	26	33	34	35	30	31	32	39

POKER-NIM [BCG p.55]

This game is played with heaps of Poker-chips. Just as in ordinary Nim, either player may reduce the size of any heap by removing some of the chips. But now we allow a player the alternative move of increasing the size of the heap by adding to it some of the chips he has acquired in earlier moves. These two kinds of moves are the only ones allowed.

Let's suppose there are three heaps, of sizes 3,4,6 ..., and that the game have been going on for some time, so that both players have accumulated substantial reserves of chips. It's Left's turn to move, and he moves to 2,4,6 since he remembers from [nimbers, above] that this is a good move in ordinary Nim. But now Right adds 50 chips to the 4 heap, making the position into 2,54,6, which well beyond those [discussed above].

This seems somewhat disconcerting, especially since Right has plenty more chips at his disposal, and doesn't seem too scared of using them to complicate the position. What does Left do? After a moment's thought, he just removes the 50 chips Right has just added and waits for Right's reply. If Right adds 1000 chips to one of the heaps, Left will remove them and restore the position to 2,4,6 once again. Sooner or later, Right must reduce one of the three heaps (since otherwise he'll run out of chips no matter how many he has), and then Left can reply with the appropriate Nim-move.

So whoever can win a position in ordinary Nim can still win in Poker-Nim, no matter how many chips his opponent has accumulated. He replies to the opponent's reducing moves just as he would in ordinary Nim, and reverses the effect of any increasing move by using a reducing move to restore the heap to the same

size again. The new moves in Poker-Nim can only postpone defeat, not avoid it indefinitely. Since the effect of any of the moves can be immediately reversed by the other player, we call them **reversible moves.**

NORTHCOTT'S GAME [BCG p.56]

The same sort of thing happens in other games, often in better disguise. Northcott's game is played on a checkerboard which has one black and one white piece on each row, as in Figure 9. You may move any piece of your own color to any other empty square in the same row, provided you do not jump over your opponent's piece in that row. If you can't move (because all your pieces are trapped at the side of the board by your opponent's), you lose.

This can seem to be an aimless game if you don't see the point, and indeed it usually goes on forever if it is played badly. But when you realize that it's only Nim in disguise once more, you'll soon be able to beat anybody pretty quickly. To the left of the board in Figure 9 we have shown the numbers of spaces between the two pieces in each row. When someone moves, just one of these numbers will be changed, and might be either increased of decreased, according as the move was retreating or advancing. But just as in Poker-Nim, any moves increasing one of the numbers can be removed by the next player and so are not much use.

Who wins in Figure 9? We can see the zero-position 2,4,6 among the numbers shown, and of course the two numbers 3 together form another zero. Neglecting the two rows that are already 0, the only other number is 5, and we maintain that the first player can win by moving so as to reduce 5 to 0. Whenever the other player enlarges some gap by retreating, the first player should reduce it again by the same extent. In fact the first player should *always advance* on his opponent, *never retreat*.

It should not be thought that the moves we advise are the only good ones. For example, from Figure 9. instead of reducing 5 to 0, we could replace 6 by 3, 4 by 1 or even 3 by 6 in the second row (for White) or 0 by 5 in the last row (for Black). In fact it will help to avoid revealing the strategy if you do not always reply to a retreating move by the corresponding advance— for similar reasons occasiona retreating moves might be desirable.

BOGUS NIM-HEAPS AND THE MEX RULE [BCG p.57]

Consider the impartial game

$$G = \{*0,*1,*2,*5,*6,*9 \mid *0,*1,*2,*5,*6,*9\}.$$

This is a new kind of Nim-heap from which either player can move to a heap of size 0, 1, 2, 5, 6, or 9. In other words, we can regard it as a rather peculiar Nim-heap of size 3 (the first missing member), from which, as well the usual moves to heaps of sizes 0, 1 or 2, we are allowed to move to a heap of size 5 or 6 or 9. However, the Poker-Nim argument shows that this extra freedom is in fact of no use whatever.

To be more precise, suppose some player has a winning strategy in the game *3 + H + K + ... Then in the same circumstances, he has one in G + H + K .... When his strategy calls for a move from any of *3,H,K, ..., that move is still available, and he need not use the new permitted moves from G to *5, *6, or *9. If his opponent tries to do so, he can immediately reverse the effect of this move by moving back to *3 (since 5, 6, and 9 are greater than 3), and revert to the original strategy. So G can be replaced by *3 without affecting either player's chances.

The same argument shows that any game of the form

$$G = \{*a,*b,*c,... \mid *a,*b,*c,...\},$$

in which the same numbers appear on both sides, is really a Nim-heap in disguise. For if m is the least number from 0,1,2,3,... that does not appear among the numbers $a,b,c,...$, then either player can still make from G any of the moves to $*0,*1,*2,...,*(m-1)$ that he could make from $*m$. If his opponent makes any other move from G, it must be to some $*n$ for which $n > m$, and can be reversed by moving back from $*n$ to $*m$. So G is really just a bogus Nim-heap $*m$. ...

THE SPRAGUE-GRUNDY THEORY FOR IMPARTIAL GAMES [BCG p.58]

The above result enables us to show that *every* impartial game can be regarded as a bogus Nim-heap. For suppose we have impartial game

$$G = \{A,B,C,... \mid A,B,C,...\}.$$

Then A, B, C, ... are simpler impartial games and therefore we can suppose they have already been shown to

be equivalent to Nim-heaps $*a,*b,*c,....$ But in this case G can be thought of as the Nim-heap $*m$ defined above. This gives us

THE BOGUS NIM-HEAP PRINCIPLE [BCG p.58]

Every impartial game is just a bogus Nim-heap (that is, a Nim-heap with reversible moves added from some of the positions). The Mex Rule gives the size of heap for G as the least possible number that is not the size of any of the heaps corresponding to the options of G.

This principle was discovered independently by R.P Sprague in 1936 and P.M. Grundy in 1939, although they did not state it in quite this way. This means that provided we can play the game of Nim, we can play *any* other impartial game give only a "dictionary" saying which **nimbers** (i.e., Nim-heaps) correspond to the positions of that game.

Definitions:

A digraph (directed graph), D(V, A), is a nonempty set of vertices, V, together with a set of arcs, A, which are ordered pairs of vertices from V. Graphs of the board positions of combinatorial games do not have multiple arcs between edges. By definition, these games do not have self-loops (an arc with the same initial and terminal vertex).

A directed path or dipath is an ordered subset of arcs in A with the property that the right-hand vertex in the preceding arc is equal to the left-hand vertex in the succeeding arc. The left-hand vertex of the first arc in the set is the initial vertex of the path; the right-hand vertex of the last arc in the set is the terminal vertex of the path. If a digraph contains a dipath with the same initial and terminal vertex then that dipath is called a cycle. The length of a cycle is the minimal number of arcs

on its dipath. If the digraph contains no cycles of length greater than zero then it is cycle free or acyclic.

A colored edge digraph, CED, is a digraph in which every edge is associated with a color.

CED = (V, A, C)

Where V is a nonempty set of vertices,

A is a set of ordered pairs (v1, v2) of vertices called arcs

and C is a function from the set of arcs to a nonempty set of colors.

GAME GRAPHS AND TREES [CG pp.5-6]

A game may be visualized as a digraph; the nodes are the positions and the arcs are the options. The arcs may be thought of as colored, say

bLue Red, or grEen

according as the option is available

to Left only, to Right only, or to either player.

Alternately, we may distinguish between different plays of the game, i.e., different dipaths in the digraph, by duplicating the nodes as necessary and representing the game by a rooted tree. The root is the starting position and the arcs are directed away from the root.

Figures 10a and 10b show the game graph and the game tree for the position {3,2} in a game of Nim: two heaps, one with three beans, the other with two. Nim is an example of an impartial game, in every position of which the same set of options is available to either player: think of the arcs in Figures 10a and 10b as being colored green. Nim is played with a number of heaps of beans. The typical option, for either player, is to choose a heap and remove from it as many beans as you wish: the whole heap maybe, but at least one bean.

> Note the difference between the **complete analysis** of a game, and a **winning strategy**. Figure 10a is a complete analysis: for a winning strategy it suffices to describe the four black arrows.
>
> You may mentally identify three seemingly different aspects of the same idea.
>
> 1. A **game**, i.e., the whole digraph or tree representing the game. For example, **the** Game of Chess, as opposed to **a** (particular) game of chess. We refer to the latter as a **play** of the game: compare *le jeu d'echecs*, *un partie d'echecs*.
>
> 2. A **position** in a game; a particular node of the digraph, perhaps the root of the tree. for example, the standard opening position in Chess, ready for a play of the game.
>
> 3. The **ordered pair** of sets of options available to two players from a given position. ...

Let v be the state of the game where v is a vertex (or node) of a colored edge digraph or CED. Left's options (the set of moves available to Left) is the set of those arcs with initial vertex v which are colored either blue or green. Right's options (the set of moves available to Right) is the set of arcs with initial vertex v which are colored either red or green. The ordered pair of sets of options available to the two players is the set $\{ LO_1, LO_2, LO_3, \ldots \mid RO_1, RO_2, RO_3, \ldots \}$ where LO_n denotes Left's nth option (the list is not ordered) and RO_n denotes Right's nth option.

If the game is impartial, both players have the same options and the graph of the game is a regular digraph.

The formal definition of a game is the ordered pair of sets of options available the two players in a given

position of the game. A given position can be the initial position (e.g., the empty board in Go) or a particular position (e.g., White to play and mate in three moves).

If the graph of a game contains a cycle then the game contains an infinite sequence of moves and does not satisfy the ending condition for combinatorial games. "A game that does not satisfy the ending condition is called a **loopy** game. Its digraph will contain a directed circuit or an infinite directed path. The outcome may be a **draw**: note that we distinguish between a *tied* game and one *drawn out* by infinite play. Chess exhibits both kinds of outcome: stalemate is a tie, but perpetual check, repetition of moves, or insufficient mating material are equivalent to draws."[CG p.7]

THE GAME-GRAPH OF A GAME

Let G = (R, C, B, I) be a combinatorial game with initial position I. The two players are Blue and Red. The game-graph of G

G = (V, A, C)

is an acyclic colored directed multigraph;

where

$V = \Delta^*(I)$;

A is the set of ordered pairs (v_i, v_t) such that v_i, $v_t \in V$ and (v_i, v_t) is a move in G;

and, C is a function from A into { Red, Blue, Green }.

A move consists of one or more applications of one or more of the rules in R as allowed by the control system which changes a board state bi in $\Delta^*(B)$ to a board state bj in $\Delta^*(B)$. For the class of games under consideration, that is, win-lose games, there is a restriction that if bk ===> bl is a move then bl ϵ $\Delta^+(bk)$. If a move uses only rules available to both players then the move is colored green. If the game is impartial then all the moves are colored green and we can eliminate the color. If the move contains one or more applications of a rule available only to Blue then it is colored Blue; and if the move contains one or more applications of a move available only to Red then it is colored Red. Clearly, a move cannot be colored both Red and Blue.

Let M(i,j) be the set of all moves from bi to bj. If M(i,j) contains only Red moves then it is colored Red. If M(i,j) contains only Blue moves then it is colored Blue. Otherwise M(i,j) is colored Green.

If S is an impartial game then

Graph (S) = { V, A }

where V = $\Delta^*(B)$

and A is the set of all ordered pairs (bi, bj) s.t. bi, bj are in V and there exists a move m1 in S s.t. bi =m1=> bj.

If S is a partizan game then

Graph (S) = { V, AC }

where V = Δ^*(B)

and AC is a set of ordered triples (bi, bj, c) such that there exists at least one move m1 s.t. bi =m1=> bj and c is the color associated with the set M(i,j).

Definition: The game-graph of a game G = (V, A) is finite if V is finite. Let G be a game then we will denote the graph of G by G′.

THE SUM OF GAMES

Let G1 and G2 be games with game-graphs G1′ and G2′. Let G be the following game: Two players, Left and Right, alternate moves; Left moves first. We will arbitrarily use the normal play condition, i.e., the first player unable to move loses and his opponent wins. A player in his turn can move in either G1 or in G2 but not both. The two games G1 and G2 are disjoint. Such a sum is called the sum of m component games [CG p.115]. G is the union of G1 and G2. The union of two disjoint digraphs G1(V1,A1) and G2(V2,A2) is the graph G3(V3,A3) where V3 = V1 + V2 and A3 = A1 + A2. The game-graph of G, G′, is the sum of the game-graphs G1′ and G2′ where V′ is the set V1′ x V2′ and the A3′ = { (x,y) such that x represents (v1,v2) and y represents (v3,v4) in V3′ and either (v1,v3) in A1 or (v2,v4) in A2 }.

NIMBLE [CG p.37]

Nimble is played with coins on a strip of squares (Figure 11a). Two players, Left and Right, take turns moving just one coin at least one square to the left. Left moves first. You can jump onto or over other coins, even clear off the strip. You can have any number of coins on a square. The first player unable to move loses, his opponent wins. [Figure 11a shows the game with coins on squares 3, 4, and 5.]

Nimgee is played with tokens on a directed one-way infinite digraph D = (V, A) with the vertices labeled 0, 1, 2, 3, ... The set of arcs, A, is the set of ordered pairs (v_i, v_j) such that Label(v_j) = Label(v_i) + 1. Two players, Left and Right, take turns moving just one token from a vertex labeled s to a vertex labeled t, s not equal to t. Left moves first. You can move tokens over or onto other tokens but not off the graph. The game ends when all the tokens are piled on the vertex labeled 0. The last player to move wins, his opponent loses.

Nimgee is isomorphic to Nimble. Figure 11b shows a position in Nimgee that is isomorphic to the position in Nimble shown in Figure 11a. In Figure 11c, the instance of Nimgee given is represented as the union of three Nim-heaps. The game-graph of all three representations is the same and its size is suggested in Figure 11d.

We use an algorithm given by Fraenkel [CG p.115] to find whether the vertices of the game-graph of the game represent N-positions or P-positions.

"There is a simple algorithm for determining the N,P-pattern of any game-graph $G = (V,E)$. Define the set of *followers* of $u \in V$ by $F_G(u) = F(u) = \{ v \in V : (u,v) \in E \}$. The definitions of N, P imply

$$u \in P \text{ if and only if } F(u) \subseteq N,$$

whereas

$$u \in N \text{ if and only if } F(u) \cap P \neq \phi.$$

Thus for normal play, all *leaves* (vertices with $F(u) = \phi$) are labeled P. Working backwards, using (1) and (2), one gets the N,P-labels in $O(|E|)$ steps (linear time)."

If the game-graph of a game does not contain a cycle then there are no draws. This does not hold in the other direction; games with no draws can contain cycles in their game-graphs. Northcott's game (a disguised form of Nim) is an example of a game that does not have draws but can continue forever. Mini-Northcott, Northcott's game on a strip of four squares with the position as shown in Figure 12, is used as an illustration. Note that the graph of the game is asymmetric.

The following result [Fraenkel, CG p.128] applies to games in which the set of outcomes are win, lose, or draw; a superset of the games under consideration.

Fundamental Theorem of Combinatorial Game Theory

> Let Gamma be a 2-person game with perfect information and no chance moves, whose game-graph may be infinite. Then for every position of Gamma there are

precisely two possibilities: (i) There exists a winning move for precisely one of the two players. (ii) There exists a winning move for neither player, but both can maintain a draw.

ANTENIM

Alice and Bob sit down to play a new game. There are three bowls on the table. Both Alice and Bob have a sack of quarters. They choose and Alice is to play first. Three random integers between 0 and 100 are drawn. The numbers chosen this time are one, two and three. The object of the game is to place one quarter in one bowl, two quarters in a second bowl and three quarters in the third bowl. The player who completes the count removes the quarters and keeps them, his or her opponent plays first in the new game. Alice plays first and the two players move alternately. Each player in turn must select a bowl and place at least one quarter in it. The number of quarters in any nonempty subset of bowls must be less than or equal to the maximal sum of the integers in a subset of integers of equal size.

Alice plays first and puts three quarters in the first bowl. Now Bob can either place one quarter or two quarters in either of the two remaining bowls. If Bob places one quarter in one of the two bowls then Alice will place two quarters in the last bowl, completing the count and winning the game and one quarter. If Bob places two quarters in a second bowl then Alice will place one quarter in the third bowl winning the game and two quarters. Bob cannot put

three quarters in another bowl as then the contents of a subset of the two bowls containing three quarters each would exceed the maximal cardinality of the contents allowed for two bowls, to wit, two and three.

Bob places a quarter in the third bowl and Alice tosses two quarters into the second bowl to win the game.

In the next game, the numbers picked are 0, 2, and 3. Bob begins by putting one quarter in bowl one. Alice puts one quarter in bowl three, leaving the position

1 0 1

Bob puts one quarter in bowl three and Alice puts two quarters into bowl one to win the game.

The integers picked are 1, 3, and 6. Alice plays first and puts 4 quarters into the first bowl. Bob puts two quarters into the second bowl and Alice adds two more quarters to the first bowl. Bob tosses one quarter into the third bowl and Alice puts one quarter into the second bowl to complete the count and win the game and three quarters.

This is, to my knowledge, a new game, which I call Antenim. If the game were played with pennies, then we can refe to it as Penny Antenim.

For the two bowl game, the following initial positions are previous player wins:

PAIR

0 0

1 1

1 2

3 3

3 4

3 5

3 6

7 7

7 8

...

Theorem 4.3.

Let A and B be two nonnegative integers and WLOG let $A \leq B$. Then a game of Antenim, for which (A, B) is the initial position is a Previous player win iff $B \leq 2*A$.

Proof.

For B equal to zero, there is nothing to do. For $B > 0$, if Next player puts $x \leq A$ coins in one bowl then place y coins in the second bowl such that $A - x = B - y$; if Next player puts $x > A$ coins in one bowl then put y coins in the second bowl such that $B - x = A - y$. Subsequently, whenever Next

player places z coins in one bowl reply by adding z coins to the other bowl, $x,y,z \in \mathbb{Z}^+$. Since the initial move left an equal number of coins to be added to each of the bowls to add up to A in one bowl and B in the other replying to Next by echoing his moves guarantees that you will make the last move, i.e., win■

For the three bowl game the situation is more complex. Let A, B, and C be nonnegative integers with $A \leq B \leq C$. If $B \leq 2*A$ then putting C coins in one bowl reduces the game to a P position two bowl game.

The initial position 1 3 x, x a nonnegative integer, is a first player win. Let Left and Right be two players who alternately drop one or more coins into exactly one of three bowls, Left plays first.

If $x = 0$ then Left puts 2 coins in one bowl. Right must either add one coin to that bowl or place one coin in a second bowl. Left then completes the count and wins the game.

If $x = 1$ then Left places three coins in one bowl reducing the game to 0 1 1. For $x = 2$, Left puts three coins in one bowl reducing the game to 0 1 2. If Right places one coin in a second bowl, Left will place two coins in the third bowl; and if Right places two coins in a second bowl, Left will put one coin in the third bowl to win.

Let $a, b, c \in \mathbb{Z}^+$

Let *a* -> *b* -> *c* denote

Left if a places a coins in one bowl then Right places b coins in a second bowl then Left places c coins in a third bowl;

Let *a* -> *a*|*b* denote

if Left places *a* coins in one bowl then Right will add *b* - *a* coins to the same bowl; and, *a* -> *b* -> *a*|*c* denote

Left places *a* coins in one bowl and Right places *b* coins in another bowl then Left will place *c* - *a* coins in a bowl that contains *a* coins.

1 3 3

3 -> 2

2 -> 3

1 -> 1 -> 1 reducing the position to 0 2 2;

a first player win.

1 3 4

4 -> 2

3 -> 3

2 -> 4

1 -> 1 -> 2 Right cannot add any coins to a bowl without reducing the game to adding coins to the third bowl;

a first player win.

1 3 5

5 -> 2

4 -> 3

3 -> 4

1 -> 1 -> 5 reducing the position to 0 1 1;
a first player win.

1 3 6

6 -> 2 reducing the position to 0 1 1

5 -> 3

4 reducing the position to the Nim-heaps 1 3 2;
a first player win.

3 -> 5

2 -> 6

1 -> 1 -> 4 reducing the position to 0 2 2;
a first player win.

1 3 7

7 -> 2

6 -> 3

5 reducing the position to the Nim-heaps 1 3 2;
a first player win.

4 -> 1 0 3 3 or 1 2 3

which are both Nim positions since the 4 coins in the one bowl fixes a Nim-heap of size 3 = 7 - 4.

3 -> 6

2 -> 7

1 -> 4

1 3 8

8 -> 2

7 -> 3

6 see 5 above

5 -> 5|6

4 -> 4|6

3 -> 7

2 -> 8

Note that 3 -> 3|6 and 2 -> 2|6 are more profitable then the moves given.

For 1 3 x, x > 8, place y coins in one bowl such that x - y = 2.

The series 2 5 x, $0 \leq x \leq 5$, are also first player wins.

2 5 0

3

2 5 [1, 2, 3, 4]

5

2 5 5

5 -> 3

4 -> 1 -> 5 a first player win

4 -> 2 -> 4

4 -> 3 -> 4|5

4 -> 4 -> 2

4 -> 5 -> 1

As is, 2 5 6.

2 5 6

6 -> 3

5 -> 4

4 -> 5

3 -> 6

2 -> 1 -> 6 -> 1|5

2 -> 1 -> 5 -> 1|6

2 -> 1 -> 4 -> 1|4

2 -> 1 -> 3 -> 1|3

2 -> 1 -> 2 -> 1|2

2 -> 1 -> 1 -> 2|6 -> 1|4

2 -> 1 -> 1 -> 2|5 -> 1|5

2 -> 1 -> 1 -> 2|4 -> 1|6

2 -> 1 -> 1 -> 2|3 -> 1|3

2 -> 1 -> 1 -> 1|6 -> 1|5

2 -> 1 -> 1 -> 1|5 -> 1|6

2 -> 1 -> 1 -> 1|4 -> 1|5

2 -> 1 -> 1 -> 1|3 -> 1|6 -> 1|5

2 -> 1 -> 1 -> 1|3 -> 1|5 -> 1|6

2 -> 1 -> 1 -> 1|3 -> 1|4 -> 3|4

2 -> 1 -> 1 -> 1|3 -> 1|3

2 -> 1 -> 1 -> 1|2 -> 2|6 -> 1|5

2 -> 1 -> 1 -> 1|2 -> 2|5 -> 1|6

2 -> 1 -> 1 -> 1|2 -> 2|4 -> 1|4

2 -> 1 -> 1 -> 1|2 -> 2|3 -> 1|3

2 -> 1 -> 1 -> 1|2 -> 1|6 -> 2|5

2 -> 1 -> 1 -> 1|2 -> 1|5 -> 2|6

2 -> 1 -> 1 -> 1|2 -> 1|4 -> 2|4

2 -> 1 -> 1 -> 1¦2 -> 1¦3 -> 2¦3

2 -> 1 -> 1 -> 1¦2 -> 2¦2

1 -> 6 -> 1¦3

1 -> 5 -> 1¦4

1 -> 4 -> 1¦5

1 -> 3 -> 1¦6

1 -> 2 -> 6 -> 1¦5

1 -> 2 -> 5 -> 1¦6

1 -> 2 -> 4 -> 1¦4

1 -> 2 -> 3 -> 1¦3

1 -> 2 -> 2 -> 1¦2

1 -> 2 -> 1

Two variations of Nim, Antonim and Synonim, are presented in Winning Ways, Vol.2, pages 459 - 462.

ANTONIM

"**Antipathetic Nim** is Nim in which no two heaps are allowed to have the same number of chips. Of course, we don't notice empty heaps, so if you want to play it with coins on a strip the conditions that no two coins may be on the same square unless the square is the moneybag (square 0)." [BCG p.459]

For two-heap Antonym with position (a, b) where a and b are natural numbers with a less than or equal to b, (a, b) is a P-position iff a is an odd number and b equals a+1. For three-heap Antonym, let a, b, c ϵ $\mathbb{N}$ then "(a,b,c) is a *P*-position in Antonym just when (a+1,b+1,c+1) is a *P*-position in Nim." [BCG p.459]

"For Antonym, The 3-heap *P*-positions are the lines in Figure 13, counting the top node as 0; the 4-heap ones are the complements of lines, counting it as 7." [BCG p.460]

SYNONIM [BCG p.460]

"In **Sympathetic Nim** all heaps of the same size must be treated alike—if you reduce one heap of a given size you must reduce all heaps of that size by the same amount (no move may affect heaps of different sizes). In the strip version the move is to take *all* coins from some square and put them on any earlier square."

The games are the same. As stated in Winning Ways [BCG p.462]:

"**SYNONIM** is just a synonym for **ANTONIM**!"

TABLE OF GRUNDY VALUES FOR TWO-HEAP SYNONIM

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
0	0	1	2	3	4	5	6	7	8	9	10
1	1	1	0	2	3	4	5	6	7	8	9
2	2	0	2	1	5	3	4	8	6	7	11
3	3	2	1	3	0	6	7	4	5	10	8
4	4	3	5	0	4	1	2	9	10	6	7
5	5	4	3	6	1	5	0	2	9	11	12
6	6	5	4	7	2	0	6	1	3	12	13

7	7	6	8	4	9	2	1	7	0	3	5
8	8	7	6	5	10	9	3	0	8	1	2
9	9	8	7	10	6	11	12	3	1	9	0
10	10	9	11	8	7	12	13	5	2	0	10

WELTER'S GAME

Welter's game is played on a strip divided into k squares on which are placed m coins, no two of which occupy the same square, $0 < m \leq k$. A move consists of moving one coin towards the left to an unoccupied square. Numbering the squares 0, 1, 2, 3, ... from the left, this is equivalent to having m heaps of stones with the number of stones in each heap corresponding to the index of each coin (including 0, the empty heap). A move consist of removing at least one coin from a single heap provided that no two heaps contain the same of stones (including zero) at the end of the move. This is the same game as antonim without the moneybag on square 0.

Example:

Let [1¦2¦3] be an instance of Welter's game played on 3 heaps with the heaps containing 1, 2, and 3 stones.
The only legal moves from this position are to

[0¦2¦3]

[1¦0¦3]

[1¦2¦0]

since 0 is the only nonnegative number not represented in [1¦2¦3]. Note that the ordering does not matter and, for example, [0¦1¦3] is the same position as [1¦0¦3].

Let [0¦1¦2¦3] represent Welter's game played on 4 heaps. There is no legal move from this position.

Let [1¦2¦20] represent Welter's game with 3 coins. This position can be represented by a strip with 21 or more squares labelled 0, 1, 2, 3, ... from the left hand edge and by three coins, one each on the squares labelled 1, 2, and 20.

A function to compute the nim-value of a position in Welter's game is as follows:

1. Mate pairs of heap sizes that are equal modulo the largest power of 2; leaving the spinster, if the number of heaps is odd, by itself.
2. Use vector addition modulo 2 to add the values of the mated pairs, then subtract 1 to find the value of the mated pair. This operation will be denoted by $\odot$.
3. Add the values obtained using vector addition modulo 2. This operation will be denoted by $\oplus$.

Let [2¦3¦7¦9¦16¦20¦25] be a seven coin Welter's game position.

POWER	VALUE MODULO 2 ** POWER						
5	2	3	7	9	16	20	25
4	2	3	7	**9**	0	4	**9**
3	2	3	7		0	4	
2	2	**3**	**3**		**0**	**0**	

The remaining heap, 2, is a spinster.

STEP 1:

(9⊙25) ⊕ (3⊙7) ⊕ (16⊙20) ⊕ 2

STEP 2:

(16-1) ⊕ (4-1) ⊕ (4-1) ⊕ 2

STEP 3:

15 ⊕ 3 ⊕ 3 ⊕ 2

12 ⊕ 3 ⊕ 2

15 ⊕ 2

13

The nim-value of the position [0¦1¦3¦9¦19¦36] is computed as follows:

POWER	POSITION MODULO 2 ** POWER					
6	0	1	3	9	17	36
5	0	1	3	9	17	4
4	0	**1**	3	9	**1**	4
3	0		3	1		4
2	**0**		3	1		**0**
1			**1**	**1**		

STEP 1.

(1⊙17) ⊕ (0⊙36) ⊕ (3⊙9)

STEP 2.

(16-1) ⊕ (36-1) ⊕ (10-1)

STEP 3.

