A COMPLETE CHARACTERIZATION OF NASH SOLUTIONS IN ORDINAL GAMES

by

Joshua M. Peterson

BS, Grove City College, 2003

Submitted to the Graduate Faculty of

the School of Engineering in partial fulfillment

of the requirements for the degree of

Master of Science in Electrical Engineering

University of Pittsburgh

2005

UNIVERSITY OF PITTSBURGH

SCHOOL OF ENGINEERING

This thesis was presented

by

Joshua M. Peterson

It was defended on

April 11, 2005

and approved by

Luis F. Chaparro, Associate Professor, Electrical Engineering Department

Ching-Chung Li, Associate Professor, Electrical Engineering Department

Marwan A. Simaan, Associate Professor, Electrical Engineering Department
Thesis Advisor

A COMPLETE CHARACTERIZATION OF NASH SOLUTIONS IN ORDINAL GAMES

Joshua M. Peterson, M.S.

University of Pittsburgh, 2005

The traditional field of cardinal game theory requires that the objective functions, which map the control variables of each player into a decision space on the real numbers, be well defined. Often in economics, business, and political science, these objective functions are difficult, if not impossible, to formulate mathematically. The theory of ordinal games has been described, in part, to overcome this problem.

Ordinal games define the decision space in terms of player preferences, rather than objective function values. This concept allows the techniques of cardinal game theory to be applied to ordinal games. Not surprisingly, an infinite number of cardinal games of a given size exist. However, only a finite number of corresponding ordinal games exist.

This thesis seeks to explore and characterize this finite number of ordinal games. We first present a general formula for the number of two-player ordinal games of an arbitrary size. We then completely characterize each 2x2 and 3x3 ordinal game based on its relationship to the Nash solution. This categorization partitions the finite space of ordinal games into three sectors, those games with a single unique Nash solution, those games with multiple non-unique Nash solutions, and those games with no Nash solution. This characterization approach, however, is not scalable to games larger than 3x3 due to the exponentially increasing dimensionality of the search space. The results for both 2x2 and 3x3 ordinal games are then codified in an algorithm capable of characterizing ordinal games of arbitrary size. The output of this algorithm,

implemented on a PC, is presented for games as large as 6x6. For larger games, a more powerful computer is needed. Finally, two applications of this characterization are presented to illustrate the usefulness of our approach.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to thank Dr. Luis Chaparro and Dr. Ching-Chung Li for devoting their time and effort to serve on my committee. I must express special thanks and sincere gratitude to Dr. Marwan Simaan, my thesis advisor. His experience, knowledge, inspiration, effort, and kindness made this thesis a reality. I am also indebted to my colleague, Robert Shapiro for his insightful editorial comments. Thanks also to my family and friends, especially to my parents. Their advice and experience was invaluable. None of this could have been possible with the love and support of my amazing wife. Thank you, Cindy. Finally, I must thank my Lord and God, who gave me the desire, knowledge, and persistence to complete this endeavor.

1.0 CARDINAL GAME THEORY AND NASH EQUILIBRIUM

The theory of games has been researched, discussed, and debated for some time. It was first introduced in the 1960's by Von Neumann and Morgenstern [1]. The theory of differential games [2] and specifically nonzero-sum differential games [3] emerged as an important area within the discipline of optimal control. Game theory has been applied to a wide field of disciplines, including electrical and mechanical control, aerospace engineering, economics, and even political science. Nash did pioneering work in the study of non-cooperative game theory [4]. Basar and Olsder also explored this field [5]. As the field of non-cooperative game theory evolved, the difficulty inherent in mathematically determining useful objective functions became clear. A method of ordinal optimization was presented by Ho, *et al* [6] in optimizing discrete event dynamic systems (DEDS). Likewise, instead of using payoff functions, Saaty developed the analytic hierarchy process (AHP) [7] and [8]. Brams [9] has explored the theory of preferential or ordinal rankings. Thanks to the work of Cruz and Simaan [10], the theory of two player ordinal games has recently emerged as an alternative theory that can be used in games where the objective function cannot be described mathematically. Building upon this work, this thesis presents a method of characterization for Nash solutions in ordinal games.