15 ⊕ 35 ⊕ 9

44 ⊕ 9

37

```
* WELTER'S GAME WITH TWO HEAPS

      INTEGER LMT
      PARAMETER ( LMT = 10 )

      INTEGER G( 0:LMT, 0:LMT )

      LOGICAL INSET( -1:100 )

      INTEGER X, Y

      INTEGER I, J, K

      INTEGER MEX

      CHARACTER*63 LINE


      DO 5 I = 0, LMT

         G(I,I) = -1

5     CONTINUE

      DO 7 I = 1, LMT

          G(I,0) = I-1
          G(0,I) = I-1

7     CONTINUE

      DO 10 X = 1, LMT
```

```
      DO 20 Y = X+1, LMT
         DO 30 I = 0, 100
            INSET(I) = .FALSE.
30       CONTINUE
         DO 40 I = 0, X-1
            INSET(G(I,Y)) = .TRUE.
40       CONTINUE
         DO 50 I = 0, Y-1
            INSET(G(X,I)) = .TRUE.
50       CONTINUE
         MEX = 0
60       IF ( INSET(MEX) ) THEN
            MEX = MEX + 1
            GOTO 60
         ENDIF
         G(X,Y) = MEX
         G(Y,X) = MEX
20    CONTINUE
10    CONTINUE

      DO 70 I = 0, LMT
         LINE = ' '
      DO 80 J = 0, LMT
         K = 3*J+2
         IF ( G(I,J).GE.10 ) THEN
            LINE(K:K) = CHAR( ICHAR('0') + G(I,J)/10 )
            LINE(K+1:K+1) = CHAR( ICHAR('0') +
     +                      MOD(G(I,J),10) )
```

```
            ELSEIF ( G(I,J).GE.0 ) THEN
               LINE(K+1:K+1) = CHAR( ICHAR('0') + G(I,J) )
            ELSE
               LINE(K+1:K+1) = '*'
            ENDIF
80       CONTINUE
             PRINT *, LINE
             PRINT *
70       CONTINUE

         END
```

```
   *  0  1  2  3  4  5  6  7  8  9
   0  *  2  1  4  3  6  5  8  7 10
   1  2  *  0  5  6  3  4  9 10  7
   2  1  0  *  6  5  4  3 10  9  8
   3  4  5  6  *  0  1  2 11 12 13
   4  3  6  5  0  *  2  1 12 11 14
   5  6  3  4  1  2  *  0 13 14 11
   6  5  4  3  2  1  0  * 14 13 12
   7  8  9 10 11 12 13 14  *  0  1
   8  7 10  9 12 11 14 13  0  *  2
   9 10  7  8 13 14 11 12  1  2  *
```

```
*  WELTER'S GAME WITH THREE HEAPS

         INTEGER LMT
         PARAMETER ( LMT = 14 )

         INTEGER G( 0:LMT, 0:LMT, 0:LMT )
```

```
      LOGICAL INSET( -1:200 )
      INTEGER X, Y, Z
      INTEGER I, J, K
      INTEGER MEX
      INTEGER TABLE(0:LMT,0:LMT)

      DO 3 I = 0, LMT
         G(I,I,I) = -1
3     CONTINUE
      DO 5 I = 0, LMT
      DO 7 J = I+1, LMT
         G(I,I,J) = -1
         G(I,J,I) = -1
         G(J,I,I) = -1

         G(J,J,I) = -1
         G(J,I,J) = -1
         G(I,J,J) = -1
7     CONTINUE
5     CONTINUE
      DO 10 X = 1, LMT
      DO 20 Y = X+1, LMT
         DO 30 I = 0, 200
            INSET(I) = .FALSE.
30       CONTINUE
         DO 40 I = 0, X-1
            INSET(G(I,Y,0)) = .TRUE.
40       CONTINUE
         DO 50 I = 0, Y-1
            INSET(G(X,I,0)) = .TRUE.
```

```
50          CONTINUE

            MEX = 0

60          IF ( INSET(MEX) ) THEN

               MEX = MEX + 1

               GOTO 60

            ENDIF

            G(O,X,Y) = MEX
            G(0,Y,X) = MEX

            G(X,0,Y) = MEX
            G(Y,0,X) = MEX

            G(X,Y,0) = MEX
            G(Y,X,0) = MEX

20       CONTINUE

10       CONTINUE


         DO 110 X = 1, LMT

         DO 120 Y = X+1, LMT

         DO 130 Z = Y+1, LMT

            DO 140 I = 0, 200

               INSET(I) = .FALSE.

140         CONTINUE

            DO 150 I = 1, X

               INSET(G(X-I,Y,Z)) = .TRUE.

150         CONTINUE

            DO 160 I = 1, Y

               INSET(G(X,Y-I,Z)) = .TRUE.

160         CONTINUE

            DO 170 I = 1, Z
```

```
            INSET(G(X,Y,Z-I)) = .TRUE.
170         CONTINUE
            MEX = 0
180         IF ( INSET(MEX) ) THEN
               MEX = MEX + 1
            ENDIF
            G(X,Y,Z) = MEX
            G(X,Z,Y) = MEX
            G(Y,X,Z) = MEX
            G(Y,Z,X) = MEX
            G(Z,X,Y) = MEX
            G(Z,Y,X) = MEX
130      CONTINUE
120      CONTINUE
110      CONTINUE

         DO 200 I = 0, 3
            DO 205 J = 0, LMT
            DO 206 K = 0, LMT
               TABLE(J,K) = -99
206         CONTINUE
205         CONTINUE
            DO 210 X = 0, LMT
            DO 220 Y = 0, LMT
            DO 230 Z = 0, LMT
               IF ( G(X,Y,Z).EQ.I ) THEN
                  TABLE(X,Y) = Z
               ENDIF
230         CONTINUE
```

```
220   CONTINUE

210   CONTINUE

      DO 290 J = 0, LMT

         WRITE  ( 6,2000 ) ( TABLE(J,K), K = 0, LMT )

2000     FORMAT ( 1X, T5, 15I4 / )

290   CONTINUE

      PRINT *
      PRINT *
      PRINT *
      PRINT *

200   CONTINUE


      END
```

The following tables are:

TABLES OF GRUNDY VALUES FOR WELTER'S GAME WITH THREE HEAPS
EACH TABLE IS A TABLE OF GRUNDY VALUES FOR THE TRIPLE
(X,Y,Z)

ROW = X, COLUMN = Y, AND ENTRY = Z

* INDICATES A POSITION NOT ALLOWED BY THE RULES OF THE GAME
-99 INDICATES A POSITION FOR WHICH THE REASON FOR HAVING NO
GRUNDY VALUE IS UNCERTAIN

GRUNDY VALUE = 0

*	2	1	4	3	6	5	8	7	10	9	12	11	14
2	*	0	5	6	3	4	9	10	7	8	13	14	11
1	0	*	6	5	4	3	10	9	8	7	14	13	12
4	5	6	*	0	1	2	11	12	13	14	7	8	9
3	6	5	0	*	2	1	12	11	14	13	8	7	10
6	3	4	1	2	*	0	13	14	11	12	9	10	7
5	4	3	2	1	0	*	14	13	12	11	10	9	8

8	9	10	11	12	13	14	*	0	1	2	3	4	5
7	10	9	12	11	14	13	0	*	2	1	4	3	6
10	7	8	13	14	11	12	1	2	*	0	5	6	3
9	8	7	14	13	12	11	2	1	0	*	6	5	4
12	13	14	7	8	9	10	3	4	5	6	*	0	1
11	14	13	8	7	10	9	4	3	6	5	0	*	2
14	11	12	9	10	7	8	5	6	3	4	1	2	*
13	12	11	10	9	8	7	6	5	4	3	2	1	0

GRUNDY VALUE = 1

*	3	4	1	2	7	8	5	6	11	12	9	10	-99
3	*	5	0	7	2	9	4	11	6	13	8	-99	10
4	5	*	7	0	1	10	3	12	13	6	-99	8	9
1	0	7	*	5	4	11	2	9	8	-99	6	13	12
2	7	0	5	*	3	12	1	10	-99	8	13	6	11
7	2	1	4	3	*	13	0	-99	10	9	12	11	6
8	9	10	11	12	13	*	-99	0	1	2	3	4	5
5	4	3	2	1	0	-99	*	13	12	11	10	9	8
6	11	12	9	10	-99	0	13	*	3	4	1	2	7
11	6	13	8	-99	10	1	12	3	*	5	0	7	2
12	13	6	-99	8	9	2	11	4	5	*	7	0	1
9	8	-99	6	13	12	3	10	1	0	7	*	5	4
10	-99	8	13	6	11	4	9	2	7	0	5	*	3
-99	10	9	12	11	6	5	8	7	2	1	4	3	*
-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99

GRUNDY VALUE = 2

*	4	3	2	1	8	7	6	5	12	11	10	9	-99
4	*	6	7	0	9	2	3	12	5	14	-99	8	-99
3	6	*	0	7	10	1	4	11	14	5	8	-99	-99
2	7	0	*	6	11	4	1	10	-99	8	5	14	-99
1	0	7	6	*	12	3	2	9	8	-99	14	5	-99
8	9	10	11	12	*	14	-99	0	1	2	3	4	-99
7	2	1	4	3	14	*	0	-99	10	9	12	11	-99
6	3	4	1	2	-99	0	*	14	11	12	9	10	-99
5	12	11	10	9	0	-99	14	*	4	3	2	1	-99
12	5	14	-99	8	1	10	11	4	*	6	7	0	-99
11	14	5	8	-99	2	9	12	3	6	*	0	7	-99
10	-99	8	5	14	3	12	9	2	7	0	*	6	-99
9	8	-99	14	5	4	11	10	1	0	7	6	*	-99
-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	*
-99	10	9	12	11	6	5	8	7	2	1	4	3	-99

GRUNDY VALUE = 3

*	5	6	7	8	1	2	3	4	13	14	-99	-99	9
5	*	3	2	9	0	7	6	13	4	11	10	-99	8
6	3	*	1	10	7	0	5	14	11	4	9	-99	-99
7	2	1	*	11	6	5	0	-99	10	9	4	-99	14
8	9	10	11	*	13	14	-99	0	1	2	3	-99	5
1	0	7	6	13	*	3	2	9	8	-99	14	-99	4
2	7	0	5	14	3	*	1	10	-99	8	13	-99	11
3	6	5	0	-99	2	1	*	11	14	13	8	-99	10

100

4	13	14	-99	0	9	10	11	*	5	6	7	-99	1
13	4	11	10	1	8	-99	14	5	*	3	2	-99	0
14	11	4	9	2	-99	8	13	6	3	*	1	-99	7
-99	10	9	4	3	14	13	8	7	2	1	*	-99	6
-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	-99	*	-99
9	8	-99	14	5	4	11	10	1	0	7	6	-99	*
10	-99	8	13	6	11	4	9	2	7	0	5	-99	3

CHAPTER 5

An Adventure in Wonderland

THE QUEEN'S GAME

This chessboard game is referenced by Martin Gardner [Gardner p.103] under the name Corner the Lady. A queen is placed on a arbitrary square of an n x n chessboard, $n \in \mathbb{Z}^+$ [Figure 14]. Two players, say, Left and Right, alternately move the queen towards the Northwest corner of the board. Left moves the queen first. Each of the two players must advance the queen by at least one square along a row, that is, towards the west; or, along a column, that is, towards the north; or, along a diagonal, that is, towards the northwest. The last player to move the queen wins and his or her opponent loses. If the rows and columns were indexed 0, 1, 2, ... starting from the Northwest corner of the chessboard; then, the final board state of the game is the position (0, 0).

By the conditions on the moves this game eventually ends and never leads to a draw.

We define P(Queen) to be all those board states in which your opponent (the Next player) does not have a winning move against perfect play. That is to say that if you (the Previous player) play correctly then the Next player must eventually be on move with the queen placed on

(0, 0). That is, P(Queen) is the set of all winning moves for the Previous player.

Lemma 5.1.

The set P(Queen), where Queen is a combinatorial game with the properties listed above is the even numbered moves from a forced win for the Previous player.

Proof.

Let zero denote the set of all positions from which there is no move. By definition of P as the set of all winning positions for the Previous player, this set of positions must be in P.

Since this set of positions has no move available it is zero moves away from a forced win for the Previous player.

Suppose the lemma is false and let Previous player be on move with an even number of forced moves to zero. Queen is a combinatorial game, i.e., each of the players has perfect information and there are no chance moves, hence, each of the two players play can play perfectly and the two players alternate moves. If Next player plays perfectly, Previous player is on move in a position zero which means that Previous player loses which contradicts zero being in P■

We can translate The Queen's Game into Wythoff's game (or Wythoff's Nim) by

letting x be the maximum rook move along a row; and,

letting y be the maximum rook move along a column. And allow the following moves:

Type-one moves: x is reduced one or more units; or, y is reduced one or more units.

Type-two moves: both x and y are reduced by one or more units subject to the restraint that both x and y be reduced by the same number of units.

On each of these moves there is the restriction that both x and y must at all times be greater or equal to zero.

It is clear that if x = 0 and y = 0 that no move is possible and that the position is a Previous player win (or a Next player loss). If the position is either (0, n), (n, 0) or (n, n), for n greater than zero, that the position is a Next player win (in the set N). Both (n, 0) and (0, n) can be reduced to (0, 0) by a Type-one move while (n, n), n greater than zero, can be reduced to (0, 0) by a Type-two move.

Wythoff's game can easily be represented as a vector addition system:

G6 = (V, C, B)

where B = (x, y), both x and y, nonnegative integers

C, the control unit, acts as referee and score keeper

$$V = \{ (0, -1), (-1, 0), (-1, -1) \}$$

L (Left) and R (Right) move alternately. Left moves first. The first player unable to move loses.

A move consists of the

1. Choosing a vector in V and
2. Applying the vector one or more times to B with the restriction that at the end of any application of the chosen vector v, i.e., B = B - v, neither x nor y is less than zero.

Wythoff's game can be reduced to a string rewriting system

$G7 = (\Sigma, R, C, B)$

where $\Sigma = \{ a, b \}$

$B = a^i \| b^j$ where $i,k \in \mathbb{N}$

and $\|$ is concatenation

$R = \{ (a, \lambda), (b, \lambda), (ab, \lambda) \}$

and C is the following:

Left and Right move alternately

Left moves first

The first player unable to move loses

A move consists of

1. Choosing a production in R
2. Applying that production one or more times.

Lemma 5.2.

Wythoff's game always ends in a win or lose, i.e., there are no draws.

Proof.

A board position in Wythoff's game consists of an ordered pair, (x, y), of two nonnegative finite integers. A move consists of reducing one or both of these integers. The lower bound on each of these integers is zero. there are a finite number of integers between a finite nonnegative integer and zero. Since each move reduces one or both of these integers the game will eventually end. The player left with the position (0, 0), that is, the player required to move and given the board (0, 0), cannot move, hence, loses. The other player wins■

A.S. Fraenkel [ASF] gives a generalization of Wythoff's game in which a Type-two move (a move in which one or more coins are removed from both heaps) is allowed providing the difference between the number of coins removed from each heap is less than A, where A is a positive integer. When A = 1, the maximum difference is zero and we have the game equivalent to The Queen's Game.

A presentation of Wythoff's game for A = 2 as a Markov System is given below.

Definition of Symbols

A - a token on heap one.

B - a token on heap two.

C, D, E are internal states.

L, R - respectively, Left to move, Right to move.

S - denotes a separate game as in the disjoint sum of games.

w - a nonempty Wythoff game.

p - zero or more separate games.

a - a string of zero or more A's.

b - a string of zero or more B's.

Rules:

1. LpSXAabp --> LpSCabp
2. LpSCAabp --> [LpSCabp¦RpSAabp]
3. LpSCabp --> RpSabp
4. LpSYabBp --> LpSDabp
5. LpSDabBp --> [LpSDabp¦RpSabBp]
6. LpSDap --> RpSap
7. LpSZAabBp --> [LpSCabBp¦LpSCAabp¦LpSCab]
8. LpSCaABbp --> [LpSCabp¦RpSaABbp]
9. LpSCabp --> RpSabp
10. RpSXAabp --> RpSCabp
11. RpSCAabp --> [RpSCab¦LpSAabp]
12. RpSCabp --> LpSabp
13. RpSYabBp --> RpSDabp
14. RpSDabBp --> [RpSDabp¦LpSabBp]

15. RpSDabp --> LpSap

16. RpSZAabBp --> [RpSCabp|RSCAabp|RpSCab]

17. RpSCaABbp --> [RpSCabp|LpSaABbp]

18. RpSCabp --> LpSabp

19. SS --> S

20. LpSwp --> [LpSXwp|LpSYwp|LpSZwp]

The initial board position is given by Lp.

I found no upper bound for the Grundy numbers of a two-heap Wythoff's game discussed in the literature. The following theorem gives a trivial upper bound; while the conjecture gives an upper bound I believe to be true but cannot prove.

Theorem 5.1. Let W be a two-heap Wythoff's game with A =1, with the game represented by the pair (X, Y), $X,Y \in \mathbb{N}$, with $0 \leq X \leq Y$, then the Grundy value of W, $g(W) \leq 2X + Y$.

Proof.

The indexing starts at zero, so, there are X positions reachable from X by a Type-one move; and, Y positions reachable from Y by a Type-one move; and, MIN(X,Y) positions reachable from the pair (X,Y) by a Type-two move. The Grundy value of position (X,Y), g(W), is given by the mex of the set of the Grundy values of all the positions reachable from (X,Y). There are at most 2X + Y Grundy values, one of which must be zero; hence, the upper bound■

Conjecture 5.1. Let W be a two-heap Wythoff's game with A =1, with the game represented by pair (X, Y), X,Y ϵ $\mathbb{N}$, with $0 \leq X \leq Y$, then the Grundy value of W,

$$g(W) \leq X + Y.$$

What we would like is to generalize Martin Gardner's [Gardner chapter 8] visual description of the set of all P-positions (Previous player wins) in The Queen's Game [Figure 15] and extend it to the cases where the difference in the coins removed from both heaps in a Type-two move is less than A, where A is a positive integer.

Let the set of P-positions suggested by the above diagrams be denoted { (A_0, B_0), (A_1, B_1), (A_2, B_2), ..., (A_n, B_n), ... } where (A_n, B_n) denotes the n^{th} pair of coordinates which is a safe position to move to. Note that there are two ordered pairs of coordinates that are symmetric with respect to A_n and B_n. Choose the pairs such that $A_n \leq B_n$ and denote by A the ordered set of all A_i, i ϵ $\mathbb{N}$, and by B the ordered set of all B_i, i ϵ $\mathbb{N}$; that is, A = { A_0, A_1, A_2, A_3, ... } and B = { B_0, B_1, B_2, B_3, ... }.

Consider the following sequences:

n	A_n	B_n
0	0	0
1	1	2
2	3	5
3	4	7
4	6	10
5	8	13

6	9	15
7	11	18
8	12	20
9	14	23
10	16	26
11	17	28

This table suggests the following theorem of Fraenkel's. [ASF p.354]

Theorem [5.2]. $P = \cup_{i=0,\infty}\{(A_i,B_i)\}$, *where* $A_n = \text{mex}\{A_i,B_i : 0 \leq i < n\}$ and $B_n = A_n + an$ $(n \geq 0)$.

Proof. Form the definition of A_n and B_n as given in the theorem it follows that if $A = \cup_{n=1,\infty} A_n$ and $B = \cup_{n=1,\infty} B_n$, then $A \cup B = \mathbb{Z}^+$ (the set of positive integers), and $A \cap B = \phi$. The last equality is true since if $A_n = B_m$, then $n > m$ implies that A_n is the mex of a set containing $B_m = A_n$, a contradiction; and $n \leq m$ is impossible since $B_m = A_m + am \geq A_m > A_n$■

In order to prove the theorem it evidently suffices to show two things: I. A player moving from some (A_n,B_n) lands in a position not of the form (A_i,B_i). II. Given any position $(x,y) \neq (A_i,B_i)$ there is a move to some (A_i,B_i). (It is useful to note that these two conditions are also necessary; the definition of *P* and *N* implies that all positions reachable in one move form a *P*-position are *N*-positions; whereas at least one *P*-position is reachable in one move from an *N*-position.)

I. A move of the first type from (A_n,B_n) clearly leads to a position not of the form (A_i,B_i). Suppose a move of the second type from (A_n,B_n) produces a position (A_i,B_i). Then $i < n$. A move of the second type satisfies $|(B_n - A_n) - (B_i - A_i)| < a$, that is, $|(n - i)a| < a$, which implies $i = n$, a contradiction.

II. Let (x, y) with $x \leq y$ be a position not of the form (A_i, B_i) ($i \geq 0$). Since A and B are complimentary, every positive integer appears in exactly one of A and B. Therefore we have either $x = B_n$ or else $x = A_n$ for some $n \geq 0$.

Case (i): $x = B_n$. Then move y --> A_n.

Case (ii): $x = An$. If $y > B_n$, then move y --> B_n. If $A_n \leq y < B_n$, let $d = y - x$, $m = \lfloor d/a \rfloor$, where $\lfloor x \rfloor$ denotes the largest integer $\leq x$. The move (x,y) --> (A_m,B_m). This is a legal move, since:

(a) $d = y - A_n < B_n - A_n = an$,
hence $m = \lfloor d/a \rfloor \leq d/a < n$.

(b) $y = A_n + d > A_m + d \Rightarrow A_m + am = B_m$.

(c) $|(y - B_m) - (x - A_m)| = |(y - x) - (B_m - A_m)| =$
$|d - am| < a$.

"In order to play a game such as a Wythoff game as best as possible, it suffices to compute two things: (A) the nature of the present position u (P or N); (B) a next move if u is in N. Reason: let u be an arbitrary game position. If (A) shows that $u \in N$, then we know that there exists some move to a position in P. Moreover, we can use (B) to find one. If, on the other hand, (A) shows that $u \in P$, we cannot do much better than an arbitrary move while exuding an air of confidence and hoping for the best, since any position reachable in one move from a P-position is necessarily an N-position, from which our opponent can win if he knows how to compute (A) and (B). Now the *statement* of Theorem [5.1] shows how to compute (A), since the statement constitutes a characterization of the P-positions, whereas the *proof* of Theorem [5.1] indicates explicitly how to compute (B).

The computation of (A) and (B) (or (B) alone when the computation of (A) is already known) will be called a *strategy* in the sequel. Summarizing our present knowledge, we can thus say that Theorem [5.1] and its proof jointly constitute a recursive strategy for Wythoff games in which each P-position can be computed from the previous ones.

We close this section by briefly considering the complexity of the indicated strategy. Given a position (x,y) with $0 \leq x \leq y$, we need, for computing the next move,

to construct the table recursively only up to the smallest n such that either $A_n = x$ or $B_n = x$. Since $A_n \leq 2n$ for every a (follows from $B_n - B_{n-1} \geq 2$), this computation requires only $O(x)$ comparisons of the table entries with x, and $O(x)$ words of memory space. We remark that once the table has been computed and stored, it takes only $O(\log x)$ steps to locate A_n such that $x = A_n$ (or B_n such that $x = B_n$), by performing a binary search on the A_n (or B_n) sequence. Since computing (B) by the method indicated in the proof of Theorem [5.1] requires at most $O(\log x)$ steps, the total number of computation steps is only $O(x)$." [ASF pp.355-356]

A second theorem of Fraenkel [ASF p.355] gives an algebraic characterization of the P-positions. "Let

$$(1) \quad \alpha = \frac{2 - a + \mathrm{Sqrt}\,(a^2 + 4)}{2}, \quad \beta = \alpha + a$$

α is the positive root of the quadratic equation $\xi + (\xi + a)^{-1}$. Thus α and β are irrational for every positive integer a, and satisfy $\alpha^{-1} + \beta^{-1} = 1$.

The following "folk-theorem" dates back at least to Beatty [S.Beatty, Problem 3173, this MONTHLY, 33 (1926) 159; 34 (1927) 159]. It has many proofs and has often been rediscovered. The proof given below seems to be one of the

most elegant ones. I have heard that it is due to Ostrowski.

Lemma [5.3]. *Let α and β be positive irrationals satisfying* $\alpha^{-1} + \beta^{-1} = 1$. *Let* $A_n' = \lfloor n\alpha \rfloor$, $B_n' = \lfloor n\beta \rfloor$ $A' = \cup_{n=1,\infty} \{A_n'\}$ *and* $B' = \cup_{n=1,\infty} \{B_n'\}$. *Then A' and B' are complementary* [in the integers].

Proof. It suffices to show that exactly one member of the sequence $\zeta = \{\alpha,\beta,2\alpha,2\beta,3\alpha,3\beta,\ldots,n\alpha,n\beta,\ldots\}$ is in the interval $[h,h+1)$ for every positive integer h. Hence it suffices to show that if $M > 1$ is any integer, then there are precisely $M - 1$ members of ζ less than M. The number of $n\alpha < M$ is $\lfloor M/\alpha \rfloor$ and the number of $n\beta < M$ $\lfloor M/\beta \rfloor$. Thus the number of members of ζ less than M is $N = \lfloor M/\alpha \rfloor + \lfloor M/\beta \rfloor$. Now

$$\frac{M}{\alpha} - 1 < \left\lfloor \frac{M}{\alpha} \right\rfloor < \frac{M}{\alpha}, \quad \frac{M}{\beta} - 1 < \left\lfloor \frac{M}{\beta} \right\rfloor < \frac{M}{\beta}.$$

Adding, $M - 2 < N < M$. Since N is an integer, we conclude [that] $N = M - 1$■

Note that $A_0' = 0 = A_0$, $B_0' = 0 = B_0$ and $B_n' = A_n'$. Moreover, $\text{mex}\{A_i', Bi' : 0 \le i < n\} = A_n' (n \ge 0)$, since A_n' and B_n' are increasing sequences and A' and B' are complementary; if the mex were not A_n', then A_n' would never be obtained! This shows that $A_n' = A_n$ and $B_n' = B_n$ $(n \ge 0)$. We have proved:

Theorem [5.2]. *If α and β are given by* (1), *then* $P = \cup_{n=0,\infty} \{(\lfloor n\alpha \rfloor, \lfloor n\beta \rfloor)\}$.

A strategy based on this observation can be realized as follows. Since α is irrational and $1 < \alpha < 2$,

$$x = \lfloor n\alpha \rfloor \Leftrightarrow x < n\alpha < x+1 \Leftrightarrow \frac{x}{\alpha} < n < \left\lfloor \frac{x+1}{\alpha} \right\rfloor < \left\lfloor \frac{x+1}{\alpha} \right\rfloor = \left\lfloor \frac{x}{\alpha} \right\rfloor + 1,$$

where (x,y) with $x \le y$ is the given game position. Therefore either $x = \lfloor n\alpha \rfloor = A_n$ where $n = \lfloor (x+1)/\alpha \rfloor$, or else, by complementarity, $x = \lfloor n\beta \rfloor = B_n$, where $n = \lfloor (x+1)/\beta \rfloor$. We have thus reduced the situation to that considered in cases (ii) and (i) in the proof of theorem [5.1], and hence the strategy presented in that proof can be followed. For example, if $x = \lfloor n\alpha \rfloor = A_n$ and $y < \lfloor n\alpha \rfloor + na = \lfloor n\beta \rfloor$, then letting $m = \lfloor (y - x)/\alpha \rfloor$, we move to $(\lfloor m\alpha \rfloor, \lfloor m\beta \rfloor) \in P$. For implementing this strategy , α has to be computed to a precision of $O(\log x)$ digits, and its storage requires $O(\log x)$ words, which is only the same order of magnitude needed for storing x itself (in binary or decimal, say)."

There is no polynomial-time algorithm for finding the Grundy values for positions in Wythoff's game with two or more heaps. That is to say, no algorithm has been found that runs in polynomial time with regard to the size of the input. For the two-heap game, the input is two nonnegative integers; for the three-heap game, the input is three

nonnegative integers. These integers can be represented as binary numbers, so the size of the input is on the order of $\log_2(N)$.

U. Blass and A. Fraenkel [Blass] have published an algorithm for finding the Grundy value of a position in the two-heap Wythoff's game. The stated complexity of the algorithm is $O(s(n)^N)$, where $s(n)$ is the game's input size. The article gives the complexity of the naive algorithm as $O(s(n)^{N**3})$.

Their algorithm is given for A = 1, that is, the number of tokens that can be removed from the heaps in a Type-two move must be the same for each of two heaps. I have extended this algorithm to $A \in \mathbf{Z}^+$. The presentation follows that of their article in Theoretical Computer Science.

Let j be a nonnegative integer and A be a positive integer. We present a recursive algorithm to construct the sequence $\mathbf{S}_j$ of pairs (a,b) of nonnegative integers such that g(a,b) = j in a two-heap Wythoff's game where A - 1 is the maximum difference between the number of stones removed from each of the two heaps in a Type-two move and g is the Sprague-Grundy Function.

Let $\mathbf{S}_j = \{ (a_0,b_0), (a_1,b_1), \ldots \}$ be the sequence of pairs (a,b) s.t. $g(a,b) = j$.

Let $\mathbf{D}_j$ be the set of allowable differences between the number of stones removed from each of the two heaps in a Type-two move for the sequence $\mathbf{S}_j$. $\mathbf{D}_j$ is defined recursively as follows:

$\mathbf{D}_0 = \phi$

$\mathbf{D}_{i+1},\ i \in \mathbb{N} \ = \ \mathbf{D}_i \cup \mathbf{V}_i$

where

$\mathbf{V}_i = \cup_{k = -\nu,\ A-1} [a_i, b_i, k]$

$\nu = \min(b_i - a_i,\ A-1,\ \lceil (a_i + b_i)/2 \rceil)$

$[a_i, b_i, k] = \{ b_i - a_i + k \}$, if $b_i - a_i + k \geq 0$

ϕ, otherwise;

and,

(a_i, b_i) is the i^{th} pair found with Grundy value j.

We assume that all sequences $\mathbf{S}_i$, $i < j$, have been constructed up to the point where $\mathbf{P}$, yet to be defined, appears in one of the pairs (a,b), $a \leq b$, in each $\mathbf{S}_i$ and $\mathbf{S}_j$ has been constructed through pair (a_{k-1}, b_{k-1}). We now give the algorithm Extended Wythoff Sprague Grundy (EWSG) for constructing pair (a_k, b_k) of sequence $\mathbf{S}_j$.

Step 1. Put $\mathbf{P}$ <-- Mex $\{ a_0, b_0, a_1, b_1, \ldots, a_{k-1}, b_{k-1} \}$.

Put $\mathbf{Q}$ <-- Mex $\mathbf{D}_{k-1}$.

Step 2. If the pair $(P,P+Q)$ does not appear in any S_i, $i < j$, constructed so far and $P + Q$ does not appear as the second term in any pair in S_j constructed so far then Put (a_k,b_k) <-- $(P,P+Q)$ end.

Step 3. Otherwise, Put Q <-- $Q+R$, where R is the smallest nonnegative integer such that $Q+R$ does not appear in D_k. Return to Step 2.

The validity of algorithm EWSG is proved using a lemma and a theorem. The lemma asserts the basic properties of the algorithm and the theorem claims that the algorithm gives the correct results.

Lemma 5.4. EWSG is an algorithm and produces sets of ordered pairs that are distinct and not reachable from another pair in the same set.

Proof.

The algorithm halts, since for a Q sufficiently large, as produced in Step 3., $(P,P+Q)$ is not in any S_i, $i < j$; and $P+Q$ did not yet appear in B_j, the set of all second terms in the pairs (a,b) in S_j.

No pair (a,b) in S_j is reachable from any other pair in S_j. This is so since the elements of D_j, (the differences $b - a$), are used at most once, thus a Type-two move is not effective. A_j, the collection of all first elements in the

ordered pairs (a,b) in $\mathbf{S}_j$, are distinct and increasing, by choice of **P**. Similarly, $\mathbf{B}_j$, the collection of second elements in the ordered pairs (a,b) in $\mathbf{S}_j$ are distinct and increasing, by choice of **P** and **Q**. Moreover, the sets $\mathbf{A}_j$ and $\mathbf{B}_j$ are complementary. Thus, no Type-one move, either (a-k,b) or (a,b-k), for some positive integer k, is effective in reducing one pair in $\mathbf{S}_j$ to another.

Theorem 5.3. For all nonnegative integers j, $\mathbf{S}_j$ contains exactly those pairs (a,b) such that $g(a,b) = j$.

Proof.

Suppose the theorem is false and let (a_n,b_n) be the minimum counterexample; that is, for all $i < j$, $\mathbf{T}_i$ contains exactly those pairs (a,b) such that $g(a,b) = i$ and for all $m < n$, $g(a_m,b_m) = j$ if (a_m,b_m) is an element of $\mathbf{S}_j$.

$j > 0$, since $\mathbf{S}_0$ is the set of all safe pairs, that is, $g(a,b) = 0$.

$m > 0$, since the first row of every two-heap Wythoff's game with $A > 0$, is the sequence 0, 1, 2, 3, ... which is the sequence of values of $g(a_0,b_0)$, for $(a_0,b_0) \in \mathbf{S}_i$, for every $i \in \mathbb{N}$.

Since (a_n,b_n) is a counterexample then either $(A_n,B_n) \in \mathbf{S}_j$ and $g(a_n,b_n) \neq j$; or, $(A_n,B_n) \notin \mathbf{S}_j$ and $g(a,b) = j$.

Suppose $(a_n,b_n) \in \mathbf{S}_j$ and $g(a_n,b_n) \neq j$. By induction hypothesis, $g(a_n,b_n) > j$. So, there exists $(a_k,b_k) \in \mathbf{S}_j$ with $g(a_k,b_k) = j$ and (a_k,b_k) is reachable from (a_n,b_n).