No matter what the field of application, optimization theory focuses on a well-defined system, the inputs which act on that system, and the outputs produced by that system. In general, the system defines an objective function,

$$J(x_1, \ldots, x_n) = C \, . \tag{1.1}$$

Note that the objective function may be a function of any, all, or none of either the inputs or the outputs. In general, the variables $x_1, \ldots, x_n$ are known as the control variables, since the controller can modify them to change the value of the objective function, represented by the real number C. The objective function maps the decision space (made up of the control variables) to a single real value. The goal of optimal control is to choose the control variables to minimize (or maximize) the value of the function J.

In some cases, the problem must be formulated in terms of multiple controllers, often with differing objectives. In other words, more than one player can affect the value of J through his control variables. Player 1 may have the goal of minimizing the value of J, while Player 2 may be interested in simultaneously maximizing J. We will explore the case where each player has his own objective function, say J_1 for Player 1 and J_2 for Player 2. Here both objective functions can be functions of the control variables for both players. As such Player 1 must make decisions about his own control variables, while understanding that decisions made by Player 2 will have an effect on his objective function J_1, and vice versa. If the players are in conflict with one another, this situation is called a non-cooperative two player game.

1.1 CARDINAL GAMES

Example 1.1: Suppose Player 1 has one control variable, x, which can take on values $x_1, x_2,$ or x_3. Likewise Player 2 has one control variable, y, which can take on values

$y_1, y_2,$ or y_3. The objective functions for players 1 and 2 are $J_1(x_i, y_j)$ and $J_2(x_i, y_j)$, respectively. The bimatrix game in Figure 1 shows the values of the objective functions for each combination of x_i, y_j.

		Player 2		
		y_1	y_2	y_3
Player 1	x_1	5.3 , 7.8	3.8 , 1.2	5.7 , 2.4
	x_2	9.6 , 3.1	8.7 , 4.0	3.5 , 2.1
	x_3	3.4 , 9.8	4.6 , 3.9	2.2 , 5.6

Figure 1: A 3x3 bimatrix game

The upper left cell of the table represents the values of the objective functions J_1 and J_2 when Player 1 chooses x_1 and Player 2 chooses y_1. In this case

$$J_1(x_1, y_1) = 5.3, \tag{1.2}$$

$$J_2(x_1, y_1) = 7.8. \tag{1.3}$$

Game theory is a wide field of study, so to correctly categorize games where the outcome is determined by the effect control variables have on objective functions, we will refer to games like that of Example 1.1 as cardinal games. We will soon find it beneficial to represent these cardinal games in terms of two separate matrices.

Example 1.2: For the cardinal game of Example 1.1, the following two matrices can be obtained by inspection.

$$J_1 = \begin{bmatrix} 5.3 & 3.8 & 5.7 \\ 9.6 & 8.7 & 3.5 \\ 3.4 & 4.6 & 2.2 \end{bmatrix} \tag{1.4}$$

$$J_2 = \begin{bmatrix} 7.8 & 1.2 & 2.4 \\ 3.1 & 4.0 & 2.1 \\ 9.8 & 3.9 & 5.6 \end{bmatrix} \tag{1.5}$$

The matrix J_1 represents the values of the objective function $J_1(x_i, y_j)$. Likewise the matrix J_2 represents the values of the objective function $J_2(x_i, y_j)$. So the value at the [1, 3] position in the matrix J_2 corresponds to the value

$$J_2(x_1, y_3) = 2.4. \tag{1.6}$$

1.2 THE NASH SOLUTION

Now that the game has been defined, and we have sampled a few important representations, can we determine the outcome of the game? In other words, what strategies should Player 1 and Player 2 use to achieve an optimal result? These questions have a number of different answers. The answers depend on the parameters of the game. For instance, can the two players cooperate with one another? Also, will both players always make the best choice, given the strategy of the other player? Can one player force the other to play in a given manner? For the duration of our discussion, we will assume the players either cannot cooperate, or they have opposing objectives, and therefore are not willing to cooperate. Also, we will assume both players are equal, i.e. neither player can forcibly exert his will over that of the other, so the game remains balanced. Since the game is defined in terms of minimization of objective functions and the value of both objective functions for any choice of control variables is known to both players, we will assume that each player will make the optimal decision. In other words, each player will choose the value of his control variable to minimize the value of his objective function in a given situation.

With these assumptions in mind, it was Nash [4] who first proposed an equilibrium point at which the game will eventually come to rest.