1. (a_k,b_k) is reachable from (a_n,b_n)
2. $a_k \leq a_n$, 1.
3. $b_k \leq b_n$, 1.
4. $a_k \neq a_n$, by choice of **P**
5. b_k not equal to b_n, by choice of **P**, **Q**
6. there does not exist a Type-one move from (a_n,b_n) to (a_k,b_k), by 4, 5
7. (a_k,b_k) not reachable from (a_n,b_n) by a Type-two move, by choice of **Q**
8. (a_k,b_k) is not reachable from (a_n,b_n), 6,7

Suppose $g(a_n,b_n) = j$ and $(a_n,b_n) \notin \mathbf{S}_j$. By induction hypothesis, $g(a_n,b_n)$ is an element of, say, $\mathbf{S}_r$, $r > j$. By choice of **P**, there exists pair (a_n,b_k), element of $\mathbf{S}_j$ such that $g(a_n,b_k) = j$. Now, (a_n,b_k) is not reachable from (a_n,b_n), for otherwise, $g(a_n,b_n)$ would be greater than j, which implies that $b_k > b_n$, which implies that there exists a pair $(a_s,b_n) \in \mathbf{S}_j$ with $s < n$, but, then, (a_s,b_n) is reachable from (a_n,b_n), which is a contradiction■

A program for finding the Grundy values of positions for the two-heap Wythoff's game with A = 1 appears below. The program uses a new algorithm which, for the two-heap game, appears similar to that of Blass and Fraenkel. The algorithm uses geometric projections, which in the two-heap game amount to linear projections of the two-dimensional chessboard, to store the information needed to determine the

Grundy values associated with each of the positions. This algorithm was developed after my algorithm for the decision problem for the three-heap Wythoff's game and prior to seeing the article of Blass and Fraenkel.

```
FINDS THE GRUNDY NUMBERS FOR POSITIONS IN THE TWO-HEAP
WYTHOFF'S GAME WITH A = 1

      INTEGER N
      PARAMETER ( N = 50 )

      INTEGER NPLUS1
      PARAMETER ( NPLUS1 = 51 )

      INTEGER SQUARE
      PARAMETER ( SQUARE = 20 )

      INTEGER G(0:NPLUS1,0:NPLUS1)

      LOGICAL R(0:NPLUS1), D(0:NPLUS1)

      INTEGER MINNY(0:N)


      INTEGER X, Y

      INTEGER V, LBX, DIF

      INTEGER I, J


      DO 2 I = 1, NPLUS1
      DO 3 J = 1, NPLUS1

         G(I,J) = -1

3     CONTINUE
2     CONTINUE

      DO 5 I = 0, N

         G(0,I) = I

5     CONTINUE

      G(0, NPLUS1) = -1
```

```
      DO 7 I = 0, N
         MINNY(I) = I
7     CONTINUE

      DO 10 V = 0, N

         DO 15 I = 0, NPLUS1
            R(I) = .TRUE.
            D(I) = .TRUE.
15       CONTINUE
         R(V) = .FALSE.
         D(V) = .FALSE.
         LBX = 1
         X = 1
         DIF = 0

20       IF ( X.LE.N ) THEN
25          IF ( G(X,MINNY(X)).GE.0 ) THEN
               MINNY(X) = MINNY(X) + 1
               GOTO 25
            ENDIF
30          IF ( .NOT. D(DIF) ) THEN
               DIF = DIF + 1
               GOTO 30
            ENDIF
            Y = MAX ( X + DIF, MINNY(X) )
50          IF ( Y.LE.N ) THEN
               IF ( R(X).AND.R(Y).AND.D(Y-X)
     +              .AND.G(X,Y).LT.0 ) THEN
```

```
                  G(X,Y) = V

                  R(X) = .FALSE.
                  R(Y) = .FALSE.

                  D(Y-X) = .FALSE.

               ELSE

                  Y = Y + 1

                  GOTO 50

               ENDIF

            ENDIF
60          IF ( .NOT. R(LBX) ) THEN

               LBX = LBX + 1

               GOTO 60

            ENDIF

            X = MAX ( X + 1, LBX )
65          IF (.NOT. R(X)) THEN

               X = X + 1

               GOTO 65

            ENDIF

            GOTO 20

         ENDIF

10    CONTINUE

      DO 70 I = 0, SQUARE

      DO 75 J = I + 1, SQUARE

         G(J,I) = G(I,J)

75    CONTINUE

70    CONTINUE
```

```
      DO 80 I = 0, SQUARE
         WRITE (6,1000) ( G(I,J), J = 0, SQUARE )
1000     FORMAT(1X, T5, 21I6 / )
80    CONTINUE

      END
```

TABLE OF GRUNDY VALUES FOR THE TWO-HEAP WYTHOFF'S GAME WITH A = 1

0	1	2	3	4	5	6	7	8
1	2	0	4	5	3	7	8	6
2	0	1	5	3	4	8	6	7
3	4	5	6	2	0	1	9	10
4	5	3	2	7	6	9	0	1
5	3	4	0	6	8	10	1	2
6	7	8	1	9	10	3	4	5
7	8	6	9	0	1	4	5	3
8	6	7	10	1	2	5	3	4
9	10	11	12	8	7	13	14	15
10	11	9	8	13	12	0	15	16
11	9	10	7	12	14	2	13	17
12	13	14	15	11	9	16	17	18
13	14	12	11	16	15	17	2	0
14	12	13	16	15	17	18	10	9
15	16	17	18	10	13	12	19	14
16	17	15	14	19	18	20	21	12
17	15	16	13	18	11	14	12	19

18	19	20	21	17	16	15	22	23
19	20	18	17	14	21	11	16	24
20	18	19	22	21	23	24	25	26

The condition G(X,Y).LT.0 states that pair (X,Y) has not been assigned a Grundy value and can be eliminated by the use of a different data structure such as a linked list. The runtime complexity of the algorithm was determined after the program was modified to separate the test G(X,Y).LT.0 from the tests of Type 2 Blocks (Type-one and Type-two moves). The runtime complexity of the algorithm is $O(s(n)^N)$. The modification is shown below; where COUNT = COUNT + 1 counts the number of tests on Type 2 Blocks.

```
101          IF ( Y.LE.YLMT ) THEN
                IF ( G(X,Y).GE.0 ) THEN
                   Y = Y + 1
                   GOTO 101
                ENDIF
             ENDIF
50              IF ( Y.LE.N ) THEN
                   COUNT = COUNT + 1
                   IF ( R(X).AND.R(Y).AND.D(Y-X) ) THEN
                      G(X,Y) = V
                      R(X) = .FALSE.
                      R(Y) = .FALSE.
                      D(Y-X) = .FALSE.
```

```
                ELSE
                   Y = Y + 1
                   GOTO 50
                ENDIF
             ENDIF
102       IF ( Y.LE.YLMT ) THEN
             IF ( G(X,Y).GE.0 ) THEN
                Y = Y + 1
                GOTO 102
             ENDIF
          ENDIF
```

The naive algorithm for the two-heap Wythoff's game is to mimic the computation of the mex of the Grundy values of all the positions reachable by a queen on a two-dimensional chessboard in The Queen's Game.

```
      SUM = 0

      DO 15 I = 1, NUMBER_OF_ROWS
      DO 25 J = I, NUMBER_OF_COLUMNS

           SUM = SUM + I + J + MIN( I, J )

25    CONTINUE
15    CONTINUE
```

The complexity of which is $O(s(n)^{N**3})$, where s(n) is the game's input size.

What follows are programs using the naive algorithm to find Grundy numbers for the two-heap game with A = 2 and A = 3.

```
* FINDS THE GRUNDY VALUES FOR THE TWO-HEAP WYTHOFF'S GAME
* FOR A=2

      INTEGER LMT

      PARAMETER ( LMT = 51 )

      INTEGER VALUES( 0:LMT, 0:LMT )

      LOGICAL INSET ( 0:LMT )

      INTEGER I, J, K, N, R, C

      INTEGER MEX

* THE VALUES FOR A SINGLE HEAP ARE KNOWN

      DO 5 I = 0, LMT

        VALUES( 0, I ) = I
        VALUES( I, 0 ) = I

5     CONTINUE

* FIND THE VALUES FOR EACH SIZE OF THE GAME N > 0

      DO 100 N = 2, LMT

        DO 10 J = 1, N/2

          K = N - J

*         FOR EACH PARTITION OF N INTO TWO NONEMPTY PARTS
*         FIND THE GRUNDY VALUE ASSOCIATED WITH THAT PAIR

*         INITIALLY THE SET OF GRUNDY VALUES ASSOCIATED WITH
*         THE OPTIONS AVAILABLE FOR THAT PAIR ARE EMPTY

          DO 15 I = 0, LMT
```

```
            INSET(I) = .FALSE.
15          CONTINUE
*           FIND THE GRUNDY VALUES ASSOCIATED WITH THE ROOK
            DO 20 R = 0, J - 1
               INSET( VALUES( R, K ) ) = .TRUE.
20          CONTINUE
            DO 25 C = 0, K - 1
               INSET( VALUES( J, C ) ) = .TRUE.
25          CONTINUE
*           AND THE VALUE OF THE EXTENDED BISHOP
            DO 45 R = 1, MIN(J,K)
               INSET ( VALUES(J-R,K-R) ) = .TRUE.
45          CONTINUE
            DO 55 R = 1, MIN(J+1,K)
               INSET ( VALUES(J-R+1,K-R) ) = .TRUE.
55          CONTINUE
            DO 65 R = 1, MIN(J,K+1)
               INSET ( VALUES(J-R,K-R+1) ) = .TRUE.
65          CONTINUE
*           THE VALUE OF THE POSITION IS THE MEX OF INSET
            MEX = 0
30          IF ( INSET( MEX ) ) THEN
               MEX = MEX + 1
               GOTO 30
            ENDIF
```

```
            VALUES( J, K ) = MEX
            VALUES( K, J ) = MEX
10        CONTINUE
100     CONTINUE

        WRITE ( 6, 1000 ) ( I, I = 0, 30 )
1000    FORMAT( 1X, T8, 31I4 / )
        DO 50 R = 0, 30
           WRITE ( 6, 2000 ) R, ( VALUES( R, C ), C = 0, 30 )
2000       FORMAT( 1X, T3, I4, T8, 31I4 )
50      CONTINUE

        END
```

TABLE OF GRUNDY VALUES FOR THE TWO-HEAP WYTHOFF'S GAME WITH A = 2

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	1	2	3	4	5	6	7	8	9	10	11	12
1	1	2	3	0	5	6	7	4	9	10	11	8	13
2	2	3	4	5	6	1	0	8	10	11	12	7	14
3	3	0	5	6	7	4	8	2	1	12	13	14	15
4	4	5	6	7	8	9	10	11	12	3	0	1	2
5	5	6	1	4	9	10	11	12	7	13	14	2	3
6	6	7	0	8	10	11	12	13	14	15	9	16	17
7	7	4	8	2	11	12	13	14	15	16	17	10	18
8	8	9	10	1	12	7	14	15	16	17	18	19	20
9	9	10	11	12	3	13	15	16	17	18	19	20	21
10	10	11	12	13	0	14	9	17	18	19	20	21	22
11	11	8	7	14	1	2	16	10	19	20	21	22	23
12	12	13	14	15	2	3	17	18	20	21	22	23	24
13	13	14	9	16	17	0	5	19	11	22	23	24	25
14	14	15	16	17	18	19	4	20	21	23	24	25	26
15	15	12	17	10	19	20	1	6	22	24	25	26	27
16	16	17	18	9	20	15	2	3	23	25	26	27	28
17	17	18	13	11	21	8	3	0	5	26	15	28	29
18	18	19	20	21	16	22	23	1	4	27	28	29	30
19	19	16	15	22	23	24	25	5	6	28	29	18	31

```
 20   20   21   22   23   14   25   19   26    0    7   30   13   16
 21   21   22   23   20   13   26   27   28    2    4   31   12   32
 22   22   23   24   25   26   21   28    9    3    1   27   32   33
 23   23   20   19   18   15   27   24   29   30    0    6   33   34
 24   24   25   26   27   28   17   29   30   31    2    4   34   35
 25   25   26   21   24   29   16   30   31   32    5    1   35    7
 26   26   27   28   19   30   31   32   33   34    6    3   36    5
 27   27   24   29   30   31   18   33   34   13    8    2    0    6
 28   28   29   30   31   22   32   34   25   26   14    5    4    8
 29   29   30   25   28   32   33   35   36   37   38    7    3    1
 30   30   31   32   33   34   23   18   27   28   35    8    6    0
```

```
* FINDS THE GRUNDY VALUES FOR THE TWO-HEAP WYTHOFF'S GAME
* FOR A=3

      INTEGER LMT

      PARAMETER ( LMT = 51 )

      INTEGER VALUES( 0:LMT, 0:LMT )

      LOGICAL INSET ( 0:LMT )


      INTEGER I, J, K, N, R, C

      INTEGER MEX


* THE VALUES FOR A SINGLE HEAP ARE KNOWN

      DO 5 I = 0, LMT

        VALUES( 0, I ) = I
        VALUES( I, 0 ) = I

5     CONTINUE


* FIND THE VALUES FOR EACH SIZE OF THE GAME N > 0

      DO 100 N = 2, LMT

        DO 10 J = 1, N/2

          K = N - J

*         FOR EACH PARTITION OF N INTO TWO NONEMPTY PARTS
*         FIND THE GRUNDY VALUE ASSOCIATED WITH THAT PAIR
```

```
*           INITIALLY THE SET OF GRUNDY VALUES ASSOCIATED WITH
*           THE OPTIONS AVAILABLE FOR THAT PAIR ARE EMPTY
            DO 15 I = 0, LMT
               INSET(I) = .FALSE.
15          CONTINUE

*           FIND THE GRUNDY VALUES ASSOCIATED WITH THE ROOK
            DO 20 R = 0, J - 1
               INSET( VALUES( R, K ) ) = .TRUE.
20          CONTINUE
            DO 25 C = 0, K - 1
               INSET( VALUES( J, C ) ) = .TRUE.
25          CONTINUE

*           AND THE VALUE OF THE EXTENDED BISHOP
            DO 45 R = 1, MIN(J,K)
               INSET ( VALUES(J-R,K-R) ) = .TRUE.
45          CONTINUE
            DO 55 R = 1, MIN(J-1,K)
               INSET ( VALUES(J-R-1,K-R) ) = .TRUE.
55          CONTINUE
            DO 65 R = 1, MIN(J,K-1)
               INSET ( VALUES(J-R,K-R-1) ) = .TRUE.
65          CONTINUE
            DO 75 R = 1, MIN(J-2,K)
               INSET ( VALUES(J-R-2,K-R) ) = .TRUE.
75          CONTINUE
```

```
            DO 85 R = 1, MIN(J,K-2)
               INSET ( VALUES(J-R,K-R-2) ) = .TRUE.
85          CONTINUE

*           THE VALUE OF THE POSITION IS THE MEX OF INSET
            MEX = 0
30          IF ( INSET( MEX ) ) THEN
               MEX = MEX + 1
               GOTO 30
            ENDIF
            VALUES( J, K ) = MEX
            VALUES( K, J ) = MEX
10        CONTINUE
100     CONTINUE

        WRITE ( 6, 1000 ) ( I, I = 0, 30 )
1000    FORMAT( 1X, T8, 31I4 / )
        DO 50 R = 0, 30
           WRITE ( 6, 2000 ) R, ( VALUES( R, C ), C = 0, 30 )
2000       FORMAT( 1X, T3, I4, T8, 31I4 )
50      CONTINUE

        END
```

TABLE OF GRUNDY VALUES FOR THE TWO-HEAP WYTHOFF'S GAME WITH A = 3

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	1	2	3	4	5	6	7	8	9	10	11	12
1	1	2	3	4	0	6	7	8	9	5	11	12	13
2	2	3	4	5	6	7	1	9	0	10	12	13	14
3	3	4	5	6	7	8	9	10	2	11	1	14	0
4	4	0	6	7	8	9	10	11	12	13	3	15	2
5	5	6	7	8	9	10	11	12	13	14	15	16	4
6	6	7	1	9	10	11	12	13	14	15	16	17	18
7	7	8	9	10	11	12	13	14	15	16	17	18	19
8	8	9	0	2	12	13	14	15	16	17	18	19	20
9	9	5	10	11	13	14	15	16	17	18	19	20	21
10	10	11	12	1	3	15	16	17	18	19	20	21	22
11	11	12	13	14	15	16	17	18	19	20	21	22	23
12	12	13	14	0	2	4	18	19	20	21	22	23	24
13	13	14	8	15	16	17	19	20	21	22	23	24	25

The next program is an implementation of a new algorithm for finding if a given position in the three-heap Wythoff's game with A = 1 is safe or unsafe, that is, respectively, in P or in N. The algorithm uses projections of the cube to answer the decision question and a sample output is provided for the position (5, 5, 30). Within the framework of the experiment, the runtime complexity of this algorithm is $O(N^{2.5})$, where N is the size of the largest

heap. The size of the input string is on the order of $\log_2(N)$.

By marking six positions in the projection DIA (in the IF statement with label 400) instead of three, for example, DIA(X, Z-Y) = .FALSE. is split into the two lines DIA(MIN(X,Z-Y), MAX(X,Z-Y)) = .FALSE. and DIA(MAX(X,Z-Y), MIN(X,Z-Y)) = .FALSE., the runtime complexity is reduced to between $O(N^2\log_2 N)$ and $O(N^{2.5})$.

```
      INTEGER XLMT, YLMT, ZLMT
      LOGICAL ROW(0:121,0:121), DIA(0:121,0:121)
      INTEGER LB(0:41), DIF(0:41)
      INTEGER I, J, K, X, Y, Z, N
      LOGICAL SAFE

2     WRITE (5,*) 'ENTER AN INTEGER BETWEEN 0 AND 40'
      READ *, XLMT
      IF (XLMT.LT.0 .OR. XLMT.GT.40) GOTO 2

3     WRITE (5,*) 'ENTER AN INTEGER BETWEEN 0 AND 40'
      READ *, YLMT
      IF (YLMT.LT.0 .OR. YLMT.GT.40) GOTO 3

4     WRITE (5,*) 'ENTER AN INTEGER BETWEEN 0 AND 40'
      READ *, ZLMT
      IF (ZLMT.LT.0 .OR. ZLMT.GT.40) GOTO 4

      IF (XLMT.GT.YLMT) THEN
```

```
      CALL SWAP( XLMT, YLMT )
   ENDIF
   IF (XLMT.GT.ZLMT) THEN
      CALL SWAP( XLMT, ZLMT )
   ENDIF
   IF ( YLMT.GT.ZLMT ) THEN
      CALL SWAP ( YLMT, ZLMT )
   ENDIF

   IF ( XLMT.EQ.YLMT .OR. YLMT.EQ.ZLMT ) THEN

      IF ( XLMT.EQ.ZLMT ) THEN
         PRINT *, XLMT, YLMT, ZLMT, 'SAFE'
      ELSE
         PRINT *, XLMT, YLMT, ZLMT, 'UNSAFE'
      ENDIF
      STOP

   ELSE

      DO 5 J = 0, 121
      DO 5 I = 0, 121
         ROW(I,J) = .TRUE.
         DIA(I,J) = .TRUE.
5     CONTINUE
      DO 7 I = 0, 41
         LB(I) = I + 1
         DIF(I) = 1
```

```
7         CONTINUE
          SAFE = .FALSE.
200       IF ( X.LE.XLMT ) THEN
20           IF ( .NOT. ROW( X, LB(X))) THEN
                LB(X) = LB(X) + 1
                GOTO 20
             ENDIF
25           IF ( .NOT. DIA( X, DIF(X)) ) THEN
                DIF(X) = DIF(X) + 1
                GOTO 25
             ENDIF
             Y = LB(X)
300          IF ( Y.LE.YLMT ) THEN
                Z = Y + DIF(X)
400             IF ( Z.LE.ZLMT ) THEN
                   IF ( ROW(X,Y) .AND. ROW(X,Z) .AND.
     +                   ROW(Y,Z) .AND. DIA(X,Z-Y) .AND.
     +                   DIA(Y,Z-X) .AND. DIA(Z,Y-X) ) THEN
                      IF ( X.EQ.XLMT .AND. Y.EQ.YLMT .AND.
     +                     Z.EQ.ZLMT ) THEN
                         SAFE = .TRUE.
                      ENDIF
                      ROW(X,Y) = .FALSE.
                      ROW(Y,Z) = .FALSE.
                      ROW(X,Z) = .FALSE.

                      DIA(X,Z-Y) = .FALSE.
                      DIA(Y,Z-X) = .FALSE.
```

```
                    DIA(Z,Y-X) = .FALSE.
30                  IF ( .NOT. DIA(X,DIF(X)) ) THEN
                       DIF(X) = DIF(X) + 1
                       GOTO 30
                    ENDIF
35                  IF ( .NOT. ROW(X,LB(X)) ) THEN
                       LB(X) = LB(X) + 1
                       GOTO 35
                    ENDIF
                    Y = MAX ( LB(Y), Y+1 )
                    GOTO 300
                 ENDIF
                 Z = Z + 1
                 GOTO 400
              ENDIF
              Y = Y + 1
              GOTO 300
           ENDIF

           X = X + 1
           GOTO 200
        ENDIF

        IF ( SAFE ) THEN
           PRINT *, XLMT, YLMT, ZLMT, 'SAFE'
        ELSE
           PRINT *, XLMT, YLMT, ZLMT, 'UNSAFE'
```

```
   ENDIF
ENDIF

END

SUBROUTINE SWAP ( A, B )
INTEGER A, B
INTEGER C
C = A
A = B
B = C
END
```

```
     5          5          30 UNSAFE
```

The output of a modified program that checks if the triples (m, m+1, m+2), $0 \leq m \leq 282$, are safe appears in Appendix A. The program was run to check the runtime complexity of the improved algorithm. The output shows that the triple (0, 1, 2) is safe; but, that the triples (m, m+1, m+2), for $1 \leq m \leq 253$, are unsafe. This lead to the following conjecture

Conjecture 5.2.

Let (m, m+1, m+2), $m \in \mathbb{N}$, represent positions in the three-heap Wythoff's game with A =1, then the triple (0, 1, 2) is safe; but, for all $m \in \mathbb{Z}^+$, the triples (m, m+1, m+2) are not safe.

The following program is an implementation of a new algorithm for finding the Grundy values of positions in the three- heap Wythoff's game with A = 1. No algorithm for the three-heap game has been published. This implementation uses a three-dimensional array to store the Grundy values

and counters to determine the runtime complexity of this algorithm. The primitive operations were considered to be: 1) the block of tests for Type-one and Type-two moves (Type 2 Blocks); and, 2) the number of triples processed (operations on z). These operations are, respectively, tests of the lower and upper bounds.

The algorithm appears to have a complexity between $O(N^3)$ and $O(N^4)$, with $O(N^3 \log_2(N))$ a likely choice. The complexity of the naive algorithm for finding the Grundy value of a position in a three-heap Wythoff's game is $O(N^4)$, where N is the size of the largest heap and the game's input size is on the order of $\log_2(N)$.

The output represents layers of a 32 x 32 x 32 cube, that is, the positions from (0, 0, 0) to (31, 31, 31). Each layer contains $3(n^2 - n) + 1$ triples, where n = 33 - layer.

A listing of triples with Grundy values 0, 1, and 2 for the three-heap Wythoff's game appears in Appendix B.

```
FINDS THE GRUNDY VALUES FOR POSITIONS IN THE THREE-HEAP
WYTHOFF'S GAME WITH A = 1.  THE OUTPUT OF THIS PROGRAM IS
DATA FOR THE RUNTIME ANALYSIS OF THE ALGORITHM.

      INTEGER UB
      PARAMETER ( UB = 40 )

      INTEGER UBP1
      PARAMETER ( UBP1 = 41 )

      INTEGER GV ( 0:UBP1, 0:UBP1, 0:UBP1 )

      INTEGER MINZ ( 0:UBP1, 0:UBP1 )
```

```
      INTEGER ROW ( 0:UBP1, 0:UBP1 ), DIA ( 0:UBP1, 0:UBP1 )

      INTEGER LB( 0:UBP1 ), DIF ( 0:UBP1 )

      INTEGER XLMT, YLMT, ZLMT, X, Y, Z, W

      INTEGER I, J, K

      INTEGER V

      INTEGER COUNT, VALUES, DONE, CC

      INTEGER LC, LCC, NEWV

      INTEGER LMT


      WRITE( 6, 4000 )
4000  FORMAT( 1X, T6, 'LAYER', T14, 'POSITIONS', T29,
     +                'OPERATIONS ON Z', T51,
     +                'TYPE 2 BLOCKS', T71, 'RATIO' / )

      ZLMT = 31
      YLMT = ZLMT

      LMT = ZLMT + 1

      VALUES = 0

      LC = 0

      LCC = 0


      DO 999 XLMT = 0, ZLMT

*     INITIALLY ALL GRUNDY VALUES ARE UNASSIGNED

      DO 15 I = 0, UBP1
      DO 25 J = 0, UBP1
      DO 35 K = 0, UBP1

         GV ( I, J, K ) = -1

35    CONTINUE
25    CONTINUE
15    CONTINUE

*     INITIALIZE MINZ:  MINIMUM Z VALUE FOR THE PAIR X, Y

      DO 40 I = 0, UB
      DO 45 J = I, UB
```

```
         MINZ ( I,J ) = J

45     CONTINUE
40     CONTINUE

       DO 47 I = 0, UBP1

          MINZ ( I, UBP1 ) = -1

47     CONTINUE

*      INITIALIZE ROW AND DIA

       DO 50 I = 0, UB
       DO 55 J = 0, UB

          ROW ( I,J ) = -1
          DIA ( I,J ) = -1

55     CONTINUE
50     CONTINUE

*      SET SENTINEL VALUES FOR DIA AND ROW

       W = XLMT + YLMT + ZLMT + 1

       DO 32 I = 0, UBP1

          DIA( I, UBP1 ) = W
          DIA( UBP1, I ) = W

          ROW( I, UBP1 ) = W
          ROW( UBP1, I ) = W

32     CONTINUE

       COUNT = 0

       VALUES = VALUES + LMT * ( LMT + 1 ) / 2

       NEWV = LMT * ( LMT + 1 ) / 2

       LMT = LMT - 1

       DONE = 0

       CC = 0

* A TRIPLE CANNOT BE ASSIGNED A GRUNDY VALUE IF IT HAS
* ALREADY BEEN ASSIGNED A ( SMALLER ) VALUE

       V = 0
```

```
105   IF ( DONE.LT.VALUES ) THEN
*     INITIALIZE THE LOCAL VALUES : LB, DIF
      DO 60 I = 0, UB
         LB(I) =  I
         DIF(I) = 0
60    CONTINUE
      X = 0
205   IF ( X.LE.XLMT ) THEN
*     LOOK FOR TRIPLES WITH GRUNDY VALUE V
         Y = LB(X)
305      IF ( Y.LE.YLMT ) THEN
*        KEEP LOOKING FOR TRIPLES WITH THE SAME VALUE FOR X
            Z = MAX ( Y + DIF(X), MINZ( X,Y ) )
            COUNT = COUNT + 1
315         IF ( GV( X,Y,Z ).GT. -1 ) THEN
               Z = Z + 1
               COUNT = COUNT + 1
               GO TO 315
            ENDIF
405         IF ( Z.LE.ZLMT ) THEN
*              ADD TO THE AMOUNT OF WORK DONE
               CC = CC + 1
*           CHECK IF THE VALUE OF ( X, Y, Z ) IS V
               IF ( ROW(X,Y).LT.V .AND. ROW(X,Z).LT.V .AND.
     +              ROW(Y,Z).LT.V .AND. DIA(X,Z-Y).LT.V
     +              .AND. DIA(Y,Z-X).LT.V .AND.
     +              DIA(Z,Y-X).LT.V ) THEN
*              COMPLETE THE PAPER WORK
```

```
                  DONE = DONE + 1
*                 UPDATE GLOBAL VALUE GV
                  GV( X, Y, Z ) = V
*                 UPDATE LOCAL VALUES ROW, DIA, LB, DIF
                  ROW(X,Y) = V
                  ROW(X,Z) = V
                  ROW(Y,Z) = V
                  DIA(X,Z-Y) = V
                  DIA(Y,Z-X) = V
                  DIA(Z,Y-X) = V
70                IF ( ROW( X,LB(X) ).EQ.V ) THEN
                     LB(X) = LB(X) + 1
                     GO TO 70
                  ENDIF
75                IF ( DIA( X,DIF(X) ).EQ.V ) THEN
                     DIF(X) = DIF(X) + 1
                     GO TO 75
                  ENDIF
415               IF ( GV( X,Y, MINZ( X,Y ) ).GT. -1 ) THEN
                     MINZ( X,Y ) = MINZ( X,Y ) + 1
                     GO TO 415
                  ENDIF
*                 AND TRY THE NEXT VALUE OF Y
                  Y = MAX ( LB(X), Y + 1 )
425               IF ( ROW( X, Y ).EQ.V ) THEN
                     Y = Y + 1
                     GOTO 425
                  ENDIF
```

```
                  GO TO 305
               ELSE
*              TRY THE NEXT VALUE OF Z
                  Z = Z + 1
                  COUNT = COUNT + 1
435               IF ( GV( X,Y,Z ).GT.-1 ) THEN
                     Z = Z + 1
                     COUNT = COUNT + 1
                     GO TO 435
                  ENDIF
                  GO TO 405
               ENDIF
            ENDIF
*           TRY THE NEXT VALUE OF Y
            Y = Y + 1
325           IF ( ROW( X, Y ).EQ.V ) THEN
                 Y = Y + 1
                 GOTO 325
              ENDIF
              GO TO 305
         ENDIF
*        GET THE NEXT VALUE OF X
         X = X + 1
         GO TO 205
      ENDIF
*     UPDATE THE GLOBAL VALUE MINZ
```

```
        DO 90 I = 0, XLMT
        DO 95 J = I, YLMT

99         IF ( GV( X, Y, MINZ(X,Y) ).GT.-1 ) THEN

              MINZ(X,Y) = MINZ(X,Y) + 1

              GO TO 99

           ENDIF

95      CONTINUE
90      CONTINUE

        V = V + 1

        GO TO 105

      ENDIF

      WRITE( 6, 5000 ) XLMT+1, NEWV, COUNT - LC, COUNT,
     +                  CC - LCC, CC, REAL( CC ) / VALUES
5000  FORMAT( 1X, T5, I6, I12, I12, I9, I11, I9, F12.3 / )

      LC = COUNT

      LCC = CC

999   CONTINUE

      END


      SUBROUTINE SWAP ( A, B )

      INTEGER A, B

      INTEGER C

      C = A

      A = B

      B = C

      END
```

LAYER	POSITIONS	OPERATIONS ON Z		TYPE 2 BLOCKS		RATIO
1	528	3124	3124	1497	1497	2.835
2	496	2974	6098	1610	3107	3.034
3	465	3127	9225	1712	4819	3.236
4	435	3117	12342	1661	6480	3.368
5	406	3534	15876	1756	8236	3.535
6	378	3602	19478	1896	10132	3.742
7	351	3308	22786	1726	11858	3.876
8	325	3379	26165	1810	13668	4.039
9	300	3292	29457	1702	15370	4.172
10	276	3360	32817	1672	17042	4.304
11	253	3270	36087	1604	18646	4.426
12	231	3393	39480	1547	20193	4.544
13	210	2792	42272	1534	21727	4.668
14	190	2849	45121	1508	23235	4.797
15	171	2782	47903	1495	24730	4.931
16	153	2509	50412	1364	26094	5.049
17	136	2825	53237	1359	27453	5.176
18	120	2164	55401	1159	28612	5.275
19	105	2024	57425	1110	29722	5.376
20	91	1792	59217	965	30687	5.460
21	78	2080	61297	831	31518	5.531
22	66	1990	63287	769	32287	5.601
23	55	1216	64503	682	32969	5.666
24	45	2106	66609	606	33575	5.726
25	36	915	67524	427	34002	5.763
26	28	557	68081	242	34244	5.777
27	21	1655	69736	305	34549	5.808
28	15	412	70148	157	34706	5.819
29	10	1411	71559	160	34866	5.836
30	6	776	72335	90	34956	5.845
31	3	105	72440	11	34967	5.844
32	1	64	72504	1	34968	5.844

The ratio of total number of Type 2 blocks to total number of positions processed approaches $\log_2(N) + 1$, where N is the size of all three of the heaps, appeared in runs with N equal to 16 and 32; and this is the source of the estimated complexity. The ratio between the number of

triples handled and the number of Type 2 blocks appears to be almost constant and suggests that the difference may be implementation dependent and that $O(N^3\log_2(N))$ represents the upper bound. The ratio of triples handled to core operations should approach 1 if the implementation used a linked list of positions yet to be processed . That at level 32 there appear 64 operations on z where there is only one triple, (32, 32, 32) and on that on level 31 there are 105 operations on z to find the triples (31, 31, 31), (31, 31, 32), and (31, 32, 32) appears to be a bug in the implementation which would be eliminated with the use of linked lists.

DEFINITION:
Let T be a mapping from $\mathbb{N} \times \mathbb{N}$ into $\mathbb{N}$ represented by a table T′. If the sum of every row and every column and the sums of both the major and minor diagonals are equal then T′ is an **infinite latin square.**

Conjecture 5.3.
The Grundy values for the two-heap Wythoff's game over $\mathbb{N} \times \mathbb{N}$ forms an infinite latin square.

The next three programs are part of an effort to find chesslike games that contain locally finite patterns within a pattern of infinite order.

LATIN ROOK

A Latin Rook [Figure 16] is placed on a chessboard. The rows of the chessboard are labelled 0, 1, 2, 3, ... starting from the Northwest corner of the board. The columns are, similarly, labelled 0, 1, 2, 3, ... starting from Northwest corner of the board. Two players, Left and Right, alternately move the Latin Rook towards the Northwest corner of the board. Left moves first. The first player unable to move loses and his or her opponent wins.

Let (R, C) be the position of the Latin Rook (R for row, C for column). A move is described as follows:

If R.gt.C then

either a) move the Latin Rook to a lower numbered row

or b) move the Latin Rook to a higher numbered column

Subject to the restriction that the final position of the Latin Rook satisfies R.ge.C

Elseif C.gt.R then

either a) move the Latin Rook to a higher numbered row

or b) move the Latin Rook to a lower numbered column

Subject to the restriction that the final position of the Latin Rook satisfies C.ge.R

Else

move the Latin Rook to a lower numbered position

along the main diagonal

Endif

The Grundy values for this game form successive latin squares of order $2^n - 1$, for $n \in \mathbf{Z}^+$.

A program for implementing this algorithm follows.

A PROGRAM FOR FINDING THE GRUNDY VALUES FOR THE TWO-HEAP LATIN ROOK'S GAME

```
      INTEGER LMT
      PARAMETER ( LMT = 31 )
      INTEGER VALUES ( 0:LMT, 0:LMT )
      LOGICAL INSET ( 0:100 )

      INTEGER I, J, K, N
      INTEGER MEX

      VALUES(0,0) = 0

      DO 5 N = 1, LMT
      VALUES(N,N) = N
      DO 10 I = N-1, 0, -1
        DO 15 K = 0, 100
           INSET(K) = .FALSE.
15      CONTINUE
        DO 20 J = 0, N-1
           INSET( VALUES(J,I) ) = .TRUE.
20      CONTINUE
        DO 25 J = N, I+1, -1
```

```
            INSET( VALUES(N,J) ) = .TRUE.
25        CONTINUE
          MEX = 0
30        IF ( INSET(MEX) ) THEN
             MEX = MEX + 1
             GO TO 30
          ENDIF
          VALUES( I, N ) = MEX
          VALUES( N, I ) = MEX
10      CONTINUE
5       CONTINUE

        DO 55 K = 0, 15
           WRITE ( 6,1000 ) ( VALUES(K,I), I = 0, 15 )
1000       FORMAT( 1X, 16I4 )
55      CONTINUE
        END
```

A TABLE OF GRUNDY VALUES FOR THE TWO-HEAP LATIN ROOK GAME

0	2	1	6	5	4	3	14	13	12	11	10	9	8
2	1	0	5	6	3	4	13	14	11	12	9	10	7
1	0	2	4	3	6	5	12	11	14	13	8	7	10
6	5	4	3	0	2	1	11	12	13	14	7	8	9
5	6	3	0	4	1	2	10	9	8	7	14	13	12
4	3	6	2	1	5	0	9	10	7	8	13	14	11
3	4	5	1	2	0	6	8	7	10	9	12	11	14
14	13	12	11	10	9	8	7	0	2	1	6	5	4
13	14	11	12	9	10	7	0	8	1	2	5	6	3
12	11	14	13	8	7	10	2	1	9	0	4	3	6
11	12	13	14	7	8	9	1	2	0	10	3	4	5
10	9	8	7	14	13	12	6	5	4	3	11	0	2
9	10	7	8	13	14	11	5	6	3	4	0	12	1
8	7	10	9	12	11	14	4	3	6	5	2	1	13
7	8	9	10	11	12	13	3	4	5	6	1	2	0
30	29	28	27	26	25	24	23	22	21	20	19	18	17

A queen is composed of a rook and a bishop. The rook represents the Type-one moves (along a row or column) and the bishop the Type-two moves (along a diagonal). An idea was to increase the power of the queen so that it attacks more squares and in doing so is forced to present some locally finite patterns.

Evenrook [Figure 17] is a game in which the bishop (on a two-dimensional chessboard) was replaced by a bishop that is allowed to move to any square as long as every heap (dimension) is either decremented or remains the same and the total number of squares the position is reduced is even. A program for this game follows. It was found that the piece was too strong in that it excluded values.

A PROGRAM FOR FINDING THE GRUNDY VALUES FOR THE TWO-HEAP EVENROOK GAME

```
      INTEGER LMT
      PARAMETER ( LMT = 126 )
      INTEGER VALUES( 0:LMT, 0:LMT )
      LOGICAL INSET ( 0:LMT )
      INTEGER I, J, K, N, R, C
      INTEGER MEX
* THE VALUES FOR A SINGLE HEAP ARE KNOWN
      DO 5 I = 0, LMT
        VALUES( 0, I ) = I
        VALUES( I, 0 ) = I
5     CONTINUE
* FIND THE VALUES FOR EACH SIZE OF THE GAME N > 0
```

```
      DO 100 N = 2, LMT

      DO 10 J = 1, N/2

         K = N - J

*        FOR EACH PARTITION OF N INTO TWO NONEMPTY PARTS
*        AND FIND THE GRUNDY VALUE ASSOCIATED WITH THAT PAIR

*        INITIALLY THE SET OF GRUNDY VALUES ASSOCIATED WITH
*        THE OPTIONS AVAILABLE FOR THAT PAIR ARE EMPTY

         DO 15 I = 0, LMT

            INSET(I) = .FALSE.

15       CONTINUE

*        FIND THE GRUNDY VALUES ASSOCIATED WITH THE ROOK

         DO 20 R = 0, J - 1

            INSET( VALUES( R, K ) ) = .TRUE.

20       CONTINUE

         DO 25 C = 0, K - 1

            INSET( VALUES( J, C ) ) = .TRUE.

25       CONTINUE

*        THEN FIND THE GRUNDY VALUE ASSOCIATED WITH EVEN
*        MOVES

         IF ( MOD(J+K,2).EQ.0 ) THEN

            DO 35 R = 0, J-1

            DO 45 C = 0, K-1

               IF ( MOD(R+C,2).EQ.0 ) THEN

                  INSET( VALUES(R,C) ) = .TRUE.

               ENDIF

45          CONTINUE

35          CONTINUE
```

```
            ELSE
               DO 55 R = 0, J-1
               DO 65 C = 0, K-1
                  IF ( MOD(R+C,2).NE.0 ) THEN
                     INSET( VALUES(R,C) ) = .TRUE.
                  ENDIF
65             CONTINUE
55             CONTINUE
            ENDIF
*           THE VALUE OF THE POSITION IS THE MEX OF INSET
            MEX = 0
30          IF ( INSET( MEX ) ) THEN
               MEX = MEX + 1
               GOTO 30
            ENDIF
            VALUES( J, K ) = MEX
            VALUES( K, J ) = MEX

10       CONTINUE
100   CONTINUE

      WRITE ( 6, 1000 ) ( I, I = 0, 15 )
1000  FORMAT( 1X, T5, 16I4 / )
      WRITE ( 6,* )
      DO 50 R = 0, 30
         WRITE ( 6, 2000 ) R, ( VALUES( R, C ), C = 0, 15 )
2000     FORMAT( 1X, T1, I2, T5, 16I4 / )
50    CONTINUE
```

```
      WRITE  ( 6,3500 ) (VALUES( I,I ), I = 0, 31 )
3500  FORMAT ( 1X, T5, 10 ( 1X, 16I3 // ) )

      END
```

OUTPUT:
A TABLE OF GRUNDY VALUES FOR THE TWO-HEAP EVENROOK GAME

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13
1	1	2	0	4	5	3	7	8	6	10	11	9	13	14
2	2	0	1	5	3	4	8	6	7	11	9	10	14	12
3	3	4	5	6	2	7	9	10	11	12	8	13	15	16
4	4	5	3	2	7	6	10	11	9	8	13	12	16	17
5	5	3	4	7	6	8	11	9	10	13	12	14	17	15
6	6	7	8	9	10	11	5	12	13	14	15	16	18	19
7	7	8	6	10	11	9	12	13	14	15	16	17	19	20
8	8	6	7	11	9	10	13	14	12	16	17	15	20	18
9	9	10	11	12	8	13	14	15	16	17	18	19	21	22
10	10	11	9	8	13	12	15	16	17	18	14	20	22	23
11	11	9	10	13	12	14	16	17	15	19	20	18	23	21
12	12	13	14	15	16	17	18	19	20	21	22	23	11	24
13	13	14	12	16	17	15	19	20	18	22	23	21	24	25
14	14	12	13	17	15	16	20	18	19	23	21	22	25	26
15	15	16	17	18	14	19	21	22	23	24	25	26	27	28
16	16	17	15	14	19	18	22	23	21	20	26	24	28	29
17	17	15	16	19	18	20	23	21	22	25	24	27	26	30
18	18	19	20	21	22	23	17	24	25	26	27	28	29	31

19	19	20	18	22	23	21	24	25	26	27	28	29	30	32
20	20	18	19	23	21	22	25	26	24	28	29	30	31	33
21	21	22	23	24	20	25	26	27	28	29	30	31	32	34
22	22	23	21	20	25	24	27	28	29	30	31	32	33	35
23	23	21	22	25	24	26	28	29	27	31	32	33	34	36
24	24	25	26	27	28	29	30	31	32	33	34	35	36	37
25	25	26	24	28	29	27	31	32	30	34	35	36	37	38
26	26	24	25	29	27	28	32	30	31	35	33	34	38	39
27	27	28	29	30	26	31	33	34	35	36	37	38	39	40
28	28	29	27	26	31	30	34	35	33	32	38	39	40	41
29	29	27	28	31	30	32	35	33	34	37	36	40	41	42
30	30	31	32	33	34	35	29	36	37	38	39	41	42	43

MISSING VALUES ALONG THE MAIN DIAGONAL

3	9	15	$6n + 3,\ n \in \mathbb{N}$
4	10	16	$6n + 4$

Another attempt to engender patterns was to extend the power of the queen to include the powers of the knight and the great knight (keima and okeima are the names of extensions in Go). Okeima is the extension 1, 3 as opposed to the keima which is the extension 1, 2). The piece is called the Commander [Figure 18].

```
* FINDS THE GRUNDY VALUES FOR A QUEEN + KEIMA + OKEIMA
[COMMANDER]

      INTEGER LMT
      PARAMETER ( LMT = 51 )


      INTEGER VALUES( 0:LMT, 0:LMT )

      LOGICAL INSET ( 0:LMT )


      INTEGER I, J, K, N, R, C

      INTEGER MEX


* THE VALUES FOR A SINGLE HEAP ARE KNOWN

      DO 5 I = 0, LMT

        VALUES( 0, I ) = I
        VALUES( I, 0 ) = I

5     CONTINUE


* FIND THE VALUES FOR EACH SIZE OF THE GAME N > 0

      DO 100 N = 2, LMT

        DO 10 J = 1, N/2


          K = N - J

*         FOR EACH PARTITION OF N INTO TWO NONEMPTY PARTS
*         FIND THE GRUNDY VALUE ASSOCIATED WITH THAT PAIR

*         INITIALLY THE SET OF GRUNDY VALUES ASSOCIATED WITH
*         THE OPTIONS AVAILABLE FOR THAT PAIR ARE EMPTY

          DO 15 I = 0, LMT

             INSET(I) = .FALSE.

15        CONTINUE


*         FIND THE GRUNDY VALUES ASSOCIATED WITH THE ROOK

          DO 20 R = 0, J - 1
```

```
            INSET( VALUES( R, K ) ) = .TRUE.
20          CONTINUE
            DO 25 C = 0, K - 1
               INSET( VALUES( J, C ) ) = .TRUE.
25          CONTINUE

*           THEN FIND THE GRUNDY VALUE ASSOCIATED WITH THE
*           KNIGHT
            IF ( J.GT.2 ) THEN
              R = J - 2
              C = K + 1
              INSET( VALUES(R,C) ) = .TRUE.

              R = J - 2
              C = K - 1
              INSET( VALUES(R,C) ) = .TRUE.

              R = J - 3
              C = K + 1
              INSET( VALUES(R,C) ) = .TRUE.

              R = J - 3
              C = K - 1
              INSET( VALUES(R,C) ) = .TRUE.
            ELSEIF ( J.GT.1 ) THEN
              R = J - 2
              C = K + 1
              INSET( VALUES(R,C) ) = .TRUE.

              R = J - 2
              C = K - 1
              INSET( VALUES(R,C) ) = .TRUE.
            ENDIF
            IF ( K.GT.2 ) THEN
              C = K - 2
              R = J + 1
              INSET( VALUES(R,C) ) = .TRUE.

              C = K - 2
              R = J - 1
```

```
              INSET( VALUES(R,C) ) = .TRUE.

              C = K - 3
              R = J + 1
              INSET( VALUES(R,C) ) = .TRUE.

              C = K - 3
              R = J - 1
              INSET( VALUES(R,C) ) = .TRUE.

            ELSEIF ( K.GT.1 ) THEN

              C = K - 2
              R = J + 1
              INSET( VALUES(R,C) ) = .TRUE.

              C = K - 2
              R = J - 1
              INSET( VALUES(R,C) ) = .TRUE.

            ENDIF

*           AND THE VALUE OF THE BISHOP

            DO 36 I = 1, J

              INSET( VALUES(J-I,K-I) ) = .TRUE.

36          CONTINUE

*           THE VALUE OF THE POSITION IS THE MEX OF INSET

            MEX = 0

30          IF ( INSET( MEX ) ) THEN

               MEX = MEX + 1

               GOTO 30

            ENDIF

            VALUES( J, K ) = MEX

            VALUES( K, J ) = MEX

10        CONTINUE

100     CONTINUE
```

```
      WRITE ( 6, 1000 ) ( I, I = 0, 30 )
1000  FORMAT( 1X, T8, 31I4 / )

      DO 50 R = 0, 30

         WRITE ( 6, 2000 ) R, ( VALUES( R, C ), C = 0, 30 )
2000     FORMAT( 1X, T3, I4, T8, 31I4 )

50    CONTINUE


      END
```

A TABLE OF GRUNDY VALUES FOR THE TWO-HEAP COMMANDER GAME

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	1	2	3	4	5	6	7	8	9	10	11	12
1	1	2	3	4	0	6	7	8	9	5	11	12	13
2	2	3	4	0	1	7	8	9	5	6	12	13	14
3	3	4	0	1	2	8	9	5	6	7	13	14	10
4	4	0	1	2	3	9	5	6	7	8	14	10	11
5	5	6	7	8	9	10	4	0	1	2	15	16	3
6	6	7	8	9	5	4	11	12	3	10	16	0	1
7	7	8	9	5	6	0	12	13	10	16	4	1	2
8	8	9	5	6	7	1	3	10	14	11	17	2	15
9	9	5	6	7	8	2	10	16	11	12	18	15	19
10	10	11	12	13	14	15	16	4	17	18	5	6	7
11	11	12	13	14	10	16	0	1	2	15	6	7	8
12	12	13	14	10	11	3	1	2	15	19	7	8	9
13	13	14	10	11	12	17	2	15	16	0	8	9	5
14	14	10	11	12	13	19	15	18	0	1	9	5	6
15	15	16	17	18	19	20	14	3	4	13	21	22	23
16	16	17	18	19	15	14	21	22	13	20	2	3	25
17	17	18	19	15	16	21	22	23	20	24	0	4	26
18	18	19	15	16	17	11	13	20	24	3	1	26	21
19	19	15	16	17	18	12	20	26	21	4	23	27	22
20	20	21	22	23	24	25	19	14	28	17	27	18	0
21	21	22	23	24	20	26	27	11	12	25	30	19	4
22	22	23	24	20	21	13	28	29	25	26	3	31	16
23	23	24	20	21	22	28	18	25	26	32	33	17	27
24	24	20	21	22	23	18	17	31	19	14	28	25	29
25	25	26	27	28	29	30	24	19	22	21	20	34	18
26	26	27	28	29	25	24	32	33	23	30	31	36	37
27	27	28	29	25	26	31	33	34	30	35	24	32	38
28	28	29	25	26	27	22	23	17	18	37	32	21	20
29	29	25	26	27	28	23	30	36	31	33	34	20	35
30	30	31	32	33	34	35	29	21	27	28	19	41	36

The following are new variants of Wythoff's game.

The first variant offered, Linear Wythoff, is one in which the heaps are represented by an ordered tuple, and a Type-two move can only be made if the two heaps are in adjacent positions. For the two-heap game, there is no difference between Linear Wythoff and the standard game. For three heaps, say, (A, B, C), a Type-two move can only reduce heaps A and B or heaps B and C.

A program for finding the Grundy values for the three heap version of this game with A = 1 follows.

```
      INTEGER LMT
      PARAMETER ( LMT = 20 )

      INTEGER VALUES(0:LMT,0:LMT,0:LMT)

      INTEGER X, Y, Z, N
      INTEGER I, J, K

      INTEGER  G
      EXTERNAL G
      VALUES(0,0,0) = 0
      DO 5 Y = 0, LMT
      DO 7 N = 1, LMT
         IF ( N.LE.LMT ) THEN
            DO 15 X = 0, N/2
               Z = N - X
               VALUES(X,Y,Z) = G(LMT,VALUES,X,Y,Z)
```

```
                VALUES(Z,Y,X) = VALUES(X,Y,Z)
15            CONTINUE
           ELSE
              DO 25 X = LMT, N/2, -1
                 Z = N - X
                 VALUES(X,Y,Z) = G(LMT,VALUES,X,Y,Z)
                 VALUES(Z,Y,X) = VALUES(X,Y,Z)
25            CONTINUE
           ENDIF
7       CONTINUE
5       CONTINUE
        DO 100 Y = 0, 5
           WRITE (6,1000) Y
1000       FORMAT(1X, T6, 'Y = ', I3 // )
           DO 200 X = 0, 20
              WRITE(6,2000) ( VALUES(X,Y,Z), Z = 0,20 )
2000          FORMAT (1X, T6, 21I6 / )
200        CONTINUE
           PRINT *
           PRINT *
           PRINT *
100        CONTINUE
        END


        INTEGER FUNCTION G(LMT,VALUES,X,Y,Z)
        INTEGER LMT
        INTEGER VALUES(0:LMT,0:LMT,0:LMT)
```

```
      INTEGER X,Y,Z
      INTEGER LL
      PARAMETER ( LL = 50 )
      LOGICAL GRUNDY(0:LL)
      INTEGER I,J
      INTEGER TEMP
      DO 5 I = 0, LL
         GRUNDY(I) = .FALSE.
5     CONTINUE
      IF ( X.GT.0 ) THEN
         DO 10 I = 0, X-1
            GRUNDY(VALUES(I,Y,Z)) = .TRUE.
10       CONTINUE
      ENDIF
      IF ( Z.GT.0 ) THEN
         DO 15 I = 0, Z-1
            GRUNDY(VALUES(X,Y,I)) = .TRUE.
15       CONTINUE
      ENDIF
      IF ( Y.GT.0 ) THEN
         DO 20 I = 0, Y-1
            GRUNDY(VALUES(X,I,Z)) = .TRUE.
20       CONTINUE
         IF ( X.GT.0 ) THEN
            DO 25 J = 1, MIN(X,Y)
               GRUNDY(VALUES(X-J,Y-J,Z)) = .TRUE.
```

```
25          CONTINUE
         ENDIF
         IF ( Z.GT.0 ) THEN
            DO 30 J = 1, MIN(Y,Z)
               GRUNDY(VALUES(X,Y-J,Z-J)) = .TRUE.
30          CONTINUE
         ENDIF
      ENDIF

      TEMP = 0
35    IF ( GRUNDY(TEMP)) THEN
         TEMP = TEMP + 1
         GOTO 35
      ENDIF
      G = TEMP

      END
```

TABLES OF GRUNDY VALUES FOR THE TRIPLES (X, Y, Z). X = ROW, Z = COLUMN, AND THE TABLE ENTRY IS THE GRUNDY NUMBER OF THE POSITION FOR Y = n ϵ { 0, 1, 2, 3, 4, 5 }.

Y = 0

0	1	2	3	4	5	6	7	8	9
1	0	3	2	5	4	7	6	9	8
2	3	0	1	6	7	4	5	10	11
3	2	1	0	7	6	5	4	11	10
4	5	6	7	0	1	2	3	12	13

5	4	7	6	1	0	3	2	13	12
6	7	4	5	2	3	0	1	14	15
7	6	5	4	3	2	1	0	15	14
8	9	10	11	12	13	14	15	0	1
9	8	11	10	13	12	15	14	1	0
10	11	8	9	14	15	12	13	2	3

Y = 1

0	2	3	1	5	6	4	8	9	7
2	3	1	4	6	7	5	9	10	11
3	1	2	5	4	8	6	7	11	9
1	4	5	2	3	9	7	6	8	12
5	6	4	3	1	2	8	10	7	14
6	7	8	9	2	3	1	4	5	10
4	5	6	7	8	1	2	3	12	13
8	9	7	6	10	4	3	2	1	5
9	10	11	8	7	5	12	1	2	3
7	11	9	12	14	10	13	5	3	2
11	12	10	13	15	16	9	17	4	1

Y = 2

0	2	1	4	3	7	5	6	10	8
2	1	0	3	7	5	6	10	4	9
1	0	3	6	2	9	7	4	5	10
4	3	6	1	8	2	10	5	7	11
3	7	2	8	4	6	1	9	11	5
7	5	9	2	6	1	4	3	8	13

5	6	7	10	1	4	3	8	2	14
6	10	4	5	9	3	8	1	14	2
10	4	5	7	11	8	2	14	3	6
8	9	10	11	5	13	14	2	6	1
9	8	11	14	10	12	15	7	1	16

Y = 3

0	2	1	5	6	4	8	3	7	11
2	1	0	6	4	8	3	11	12	5
1	0	4	7	8	3	2	9	13	6
5	6	7	8	9	10	1	2	3	4
6	4	8	9	2	5	11	12	10	1
4	8	3	10	5	2	9	7	6	14
8	3	2	1	11	9	6	5	4	7
3	11	9	2	12	7	5	6	8	15
7	12	13	3	10	6	4	8	1	9
11	5	6	4	1	14	7	15	9	3
12	7	14	15	13	1	10	4	5	2
10	14	5	13	3	15	12	1	2	8

Y = 4

0	2	1	5	7	8	9	4	11	3
2	1	0	6	8	9	10	7	3	13
1	0	3	2	5	4	8	10	14	12
5	6	2	4	0	7	3	11	1	8
7	8	5	0	3	11	12	13	4	2

8	9	4	7	11	6	5	14	2	15
9	10	8	3	12	5	11	2	7	1
4	7	10	11	13	14	2	8	5	6
11	3	14	1	4	2	7	5	6	16
3	13	12	8	2	15	1	6	16	4
13	14	15	10	6	17	4	3	8	11

Y = 5

0	2	1	5	7	3	10	11	6	12
2	1	0	6	3	10	8	5	13	4
1	0	3	2	5	6	9	12	8	15
5	6	2	3	0	1	4	7	9	13
7	3	5	0	6	8	13	1	2	10
3	10	6	1	8	4	11	15	7	5
10	8	9	4	13	11	1	16	3	6
11	5	12	7	1	15	16	3	17	9
6	13	8	9	2	7	3	17	10	11
12	4	15	13	10	5	6	9	11	7
8	9	16	11	4	2	5	14	18	17

THIS IS A TABLE OF THE GRUNDY VALUES ASSOCIATED WITH THE PAIR (X,Y) AND COLUMN = Z; THE PAIRS (X,Y) WERE CHOSEN SINCE THESE PAIRS TAKEN ALONE, I.E., (X,Y,0), HAVE A GRUNDY VALUE OF ZERO

PAIR		0	1	2	3	4	5	6	7	8
0	0	0	1	2	3	4	5	6	7	8
1	2	2	1	0	3	7	5	6	10	4
3	5	5	6	2	3	0	1	4	7	9

4	7	7	3	5	0	9	12	14	11	1
6	10	11	4	10	8	9	0	5	6	1
8	13	4	7	9	10	13	12	1	0	8
9	15	13	16	4	7	8	2	12	3	14
11	18	17	15	16	12	18	13	2	4	14
12	20	20	17	7	19	11	8	14	15	16

The next game, Designer Wythoff, is based on Synonim, that is, if there are two heaps containing the same number of tokens then if one of those heaps is reduced as part of a Wythoff's move then all the heaps having the same number of tokens are reduced. A problem arises if A > 1 in that this would allow the removal of an unequal number of tokens from each of two heaps in a Type-two move. For example, let A = 2, that is, the number of tokens removed from each of the two heaps in a Type-two move can differ by at most one. Given a three-heap game with the position (5, 5, 8), is the reduction (1, 2, 0) allowed, and is the resulting position (3, 4, 8) before reordering? Again, for a three-heap game with A = 2 and position (5, 5, 5), what would be the effect of the move (0, 1, 2)?

The following program finds the safe positions for the four-heap Designer Wythoff's game with A = 1. The program produces a list of safe 4-tuples by the following algorithm.

Step 1. Set N = 0

Step 2. Partition N into four parts: $0 \le H \le I \le J \le K \le N$

Step 3. If the 4-tuple (H, I, J, K) is not reducible to a 4-tuple on the list then it is safe. Put it on the list.

Step 4. Set N = N + 1 and go to Step 2.

The list is initially empty, so (0, 0, 0, 0) is a safe 4-tuple.

A PROGRAM FOR FINDING THE SAFE 4-TUPLES IN THE FOUR HEAP DESIGNER WYTHOFF'S GAME WITH A = 1

```
      INTEGER HEAP(2500,4), PTR2, PTR3
      CHARACTER LINE31(800), LINE32(800), LINE21(800),
     +          LINE22(800)
      INTEGER GAP (2500), Z1(800,2), FST, SND
      INTEGER TUPLE (4)
      INTEGER COUNT, SUM
      INTEGER A, N, P, M, R, S
      INTEGER H, I, J, K
      LOGICAL OK
      LOGICAL CHECK
      EXTERNAL CHECK

      M = 4
      A = 1
      P = 1
      PTR2 = 0
      PTR3 = 0
      DO 999 R = 1, 800
```

```
          LINE21(R) = ' '

          LINE22(R) = ' '

          LINE31(R) = ' '

          LINE32(R) = ' '

999    CONTINUE

* ( 0, 0, 0, 0 ) IS A SAFE 4-TUPLE

       HEAP(P,1) = 0
       HEAP(P,2) = 0
       HEAP(P,3) = 0
       HEAP(P,4) = 0

       WRITE(6,*) 'GOOD', 1, (HEAP(P,R),R=1,M)

       GAP(P) = 0

       COUNT  = 0


       DO 10 N = 1, 15

* PARTITION N INTO FOUR PARTS, 0 ≤ H ≤ I ≤ J ≤ K ≤ N

          DO 15 K = N, (N+3)/4, -1

          DO 20 J = MIN(N-K,K), (N-K+2)/3, -1

          DO 25 I = MIN(N-(J+K),J), (N-K-J+1)/2, -1

             H = N - ( K + J + I )

             TUPLE(1) = H
             TUPLE(2) = I
             TUPLE(3) = J
             TUPLE(4) = K

             OK = CHECK (P, M, HEAP, TUPLE, A)


             IF (OK) THEN

                P = P + 1

                HEAP(P,1) = TUPLE(1)
                HEAP(P,2) = TUPLE(2)
                HEAP(P,3) = TUPLE(3)
                HEAP(P,4) = TUPLE(4)
```

```
                GAP(P) = COUNT
                COUNT =  0
              ELSE
                COUNT = COUNT + 1
              ENDIF
25          CONTINUE
20          CONTINUE
15          CONTINUE

10      CONTINUE

        END

        LOGICAL FUNCTION CHECK( P, M, HEAP, TUPLE, A )
        INTEGER P, M
        INTEGER HEAP(2500,4), TUPLE(4)
        INTEGER A
        INTEGER OLD(4), NEW(4)
        INTEGER OLDPTR, NEWPTR
        INTEGER I, J, K, W, X, Y, Z, D1, D2
        INTEGER T, R, S
        INTRINSIC MIN

        CHECK = .FALSE.
        DO 5 K = 1, P

        OLDPTR = 0
        NEWPTR = 0
```

```
      I = 4
      J = 4

25    CONTINUE

        IF ( HEAP(K,I) .GT. TUPLE(J) ) THEN
          OLD(OLDPTR) = HEAP(K,I)
          I = I - 1
        ELSEIF( TUPLE(J) .GT. HEAP(K,I) ) THEN
          NEWPTR = NEWPTR + 1
          NEW(NEWPTR) = TUPLE(J)
          J = J - 1
        ELSE
          I = I - 1
          J = J - 1
        ENDIF

      IF ( I .GE. 1 .AND. J .GE. 1 ) GO TO 25

********** CHECK FOR EQUALITY **************
      IF ( OLDPTR .EQ. 0 .AND. NEWPTR .EQ. 0 ) THEN
         WRITE(6,1500) K, (TUPLE(R),R=1,4)
1500     FORMAT( 1X, 'K = ', I3, 4X, 4I4/)
         RETURN
      ENDIF
*********************************************

      IF ( I.GE.1 ) THEN
         DO 125 R = I, 1, -1
```

```
            OLDPTR = OLDPTR + 1
            OLD(OLDPTR) = HEAP(K,R)
125         CONTINUE
         ELSEIF ( J .GE. 1 ) THEN
            DO 35 R = J, 1, -1
               NEWPTR = NEWPTR + 1
               NEW(NEWPTR) = TUPLE(R)
35          CONTINUE
         ENDIF
         DO 45 R = 1, OLDPTR
            IF ( OLD(R).GT.NEW(R) ) THEN
               GOTO 5
            ENDIF
45       CONTINUE
         IF ( OLDPTR .LE. 1 ) THEN
            RETURN
         ENDIF
         IF ( OLDPTR .EQ. 3 ) THEN
            GOTO 5
         ENDIF
         IF ( OLDPTR .EQ. 2 ) THEN
            X = NEW(1) - OLD(1)
            Y = NEW(2) - OLD(2)
            Z = ABS( Y - X )
            IF (  Z.LT.A ) THEN
               RETURN
            ENDIF
```

```
      ENDIF

5     CONTINUE

      CHECK = .TRUE.
      WRITE(6,*) 'GOOD', P+1, (TUPLE(S), S=1,4)


      END
```

A LIST OF SAFE 4-TUPLES FOR DESIGNER WYTHOFF WITH A = 1

POSITION ON LIST	HEAP 1	HEAP 2	HEAP 3	HEAP 4
1	0	0	0	0
2	0	0	1	2
3	0	1	1	1
4	0	2	2	2
5	1	1	2	2
6	0	0	3	5
7	0	1	3	4
8	0	3	3	3
9	0	0	4	7
10	0	2	3	6
11	1	1	3	6
12	1	2	3	5
13	2	2	3	4
14	0	4	4	4
15	0	1	5	8
16	0	2	4	8
17	1	1	4	8
18	0	2	5	7
19	1	1	5	7
20	1	2	4	7
21	1	4	4	6
22	0	5	5	5

Welter's game, finds its Wythoff's equivalent in WW2 and WW3, respectively, Welter's Wythoff for two heaps and Welter's Wythoff for three heaps. Recall that in this game, no two heaps can have the same number of tokens.

A PROGRAM FOR FINDING THE GRUNDY VALUES OF TWO-HEAP WW2

```
      INTEGER LMT
      PARAMETER ( LMT = 10 )

      INTEGER G( 0:LMT, 0:LMT )
      LOGICAL INSET( -1:100 )

      INTEGER X, Y
      INTEGER I, J, K
      INTEGER MEX
      CHARACTER*63 LINE

      DO 5 I = 0, LMT
         G(I,I) = -1
5     CONTINUE
      DO 7 I = 1, LMT
          G(I,0) = I-1
          G(0,I) = I-1
7     CONTINUE

      DO 10 X = 1, LMT
      DO 20 Y = X+1, LMT

         DO 30 I = 0, 100
            INSET(I) = .FALSE.
30       CONTINUE
         DO 40 I = 0, X-1
            INSET(G(I,Y)) = .TRUE.
40       CONTINUE
```

```
            DO 50 I = 0, Y-1
               INSET(G(X,I)) = .TRUE.
            CONTINUE
            DO 55 I = 1, X
               INSET( G(X-I,Y-I) ) = .TRUE.
55          CONTINUE
            MEX = 0
60          IF ( INSET(MEX) ) THEN
               MEX = MEX + 1
               GOTO 60
            ENDIF
            G(X,Y) = MEX
            G(Y,X) = MEX

20       CONTINUE
10       CONTINUE

         DO 70 I = 0, LMT
            LINE = ' '
         DO 80 J = 0, LMT
            K = 3*J+2
            IF ( G(I,J).GE.10 ) THEN
               LINE(K:K) = CHAR( ICHAR('0') + G(I,J)/10 )
               LINE(K+1:K+1) = CHAR( ICHAR('0') +
     +                               MOD(G(I,J),10) )
            ELSEIF ( G(I,J).GE.0 ) THEN
               LINE(K+1:K+1) = CHAR( ICHAR('0') + G(I,J) )
            ELSE
```

```
            LINE(K+1:K+1) = '*'
         ENDIF
80     CONTINUE

       PRINT *, LINE
       PRINT *

70     CONTINUE

       END
```

OUTPUT:
A TABLE OF GRUNDY VALUES FOR WW2

*	0	1	2	3	4	5	6	7	8	9
0	*	2	3	1	5	6	4	8	9	7
1	2	*	4	0	3	7	5	6	10	8
2	3	4	*	5	6	0	1	9	7	11
3	1	0	5	*	7	2	8	4	11	10
4	5	3	6	7	*	1	9	10	0	2
5	6	7	0	2	1	*	3	11	4	12
6	4	5	1	8	9	3	*	12	13	14
7	8	6	9	4	10	11	12	*	14	5
8	9	10	7	11	0	4	13	14	*	6
9	7	8	11	10	2	12	14	5	6	*

A PROGRAM FOR FINDING THE GRUNDY VALUES FOR WW3

```
      INTEGER LMT
      PARAMETER ( LMT = 50 )

      INTEGER TLMT
```

```
      PARAMETER ( TLMT = 14 )

      INTEGER G( 0:LMT, 0:LMT, 0:LMT )
      LOGICAL INSET( -1:200 )

      INTEGER X, Y, Z
      INTEGER I, J, K
      INTEGER MEX
      INTEGER TABLE(0:TLMT,0:TLMT)

      DO 3 I = 0, LMT
         G(I,I,I) = -1
3     CONTINUE
      DO 5 I = 0, LMT
      DO 7 J = I+1, LMT
         G(I,I,J) = -1
         G(I,J,I) = -1
         G(J,I,I) = -1

         G(J,J,I) = -1
         G(J,I,J) = -1
         G(I,J,J) = -1
7     CONTINUE
5     CONTINUE

      DO 10 X = 1, LMT
      DO 20 Y = X+1, LMT

         DO 30 I = 0, 200
            INSET(I) = .FALSE.
30       CONTINUE
         DO 40 I = 0, X-1
```

```
            INSET(G(I,Y,0)) = .TRUE.
40          CONTINUE
            DO 50 I = 0, Y-1
               INSET(G(X,I,0)) = .TRUE.
50          CONTINUE
            DO 57 I = 1, X-1
               INSET( G(X-I,Y-I,0) ) = .TRUE.
57          CONTINUE
            MEX = 0
60          IF ( INSET(MEX) ) THEN
               MEX = MEX + 1
               GOTO 60
            ENDIF
            G(O,X,Y) = MEX
            G(0,Y,X) = MEX
            G(X,0,Y) = MEX
            G(Y,0,X) = MEX
            G(X,Y,0) = MEX
            G(Y,X,0) = MEX

20       CONTINUE
10       CONTINUE

         DO 110 X = 1, LMT
         DO 120 Y = X+1, LMT
         DO 130 Z = Y+1, LMT
            DO 140 I = 0, 200
```

```
            INSET(I) = .FALSE.
140         CONTINUE
            DO 150 I = 1, X
               INSET(G(X-I,Y,Z)) = .TRUE.
150         CONTINUE
            DO 160 I = 1, Y
               INSET(G(X,Y-I,Z)) = .TRUE.
160         CONTINUE
            DO 170 I = 1, Z
               INSET(G(X,Y,Z-I)) = .TRUE.
170         CONTINUE
            DO 175 I = 1, X-1
               INSET( G(X-I,Y-I,Z) ) = .TRUE.
               INSET( G(X-I,Y,Z-I) ) = .TRUE.
175         CONTINUE
            DO 177 I = 1, Y-1
               INSET( G(X,Y-I,Z-I) ) = .TRUE.
177         CONTINUE
            MEX = 0
180         IF ( INSET(MEX) ) THEN
               MEX = MEX + 1
               GO TO 180
            ENDIF
            G(X,Y,Z) = MEX
            G(X,Z,Y) = MEX
            G(Y,X,Z) = MEX
```

```
            G(Y,Z,X) = MEX
            G(Z,X,Y) = MEX
            G(Z,Y,X) = MEX
130      CONTINUE
120      CONTINUE
110      CONTINUE

         DO 200 I = 0, 6
            WRITE  ( 6, 3500 ) I
3500        FORMAT ( 1X, T5, 'GRUNDY VALUE = ', I1 // )
            DO 205 J = 0, TLMT
            DO 206 K = 0, TLMT
               TABLE(J,K) = -99
206         CONTINUE
205         CONTINUE
            DO 210 X = 0, TLMT
            DO 220 Y = 0, TLMT
            DO 230 Z = 0, LMT
               IF ( G(X,Y,Z).EQ.I ) THEN
                  TABLE(X,Y) = Z
               ENDIF
230         CONTINUE
220         CONTINUE
210         CONTINUE
            DO 290 J = 0, TLMT
               WRITE  ( 6,2000 ) ( TABLE(J,K), K = 0, TLMT )
2000           FORMAT ( 1X, T5, 15I4 / )
```

```
290       CONTINUE
          PRINT *
          PRINT *
          PRINT *
          PRINT *
200       CONTINUE

      END
```

THE FOLLOWING TABLES ARE OF THE FORM (X, Y, Z), X = ROW, Y = COLUMN, AND Z IS THE TABLE ENTRY. * INDICATES THAT THE POSITION IS ALLOWED BY THE RULES.