Definition 1.1: A Nash solution for a cardinal game involving two players is a set of values of the control variables $\{x_N, y_N\}$ that satisfies the following set of inequalities

$$J_1(x_N, y_N) \le J_1(x, y_N), \forall x \in X, \tag{1.7}$$

$$J_2(x_N, y_N) \le J_2(x_N, y), \forall y \in Y. \tag{1.8}$$

Here X is the set of all possible values of the control variable x and Y is the set of all possible values of the control variable y. Note in this solution, each player minimizes his objective function, given that the other player also uses a Nash strategy. This definition is illustrated by the following example.

Example 1.3: Suppose we are witnessing the game of Example 1.1. Both Player 1 and Player 2 are using Nash strategies, that is, they cannot cooperate and they both attempt to minimize their respective objective functions at all times. Suppose Player 2 begins the game by playing y_3, then Player 1 will undoubtedly play x_3, since

$$J_1(x_3, y_3) = 2.2 < J_1(x_2, y_3) = 3.5 < J_1(x_1, y_3) = 5.7. \tag{1.9}$$

Once Player 2 notices that Player 1 has used x_3, he will certainly change his strategy and play y_2, since

$$J_2(x_3, y_2) = 3.9 < J_2(x_3, y_3) = 5.6 < J_2(x_3, y_1) = 9.8. \quad (1.10)$$

Finally, Player 1 will respond to strategy y_2 from Player 2 by switching x_1, since

$$J_1(x_1, y_2) = 3.8 < J_1(x_3, y_2) = 4.6 < J_1(x_2, y_2) = 8.7. \quad (1.11)$$

After this move, Player 2 cannot produce a lower value of his objective function by changing to either y_1 or y_3, so he is content to stay with strategy y_2. At this point the game has reached an equilibrium state, where neither player is willing to change his strategy. Thus $\{x_1, y_2\}$ is the Nash solution for the game of Example 1.1 since it is the only $\{x_i, y_j\}$ pair that satisfies the inequalities (1.7) and (1.8).

The first choice of Player 2, namely y_3, was arbitrary. In this game, any initial choice by either player will lead to the same outcome. The game will come to rest with Player 1 choosing x_1 and Player 2 chosing y_2. So if both players have complete knowledge of the values of J_1 and J_2 shown in Figure 1, and both players always attempt to minimize their respective objective functions, then both players can immediately implement their respective choices indicated by the Nash solution. Under these assumptions, the result of the game will not change.

To this point we have not discussed the lack of player cooperation necessary to make the Nash solution useful. A classical game theory problem, the prisoners' dilemma, illustrates the need for this condition.

Example 1.4: Suppose that two prisoners are being held in separate cells. The judge meets with both of them individually, and informs each that he has been charged with a major crime that carries an eight year prison sentence, and a more minor crime, which carries a two year prison sentence. He has evidence to convict both of the minor crime. If, however, either prisoner divulges information that could implicate the other prisoner in the major crime, the prisoner who divulged the information will have his total sentence cut in half. The bimatrix for this game is given below.

<table>
<tr><td colspan="2"></td><td colspan="2">Prisoner 2</td></tr>
<tr><td colspan="2"></td><td>divulge</td><td>do not divulge</td></tr>
<tr><td rowspan="2">Prisoner 1</td><td>divulge</td><td>5 , 5</td><td>1, 10</td></tr>
<tr><td>do not divulge</td><td>10 , 1</td><td>2 , 2</td></tr>
</table>

Figure 2: The prisoners' dilemma

The Nash solution for this game states that both prisoners will in fact divulge information, ensuring that they each receive a five year prison sentence. If they could have communicated, and could trust each other, they could have agreed to not divulge any information, and serve only two years each. However that situation is subject to cheating, since either player could shorten his prison sentence by divulging information. Since they cannot trust each other, each acts in his own best interest and they will both serve five year sentences. So a lack of communication, or as is more often the case, a lack of trust between adversaries motivates the usefulness of the Nash solution technique.

We have shown that the games in Example 1.1 and Example 1.4 both have clear Nash solutions. However it is not difficult to imagine a game which does not have a Nash solution.

Example 1.5: Suppose we have the following cardinal game, which is a slight modification of the game in Figure 1.