GRUNDY VALUE = 0

*	2	1	5	7	3	10	4	13	15	6	18	20	8
2	*	0	4	3	7	9	5	12	6	15	17	8	20
1	0	*	6	5	4	3	11	10	14	8	7	18	22
5	4	6	*	1	0	2	8	7	11	16	9	17	21
7	3	5	1	*	2	11	0	9	8	14	6	19	25
3	7	4	0	2	*	14	1	11	16	20	8	13	12
10	9	3	2	11	14	*	12	18	1	0	4	7	15
4	5	11	8	0	1	12	*	3	10	9	2	6	23
13	12	10	7	9	11	18	3	*	4	2	5	1	0
15	6	14	11	8	16	1	10	4	*	7	3	21	19
6	15	8	16	14	20	0	9	2	7	*	19	23	18
18	17	7	9	6	8	4	2	5	3	19	*	16	14
20	8	18	17	19	13	7	6	1	21	23	16	*	5
8	20	22	21	25	12	15	23	0	19	18	14	5	*
23	22	9	25	10	6	5	17	16	2	4	13	24	11

GRUNDY VALUE = 1

*	3	5	1	8	2	7	6	4	14	16	19	21	20
3	*	4	0	2	6	5	10	13	15	7	18	16	8
5	4	*	7	1	0	8	3	6	10	9	16	19	21
1	0	7	*	5	4	10	2	11	16	6	8	14	23
8	2	1	5	*	3	9	11	0	6	15	7	18	22
2	6	0	4	3	*	1	13	12	11	17	9	8	7
7	5	8	10	9	1	*	0	2	4	3	12	11	16
6	10	3	2	11	13	0	*	18	12	1	4	9	5
4	13	6	11	0	12	2	18	*	19	23	3	5	1
14	15	10	16	6	11	4	12	19	*	2	5	7	17
16	7	9	6	15	17	3	1	23	2	*	20	22	14
19	18	16	8	7	9	12	4	3	5	20	*	6	24
21	16	19	14	18	8	11	9	5	7	22	6	*	26
20	8	21	23	22	7	16	5	1	17	14	24	26	*
9	23	20	12	21	24	22	19	28	0	13	15	3	10

GRUNDY VALUE = 2

*	4	3	2	1	7	11	5	12	16	18	6	8	19
4	*	5	7	0	2	8	3	6	10	9	16	18	21
3	5	*	0	6	1	4	10	9	8	7	18	17	23
2	7	0	*	8	10	9	1	4	6	5	12	11	15
1	0	6	8	*	12	2	13	3	11	19	9	5	7
7	2	1	10	12	*	15	0	14	13	3	22	4	9
11	8	4	9	2	15	*	16	1	3	22	0	13	12

5	3	10	1	13	0	16	*	11	20	2	8	19	4
12	6	9	4	3	14	1	11	*	2	21	7	0	16
16	10	8	6	11	13	3	20	2	*	1	4	22	5
18	9	7	5	19	3	22	2	21	1	*	14	16	20
6	16	18	12	9	22	0	8	7	4	14	*	3	25
8	18	17	11	5	4	13	19	0	22	16	3	*	6
19	21	23	15	7	9	12	4	16	5	20	25	6	*
24	25	22	21	26	8	17	15	5	18	11	10	28	27

GRUNDY VALUE = 3

*	5	4	6	2	1	3	8	7	17	15	20	18	23
5	*	3	2	6	0	4	11	14	12	17	7	9	18
4	3	*	1	0	8	7	6	5	15	14	19	21	20
6	2	1	*	7	11	0	4	10	13	8	5	20	9
2	6	0	7	*	13	1	3	15	18	12	17	10	5
1	0	8	11	13	*	9	12	2	6	22	3	7	4
3	4	7	0	1	9	*	2	16	5	19	13	22	11
8	11	6	4	3	12	2	*	0	14	13	1	5	10
7	14	5	10	15	2	16	0	*	11	3	9	13	12
17	12	15	13	18	6	5	14	11	*	16	8	1	3
15	17	14	8	12	22	19	13	3	16	*	21	4	7
20	7	19	5	17	3	13	1	9	8	21	*	15	6
18	9	21	20	10	7	22	5	13	1	4	15	*	8
23	18	20	9	5	4	11	10	12	3	7	6	8	*
21	8	10	19	24	15	20	9	1	7	2	22	17	25

GRUNDY VALUE = 4

*	6	8	4	3	9	1	10	2	5	7	13	19	11
6	*	7	5	8	3	0	2	4	16	11	10	15	19
8	7	*	10	12	6	5	1	0	11	3	9	4	16
4	5	10	*	0	1	7	6	13	15	2	14	21	8
3	8	12	0	*	11	10	9	1	7	6	5	2	14
9	3	6	1	11	*	2	15	17	0	16	4	22	18
1	0	5	7	10	2	*	3	19	14	4	18	25	21
10	2	1	6	9	15	3	*	12	4	0	17	8	24
2	4	0	13	1	17	19	12	*	18	14	22	7	3
5	16	11	15	7	0	14	4	18	*	17	2	13	12
7	11	3	2	6	16	4	0	14	17	*	1	26	29
13	10	9	14	5	4	18	17	22	2	1	*	20	0
19	15	4	21	2	22	25	8	7	13	26	20	*	9
11	19	16	8	14	18	21	24	3	12	29	0	9	*
22	24	18	11	13	25	9	26	10	6	8	3	27	4

GRUNDY VALUE = 5

*	7	6	8	5	4	2	1	3	11	13	9	22	10
7	*	8	6	9	12	3	0	2	4	14	13	5	11
6	8	*	4	3	7	0	5	1	12	16	15	9	18
8	6	4	*	2	9	1	10	0	5	7	17	13	12
5	9	3	2	*	0	8	12	6	1	20	14	7	17
4	12	7	9	0	*	10	2	15	3	6	21	1	22
2	3	0	1	8	10	*	15	4	18	5	23	19	24

1	0	5	10	12	2	15	*	13	21	3	22	4	8
3	2	1	0	6	15	4	13	*	14	17	19	16	7
11	4	12	5	1	3	18	21	14	*	19	0	2	20
13	14	16	7	20	6	5	3	17	19	*	18	15	0
9	13	15	17	14	21	23	22	19	0	18	*	30	1
22	5	9	13	7	1	19	4	16	2	15	30	*	3
10	11	18	12	17	22	24	8	7	20	0	1	3	*
25	10	23	22	11	16	28	20	9	8	1	4	26	21

GRUNDY VALUE = 6

*	8	7	9	6	13	4	2	1	3	11	10	15	5
8	*	6	10	5	4	2	9	0	7	3	14	17	22
7	6	*	5	8	3	1	0	4	16	13	12	11	10
9	10	5	*	11	2	8	12	6	0	1	4	7	17
6	5	8	11	*	1	0	10	2	14	7	3	21	15
13	4	3	2	1	*	12	16	18	17	15	23	6	0
4	2	1	8	0	12	*	13	3	11	14	9	5	7
2	9	0	12	10	16	13	*	21	1	4	15	3	6
1	0	4	6	2	18	3	21	*	10	9	17	19	23
3	7	16	0	14	17	11	1	10	*	8	6	23	21
11	3	13	1	7	15	14	4	9	8	*	0	18	2
10	14	12	4	3	23	9	15	17	6	0	*	2	20
15	17	11	7	21	6	5	3	19	23	18	2	*	16
5	22	10	17	15	0	7	6	23	21	2	20	16	*
18	11	24	15	9	21	10	28	22	4	6	1	29	19

A program for playing Wythoff's game with blocked positions is the final Wythoff variation to be presented. In the two-heap version, a queen is placed on a two-dimensional chessboard and moves towards the origin as in the standard two-heap Wythoff's game. What is different is that one or more squares have been removed from the chessboard. The queen may not land on or pass through one of these holes, hence, the queen's progress along a row, column, or diagonal may be blocked by a removed square. A program which analyzes the two-heap game is given below and will be followed by a program for the three-heap game.

```
* TWO-DIMENSIONAL WYTHOFF'S GAME WITH BLOCKED SQUARES
* A MOVE CANNOT START FROM OR PASS THROUGH A BLOCKED SQUARE

      INTEGER LMT
      PARAMETER ( LMT = 50 )

      INTEGER BLOCK
      PARAMETER ( BLOCK = -99 )

      INTEGER SLMT
      PARAMETER ( SLMT = 100 )

      CHARACTER MORE

      INTEGER X, Y

      INTEGER N, VAL

      INTEGER I, J, K

      INTEGER G( -1:LMT, -1:LMT )

      LOGICAL INSET ( -1:SLMT )

* MARK THE BORDER AS BLOCKED
```

```
      G( -1, -1 ) = BLOCK
      DO 5 I = 0, LMT
         G( -1, I ) = BLOCK
         G( I, -1 ) = BLOCK
5     CONTINUE
* AND GIVE ALL THE OTHER SQUARES A NON-OPERATIVE DUMMY VALUE
      DO 10 I = 0, LMT
      DO 12 J = 0, LMT
         G( I, J ) = -1
12    CONTINUE
10    CONTINUE

* ENTER ONE OR MORE BLOCKED SQUARES ON THE PLAYING FIELD
15    CONTINUE

         WRITE ( 5, * ) 'ENTER A BLOCKED SQUARE'
25       CONTINUE
            WRITE ( 5, * ) 'ENTER X BETWEEN 0 AND ', LMT
            READ *, X
         IF ( X.LT.0 .OR. X.GT.LMT ) GO TO 25
35       CONTINUE
            WRITE ( 5, * ) 'ENTER Y BETWEEN 0 AND ', LMT
            READ *, Y
         IF ( Y.LT.0 .OR. Y.GT.LMT ) GO TO 35
         G( X, Y ) = BLOCK
         WRITE ( 5, * ) 'DO YOU WANT TO ENTER ANOTHER
     +                   POINT?'
         WRITE ( 5, * ) 'ENTER Y OR N'
```

```
      READ  ( 5, '(A1)' ) MORE

      IF ( MORE.EQ.'Y' ) GO TO 15

      DO 100 N = 0, 2 * LMT
      DO 110 J = 0, MIN ( N, LMT )

         K = MIN ( N - J, LMT )
         DO 120 I = -1, SLMT
            INSET(I) = .FALSE.
120      CONTINUE

         IF ( G(J,K).NE.BLOCK ) THEN
            I = 1
            VAL = G( J-I, K-I )
130         IF ( VAL.NE.BLOCK ) THEN
               INSET( VAL ) = .TRUE.
               I = I + 1
               VAL = G( J-I, K-I )
               GO TO 130
            ENDIF
            I = 1
            VAL = G( J-I, K )
140         IF ( VAL.NE.BLOCK ) THEN
               INSET( VAL ) = .TRUE.
               I = I + 1
               VAL = G( J-I, K )
               GO TO 140
```

```
            ENDIF
            I = 1
            VAL = G( J, K-I )
150         IF ( VAL.NE.BLOCK ) THEN
               INSET( VAL ) = .TRUE.
               I = I + 1
               VAL = G( J, K-I )
               GO TO 150
            ENDIF
            MEX = 0
160         IF ( INSET(MEX) ) THEN
               MEX = MEX + 1
               GO TO 160
            ENDIF
            G( J, K ) = MEX
         ENDIF

110   CONTINUE
100   CONTINUE

* PRINT THE RESULTS
      DO 200 I = 0, 20
         WRITE  ( 6, 1000 ) ( G(I,J), J = 0, 20 )
1000     FORMAT ( 1X, T5, 21I3 / )
200   CONTINUE

      END
```

A TABLE OF GRUNDY VALUES FOR THE TWO-DIMENSIONAL BLOCKED WYTHOFF GAME WITH BLOCKED SQUARES NOTED BY *

*	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	*	2	0	1	5	3	4	8	6	7	11	9	10	14	12	13	17
1	2	*	1	2	0	4	5	3	7	8	6	10	11	9	13	14	12
2	0	1	*	4	3	6	0	1	2	5	9	12	7	8	15	10	11
3	1	2	4	0	6	7	8	9	10	11	12	5	13	15	16	17	18
4	5	0	3	6	1	8	9	10	11	2	7	13	14	16	17	12	19
5	3	4	6	7	8	2	10	11	9	1	0	14	15	12	18	16	13
6	4	5	0	8	9	10	3	12	13	14	2	7	1	11	19	20	21
7	8	3	1	9	10	11	12	4	5	6	13	15	16	0	2	18	14
8	6	7	2	10	11	9	13	5	12	3	4	16	17	18	20	0	1
9	7	8	5	11	2	1	14	6	3	10	15	17	4	19	21	22	20
10	11	6	9	12	7	0	2	13	4	15	5	18	8	3	22	19	23
11	9	10	12	5	13	14	7	15	16	17	18	6	19	20	23	8	22
12	10	11	7	13	14	15	1	16	17	4	8	19	9	21	5	24	6
13	14	9	8	15	16	12	11	0	18	19	3	20	21	7	24	23	5
14	12	13	15	16	17	18	19	2	20	21	22	23	5	24	8	7	10
15	13	14	10	17	12	16	20	18	0	22	19	8	24	23	7	11	9
16	17	12	11	18	19	13	21	14	1	20	23	22	6	5	10	9	15
17	15	16	18	19	20	21	22	23	24	0	1	25	26	10	11	27	28
18	16	17	13	20	15	19	23	21	14	25	24	3	27	6	26	28	29
19	20	15	14	21	22	17	16	24	23	12	26	2	18	28	9	25	7

A program for analyzing three-heap Wythoff with blocked squares appears below.

```
* THREE-DIMENSIONAL WYTHOFF WITH BLOCKED SQUARES

      INTEGER LMT
      PARAMETER ( LMT = 20 )

      INTEGER BLOCK
      PARAMETER ( BLOCK = -99 )

      INTEGER SLMT
      PARAMETER ( SLMT = 100 )

      CHARACTER MORE

      INTEGER X, Y, Z

      INTEGER N, VAL

      INTEGER I, J, K, L

      INTEGER G( -1:LMT, -1:LMT, -1:LMT )

      LOGICAL INSET( -1:SLMT )

      INTEGER MEX

      INTEGER PLMT
      PARAMETER ( PLMT = 10 )

      INTEGER TABLE( 0:LMT, 0:LMT )


      DO  5 I = -1, LMT

      DO 10 J = -1, LMT

         G( I, J, -1 ) = BLOCK

         G( -1, I, J ) = BLOCK

         G( I, -1, J ) = BLOCK

10    CONTINUE

5     CONTINUE


      DO 15 I = 0, LMT

      DO 16 J = 0, LMT

      DO 17 K = 0, LMT
```

```
          G( I, J, K ) = -1
17      CONTINUE
16      CONTINUE
15      CONTINUE

* ENTER THE BLOCKED SQUARES
20      CONTINUE

          WRITE ( 5, * ) 'ENTER A BLOCKED SQUARE'
25        CONTINUE
             WRITE ( 5, * ) 'ENTER X BETWEEN 0 AND ', LMT
             READ *, X
          IF ( X.LT.0 .OR. X.GT.LMT ) GO TO 25
35        CONTINUE
             WRITE( 5, * ) 'ENTER Y BETWEEN 0 AND ', LMT
             READ *, Y
          IF ( Y.LT.0 .OR. Y.GT.LMT ) GO TO 35
45        CONTINUE
             WRITE( 5, * ) 'ENTER Z BETWEEN 0 AND ', LMT
             READ *, Z
          IF ( Z.LT.0 .OR. Z.GT.LMT ) GO TO 45
          G(X,Y,Z) = BLOCK
          WRITE ( 5, * ) 'DO YOU WANT TO ENTER ANOTHER
     +                    POINT?'
          WRITE ( 5, * ) 'ENTER Y OR N'
          READ  ( 5, '(A1)' ) MORE

        IF ( MORE.EQ.'Y' ) GO TO 20
```

```
      DO 100 N = 0, 3 * LMT
      DO 110 J = 0, MIN ( N, LMT )
      DO 115 K = 0, MIN ( N - J, LMT )

         L = MIN ( N - J - K, LMT )
         DO 120 I = -1, SLMT
            INSET(I) = .FALSE.
120      CONTINUE

         IF ( G(J,K,L).NE.BLOCK ) THEN

            I = 1
            VAL = G( J-I, K-I, L )
130         IF ( VAL.NE.BLOCK ) THEN
               INSET( VAL ) = .TRUE.
               I = I + 1
               VAL = G( J-I, K-I, L )
               GO TO 130
            ENDIF
            I = 1
            VAL = G( J-I, K, L-I )
140         IF ( VAL.NE.BLOCK ) THEN
               INSET( VAL ) = .TRUE.
               I = I + 1
               VAL = G( J-I, K, L-I )
               GO TO 140
            ENDIF
            I = 1
```

```
            VAL = G( J, K-I, L-I )
150         IF ( VAL.NE.BLOCK ) THEN
               INSET( VAL ) = .TRUE.
               I = I + 1
               VAL = G( J, K-I, L-I )
               GO TO 150
            ENDIF
            I = 1
            VAL = G( J-I, K, L )
160         IF ( VAL.NE.BLOCK ) THEN
               INSET( VAL ) = .TRUE.
               I = I + 1
               VAL = G( J-I, K, L )
               GO TO 160
            ENDIF
            I = 1
            VAL = G( J, K-I, L )
170         IF ( VAL.NE.BLOCK ) THEN
               INSET( VAL ) = .TRUE.
               I = I + 1
               VAL = G( J, K-I, L )
               GO TO 170
            ENDIF
            I = 1
            VAL = G( J, K, L-I )
180         IF ( VAL.NE.BLOCK ) THEN
```

```
                INSET( VAL ) = .TRUE.
                I = I + 1
                VAL = G( J, K, L-I )
                GO TO 180
             ENDIF
             MEX = 0
190          IF ( INSET(MEX) ) THEN
                MEX = MEX + 1
                GO TO 190
             ENDIF
             G(J,K,L) = MEX

          ENDIF

115    CONTINUE
110    CONTINUE
100    CONTINUE

       DO 300 N = 0, 2

          DO 200 I = 0, LMT
          DO 210 J = 0, LMT
             TABLE(I,J) = -1
210       CONTINUE
200       CONTINUE
          DO 310 X = 0, LMT
          DO 320 Y = X, LMT
          DO 330 Z = Y, LMT
```

```
            IF ( G(X,Y,Z).EQ.N ) THEN

               TABLE(X,Y) = Z
               TABLE(Y,X) = Z

            ENDIF

330      CONTINUE

320      CONTINUE

310      CONTINUE

         WRITE  ( 6, 1000 )
1000     FORMAT ( 1X, T5, 'GRUNDY VALUE ', I1 // )

         DO 400 I = 0, PLMT

            WRITE  ( 6, 2000 ) ( TABLE(I,J), J = 0, PLMT )
2000        FORMAT ( 1X, T5, 15I4 / )

400      CONTINUE

300   CONTINUE


      END
```

TABLES OF GRUNDY VALUES 0, 1, AND 2 FOR THE ORDERED TRIPLES (X, Y, Z), WHERE X, Y, Z ϵ $\mathbb{N}$ AND THE POSITION IS GIVEN IN THE TABLE AS (ROW, COLUMN, ENTRY). IF THE ENTRY IS * THEN NO POSITION WAS FOUND STARTING WITH THE ORDERED PAIR (X, Y) WITH THE SPECIFIED GRUNDY NUMBER.

THE TABLES ARE FOR THREE-DIMENSIONAL BLOCKED WYTHOFF WITH THE TWO BLOCKED SQUARES (1, 3, 4) AND (2, 4, 5)

GRUNDY VALUE = 0

0	2	*	5	7	*	10	*	13	15	*
2	1	*	4	*	8	12	11	*	14	17
*	*	2	6	8	7	*	*	*	16	15
5	4	6	9	*	*	*	14	8	*	11
7	*	8	*	4	10	9	*	*	*	*
*	8	7	*	10	5	14	*	*	11	*
10	12	*	*	9	14	6	13	15	*	*
*	11	*	14	*	*	13	7	17	10	*
13	*	*	8	*	*	15	17	*	9	18
15	14	16	*	*	11	*	10	9	*	*
*	17	15	11	*	*	*	*	18	*	10

GRUNDY VALUE = 1

1	*	2	6	8	7	*	*	*	14	16
*	2	*	5	7	*	10	*	13	16	*
2	*	*	4	*	8	12	11	*	18	17
6	5	4	8	*	*	*	7	*	10	*
8	7	*	*	6	5	*	*	*	12	14
7	*	8	*	5	*	6	*	*	17	13
*	10	12	*	*	6	*	14	16	11	*
*	*	11	7	*	*	14	*	8	20	*
*	13	*	*	*	*	16	8	*	*	12
14	16	18	10	12	17	11	20	*	9	*
16	*	17	*	14	13	*	*	12	*	11

GRUNDY VALUE = 2

2	1	*	4	*	8	11	13	*	16	14
1	*	2	6	8	7	*	*	*	15	18
*	2	*	5	7	*	10	*	13	17	*
4	6	5	10	*	*	*	8	*	9	*
*	8	7	*	5	*	14	*	*	11	10
8	7	*	*	*	9	17	*	*	*	12
11	*	10	*	14	17	7	*	8	18	*
13	*	*	8	*	*	*	14	*	12	16
*	*	13	*	*	*	8	*	*	10	*
16	15	17	9	11	*	18	12	10	*	*
14	18	*	*	10	12	*	16	*	*	*

CHAPTER 6

Games on Words

In this chapter several new combinatorial games based on string rewriting are introduced. The first game, Literary Wythoff, is actually a schema for a family of games in which each heap is a heap containing a distinct letter and the game consists of using these letters to build a string. The letters are removed from the heaps subject to the restrictions imposed by the Type-one and Type-two rules of Wythoff's game.

Let W(h1,h2, ... ,hk) be a k-heap Wythoff's game and let N = |h1| + |h2| + ... + |hk|.

This represents a canonical form for Wythoff's game. The heaps in Wythoff's game have a representation as a bag of nonnegative integers and placing an ordering on the heaps does not alter the game.

Literary Wythoff, LW(h1,h2, ... ,hk), is a transformation of W(h1,h2, ... ,hk) that preserves the Type-one and Type-two moves of the underlying Wythoff's game.

Literary Wythoff is played over a k-letter alphabet. Each of the heaps in Literary Wythoff contains a distinct letter. The assignment of letters to each of the heaps in Literary Wythoff are made in lexicographic order. The size of the heaps, i.e., the number of letters in the heap,

corresponds to the number of stones in the underlying Wythoff's game.

For example, let WY(3) be a three heap Wythoff's game with a = 1 and the normal ending condition and let LW(3) be a three heap Literary Wythoff's game which corresponds to W3.

WY(3) = (3, 4, 5)

LW(3) = (AAA, BBBB, CCCCC)

The rules for Literary Wythoff are as follows:

1) There are two players, Left and Right.

2) Left moves first.

3) The two players alternate moves until one of the players makes the ending move.

4) The player making the ending move loses, his or her opponent wins.

To make a move, a player

1) Removes letters from the heaps only if the move preserves the Type-one and Type-two moves in the underlying Wythoff's game.

2) Sorts the letters lexicographicly, and,

3) Concatenates this word to the end of the Wythoff string.

Initially, the Wythoff string is empty.

The ending move is concatenating the empty string to the Wythoff string.

In the game LW(3) given

(AAA, BBBB, CCCCC)

Left removes the four B's from the first heap , corresponding to removing four stones from the first heap in WY(3). They are already sorted, so, she adds them to the Wythoff string giving

BBBB

and leaving

(AAA, λ, CCCCC)

for Right.

Right removes a C from the third heap and adds it to the Wythoff string, giving

BBBBC

and leaving

(AAA, λ, CCCC)

for Left.

Left removes two A's and two C's, sorts them and adds them to the string, giving

BBBBCAACC

and leaving

(A, λ, CC)

for Right.

Right removes a C giving

BBBBCAACCC

and leaving

(A, λ, C)

for Left.

Left removes the A and the C giving

BBBBCAACCCAC

and leaving

(λ, λ, λ)

for Right.

Right's only move is to add the empty string to the Wythoff string, ending the game. Right loses and his opponent wins.

The set of all Wythoff strings for LW(3) is the game space for LW(3). The set of all prefixes from which Left cannot force a win is the set P (for Previous player). The set of all prefixes from which Right cannot force a win in the set N (for Next player). If Left cannot force a win from the prefix λ, we say that the initial position was safe, otherwise, we say that the initial position was unsafe. A player making a move from a position in which there is at least one winning move is referred to as a P-player, a player making a move from a position in which all the moves are losing moves is referred to as an N-player.

The ending move, concatenating λ to the Wythoff string, along with the condition that the person making this move loses and his or her opponent wins corresponds to the normal ending condition. Note that the set of Wythoff strings from

a prefix in P is not unique. For example, if the initial position were

(AA, BB)

Left has a choice of three winning moves

1) (λ, λ) AABB

2) (A, BB) A

3) (AA, B) B

with 1) and 3) or 2) and 3) always leading to different Wythoff strings.

A position in Literary Wythoff is said to be ambiguous if there are moves leading to different Wythoff strings from the given position that do not reflect a difference in the underlying Wythoff's game, e.g., (AAA, BBB, CCC). A move that can lead to different Wythoff strings that do not reflect a difference is the underlying Wythoff's game are said to be an ambiguous move. The set of all ambiguous moves leading to the same underlying position are an equivalence class. An ambiguous move must conform to the following restriction: A player making a move from an equivalence class must choose that move which creates the lexographically smallest suffix for the Wythoff string.

In the example given moves 2 and 3 fall in one equivalence class and move 1 in another. The moves to (AA, λ) and (λ, BB) fall into a third. (Technically there is a forth equivalence class, λ, the empty string, and we must add the rule that the empty string can be chosen only if it

is the only move available.) If Left chooses to make a Type-one move then Left's choices are to either (A, BB) or (λ, BB).

The imposition of another restriction on the P-player, that she make a move that leads to the shortest sequence of moves for the game; and, a similar restriction on N-player, that he make a move that leads to the longest sequence of moves offers an interesting heuristic for playing Wythoff's game.

We introduce the notion of forbidden substrings. The ending condition can be stated as:

1) λ is a forbidden substring.

2) A player may not add a forbidden substring to the Wythoff string.

3) A player unable to add a suffix to the Wythoff string loses and his or her opponent wins.

A forbidden substring is a restriction on the formation of the Wythoff string that reflects a restriction on the underlying Wythoff's game.

The following are some possible restrictions:

1) A player is forbidden to add a forbidden substring

2) The Wythoff string is forbidden to contain

    a) a forbidden substring

    b) a forbidden suffix

    c) a forbidden prefix

3) No move may contain a specified

a) number of letters

e.g., no move may remove 4 letters from the heaps

b) combination of letters

e.g., no move may contain 3 A's

e.g., no move may contain a total of 5 A's and B's

The next game is Abel, in which, each player, in turn reduces a string of a's and b's (or any two symbol alphabet) until the empty string is reached.
A formal representation of the game Abel:

Abel := (C, M, B)

Where C is the control structure [set of metarules]
M is the set of moves [rules]
and B is the (initial) position

B is a string $\alpha \circ \beta$ where $\alpha \in \Sigma^*$

$\Sigma = \{ a,b \}$

$\beta \in T^*$

$T = \{ L,R \}$

and $\circ$ is concatenation.

M is a set of moves or (rewrite) rules on $\Sigma^* \circ T \times \Sigma^* \circ T$

$p(g(h(w)))$ for $w \in \Sigma^* \circ T$

h is a translation rule

$h := \{ (ab,ba), (ba,ab), (a,a), (b,b) \}$

g is a reduction rule

$g := \{ ((ab)^n, (ab)^m), ((ba)^n, (ba)^m), (a^n, a^m), (b^n, b^m) \}$

$n \in \mathbb{Z}+$, $m \in \mathbb{N}$ and $n > m$

p is a marking rule to determine the player whose turn it is

$p := \{ (L,R), (R,L) \}$

C is a set of metarules that determine the winner and loser; whether the game is played under the normal ending condition or under the misère ending condition; and, if a move is legal.