		Player 2		
		y_1	y_2	y_3
Player 1	x_1	5.3 , 7.8	6.8 , 1.2	5.7 , 2.4
	x_2	9.6 , 3.1	8.7 , 4.0	9.4 , 2.1
	x_3	3.4 , 9.8	4.6 , 3.9	6.3 , 2.5

Figure 3: A cardinal game with no Nash solution

In this game, no combination of the control variables will satisfy the Nash inequalities (1.7) and (1.8). Suppose Player 2 starts by choosing y_2. Player 1 will then play x_3, followed by a choice of y_3 by Player 2, and then x_1 by Player 1. Player 2 will return to strategy y_2, and the sequence will repeat. So this game has no Nash solution.

Likewise it is possible to imagine a game with multiple Nash solutions. In other words, the equilibrium point of the game depends on the starting point of the game.

Example 1.6: Suppose we have the following cardinal game, another slight modification of the game in Figure 1.

		Player 2		
		y_1	y_2	y_3
Player 1	x_1	5.3 , 7.8	3.8 , 1.2	5.7 , 2.4
	x_2	9.6 , 3.1	8.7 , 4.0	3.5 , 2.1
	x_3	3.4 , 3.9	4.6 , 9.8	2.1 , 5.6

Figure 4: A cardinal game with multiple Nash solutions

If Player 2 begins by choosing y_3, the outcome will see Player 1 choose x_3 and Player 2 choose y_1. However if Player 2 begins instead by choosing y_2, Player 1 will immediately choose x_1, bringing the game to rest at a different point.

So given an arbitrary cardinal game, we can expect to charactorize it in one of three ways, based on the existence of zero, one, or more than one Nash solutions for the game.

Definition 1.2: A cardinal game is said to have a unique Nash solution if exactly one pair of values of the control variables $\{x_N, y_N\}$ satisfies inequalities (1.7) and (1.8). If more than one pair of values of the control variables $\{x_N, y_N\}$ satisfies the inequalities, the game is said to have multiple non-unique Nash solutions. If no pair of values of the control variables $\{x_N, y_N\}$ satisfies the inequalities, the game is said to have no Nash solution.

We have given examples of the existence of each of these three types of cardinal games for the 3x3 case. We will assume that cardinal games of all three types exist in general, for games of size 2x2 and greater.

2.0 ORDINAL GAME THEORY

Up to this point, we have assumed the values of the objective functions J_1 and J_2 are known for each combination of the control variables. Often this is the case. However, more often these values are difficult to calculate. Sometimes we cannot begin to formulate an effective objective function. Often in disciplines like economics, political science, and international affairs, far too many factors exist to be considered in an objective function. Cardinal game theory is unable to effectively handle many of these situations. Following the lead of Cruz and Simaan [10], we can use a different approach to solve these problems, allowing us to formulate a game and apply the principles of cardinal game theory. This different approach is known as ordinal game theory.

Suppose instead of determining objective functions for the players, we ask them to rate the possible outcomes of the game, in order of preference. We can assign numerical values to these preferences. To maintain consistency with the discussion above, we will assume the outcome a player most prefers will be rated 1, the second most preferred outcome will be rated 2, and so on, until all of the possible outcomes have been so ordered. Now the values of the objective functions shown in the matrix can be replaced by these preferential rankings, and the principles of cardinal game theory, most notably the Nash solution technique, can be applied.

Example 2.1: At the height of the Cold War, both the United States and Soviet Union were deadlocked in an arms race[1]. Certain factions in each country pushed their leaders to add more weapons, while others were in favor of reducing the size of each military. The ordinal game for this situation can be formulated as shown below.

		Soviet Strategy (Player 2)	
		Arm	Disarm
US Strategy (Player1)	Arm	3 ,3	1 , 4
	Disarm	4 , 1	2 , 2

Figure 5: The US-Soviet arms race

Certainly it would be difficult, if not impossible for even the most versed international scholars and mathematicians to determine suitable objective functions encompassing all of the factors under consideration by both the United States and the Soviet Union at the time. However, it is clear that the United States would most prefer that it arm and the Soviet Union disarm. The second choice for the United States would have both countries disarming and ensuring peace. The third best option for the United Statess would entail both countries arming, so that if the Soviet Union attacked the United States, the United States could defend itself. In the worst-case scenario, United States policy makers would blunder by disarming while the Soviet Union builds

[1] This practical application of game theory was presented in a Newsweek article entitled "The Games Scholars Play" by authors Sharon Begley and David Grant on September 6, 1982.

up its military. The preferences for the Soviet Union are similar. Since neither country can trust the other, both countries will undoubtedly act in their own best interest. This formulation of the game presents us with values for all of the possible combinations of choices, so we can apply the Nash solution techniques. Although it may be better if both countries disarm, so that war is not an option, that outcome will not occur. The Nash solution to this game illustrates that both countries will attempt to arm, which, as history testifies, is precisely what happened.