A program for analyzing the Grundy values of strings in Abel follows. The first seven values were calculated using the Mex rule.

```
INTEGER LMT
PARAMETER ( LMT = 10 )

INTEGER SLMT
PARAMETER ( SLMT = 200 )

INTEGER GLMT
PARAMETER ( GLMT = 2048 )
INTEGER GV( 0:GLMT )
LOGICAL INSET( 0:SLMT )
INTEGER SIZE
PARAMETER ( SIZE = 15 )

INTEGER ZEROS( 200 )
INTEGER P
CHARACTER*100 STR, COPY, COPY1
INTEGER H, I, J, K, L, M, N
INTEGER NUM

GV(0) = 0
GV(1) = 1
GV(2) = 2
GV(3) = 2
GV(4) = 0
GV(5) = 0
GV(6) = 0
```

```
      GV(7) = 3

      DO 10 N = 4, LMT
         I = 2**N-1
         GV(I) = N
10    CONTINUE

      DO 100 N = 3, LMT
         DO 200 NUM = 2**N, 2**(N+1)-2

            CALL CREATE( STR, NUM )
            DO 210 I = 0, SLMT
               INSET(I) = .FALSE.
210         CONTINUE
            CALL RS3( STR, COPY )

* THE IDENTITY TRANSFORMATION

            COPY1 = COPY
            CALL SETUP( COPY1, INSET, SLMT, GV, LMT )

* AND THE OTHER TRANSFORMATIONS

            L = INDEX( COPY(2: ), ' ' )

            DO 220 I = 2, L
```

```
                J = I + 1
                IF ( COPY(I:I).NE.COPY(J:J) ) THEN
                   H = I - 1
                   K = J + 1
                   COPY1 = COPY( :H)//COPY(J:J)//COPY(I:I)//
     +                    COPY(K: )
                   CALL SETUP( COPY1, INSET, SLMT, GV, LMT )
                ENDIF
220          CONTINUE

             MEX = 0
230          IF ( INSET(MEX) ) THEN
                MEX = MEX + 1
                GO TO 230
             ENDIF
             GV(NUM) = MEX

200       CONTINUE
100    CONTINUE


       DO 400 I = 0, 31
          WRITE  ( 6, 2000 ) I, GV(I), I+32, GV(I+32), I+64,
     +                       GV(I+64), I+96, GV(I+96)
2000      FORMAT ( 1X, T5, I3, T11, I3, T21, I3, T27, I3,
T37, I3,
```

```
     +            T43, I3, T53, I3, T59, I3 )
400   CONTINUE

      P = 0
      DO 905 I = 0, 2046
         IF ( GV(I).EQ.0 ) THEN
            P = P + 1
            ZEROS(P) = I
         ENDIF
905   CONTINUE
      PRINT *
      PRINT *
      PRINT *
      WRITE  ( 6, 3000 ) ( ZEROS(I), I = 1, P )
3000  FORMAT ( 1X, ( T5, 10I5 / ) )

      END

      SUBROUTINE CREATE ( STR, N )
      CHARACTER*(*) STR
      INTEGER N
      INTEGER I
      INTEGER COPY

      STR = ' '
      COPY = N
```

```
      I = 0
15    IF ( COPY.GT.0 ) THEN
         I = I + 1
         STR(I:I) = CHAR( ICHAR('0') + MOD( COPY, 2 ) )
         COPY = COPY/2
         GO TO 15
      ENDIF

      END

      SUBROUTINE RS3( STR, COPY )
      CHARACTER*(*) STR, COPY
      INTEGER I, J, K, N

      N = INDEX( STR, ' ' ) - 1
      COPY = ' '
      DO 5 I = 1, N
         J = I + 1
         K = N - I + 1
         COPY(J:J) = STR(K:K)
5     CONTINUE

      END

      SUBROUTINE SETUP ( STR, INSET, SLMT, GV, LMT )
      CHARACTER*(*) STR
```

```
      INTEGER LMT, SLMT
      INTEGER GV( 0:LMT )
      LOGICAL INSET(0:SLMT)
      INTEGER FB, UB, LB
      INTEGER I, J

      FB = INDEX( STR(2: ), ' ' ) + 1

* REDUCE BY 1'S
      UB = 0
      I = INDEX( STR, '1' )
100   IF ( I.GT.0 ) THEN
         LB = UB + I - 1
         UB = INDEX( STR(LB+1: ), '0' ) + LB
         IF ( UB.GT.LB ) THEN
            DO 110 J = LB+2, UB
               CALL EVAL( STR( :LB)//STR(J: ), GV, LMT,
INSET,
     +                    SLMT )
110         CONTINUE
            I = INDEX( STR(UB: ), '1' ) - 1
            GO TO 100
         ELSE
            DO 120 J = LB+2, FB
               CALL EVAL( STR( :LB)//STR(J: ), GV, LMT,
                          INSET, SLMT )
```

```
120         CONTINUE
         ENDIF
      ENDIF

* REDUCE BY 0'S

      UB = 0
      I = INDEX( STR, '0' )
200   IF ( I.GT.0 ) THEN
         LB = UB + I - 1
         UB = INDEX( STR(LB+1: ), '1' ) + LB
         IF ( UB.GT.LB ) THEN
            DO 210 J = LB+2, UB
               CALL EVAL( STR( :LB)//STR(J: ), GV, LMT,
INSET,
     +                    SLMT )
         CONTINUE
            I = INDEX( STR(UB: ), '0' ) - 1
            GO TO 200
         ELSE
            DO 220 J = LB+2, FB
               CALL EVAL( STR( :LB)//STR(J: ), GV, LMT,
INSET,
     +                    SLMT )
220         CONTINUE
         ENDIF
```

```
      ENDIF

* REDUCE BY 10'S
      UB = 0

      I = INDEX( STR, '10' )
300   IF ( I.GT.0 ) THEN
         LB = UB + I - 1
         UB = LB + 3
310      CONTINUE
            CALL EVAL( STR( :LB)//STR(UB:), GV, LMT, INSET,
     +                 SLMT )
            I = INDEX( STR(UB: ), '10' )
         IF ( I.EQ.1 ) THEN
             UB = UB + 2
             GO TO 310
          ELSE
             GO TO 300
          ENDIF
      ENDIF

* REDUCE BY 01'S
      UB = 0
      I = INDEX( STR, '01' )
400   IF ( I.GT.0 ) THEN
         LB = UB + I - 1
```

```
          UB = LB + 3

410       CONTINUE
             CALL EVAL( STR( :LB)//STR(UB:), GV, LMT, INSET,
     +                   SLMT )
             I = INDEX( STR(UB: ), '01' )
          IF ( I.EQ.1 ) THEN
             UB = UB + 2
             GO TO 410
          ELSE
             GOTO 400
          ENDIF
       ENDIF

       END

       SUBROUTINE EVAL( STR1, GV, LMT, INSET, SLMT )
       CHARACTER*(*) STR1
       INTEGER LMT, SLMT
       INTEGER GV( 0:LMT )
       LOGICAL INSET(0:LMT)
       CHARACTER*100 COPY
       INTEGER I, NUM

       COPY = STR1(2: )
       IF ( COPY(1:1).EQ.'1' ) THEN
```

```
            NUM = 1
            I = 2
100         IF ( COPY(I:I).NE.' ' ) THEN
               NUM = NUM + NUM
               IF ( COPY(I:I).EQ.'1' ) THEN
                  NUM = NUM + 1
               ENDIF
               I = I + 1
               GO TO 100
            ENDIF
         ELSE
            NUM = 1
            I = 2
200         IF ( COPY(I:I).NE.' ' ) THEN
               NUM = NUM + NUM
               IF ( COPY(I:I).EQ.'0' ) THEN
                  NUM = NUM + 1
               ENDIF
               I = I + 1
               GO TO 200
            ENDIF
         ENDIF
         INSET( GV(NUM) ) = .TRUE.

         END
```

I	GV(I)	I	GV(I)	I	GV(I)	I	GV(I)
0	0	32	6	64	3	96	7
1	1	33	6	65	7	97	7
2	2	34	6	66	7	98	2
3	2	35	1	67	7	99	2
4	0	36	6	68	2	100	2
5	0	37	6	69	2	101	2
6	0	38	6	70	2	102	2
7	3	39	6	71	2	103	0
8	4	40	6	72	2	104	7
9	3	41	6	73	2	105	7
10	3	42	6	74	7	106	7
11	4	43	6	75	7	107	0
12	3	44	6	76	2	108	2
13	4	45	6	77	2	109	0
14	4	46	6	78	2	110	2
15	4	47	6	79	2	111	3
16	5	48	1	80	7	112	2
17	5	49	1	81	7	113	2
18	5	50	6	82	7	114	2
19	5	51	6	83	7	115	2
20	5	52	6	84	7	116	7
21	5	53	6	85	4	117	7
22	5	54	6	86	7	118	2
23	5	55	6	87	7	119	2
24	5	56	1	88	2	120	7

25	5	57	6	89	2	121	1
26	5	58	6	90	2	122	1
27	1	59	6	91	0	123	3
28	5	60	6	92	2	124	7
29	5	61	6	93	2	125	3
30	5	62	6	94	1	126	7
31	5	63	6	95	3	127	7

The following numbers are the decimal representation of the first sixty one safe strings over the binary alphabet { 0, 1 }. The value given to the empty string is zero. The other values are the values of the canonical form of a particular safe string, that is, a string stating with a 1. For example, the strings 001 and 100 are equivalent in that the pattern is identical, but the value of the string is 4, the decimal equivalent of the string 100.

0	4	5	6	91	103	107	109	134	135
141	142	143	147	153	154	156	176	177	178
180	184	194	195	197	198	200	201	204	216
224	225	226	228	236	240	241	264	289	350
478	495	757	762	871	877	997	998	1189	1210
1277	1301	1325	1343	1349	1350	1375	1661	1684	1866
1873									

The last three games are variations on ABEL and are given without analysis.

A variation of Abel, Baker, is to allow either a translation or a reduction with the provision that no position can be repeated.

A formal representation of the game Baker

Baker := (C, M, B)

Where C is the control structure [set of metarules]

M is the set of moves [rules]

and B is the (initial) position

B is a string $\alpha \circ \beta$ where $\alpha \in \Sigma^*$

$\Sigma = \{ a,b \}$

$\beta \in T^*$

$T = \{ L,R \}$

and $\circ$ is concatenation.

M is a set of moves or (rewrite) rules on $\Sigma^* \circ T \times \Sigma^* \circ T$

$p(g(w) \mid h(w))$ for $w \in \Sigma^* \circ T$

h is a translation rule

h := { (ab,ba), (ba,ab), (a,a), (b,b) }

g is a reduction rule

g := { $((ab)^n, (ab)^m)$, $((ba)^n, (ba)^m)$, (a^n, a^m),

(b^n, b^m) }

n a positive integer, $m \in \mathbb{N}$ and $n > m$

p is a marking rule to determine the player whose turn it is

$p := \{ (L,R), (R,L) \}$

C is a set of metarules that determine the winner and loser; whether the game is played under the normal ending condition or under the misère ending condition; and, if a move is legal.

In Baker we have the metarule that no position may be repeated.

Charlie is a variation of Baker that uses string reversals as the translation operation.

A formal representation of the game Charlie

Charlie := (C, M, B)

Where C is the control structure [set of metarules]

M is the set of moves [rules]

and B is the (initial) position

B is a string $\alpha \circ \beta$ where $\alpha \in \Sigma^*$

$\Sigma = \{ a,b \}$

β is an element of T^*

$T = \{ L,R \}$

and $\circ$ is concatenation.

M is a set of moves or (rewrite) rules on $\Sigma^* \circ T \times \Sigma^* \circ T$

$p(g(w) \mid h(w))$ for $w \in \Sigma^* \circ T$

h is a translation rule

$h := \{ w{+}{+}, rev(w{+}{+}) \}$

where w++ is a substring of α with length 2 or more

and REV is defined as follows

$REV(e) = e$ where e := the empty string

$REV(z) = z$ and $z \in \Sigma$

$REV(z1wz2) = z2 \circ REV(w) \circ z1$

where $z1, z2 \in \Sigma$

and $w \in \Sigma^*$

g is a reduction rule

$g := \{ ((ab)^n, (ab)^m), ((ba)^n, (ba)^m), (a^n, a^m),$

$(b^n, b^m) \}$

n a positive integer, m a natural number and $n > m$

p is a marking rule to determine the player whose turn it is

$p := \{ (L,R), (R,L) \}$

C is a set of metarules that determine the winner and loser; whether the game is played under the normal ending condition or under the misère ending condition; and, if a move is legal.

In Charlie as in Baker we have the metarule that no position may be repeated.

Delta is a game played on an alphabet of order 3. Instead of deleting a's and b's Delta replaces them with c's. A player may either transform the string or replace part of the string with c's. To avoid ties we add the metarule that repetitions are not allowed.

A formal representation of the game Delta

Delta := (C, M, B)

Where C is the control structure [set of metarules]

M is the set of moves [rules]

and B is the (initial) position

B is a string $\alpha \circ \beta$ where $\alpha \in \Sigma^*$

$\Sigma = \{ a,b,c \}$

$\beta \in T^*$

$T = \{ L,R \}$

and $\circ$ is concatenation.

M is a set of moves or (rewrite) rules on $\Sigma^* \circ T \times \Sigma^* \circ T$

$p(g(w) \mid h(w))$ for $w \in \Sigma^* \circ T$

h is a translation rule

h := { (ab,ba), (ba,ab) }

g is a replacement rule

g := { $((ab)^n, (cc)^n)$, $((ba)^n, (cc)^n)$,

(a^n, c^n), (b^n, c^n) }

where $n \in \mathbb{Z}^+$

p is a marking rule to determine the player whose turn it is

$p := \{ (L,R), (R,L) \}$

C is a set of metarules that determine the winner and loser; whether the game is played under the normal ending condition or under the Misère ending condition; and, if a move is legal.

APPENDIX A

A LIST OF THREE-TUPLES AND THE COSTS ASSOCIATED WITH THE THREE-HEAP DECISION PROBLEM ALGORITHM

X	Y	Z	P?	BLOCK COUNT	$N^2LOG_2(N)$	$N^{2.5}$
0	1	2	T	1	0.83048218	0.17677665
1	2	3	F	1	0.23287815	0.06414998
2	3	4	F	3	0.31143069	0.09375000
3	4	5	F	6	0.34336251	0.10733122
4	5	6	F	10	0.35697144	0.11340231
5	6	7	F	18	0.43467981	0.13884407
6	7	8	F	27	0.46714610	0.14915532
7	8	9	F	40	0.51750696	0.16460901
8	9	10	F	57	0.56999999	0.18024981
9	10	11	F	63	0.49996638	0.15698522
10	11	12	F	121	0.77862495	0.24256730
11	12	13	F	159	0.84459287	0.26093882
12	13	14	F	171	0.76121444	0.23317182
13	14	15	F	196	0.74068373	0.22491992
14	15	16	F	238	0.77208894	0.23242187
15	16	17	F	291	0.81833607	0.24421400
16	17	18	F	296	0.72779441	0.21533293
17	18	19	F	334	0.72352332	0.21225721
18	19	20	F	385	0.73979861	0.21522152
19	20	21	F	457	0.78374422	0.22613508

20	21	22	F	525	0.80802488	0.23126107
21	22	23	F	626	0.86901742	0.24674863
22	23	24	F	718	0.90314299	0.25444639
23	24	25	F	825	0.94424689	0.26400000
24	25	26	F	958	1.00154495	0.27792788
25	26	27	F	1139	1.09155560	0.30068672
26	27	28	F	1097	0.96688479	0.26443040
27	28	29	F	1147	0.93261415	0.25326115
28	29	30	F	1330	1.00044441	0.26980400
29	30	31	F	1687	1.17708683	0.31529045
30	31	32	F	1607	1.04264355	0.27742207
31	32	33	F	1656	1.00141335	0.26471311
32	33	34	F	1982	1.11952686	0.29403985
33	34	35	F	2178	1.15147686	0.30052984
34	35	36	F	2403	1.19139194	0.30902773
35	36	37	F	2663	1.24041080	0.31979150
36	37	38	F	2963	1.29887295	0.33286846
37	38	39	F	2894	1.19586277	0.30467516
38	39	40	F	2949	1.15047169	0.29142362
39	40	41	F	3636	1.34115791	0.33780354
40	41	42	F	3549	1.23943043	0.31044358
41	42	43	F	4158	1.37669182	0.34293622
42	43	44	F	4456	1.40049839	0.34698719
43	44	45	F	4070	1.21573925	0.29961467
44	45	46	F	4238	1.20452595	0.29530197
45	46	47	F	4779	1.29383659	0.31556755

46	47	48	F	5148	1.32900333	0.32250416
47	48	49	F	6510	1.60417461	0.38733858
48	49	50	F	5952	1.40132046	0.33669597
49	50	51	F	6455	1.45337486	0.34751278
50	51	52	F	5986	1.29006577	0.30699289
51	52	53	F	6623	1.36740208	0.32386577
52	53	54	F	7188	1.42289829	0.33544677
53	54	55	F	8370	1.58986473	0.37309438
54	55	56	F	8185	1.49298191	0.34877759
55	56	57	F	8473	1.48523045	0.34542203
56	57	58	F	8729	1.47146797	0.34071749
57	58	59	F	9012	1.46195793	0.33704752
58	59	60	F	9722	1.51874352	0.34864020
59	60	61	F	11195	1.68517876	0.38521171
60	61	62	F	11357	1.64834118	0.37521857
61	62	63	F	11208	1.56940079	0.35577607
62	63	64	F	12668	1.71232796	0.38659668
63	64	65	F	12271	1.60205078	0.36024380
64	65	66	F	13330	1.68182087	0.37667799
65	66	67	F	14049	1.71386814	0.38234764
66	67	68	F	14245	1.68111897	0.37358558
67	68	69	F	14795	1.68993759	0.37410390
68	69	70	F	15796	1.74715519	0.38530266
69	70	71	F	16709	1.79046822	0.39337301
70	71	72	F	16916	1.75688362	0.38456202
71	72	73	F	17743	1.78687000	0.38969052

72	73	74	F	18555	1.81273460	0.39389604
73	74	75	F	18994	1.80085373	0.38990897
74	75	76	F	19156	1.76332283	0.38042653
75	76	77	F	20592	1.84103584	0.39579624
76	77	78	F	21825	1.89592934	0.40617925
77	78	79	F	22631	1.91090393	0.40797728
78	79	80	F	24670	2.02548981	0.43096715
79	80	81	F	24623	1.96644592	0.41699266
80	81	82	F	25267	1.96347904	0.41497219
81	82	83	F	26972	2.04016018	0.42975199
82	83	84	F	27204	2.00357723	0.42066318
83	84	85	F	28322	2.03170013	0.42518383
84	85	86	F	29146	2.03710556	0.42494506
85	86	87	F	28984	1.97435665	0.41054440
86	87	88	F	31376	2.08366585	0.43190747
87	88	89	F	31906	2.06630039	0.42696995
88	89	90	F	32584	2.05845261	0.42403132
89	90	91	F	33477	2.06357384	0.42378259
90	91	92	F	34688	2.08693504	0.42727727
91	92	93	F	33040	1.94062901	0.39612532
92	93	94	F	34187	1.96087551	0.39906293
93	94	95	F	39469	2.21127701	0.44869077
94	95	96	F	37774	2.06769943	0.41832596
95	96	97	F	41891	2.24093533	0.45205522
96	97	98	F	42178	2.20553112	0.44362986
97	98	99	F	42951	2.19594574	0.44043863

98	99	100	F	44954	2.24769974	0.44953996
99	100	101	F	44802	2.19122696	0.43701273
100	101	102	F	44913	2.14920616	0.42743647
101	102	103	F	46211	2.16402435	0.42919278
102	103	104	F	47739	2.18823242	0.43280262
103	104	105	F	49464	2.21974754	0.43784058
104	105	106	F	54016	2.37366867	0.46693647
105	106	107	F	54355	2.33941746	0.45896548
106	107	108	F	54186	2.28460789	0.44702083
107	108	109	F	56547	2.33600903	0.45587230
108	109	110	F	61597	2.49372005	0.48537546
109	110	111	F	59480	2.36027718	0.45820880
110	111	112	F	61532	2.39373875	0.46350676
111	112	113	F	64002	2.44135571	0.47151715
112	113	114	F	68066	2.54628086	0.49053282
113	114	115	F	65348	2.39785194	0.46077371
114	115	116	F	67806	2.44087791	0.46786773
115	116	117	F	68356	2.41443348	0.46164900
116	117	118	F	72851	2.52526188	0.48164874
117	118	119	F	70176	2.38760281	0.45427752
118	119	120	F	74184	2.47773743	0.47028059
119	120	121	F	75522	2.47661304	0.46893215
120	121	122	F	75737	2.43892860	0.46068978
121	122	123	F	79857	2.52566814	0.47593772
122	123	124	F	83429	2.59189129	0.48726231
123	124	125	F	83104	2.53642464	0.47571504

124	125	126	F	86464	2.59297466	0.48518670
125	126	127	F	88238	2.60041714	0.48545194
126	127	128	F	90961	2.63467693	0.49071604
127	128	129	F	92540	2.63479137	0.48961550
128	129	130	F	95280	2.66699219	0.49447393
129	130	131	F	95816	2.63705349	0.48782021
130	131	132	F	99425	2.69087982	0.49666184
131	132	133	F	96253	2.56204510	0.47182953
132	133	134	F	100929	2.64251518	0.48557228
133	134	135	F	104427	2.68966103	0.49314922
134	135	136	F	109752	2.78121090	0.50882119
135	136	137	F	109543	2.73146534	0.49863553
136	137	138	F	107799	2.64525318	0.48185563
137	138	139	F	111576	2.69473457	0.48981684
138	139	140	F	114830	2.72988033	0.49514830
139	140	141	F	115682	2.70736408	0.49002469
140	141	142	F	117222	2.70104599	0.48785210
141	142	143	F	125290	2.84268856	0.51236123
142	143	144	F	121024	2.70409584	0.48636830
143	144	145	F	129008	2.83891106	0.50956154
144	145	146	F	127773	2.76952553	0.49608600
145	146	147	F	136490	2.91436291	0.52096373
146	147	148	F	131946	2.77562428	0.49515581
147	148	149	F	136488	2.82894802	0.50364989
148	149	150	F	145279	2.96717548	0.52719903
149	150	151	F	145742	2.93344593	0.52016646

150	151	152	F	145069	2.87782097	0.50929052
151	152	153	F	146957	2.87353992	0.50752997
152	153	154	F	143053	2.75742245	0.48606592
153	154	155	F	152738	2.90251064	0.51064354
154	155	156	F	153037	2.86737061	0.50348312
155	156	157	F	160522	2.96566868	0.51973915
156	157	158	F	155329	2.82996368	0.49500525
157	158	159	F	157331	2.82697296	0.49353904
158	159	160	F	164802	2.92070198	0.50893515
159	160	161	F	160185	2.80028248	0.48703158
160	161	162	F	169840	2.92895699	0.50845492
161	162	163	F	173194	2.94669819	0.51057994
162	163	164	F	184110	3.09062672	0.53452462
163	164	165	F	177905	2.94686222	0.50871938
164	165	166	F	178464	2.91716003	0.50266695
165	166	167	F	191985	3.09706497	0.53269190
166	167	168	F	187051	2.97818089	0.51131290
167	168	169	F	193741	3.04477310	0.52180082
168	169	170	F	188463	2.92372131	0.50015408
169	170	171	F	204691	3.13486290	0.53531384
170	171	172	F	200868	3.03720379	0.51771367
171	172	173	F	205291	3.06484795	0.52150035
172	173	174	F	208838	3.07862568	0.52292132
173	174	175	F	219677	3.19795227	0.54223734
174	175	176	F	212234	3.05122375	0.51645595
175	176	177	F	223249	3.16994572	0.53561944

176	177	178	F	217070	3.04436302	0.51351100
177	178	179	F	223565	3.09716988	0.52152026
178	179	180	F	234990	3.21592140	0.54059011
179	180	181	F	238098	3.21911430	0.54020584
180	181	182	F	239833	3.20364380	0.53669858
181	182	183	F	241771	3.19097233	0.53367448
182	183	184	F	244893	3.19379997	0.53325117
183	184	185	F	257783	3.32220459	0.55376428
184	185	186	F	250775	3.19393158	0.53149825
185	186	187	F	257684	3.24359226	0.53886926
186	187	188	F	252560	3.14216042	0.52115864
187	188	189	F	256855	3.15867043	0.52303827
188	189	190	F	258790	3.14588737	0.52007198
189	190	191	F	268987	3.23245811	0.53351653
190	191	192	F	284017	3.37426281	0.55602109
191	192	193	F	288497	3.38871193	0.55750406
192	193	194	F	271750	3.15607834	0.51840025
193	194	195	F	292057	3.35395050	0.55002326
194	195	196	F	283802	3.22284985	0.52768558
195	196	197	F	298297	3.34992409	0.54762506
196	197	198	F	302751	3.36246300	0.54881072
197	198	199	F	304047	3.33982468	0.54426199
198	199	200	F	311104	3.38005161	0.54995930
199	200	201	F	314084	3.37538242	0.54834718
200	201	202	F	327965	3.48648548	0.56552142
201	202	203	F	324224	3.40966892	0.55221099

202	203	204	F	323524	3.36591911	0.54429090
203	204	205	F	319669	3.29041958	0.53127068
204	205	206	F	321152	3.27067757	0.52728152
205	206	207	F	334506	3.37077904	0.54259777
206	207	208	F	337614	3.36642075	0.54108071
207	208	209	F	337951	3.33461475	0.53516537
208	209	210	F	344931	3.36814404	0.53973925
209	210	211	F	358543	3.46487331	0.55441517
210	211	212	F	367353	3.51349640	0.56136316
211	212	213	F	366572	3.47013378	0.55361807
212	213	214	F	377958	3.54146481	0.56416881
213	214	215	F	376769	3.49452496	0.55587733
214	215	216	F	384157	3.52708530	0.56024027
215	216	217	F	385034	3.49962234	0.55507249
216	217	218	F	387540	3.48718071	0.55230033
217	218	219	F	395231	3.52098846	0.55685318
218	219	220	F	402444	3.54972744	0.56059438
219	220	221	F	398363	3.47907829	0.54865366
220	221	222	F	417823	3.61320972	0.56899691
221	222	223	F	419994	3.59649086	0.56556290
222	223	224	F	430721	3.65246773	0.57355618
223	224	225	F	442404	3.71520805	0.58258963
224	225	226	F	443802	3.69101524	0.57798707
225	226	227	F	455392	3.75105762	0.58657116
226	227	228	F	452360	3.69047832	0.57629794
227	228	229	F	460460	3.72082424	0.58023405

228	229	230	F	474932	3.80142117	0.59198648
229	230	231	F	474931	3.76556683	0.58559930
230	231	232	F	472116	3.70810509	0.57587570
231	232	233	F	464254	3.61227036	0.56022930
232	233	234	F	472891	3.64522839	0.56457466
233	234	235	F	484050	3.69666862	0.57176888
234	235	236	F	495728	3.75092030	0.57937992
235	236	237	F	497691	3.73117542	0.57555777
236	237	238	F	494196	3.67107677	0.56553149
237	238	239	F	506372	3.72725773	0.57342273
238	239	240	F	523551	3.81874847	0.58671993
239	240	241	F	526939	3.80874157	0.58441013
240	241	242	F	528140	3.78308105	0.57970977
241	242	243	F	540143	3.83440113	0.58680397
242	243	244	F	542244	3.81497574	0.58306926
243	244	245	F	553606	3.86031246	0.58923095
244	245	246	F	560974	3.87707996	0.59102374
245	246	247	F	555547	3.80574036	0.57939988
246	247	248	F	556333	3.77768135	0.57438833
247	248	249	F	557773	3.75436020	0.57011062
248	249	250	F	595595	3.97404575	0.60269976
249	250	251	F	584826	3.86836147	0.58592540
250	251	252	F	587593	3.85310650	0.58287472
251	252	253	F	605333	3.93529987	0.59455633
252	253	254	F	614885	3.96315765	0.59801161
253	254	255	F	621813	3.97362041	0.59883797

254	255	256	F	633034	4.01095009	0.60370827
255	256	257	F	638903	4.01387405	0.60339558
256	257	258	F	648295	4.03853989	0.60635006
257	258	259	F	636750	3.93331051	0.58981979
258	259	260	F	658618	4.03435898	0.60422724
259	260	261	F	663871	4.03265095	0.60322952
260	261	262	F	653860	3.93887043	0.58847988
261	262	263	F	682570	4.07782078	0.60849595
262	263	264	F	681253	4.03643131	0.60158718
263	264	265	F	708886	4.16569328	0.62010002
264	265	266	F	674874	3.93340588	0.58481532
265	266	267	F	702988	4.06390190	0.60348946
266	267	268	F	721905	4.13940334	0.61396396
267	268	269	F	708385	4.02905178	0.59688205
268	269	270	F	717393	4.04743195	0.59889102
269	270	271	F	738593	4.13361549	0.61091685
270	271	272	F	744692	4.13444138	0.61031568
271	272	273	F	744258	4.09913254	0.60438955
272	273	274	F	748857	4.09174442	0.60259074
273	274	275	F	764747	4.14554310	0.60979789
274	275	276	F	772148	4.15270138	0.61013752
275	276	277	F	793040	4.23159981	0.62100583
276	277	278	F	801195	4.24169445	0.62176484
277	278	279	F	804535	4.22620296	0.61877739
278	279	280	F	825310	4.30169010	0.62910330
279	280	281	F	810357	4.19108963	0.61222410

280	281	282	F	816202	4.18879604	0.61118811
281	282	283	F	835345	4.25412750	0.62001133
282	283	284	F	854690	4.31934643	0.62880033

APPENDIX B

A LIST OF THREE TUPLES AND THEIR GRUNDY VALUES COSTS ASSOCIATED WITH THE NEW ALGORITHM

X	Y	Z	V	OPERATIONS ON TYPE 2 BLOCKS TOTAL	THIS TUPLE
0	0	0	0	1	1
0	1	2	0	2	1
0	3	5	0	3	1
0	4	7	0	4	1
0	6	10	0	5	1
0	8	13	0	6	1
0	9	15	0	7	1
0	11	18	0	8	1
0	12	20	0	9	1
0	14	23	0	10	1
0	16	26	0	11	1
0	17	28	0	12	1
0	19	31	0	13	1
1	3	4	0	45	32
1	5	8	0	46	1
1	6	12	0	49	3
1	7	11	0	50	1
1	9	14	0	51	1
1	10	17	0	52	1
1	13	21	0	53	1
1	15	24	0	54	1

1	16	28	0	57	3
1	18	29	0	59	2
1	20	30	0	63	4
2	3	6	0	96	33
2	4	8	0	99	3
2	5	7	0	100	1
2	9	16	0	103	3
2	10	15	0	104	1
2	11	19	0	107	3
2	12	18	0	108	1
2	13	23	0	110	2
2	14	26	0	114	4
2	17	31	0	120	6
2	20	29	0	121	1
3	7	9	0	152	31
3	8	14	0	158	6
3	10	11	0	159	1
3	12	21	0	162	3
3	13	20	0	163	1
3	15	23	0	164	1
3	16	27	0	166	2
3	17	29	0	169	3
4	5	10	0	209	40
4	6	9	0	212	3
4	11	12	0	213	1
4	13	22	0	219	6

4	14	24	0	226	7
4	15	19	0	227	1
4	16	29	0	233	6
4	17	25	0	234	1
4	18	30	0	236	2
5	6	14	0	272	36
5	9	11	0	274	2
5	12	13	0	275	1
5	15	26	0	283	8
5	16	20	0	284	1
5	17	27	0	286	2
5	19	28	0	292	6
6	7	13	0	324	32
6	8	11	0	326	2
6	15	28	0	338	12
6	16	23	0	344	6
6	17	21	0	347	3
6	20	22	0	371	24
7	8	16	0	404	33
7	10	12	0	406	2
7	14	19	0	411	5
7	15	31	0	427	16
7	17	30	0	440	13
7	18	27	0	449	9
7	20	21	0	450	1
8	9	17	0	482	32

8	10	20	0	492	10
8	12	19	0	499	7
8	15	18	0	502	3
8	21	22	0	503	1
8	23	29	0	504	1
9	10	18	0	535	31
9	12	22	0	545	10
9	13	24	0	556	11
9	21	28	0	586	30
9	23	26	0	589	3
9	25	30	0	594	5
10	13	16	0	625	31
10	14	21	0	631	6
10	19	23	0	634	3
10	25	27	0	649	15
11	13	14	0	671	22
11	15	25	0	679	8
11	16	21	0	682	3
11	20	31	0	703	21
12	14	15	0	748	45
12	16	30	0	761	13
12	17	23	0	766	5
12	29	31	0	787	21
13	15	17	0	808	21
13	18	28	0	815	7
13	19	30	0	823	8

13	26	31	0	828	5
14	16	22	0	853	25
14	18	31	0	874	21
14	20	27	0	878	4
14	25	29	0	879	1
15	22	27	0	940	61
16	17	19	0	961	21
16	18	24	0	964	3
17	18	22	0	986	22
17	20	26	0	988	2
18	19	21	0	1007	19
18	20	23	0	1008	1
19	20	25	0	1031	23
19	22	24	0	1032	1
20	24	28	0	1048	16
21	23	25	0	1061	13
21	24	27	0	1062	1
22	23	30	0	1079	17
22	25	31	0	1082	3
23	28	31	0	1109	27
24	29	30	0	1131	22
26	27	28	0	1154	23
0	0	1	1	1178	1
0	2	2	1	1179	1
0	3	6	1	1180	1
0	4	8	1	1182	2

0	5	7	1	1183	1
0	9	14	1	1184	1
0	10	16	1	1185	1
0	11	19	1	1186	1
0	12	21	1	1188	2
0	13	20	1	1189	1
0	15	25	1	1190	1
0	17	29	1	1191	1
0	18	31	1	1194	3
1	1	2	1	1195	1
1	3	5	1	1196	1
1	4	7	1	1197	1
1	6	10	1	1198	1
1	8	13	1	1199	1
1	9	16	1	1201	2
1	11	17	1	1202	1
1	12	20	1	1203	1
1	14	23	1	1204	1
1	15	27	1	1207	3
1	18	28	1	1208	1
1	19	30	1	1209	1
2	3	4	1	1240	31
2	5	8	1	1241	1
2	6	12	1	1244	3
2	7	11	1	1245	1
2	9	18	1	1249	4