2.1 CONVERSION FROM CARDINAL GAMES TO ORDINAL GAMES

Example 2.1 is at least one case where ordinal game theory greatly simplified what would be an extremely complex, if not impossible, problem in cardinal game theory. In fact, any cardinal game can be converted to an equivalent ordinal game. By equivalent, we mean the characterization of the cardinal game based on the existence of Nash solutions, as well as the location of any Nash solutions, will be preserved during this conversion. This conversion can be accomplished using the following definition.

Definition 2.1: Suppose we have an $n \times m$ matrix M made up of real numbers. The associated rank-ordered matrix M^o is the $n \times m$ matrix where each number $\left[m_{ij}^o\right]$ is the rank of the number $\left[m_{ij}\right]$ in the set $\left\{\left[m_{ij}\right], \text{for } i = 1, \ldots, n \text{ and } j = 1, \ldots, m\right\}$. If two entries in M have an equal rank, they are assigned the same value in M^o.

Example 2.2: The following are some sample matrices and their associated rank-ordered matrices.

$$a = \begin{bmatrix} 2.4 & 1.3 & 9.8 & 5.6 \end{bmatrix} \qquad a^o = \begin{bmatrix} 2 & 1 & 4 & 3 \end{bmatrix}$$

$$K = \begin{bmatrix} 4.5 & 3.2 \\ 2.0 & 1.9 \\ 8.0 & 3.7 \end{bmatrix} \qquad K^o = \begin{bmatrix} 5 & 3 \\ 2 & 1 \\ 6 & 4 \end{bmatrix}$$

$$T = \begin{bmatrix} 3.4 & 8.7 & 5.2 \\ 7.6 & 4.3 & 8.7 \\ 1.7 & 2.0 & 1.3 \end{bmatrix} \quad T^o = \begin{bmatrix} 4 & 8 & 6 \\ 7 & 5 & 8 \\ 2 & 3 & 1 \end{bmatrix}$$

Example 2.3: The matrices from Example 1.2, J_1 and J_2, which correspond to the game in Figure 1, have the following associated rank-ordered matrices, J_1^o and J_2^o

$$J_1^o = \begin{bmatrix} 6 & 4 & 7 \\ 9 & 8 & 3 \\ 2 & 5 & 1 \end{bmatrix} \qquad J_2^o = \begin{bmatrix} 8 & 1 & 3 \\ 4 & 6 & 2 \\ 9 & 5 & 7 \end{bmatrix}.$$

Example 2.4: The cardinal game from Figure 1 can then be converted to an ordinal game. The cardinal game of Figure 1 is repeated here, followed by its corresponding ordinal game.

		Player 2		
		y_1	y_2	y_3
Player 1	x_1	5.3 , 7.8	3.8 , 1.2	5.7 , 2.4
	x_2	9.6 , 3.1	8.7 , 4.0	3.5 , 2.1
	x_3	3.4 , 9.8	4.6 , 3.9	2.2 , 5.6

Figure 6: The cardinal game of Figure 1

		Player 2		
		y_1	y_2	y_3
Player 1	x_1	6 , 8	4 , 1	7 , 3
	x_2	9 , 4	8 , 6	3 , 2
	x_3	2 , 9	5 , 5	1 , 7

Figure 7: The cardinal game of Figure 1 converted to an ordinal game

As in Example 1.3, the Nash solution remains $\{x_1, y_2\}$. We have neither introduced nor eliminated any Nash solutions by converting the cardinal game into the ordinal game, so the above Nash solution is still a unique Nash solution.

Note that, like cardinal games, ordinal games can have more than one Nash solution. Take for example the dating game.

Example 2.5: Suppose Lynn and Michael are attempting to decide what movie to watch on a Friday night. Lynn would rather see the romantic comedy, while Michael would prefer an action movie. If asked to rate their respective preferences, Lynn and Michael would fill out the table below.

Table 1: Lynn’s and Michael’s preferences

Movie Option	Lynn's Preference	Michael's Preference
Both see the romantic comedy	1	2
Both see the action film	2	1
Lynn sees the romantic comedy, Michael sees the action film	3	3
Lynn sees the action film, Michael sees the romantic comedy	4	4

Their situation can be described by the game below.