2	10	17	1	1251	2
2	13	21	1	1255	4
2	14	19	1	1256	1
2	15	26	1	1258	2
2	16	28	1	1261	3
3	3	8	1	1268	7
3	7	7	1	1269	1
3	9	10	1	1270	1
3	11	14	1	1271	1
3	12	19	1	1272	1
3	13	24	1	1276	4
3	15	30	1	1283	7
3	16	25	1	1285	2
3	17	31	1	1291	6
3	18	26	1	1292	1
4	4	6	1	1297	5
4	5	5	1	1298	1
4	9	12	1	1299	1
4	10	14	1	1300	1
4	11	16	1	1301	1
4	13	26	1	1307	6
4	15	22	1	1308	1
4	17	27	1	1310	2
4	18	29	1	1313	3
4	19	28	1	1314	1
5	6	6	1	1341	27

5	9	17	1	1347	6
5	10	13	1	1348	1
5	11	15	1	1349	1
5	12	22	1	1355	6
5	14	25	1	1362	7
5	16	29	1	1371	9
5	19	24	1	1381	10
5	21	30	1	1385	4
6	7	14	1	1417	32
6	8	16	1	1423	6
6	9	11	1	1424	1
6	13	17	1	1425	1
6	15	20	1	1426	1
6	18	24	1	1427	1
6	19	31	1	1429	2
7	8	8	1	1454	25
7	9	20	1	1465	11
7	10	24	1	1478	13
7	12	18	1	1484	6
7	13	23	1	1494	10
7	15	28	1	1507	13
7	17	19	1	1524	17
7	21	22	1	1525	1
8	9	25	1	1564	39
8	10	12	1	1565	1
8	11	22	1	1572	7

8	14	20	1	1574	2
8	15	23	1	1578	4
8	17	24	1	1581	3
8	18	27	1	1586	5
9	9	13	1	1605	19
9	15	21	1	1612	7
9	19	22	1	1616	4
9	23	23	1	1617	1
9	26	27	1	1625	8
9	28	30	1	1626	1
10	10	11	1	1627	1
10	15	15	1	1628	1
10	18	20	1	1629	1
10	21	28	1	1643	14
10	22	25	1	1644	1
11	11	12	1	1645	1
11	13	28	1	1659	14
11	18	25	1	1665	6
11	20	29	1	1673	8
11	23	27	1	1685	12
11	24	30	1	1690	5
12	12	15	1	1697	7
12	13	27	1	1711	14
12	14	26	1	1722	11
12	16	23	1	1729	7
12	17	25	1	1736	7

12	29	30	1	1747	11
13	13	14	1	1748	1
13	15	29	1	1760	12
13	16	18	1	1761	1
13	22	30	1	1776	15
13	25	31	1	1780	4
14	14	16	1	1782	2
14	15	18	1	1783	1
14	17	28	1	1791	8
14	21	31	1	1798	7
14	22	27	1	1800	2
15	16	19	1	1824	24
15	24	31	1	1845	21
16	16	17	1	1846	1
16	20	26	1	1850	4
16	21	24	1	1851	1
17	17	18	1	1853	2
17	23	26	1	1881	28
18	18	22	1	1885	4
18	21	23	1	1896	11
19	19	21	1	1898	2
19	20	20	1	1899	1
19	23	29	1	1903	4
20	21	21	1	1924	21
20	24	27	1	1938	14
20	25	30	1	1940	2

21	26	29	1	1961	21
22	22	23	1	1965	4
24	24	25	1	2008	43
24	28	28	1	2015	7
25	25	27	1	2018	3
26	26	31	1	2027	9
26	30	30	1	2032	5
27	27	29	1	2034	2
28	29	29	1	2039	5
30	31	31	1	2044	5
0	0	2	2	2045	1
0	1	1	2	2046	1
0	3	4	2	2047	1
0	5	8	2	2048	1
0	6	11	2	2049	1
0	7	13	2	2052	3
0	9	16	2	2054	2
0	10	14	2	2055	1
0	12	22	2	2056	1
0	15	23	2	2057	1
0	17	26	2	2058	1
0	18	29	2	2059	1
1	2	2	2	2089	30
1	3	6	2	2090	1
1	4	8	2	2092	2
1	5	7	2	2093	1

1	9	15	2	2094	1
1	10	18	2	2097	3
1	11	16	2	2098	1
1	12	19	2	2099	1
1	13	23	2	2101	2
1	14	25	2	2103	2
1	17	29	2	2107	4
1	21	30	2	2110	3
2	3	5	2	2141	31
2	4	7	2	2142	1
2	6	10	2	2143	1
2	8	13	2	2144	1
2	9	17	2	2146	2
2	11	20	2	2149	3
2	12	23	2	2154	5
2	14	21	2	2156	2
2	15	27	2	2162	6
2	16	22	2	2163	1
2	18	28	2	2164	1
3	3	9	2	2169	5
3	7	8	2	2170	1
3	10	10	2	2171	1
3	11	13	2	2172	1
3	12	26	2	2178	6
3	14	23	2	2181	3
3	15	25	2	2184	3

3	16	24	2	2186	2
3	17	28	2	2191	5
3	18	30	2	2196	5
3	20	27	2	2203	7
4	4	5	2	2204	1
4	6	14	2	2212	8
4	9	9	2	2213	1
4	10	12	2	2214	1
4	11	15	2	2215	1
4	13	19	2	2216	1
4	16	26	2	2218	2
4	17	31	2	2223	5
4	18	27	2	2224	1
5	5	10	2	2230	6
5	6	9	2	2232	2
5	11	18	2	2237	5
5	12	14	2	2238	1
5	13	17	2	2239	1
5	15	24	2	2240	1
5	16	28	2	2243	3
5	19	29	2	2244	1
5	20	31	2	2245	1
6	6	7	2	2246	1
6	8	8	2	2247	1
6	12	17	2	2250	3
6	13	16	2	2251	1

6	15	21	2	2252	1
6	18	31	2	2259	7
6	19	26	2	2260	1
7	7	14	2	2271	11
7	9	10	2	2272	1
7	11	19	2	2278	6
7	12	15	2	2279	1
7	16	25	2	2283	4
7	17	27	2	2288	5
7	18	24	2	2289	1
8	9	18	2	2322	33
8	10	11	2	2324	2
8	12	12	2	2325	1
8	14	22	2	2327	2
8	15	28	2	2334	7
8	16	23	2	2336	2
8	19	25	2	2345	9
8	20	30	2	2346	1
9	11	23	2	2380	34
9	12	21	2	2388	8
9	13	20	2	2394	6
9	14	27	2	2406	12
9	19	30	2	2415	9
9	22	28	2	2420	5
9	24	26	2	2421	1
10	13	24	2	2454	33

10	15	16	2	2455	1
10	17	22	2	2456	1
10	19	28	2	2460	4
10	20	26	2	2461	1
10	21	31	2	2462	1
11	11	27	2	2477	15
11	12	31	2	2497	20
11	14	14	2	2498	1
11	17	21	2	2502	4
11	22	29	2	2509	7
11	25	26	2	2516	7
12	13	13	2	2536	20
12	16	18	2	2538	2
12	20	25	2	2543	5
12	24	27	2	2546	3
13	14	30	2	2584	38
13	15	15	2	2585	1
13	18	21	2	2587	2
13	25	29	2	2597	10
14	15	26	2	2632	35
14	16	17	2	2633	1
14	19	24	2	2644	11
14	20	28	2	2648	4
15	17	18	2	2666	18
15	19	22	2	2670	4
15	20	20	2	2671	1

16	16	20	2	2674	3
16	19	19	2	2675	1
16	21	29	2	2682	7
17	17	30	2	2699	17
17	23	23	2	2724	25
17	24	25	2	2725	1
18	18	23	2	2729	4
18	26	26	2	2770	41
19	20	21	2	2782	12
19	27	27	2	2791	9
20	23	29	2	2819	28
21	21	25	2	2828	9
21	22	22	2	2829	1
21	26	28	2	2841	12
22	23	25	2	2852	11
22	24	24	2	2853	1
23	26	27	2	2876	23
24	30	30	2	2888	12
25	25	28	2	2890	2
25	27	31	2	2893	3
26	31	31	2	2902	9
27	28	29	2	2907	5
28	28	31	2	2910	3

APPENDIX C

Games and Rewriting Systems

The most natural relationship between games and rewriting systems is that rewriting systems can be used to represent games.

A game is a triple

G = (R, C, B)

where R is a set of moves
C is a control unit
and B is the board

The control unit acts as referee and scorekeeper and where convenient serves to simplify the structure of the rules.

Functions of the control unit include

1) Determining who is on move

2) Determining if a move is legal

3) Determining who won the game.

If the game includes chance moves, the control unit might

1) Flip a fair coin

or 2) Roll a pair of dice

or 3) Shuffle and deal cards

or 4) Generate a random number.

A definition of a general rewriting system is given below. The definitions are from [Jantzen] and are supplemented with comments related to games. This is followed by definitions of Thue systems as rewriting systems taken from [Book].

GENERAL REDUCTION SYSTEMS [Jantzen pp.7-9]

A reduction system is generally characterized by a set *B* of objects together with a (by definition) meaning preserving one-step transformation relation ==> on *B*.

Definition 1.1.1 Let B be an arbitrary set of objects and ==> $\subset$ *B* X *B* a binary relation on *B*. Then (*B*, ==>) is

called a *reduction system*, and ==> is the one-step transformation relation.

Note that if we have b_1 ==> b_2 in some reduction system (B, ==>) then this does not necessarily mean that the object b_2 is simpler than the object b_1 or that b_1 is reduced modulo some metric defined on B. It can also mean that b_2 is more general than b_1 or that b_1 is the basis or root of the offspring b_2. However, the goal is usually to find a reduction ordering with respect to which the objects are reduced in each and every step, so that the term 'reduction system' makes more sense.

A combinatorial game can be represented by the following rewriting system:

G = (B, P, C)

where

B is a set of board states

P $\subset$ B x B is a set of productions or rewriting rules

and C is a control unit.

The control unit has three functions

1) Determining which player is on move

2) Inputting and executing a legal move

3) Determining which of the two players won the game.

The control unit is not part of the rewriting system but represents a metasystem which interprets the rewriting system as well as providing both input and output functions.

If the game is not the null game then there is a board state b ϵ B designated as the initial state of the game. Note that if the game is the null game or if the set of rewriting rules is empty then the first player to play always loses under the normal ending condition and always wins under the misère ending condition.

The moves are from board state to board state. A restriction imposed by G being a combinatorial game is that no nonempty sequence of productions can take a state b1 ϵ B to the same state b1 ϵ B, as this would violate the rule against draws. The fact that, say, Right is on play in the

first instance and Left is on play in the second has no significance as the question of who is on play is part of the metasystem and not represented in the rewriting system.

Every combinatorial game is representable as a string rewriting system. This follows from the fact that a combinatorial game is a position (finitely representable) together with a finite set of rules. The position has a representation as a string of symbols over a finite alphabet and the rules are transformations of that string. Combinatorial games are a subset of context sensitive languages.

There exist string rewriting systems that do not represent combinatorial games. As an example consider the string AB and the following set of rewriting rules { AB --> BA, BA --> AB }. This system leads to a draw by repetition.

Based on the binary reduction relation ==> one defines a number of further relations, mainly of technical interest.

Definition 1.1.2 Let $(B, ==>)$ be a reduction system.

(a) $=0=>$ is the *identity* on B.

(b) $=(-1)=>$ is the *inverse* of ==>, usually written <==.

This would occur in Go if there were no ko rule (requiring a player to make at least one other move before retaking a ko).

(c) $=n+1=> := =n=> \circ ==> \forall\ n \in \mathbb{N}$, where $\mathbb{N}$ is the set of nonnegative integers and $\circ$ is the usual *composition* of relations.

(d) $=+=> := \bigcup =n=> \forall\ n \in \mathbb{N}_0$ is the *transitive closure* of ==> and $\mathbb{N}_0 := \mathbb{N} - \{0\}$.

(e) $=*=> := =+=> \cup =0=>$ is the *reflexive transitive closure* of ==>, and thus a *quasi order* on B.

(f) <==> := ==> $\cup$ $=(-1)=>$ is the *symmetric closure of* ==>.

(g) $<=0=>$ is the identity on B.

(h) $<=n+1=> := <=n=> \circ <==>$ for all $n \in \mathbb{N}$.

(i) $<=+=> := \bigcup_{n \in \mathbb{N}} <=n=>$ is the *symmetric, transitive closure* of ==>.

(j) <=*=> := <=+=> ∪ <=0=> is the symmetric, transitive and reflexive closure of ==>, and thus an *equivalence relation* on B.

Let us recall the definitions of various ordering relations as we use them here.

Definition 1.1.3 Let R be a binary relation on a set B; then

(a) R is a *quasi order* if R is *transitive* (i.e., $R \circ R \subset R$) and *reflexive* (i.e., xRx for all $x \in B$).

(b) R is a *partial order* if R is a quasi order and *antisymmetric* (i.e., xRy and yRx implies $x=y$).

(c) R is a *strict partial order* if R is transitive and *irreflexive* (i.e., xRx for no $x \in B$).

(d) R is a *total order* if R is a partial order and xRy or yRx for all $x, y \in B$.

(e) R is *acyclic* if its transitive closure R^+ is irreflexive.

Thus, given a reduction system $(B, ==>)$ the derived relation =+=> is a strict partial order if and only if ==> is acyclic, in which case =+=> is a partial order.

Definition 1.1.4 If in some reduction system $(B, ==>)$ we have $b_1 =n=> b_2$ for some $n \in \mathbb{N}$ then we say that b_1 is an *ancestor* of b_2 and b_2 is a *descendant* of b_1.

If $n = 1$ the b_1 is a *direct ancestor* of b_2 and b_2 is a *direct descendant* of b_1.

If $n \neq 0$ then we speak of a *proper ancestor* and a *proper descendant* respectively.

Given a reduction system $(B, ==>)$, then, among others, the following problems are of major importance.

The common descendant problem (CDP)
Problem 1.1.5 Determine, given elements $b_1, b_2 \in B$, whether there exists some $b \in B$ which is a descendant of both b_1 and b_2.

The common ancestor problem (CAP)

Problem 1.1.6 Determine, given elements $b_1, b_2 \in B$, whether there exists some $b \in B$ which is a common ancestor of both b_1 and b_2.

The word (equivalence) problem (WP)
Problem 1.1.7 Determine, given $b_1, b_2 \in B$, whether b_1 and b_2 are equivalent, i.e., $b_1 <=*=> b_2$.

Problem 1.1.7 is usually the question of whether two syntactically different objects do have the same meaning, where the reduction relation is considered a meaning preserving transformation.

In Computer Science one is mainly interested in having an effective method for resolving these questions on the basis of the given input of $(B, ==>)$, b_1, and b_2. In order that this may be possible, we shall assume that in general,

(a) the elements $b \in B$ do have a finite description;
(b) the set B is recursively enumerable;
(c) the relation ==> is recursive.

Of course, these three properties by no means guarantee the decidability of any of these problems, ...

With respect to impartial games we find that the set of all positions are divided into two equivalence classes, P and N (respectively, Previous player wins and Next player wins), which will be discussed later. The word problem for impartial games can be stated as whether a particular board state is in P or in N.

Definition 1.1.8 Let R := $(B, ==>)$ be a reduction system and $A \subset B$, $x \in B$; then

(a) $\Delta(x) := \{y \in B | x ==> y\}$

(b) $\Delta^+(x) := \{y \in B | x =+=> y\}$

(c) $\Delta^*(x) := \{y \in B | x =*=> y\}$

(d) $\Delta(A) := \cup_{x \in A} \Delta(x)$

(e) $\Delta^+(A) := \cup_{x \in A} \Delta^+(x)$

(f) $\Delta^*(A) := \cup_{x \in A} \Delta^*(x)$

In this and the following definitions we shall add the notation (mod R), or equivalently w.r.t. R, whenever it is necessary to avoid confusion.

Thus $\Delta^*(x)$ is the set of descendants of x and $\Delta^+(x)$ is the set of all proper descendants. Notice that it may happen that some element is its own proper ancestor, namely when $x =n=> x$ for some $n \in \mathbb{N}$. [e.g., A triple ko in Go]

We shall define in a similar way various sets of ancestors where we also distinguish between direct, proper, and arbitrary ancestors.

Definition 1.1.9 Let R := $(B,==>)$ be a reduction system and $A \subset B$, $x \in B$; then

(a) $\langle x \rangle := \{ y \in B \mid y ==> x \}$

(b) $\langle x \rangle + := \{ y \in B \mid y =+=> x \}$

(c) $\langle x \rangle * := \{ y \in B \mid y =*=> x \}$

(d) $\langle A \rangle := \bigcup_{x \in A} \langle x \rangle$

(e) $\langle A \rangle + := \bigcup_{x \in A} \langle x \rangle +$

(f) $\langle A \rangle * := \bigcup_{x \in A} \langle x \rangle *$

Definition 1.1.10 Let R := $(B,==>)$ be a reduction system and $A \subset B$, $x \in B$; then

(a) $[x]_R := \{ y \in B \mid x <=*=> y \}$ is the *equivalence* class of the element x, (mod R).

(b) $[A]R := \bigcup_{x \in A} [x]_R$

Here, too, the subscript R will be omitted if possible.

Definition 1.1.11 Two reduction systems $(B_1,=1=>)$ and $(B_2,=2=>)$ are *equivalent* iff $B_1 = B_2$ and $<=1*=> = <=2*=>$. They are called *isomorphic* if there exists a bijection f: $B_1 \dashrightarrow B_2$ such that $x <=1*=> y$ iff $f(x) <=1*=> f(y)$.

Definition 1.1.12 Let R := $(B,==>)$ be a reduction system. Then $x \in B$ is *irreducible* (mod R) if $\Delta(x) = \phi$. Let $x \in B$, $A \subset B$; then

(a) $IRR(x) := \{ y \in \Delta^*(x) \mid y \text{ is irreducible} \}$

(b) $IRR(A) := \bigcup_{x \in A} IRR(x)$ and $IRR(R) := IRR(B)$.

Definition 1.1.13 Let $R := (B,==>)$ be a reduction system. Then $Y \in B$ is called a *normal form* of $x \in B$ (mod R) if $y \in [x]$ and y is irreducible.

Let $A \subset B$, $x \in B$; then

(a) $NORM(x) := \{ y \in [x] \mid y \text{ is irreducible} \}$

(b) $NORM(A) := \cup_{x \in A} NORM(x)$ and $NORM(B) := NORM(R)$.

Definition 1.1.14 A binary relation ==> subset of $B \times B$ is called *Noetherian* or else *well founded* if there is no infinite chain

$$b_0 ==> b_1 ==> b_2 ==> \ldots ==> b_i ==> b_{i+1} ==> \ldots$$

with $b_i \in B$. A reduction system $(B,==>)$ in which ==> is a Noetherian relation will be called a *terminating* reduction system.

Definition 1.1.15 Given a reduction system R := (B,==>) then ==>, as well as R, is called

(a) locally finite iff $\forall x \in B$: $\Delta(x)$ is finite;

(b) globally finite iff $\forall x \in B$: $\Delta^*(x)$ is finite.

Note that by König's Lemma [J.König, Einleitung in die Allgemeine Theorie der Algebraischen Größen. Teubner, Leipzig (1903).] a locally finite system R := (B,==>) is terminating iff R is globally finite and acyclic.

Definition 1.1.17 a reduction system R := (B,==>) is

(a) confluent iff $\forall x,y \in B$: $\langle x\rangle^* \cap \langle y\rangle^* \neq \phi$ implies $\Delta^*(x) \cap \Delta^*(y) \neq \phi$;

(b) locally confluent iff $\forall x,y \in B$: $\langle x\rangle \cap \langle y\rangle \neq \phi$ implies $\Delta^*(x) \cap \Delta^*(y) \neq \phi$;

(c) Church-Rosser iff for all $x,y \in B$: $[x] = [y]$ implies $\Delta^*(x) \cap \Delta^*(y) \neq \phi$.

[Book pp.172-173]

1. Preliminaries

... Some definitions and notation for Thue systems and their congruences are established here.

If Σ is a finite alphabet, then Σ^* is the free semigroup with identity e generated by Σ. If w is a string, then the *length* of w is denoted by $|w|$: $|e| = 0$, $|a| = 1$ for $a \in \Sigma$, and $|wa| = |w| + 1$ for $w \in \Sigma^*$, $a \in \Sigma$.

A *Thue system* S on a finite alphabet Σ is a subset of $\Sigma^* \times \Sigma^*$. Each pair in S is a *relation*. The *Thue congruence generated by S* is the reflexive transitive closure, $<-*->_S$, of the relation $<--->_S$ defined as follows: For any u, v such that $(u, v) \in S$ or $(v, u) \in S$ and any $x, y \in \Sigma^*$, $xuy <--->_S xvy$. Two strings w, z are *congruent (mod S)* if $w <-*->_S z$; the *congruence class of z (mod S)* is $[z]_S = \{ w \mid w <-*-> z \}$. (whenever possible the subscript S will be omitted.)

If S_1 and S_2 are Thue systems such that for all x, $x <-*->_{S_1} y$ implies $x <-*->_{S_2} y$, then S_1 refines S_2; if S_1 refines S_2 and S_2 refines S_1, then S_1 and S_2 are equivalent. Clearly S_1 refines S_2 if and only if for every $(u, v) \in S_1$, $u <-*->_{S_2} v$.

Let S be a system of relations.

(a) Write $x ---> y$ provided $x <---> y$ and $|x| > |y|$.
(b) Write $x \,|---|\, y$ provided $x <---> y$ and $|x| = |y|$.
(c) Write $x \,|---> y$ provided $x ---> y$ or $x \,|---|\, y$.

The reflexive transitive closure of ---> ($|--->$, $|---|$) is denoted -*-> (respectively, $|-*-|$, $|-*->$).

If S as a Thue system, then a string x is *irreducible* (mod S) if for all y, $x <---> y$ implies $|x| \leq |y|$, and is *minimal* if $x <-*-> y$ implies $|x| \leq |y|$.

A transformation $x -*-> y$ is a *reduction*.

...

2. Types of Thue Systems

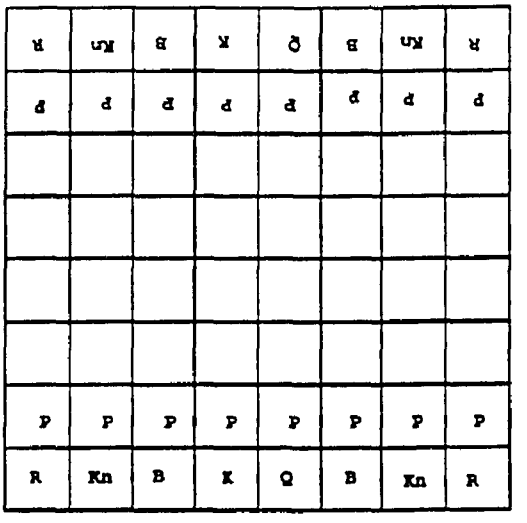
...

Let S be a Thue system.

(a) S is *Church-Rosser* if for all x, y, if $x <-*-> y$, then there exists a z such that $x -*-> z$ and $y -*-> z$ (see Figure 19a).
(b) S is *confluent* if for all w,x,y, if $w -*-> x$ and $w -*-> y$, then there exists a z such that $x -*-> z$ and $y -*-> z$ (see Figure 19b).

(c) *S* is *almost-confluent* if for all x, y, if x <-*-> y, then there exists g, h such that x -*-> g, y -*-> h, and g ¦-*-¦ h (see Figure 19c).
(d) *S* is *quasi-perfect* if for all w, x, y, if w ¦-*-> x and w ¦-*-> y, then there exists a z such that x ¦-*-> z and y ¦-*-> z.
(e) *S* is *preperfect* if for all x, y, if x <-*-> y, then there exists a z such that x ¦-*-> z and y ¦-*-> z (see Figure 19d).

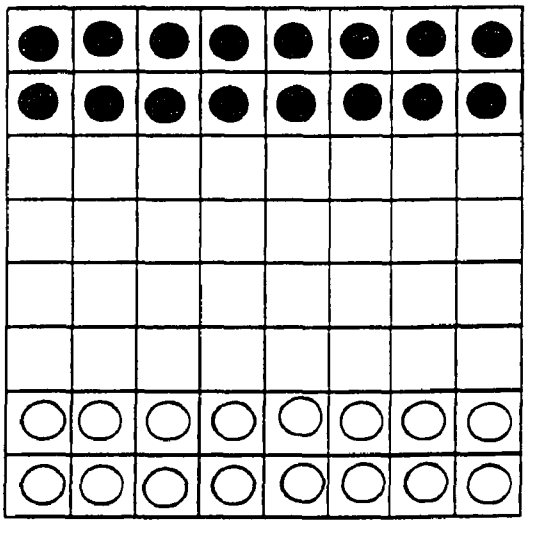
ILLUSTRATIONS



CHESS

CHESS AND CHECKERS

FIGURE 1



CHECKERS

THE POSITION (1, 2) IN WYTHOFF'S GAME WITH A = 1

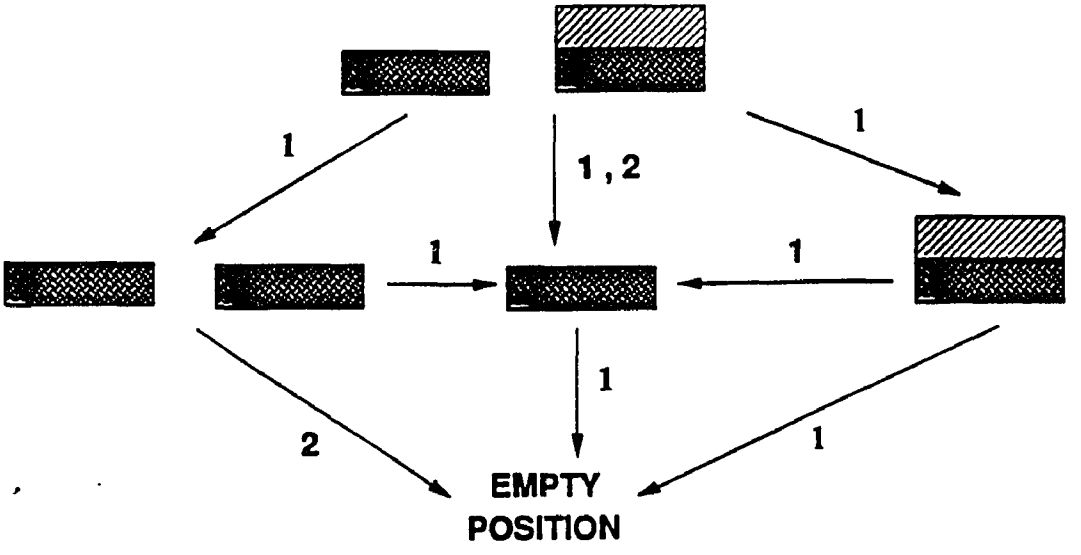
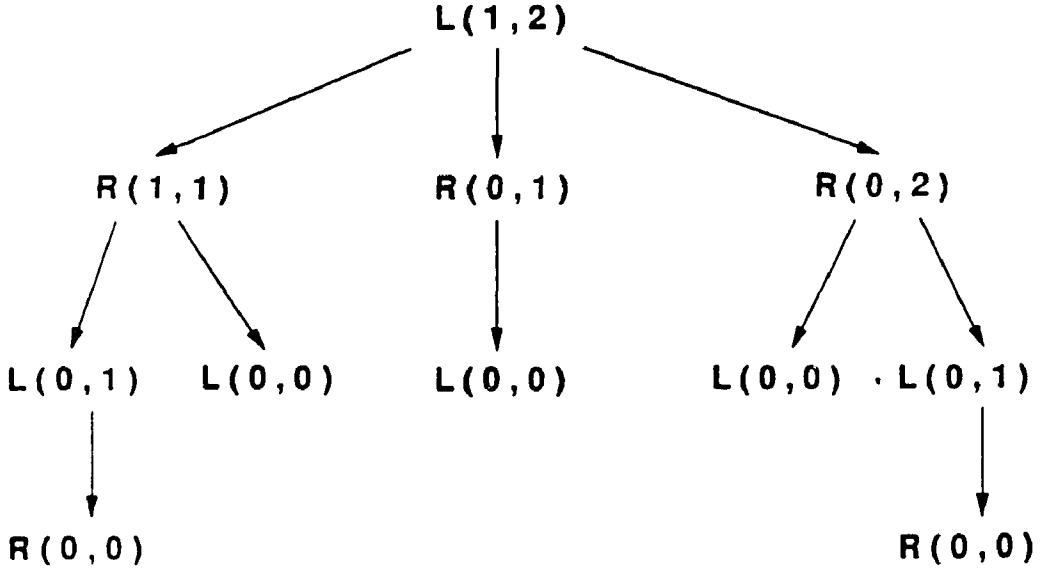
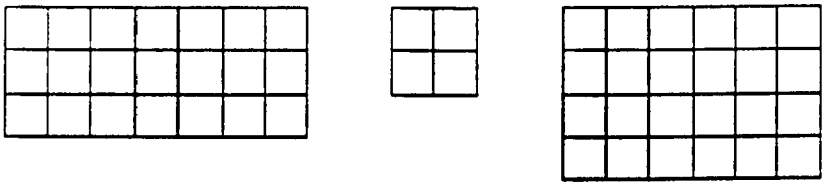


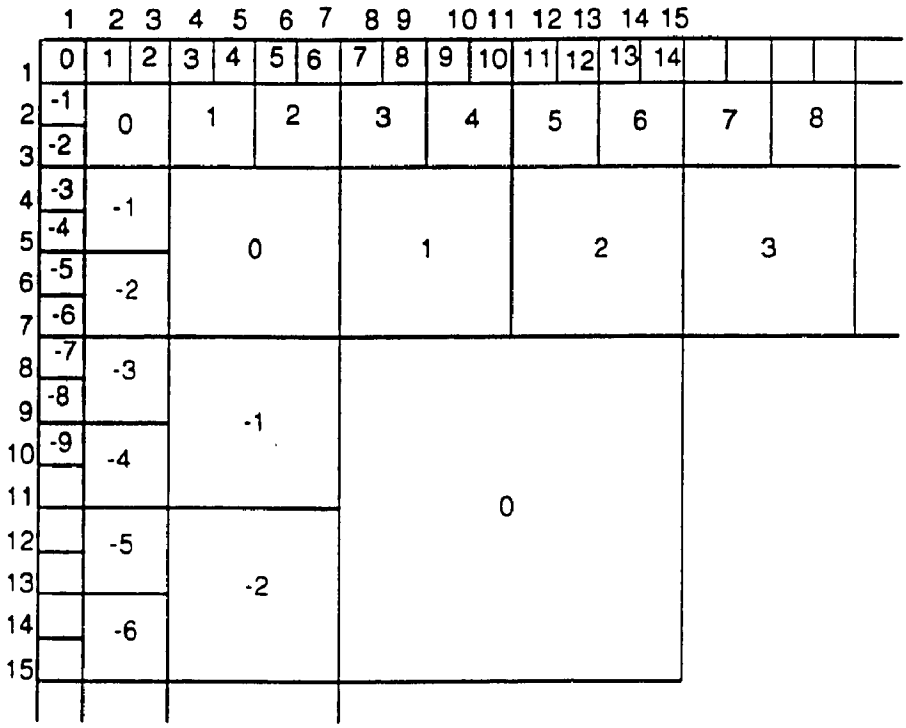
FIGURE 2 a The Graph of the Game

FIGURE 2 b The Game Tree associated with the position (1,2) with Left on move





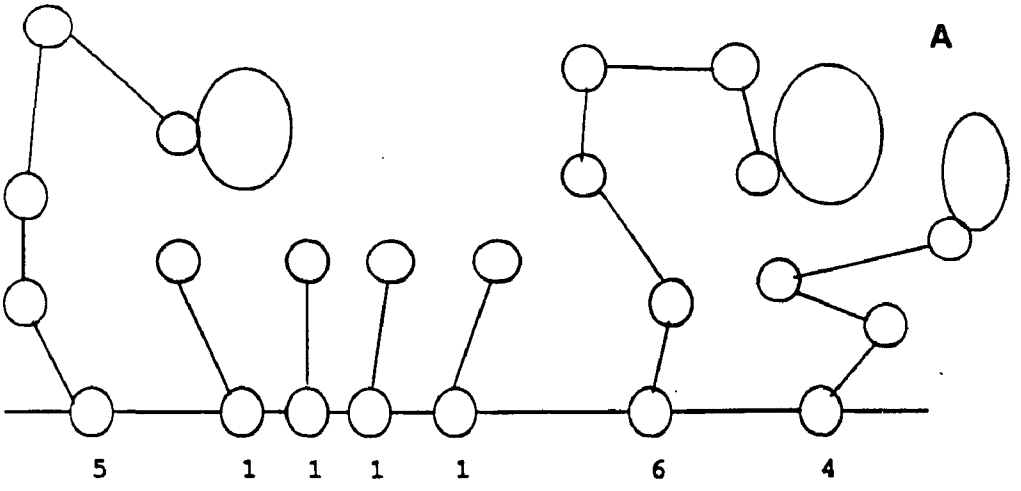
A. THREE PIECES READY TO BE CUT



B. VALUES OF RECTANGLES IN CUTCAKE

CUTCAKE

FIGURE 3 (after Figure 3 [BCG] p.26)



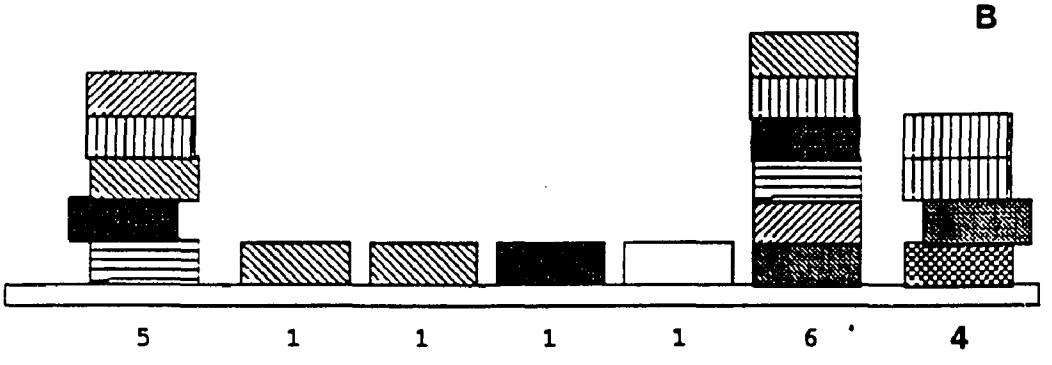
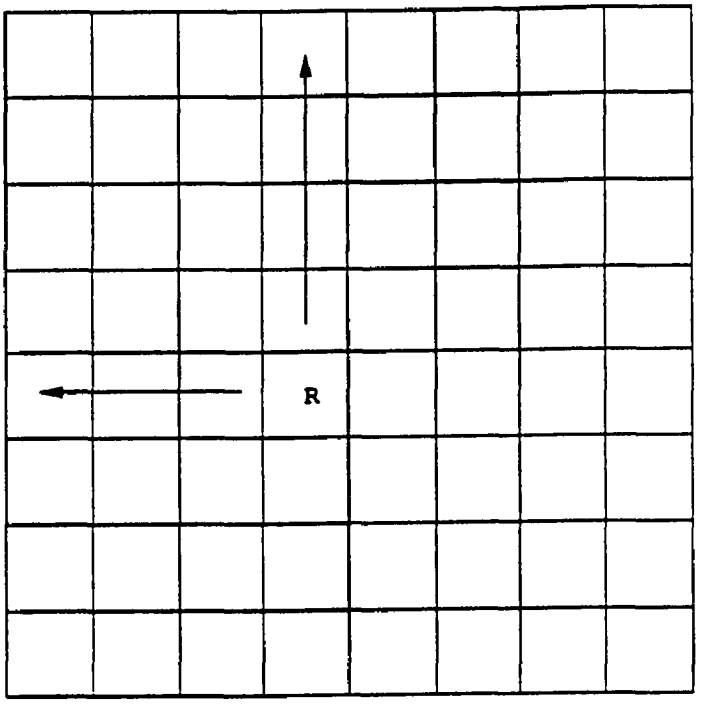


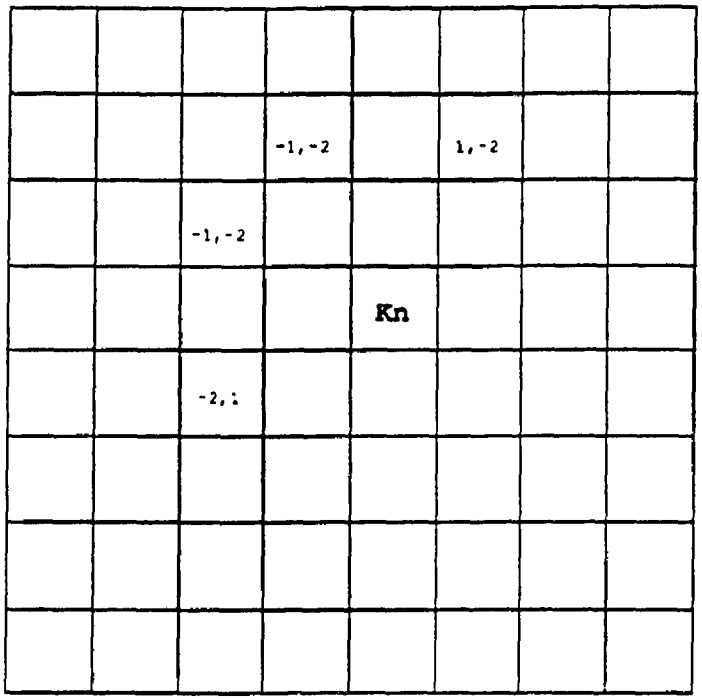
FIGURE 4 after [BCG] p.43



A ROOK AND ITS MOVES

THE ROOK'S GAME

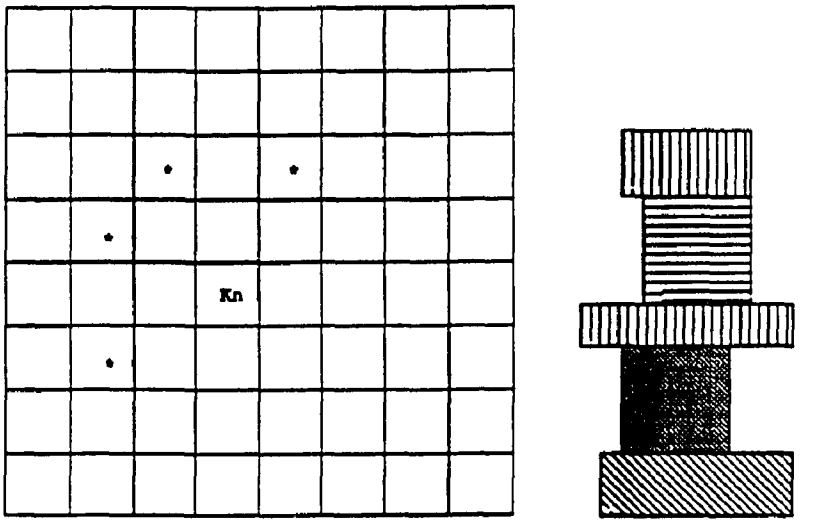
FIGURE 5



A KNIGHT AND ITS MOVES

KNIGHT

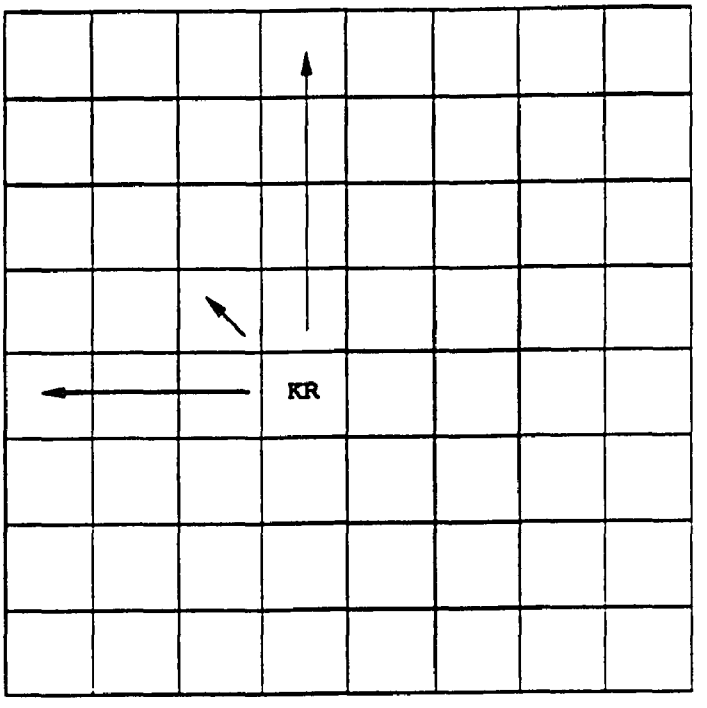
FIGURE 6



A POSITION IN WHITE KNIGHT

WHITE KNIGHT

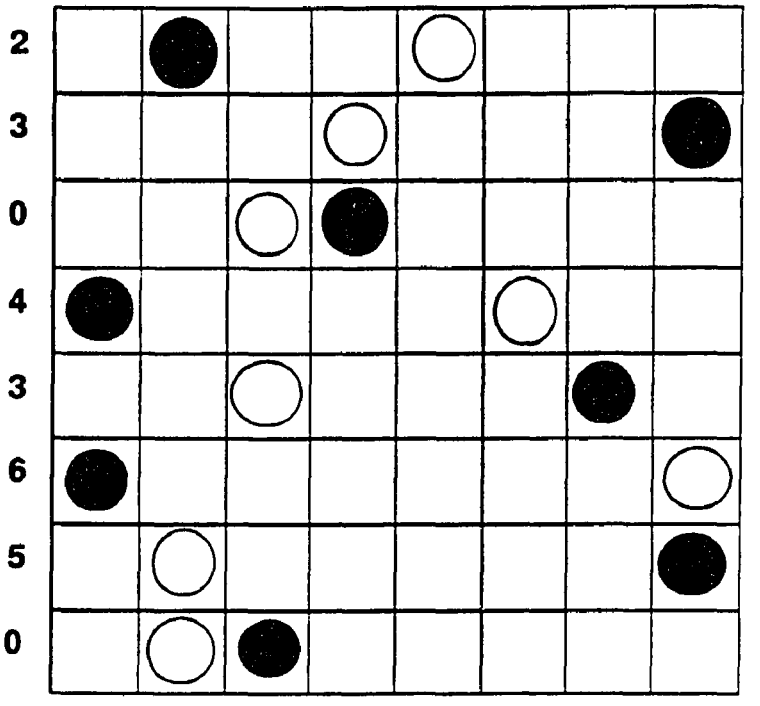
FIGURE 7



A KING-ROOK AND ITS MOVES

THE KING-ROOK GAME

FIGURE 8

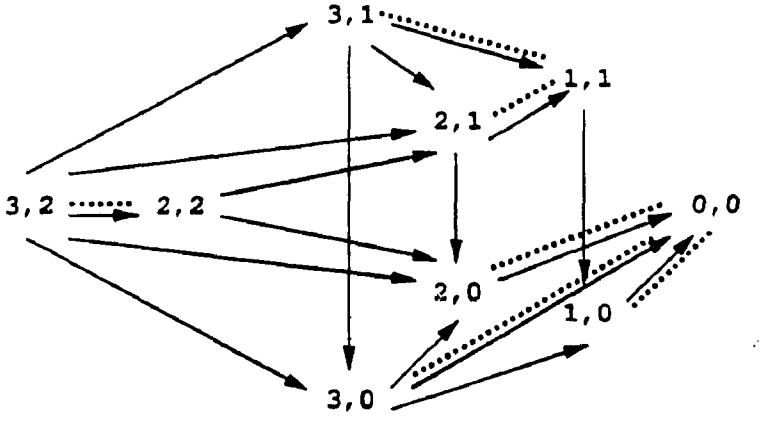


A POSITION IN NORTHCOTT'S GAME

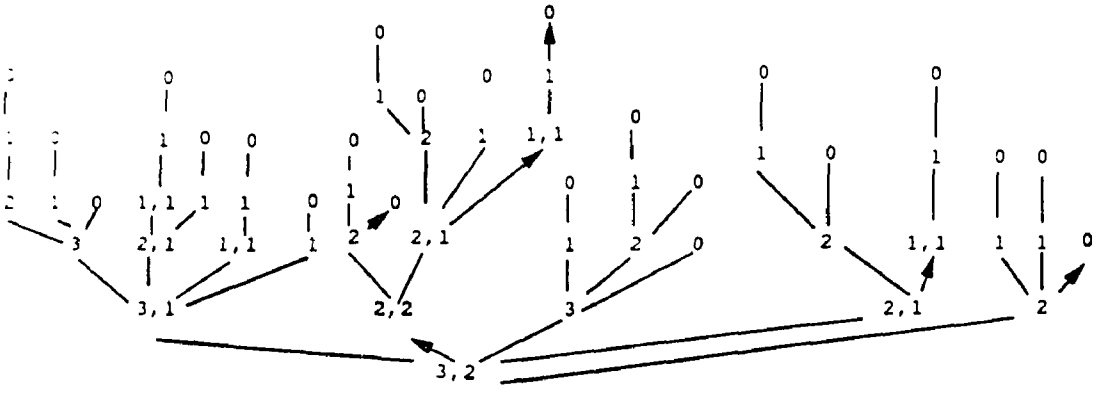
BCG P.56

FIGURE 9

FIGURE 10

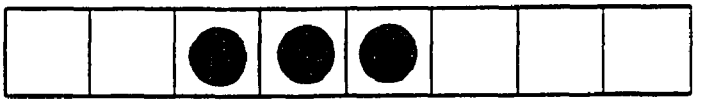


A The game graph for the Nim position {3,2}.
From the leftmost position the next player can win by adopting the strategy indicated by the ····· ⟶ double line.

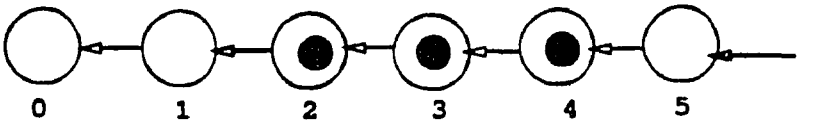


B The game tree for the same Nim position.

Reference: [CG] p.5, p.6

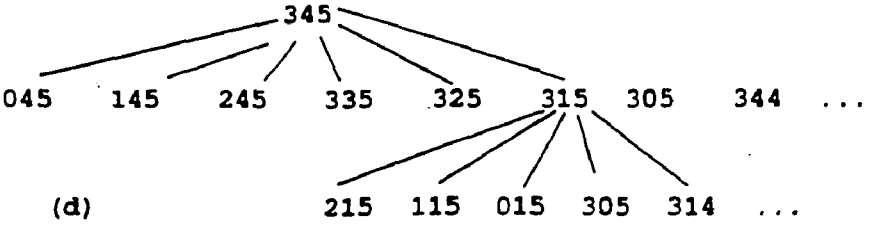


(a) NIMBLE



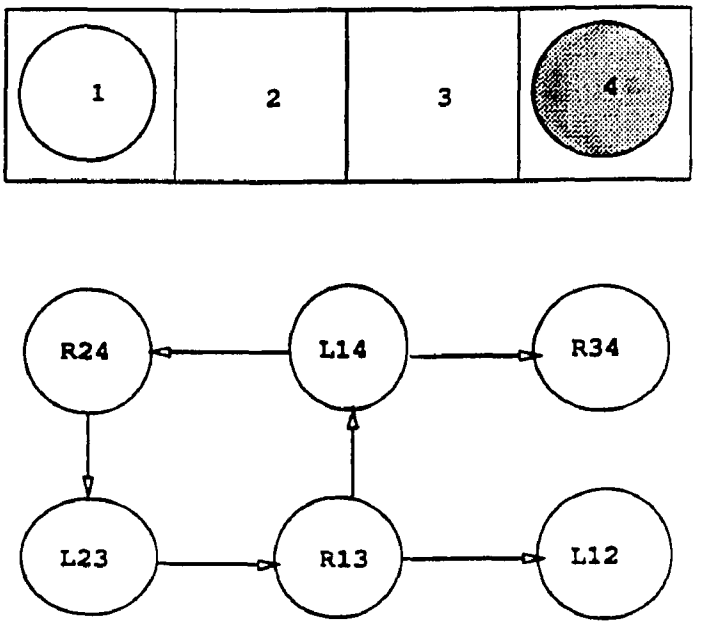
(b) NIMGEE

(c) NIM



(d)

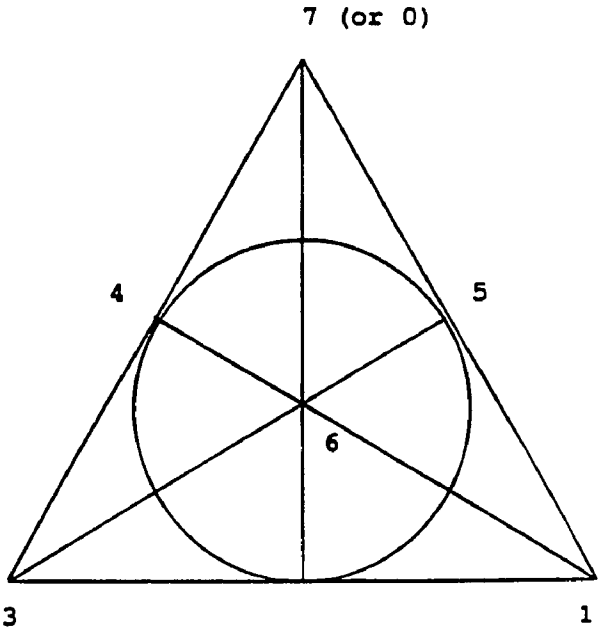
FIGURE 11



A GAME AND ITS GRAPH

MINI NORTHCOTT

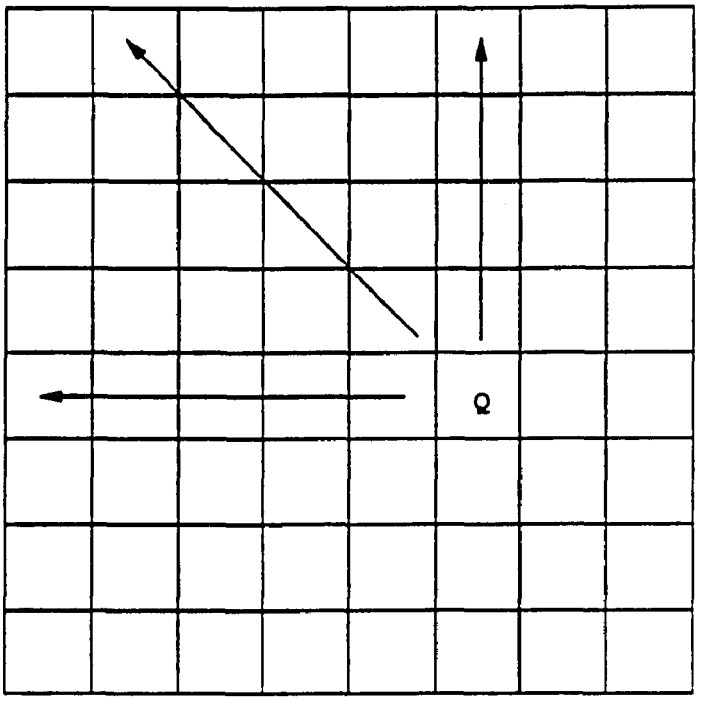
FIGURE 12



FANO'S FANCY ANTONIM FINDER

BCG P.461

FIGURE 13



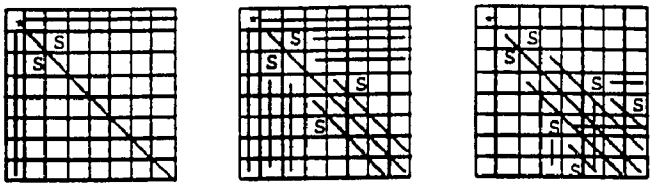
A QUEEN AND ITS MOVES

THE QUEEN'S GAME

FIGURE 14

A,B,C recursively define the first three safe pairs

D shows the first nine safe pairs



A B C

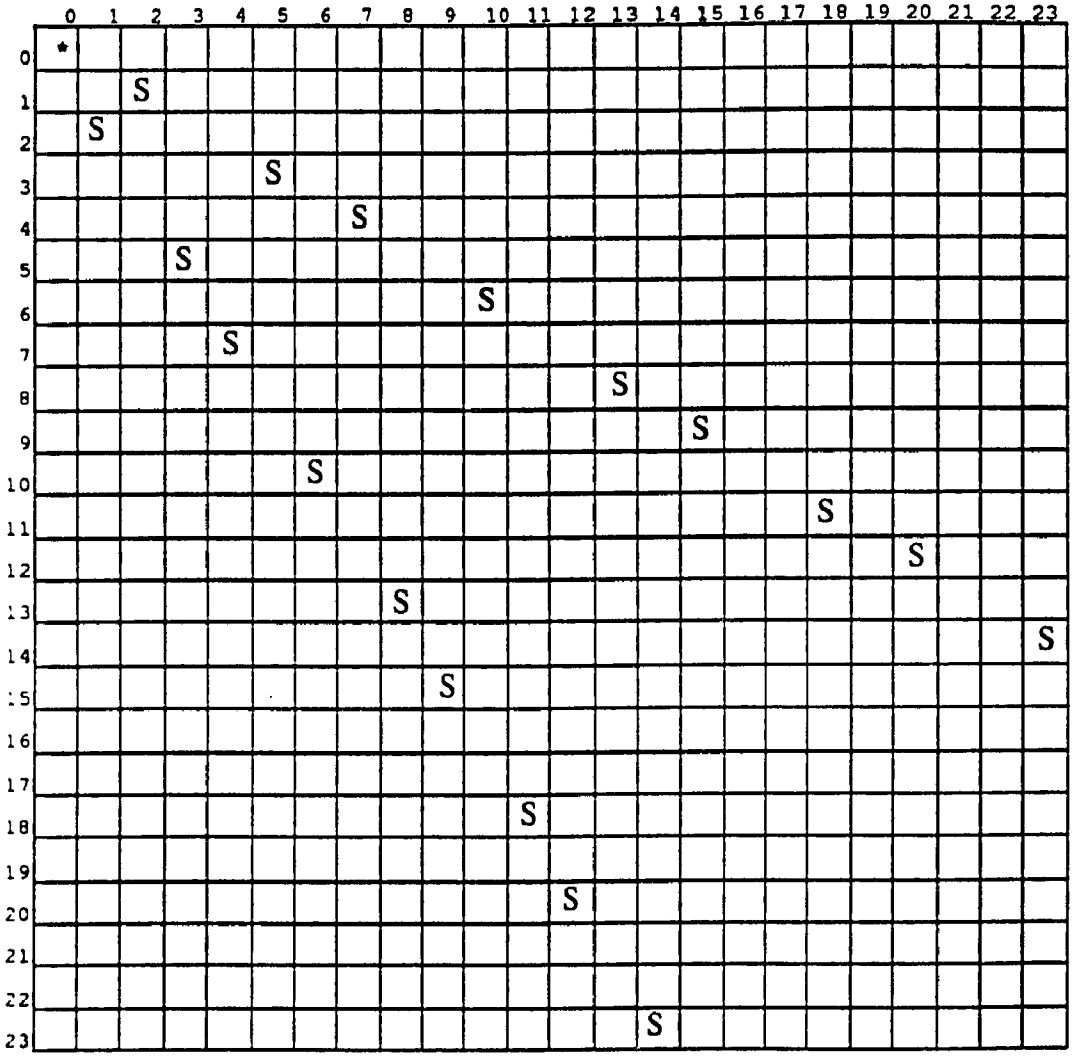
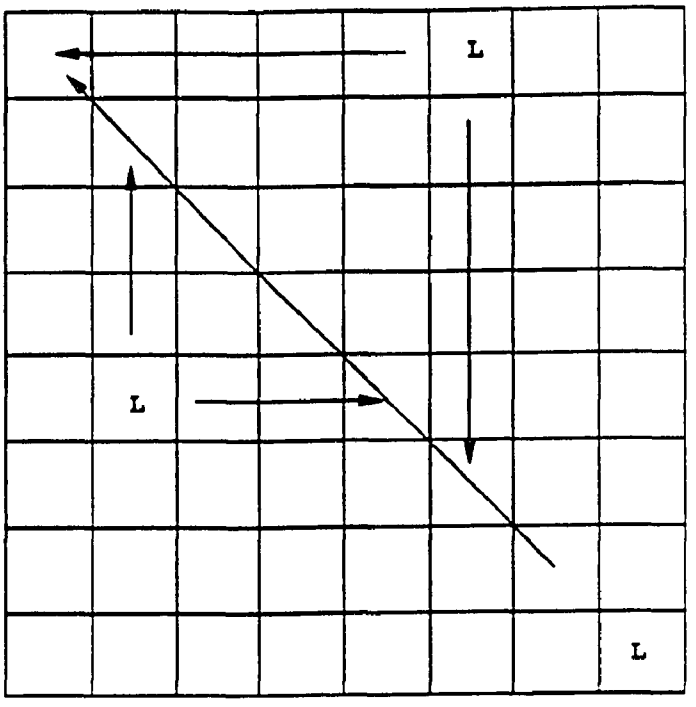


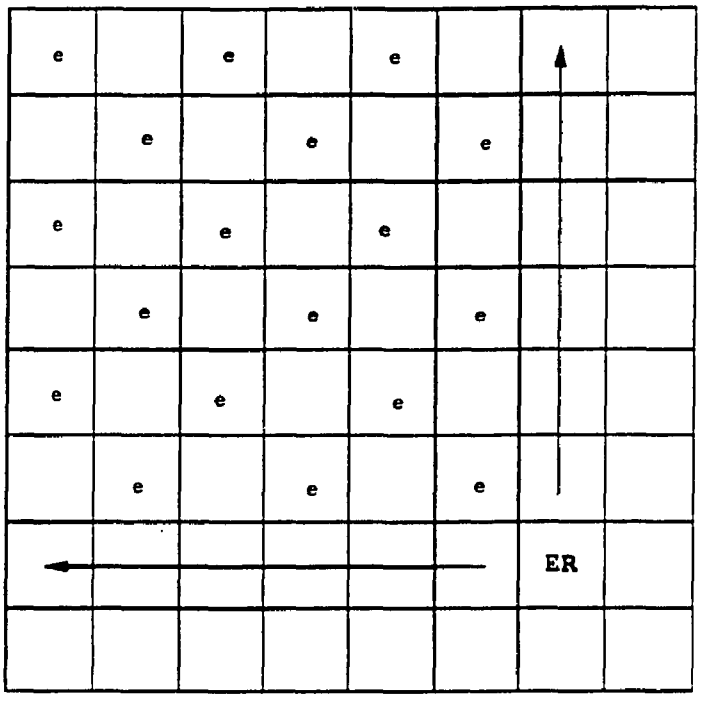
FIGURE 15 (after Figure 49 [GARDNER] p.105) D



THREE LATIN ROOKS AND THEIR MOVES

LATIN ROOK

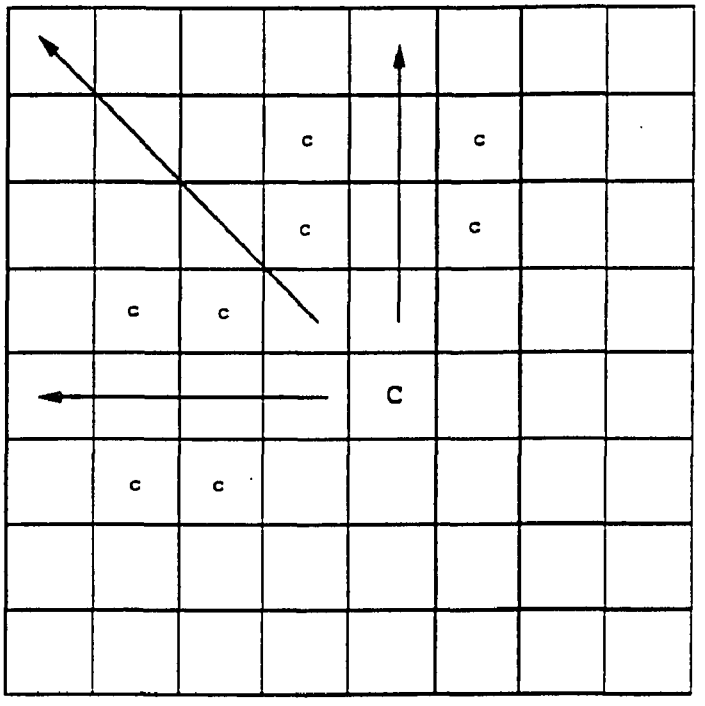
FIGURE 16



AN EVENROOK AND ITS MOVES

EVENROOK

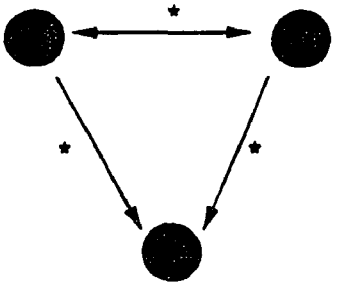
FIGURE 17



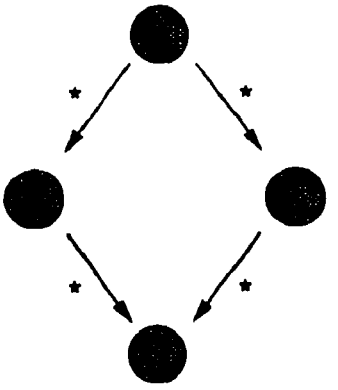
A COMMANDER AND ITS MOVES

COMMANDER

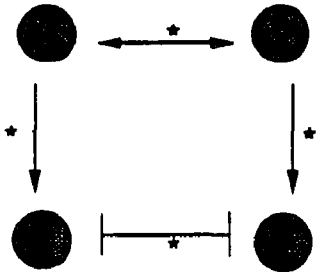
FIGURE 18



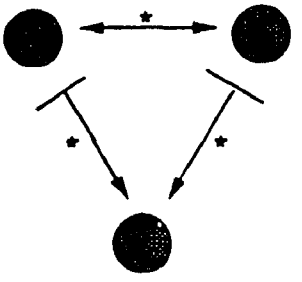
(a) Church-Rosser



(b) Confluent



(c) Almost-confluent



(d) Preperfect

BOOK P.173
FIGURE 19

SELECTED BIBLIOGRAPHY

[BCG] Elwin R. Berlekamp, John H. Conway, Richard K. Guy, Winning Ways for your mathematical plays, Volume 1: Games In General, Volume 2: Games In Particular, Academic Press, London, 1982.

[BOOK] Ronald V. Book, "Confluent and Other Types of Thue Systems", Journal of the Association for Computing Machinery, Vol. 29, No. 1, January 1982, pp. 171 - 182.

[Blass] Uri Blass and Aviezri S. Fraenkel, "The Sprague-Grundy Function for Wythoff's Game," Theoretical Computer Science, 75(3), 1990, pp. 311 - 333.

[CG] Combinatorial Games, Richard K. Guy, Editor, Vol. 43, Preceedings of Symposia in Applied Mathematics, American Mathematical Society, Providence, Rhode Island, 1991.

[ASF] Aviezri S. Fraenkel, "How to beat your opponent on three fronts," Amer. Math. Monthly, 89(1982), pp. 353 - 361.

[ASF2] Aviezri S. Fraenkel and Mordechai Lorberbom, "Nimhoff Games," Journal of Combinatorial Theory, Series A, Volume 58, Number 1, September 1991, pp. 1 - 25.

[ASF3] Aviezri S. Fraenkel, "Recreation and Depth in Combinatorial Games," The Lighter Side of Mathematics, Richard K. Guy and Robert E. Woodstock, Editiors, Proceedings of the Eugène Strens Memorial Conference on Recreational Mathematics and its History, The Mathematical Association of America, Washington, D.C., 1994.

[Gardner] Martin Gardner, Penrose Tiles To Trapdoor Ciphers, W.H. Freeman and Company, New York, 1989.

[Jantzen] Matthias Jantzen, Confluent String Rewriting, EATCS Monographs on Theoretical Computer Science:v.14, Springer-Verlag, Berlin, 1988.

[MAT] Hilbert's Tenth Problem, Yuri V. Matiyasevich, The MIT Press, Cambridge, Mass., 1993.

[Peterson] James L. Peterson, Petri Net Theory And The Modeling of Systems, Prentice Hall, Englewood Cliffs, 1981.

[Wilson] Robin J. Wilson, Introduction to Graph Theory, Second Edition, Longman Group Ltd, London, 1979.

WORKS NOT CITED

John D. Beardsley, The Mathematics of Games, Oxford University Press, Oxford, 1989.

H. S. M. Coxeter, "The Golden Section, Phyllotaxis, and Wythoff's Game", in Scripta Mathematica, 19, June/September 1953, pp. 135 - 143.

A.S. Fraenkel and I. Borosh, "A Generalization of Wythoff's Game", Journal of Combinatorial Theory (A), 15, 1973, pp. 175 - 191.

Aviezri S. Fraenkel, "Wythoff Games, Continued Fractions, Cedar Trees and Fibonacci Searches," Theoretical Computer Science, 29, 1984, pp. 49 - 73.