		Michael	
		Romantic Comedy	Action
Lynn	Romantic Comedy	1 , 2	3 , 3
	Action	4 , 4	2 , 1

Figure 8: The dating game

This game actually has two Nash solutions. If Lynn first refuses to see the action film, Michael will undoubtedly see the romantic comedy with her. Likewise if Michael first refuses to see the romantic comedy, Lynn will likely go with him to see the action film. So in this case, the outcome of the game is uncertain. In fact, the player who acts first has an advantage, and can control which of the two Nash solutions occurs.

2.2 CONVERSION FROM ORDINAL GAMES TO MINIMAL ORDINAL GAMES

After converting the cardinal game of Figure 1 into the ordinal game of Figure 7, we can simplify the game even more, and obtain the Nash solution more quickly by implementing row and column rank-ordering.

Definition 2.2: Suppose we have an $n \times m$ matrix K. If we say

$$K = \left[k_{c_1} \vdots k_{c_2} \vdots \cdots \vdots k_{c_m} \right] \tag{2.1}$$

where k_{c_i} are column vectors representing the columns of K, we can generate the column rank-ordered matrix K^{co} by replacing each column of K with its corresponding rank-ordered column vector such that

$$K^{co} = \left[k^o_{c_1} \vdots k^o_{c_2} \vdots \cdots \vdots k^o_{c_m} \right]. \tag{2.2}$$

Example 2.6: This example illustrates the process of generating a column rank-ordered matrix using matrix T from Example 2.2.

$$T = \begin{bmatrix} 3.4 & 8.7 & 5.2 \\ 7.6 & 4.3 & 8.7 \\ 1.7 & 2.0 & 1.3 \end{bmatrix} \quad T^o = \begin{bmatrix} 4 & 8 & 6 \\ 7 & 5 & 8 \\ 2 & 3 & 1 \end{bmatrix} \quad T^{co} = \begin{bmatrix} 2 & 3 & 2 \\ 3 & 2 & 3 \\ 1 & 1 & 1 \end{bmatrix}$$

Definition 2.3: Suppose we have an $n \times m$ matrix K. If we say

$$K = \begin{bmatrix} k_{r_1} \\ \cdots \\ k_{r_2} \\ \cdots \\ \vdots \\ \cdots \\ k_{r_n} \end{bmatrix} \tag{2.3}$$

where k_{r_i} are row vectors representing the rows of K, we can generate the row rank-ordered matrix K^{ro} by replacing each row of K with its corresponding rank-ordered row vector such that

$$K^{ro} = \begin{bmatrix} k^o_{r_1} \\ \cdots \\ k^o_{r_2} \\ \cdots \\ \vdots \\ \cdots \\ k^o_{r_n} \end{bmatrix}. \tag{2.4}$$

Example 2.7: This example illustrates the process of generating a row rank-ordered matrix using matrix T from Example 2.2.

$$T = \begin{bmatrix} 3.4 & 8.7 & 5.2 \\ 7.6 & 4.3 & 8.7 \\ 1.7 & 2.0 & 1.3 \end{bmatrix} \quad T^o = \begin{bmatrix} 4 & 8 & 6 \\ 7 & 5 & 8 \\ 2 & 3 & 1 \end{bmatrix} \quad T^{ro} = \begin{bmatrix} 1 & 3 & 2 \\ 2 & 1 & 3 \\ 2 & 3 & 1 \end{bmatrix}$$

The Nash inequalities (1.7) and (1.8) actually apply only to one row or one column of the game matrix at a time. Hence we can use row and column rank-ordered matrices to simplify our search for Nash solutions.

Example 2.8: Using the ordinal game from Example 2.3 and Example 2.4, we can implement row and column rank-ordering to simplify the problem. First we find the column rank-ordered matrix associated with J_1^o and the row rank-ordered matrix associated with J_2^o.

$$J_1^{co} = \begin{bmatrix} 2 & 1 & 3 \\ 3 & 3 & 2 \\ 1 & 2 & 1 \end{bmatrix} \tag{2.5}$$

$$J_2^{ro} = \begin{bmatrix} 3 & 1 & 2 \\ 2 & 3 & 1 \\ 3 & 1 & 2 \end{bmatrix} \tag{2.6}$$

The matrix for Player 1, J_1^o, needs to be column rank-ordered. For any given choice made by Player 2, Player 1 will rank his preferences, and choose to play the strategy with the rank of 1. So if Player 2 selects y_1, Player 1 will use strategy x_3. Likewise if Player 2 chooses y_2, Player 1 will respond with x_1, and so on. The matrix for Player 2 needs to be row rank-ordered for similar reasons. If we formulate the game bimatrix using J_1^{co} and J_2^{ro} we have the following game.

		Player 2		
		y_1	y_2	y_3
Player 1	x_1	2 , 3	1 , 1	3 , 2
	x_2	3 , 2	3 , 3	2 , 1
	x_3	1 , 3	2 , 1	1 , 2

Figure 9: The minimal ordinal game from Figure 7

Since we can now be sure that each player will always choose the strategy indicated by a 1, we can quickly see by inspection that the unique Nash solution for this game is $\{x_1, y_2\}$. In other words, any time we recognize the pattern {1,1} in a given entry, we can be sure that entry corresponds to a Nash solution for the game.

Definition 2.4: We will refer to ordinal games that have been column and row rank-ordered as minimal ordinal games.

You may have noticed some similarities between prisoners' dilemma (Example 1.4) and the Cold War arms race (Example 2.1). If we apply column and row rank-ordering to both games, it becomes clear that they are actually the same game.

Example 2.9: The game below is the minimal ordinal game for both the prisoners' dilemma and the Cold War arms race.

		Player 2	
		y_1	y_2
Player 1	x_1	1 , 1	1 , 2
	x_2	2 , 1	2 , 2

Figure 10: Minimal ordinal game for the prisoners' dilemma and the Cold War arms race

We can begin now to see the power of ordinal games, and specifically minimal ordinal games. In this case, two 2x2 ordinal games, each describing very different situations, can both be represented by one minimal ordinal game, without any change to the characterization of the games based on their Nash solutions. Each ordinal game encapsulates an infinite continuum of cardinal games, and each minimal ordinal game encapsulates a finite number of ordinal games. The following example illustrates a method of representing the former of these two concepts.

Example 2.10: Suppose an international scholar determined the objective functions to describe the strategies of both the United States and the Soviet Union during the Cold War arms race. The cardinal game, using the scholar's objective functions, is given below.

		USSR Strategy	
		Arm	Disarm
US Strategy	Arm	6.9 , 7.6	2.5 , 9.3
	Disarm	8.8 , 0.4	3.7 , 3.2

Figure 11: A cardinal game representation of the Cold War arms race

Since we defined the situation as an ordinal game in Figure 5, we can replace each of the values in the game of Figure 11 with a range of values without affecting the outcome of the game.

		USSR Strategy	
		Arm	Disarm
US Strategy	Arm	(5.3,7.9] , (5.4,8.3]	[0,3.1] , (8.3,∞)
	Disarm	(7.9,∞) , [0, 2.0]	(3.1,5.3] , (2.0,5.4]

Figure 12: The cardinal game of Figure 11 expressed using a range of values

Thanks to this simplification, the scholar is now free to modify the objective functions with confidence, knowing that he will not change the outcome of the game so long as the values of the objective functions in each case remain within the specified ranges. If the value of one of the objective functions does stray outside one of the given ranges, that may serve as a warning about the effectiveness of the objective function. The objective function should adhere to the simple dynamics of the game defined in Figure 5, since the preference rankings in the ordinal game represent each player's final objective. Figure 12 is a representation of the infinite continuum of cardinal games represented by the ordinal game of Figure 5.

3.0 2X2 ORDINAL GAMES

3.1 A GENERAL 2X2 ORDINAL GAME

To better understand the simplifications possible when we implement column and row rank-ordering to convert ordinal games into minimal ordinal games, we will first explore the simplest non-trivial case, 2x2 ordinal games. Both Brams [9] and Rapoport, *et al.* [11] recognize 78 different 2x2 ordinal games. Rapoport, *et al.* further categorize these games into 24 groups. Based on our Nash solution characterization in Definition 1.2, we will show that only 16 different 2x2 ordinal games exist. To understand this characterization, we need to imagine a general 2x2 ordinal game. We have already demonstrated that at least two specific 2x2 ordinal games correspond to the same minimal ordinal game (the prisoners' dilemma and the Cold War arms race). Suppose we have the following general 2x2 ordinal game.

		Player 2	
		y_1	y_2
Player 1	x_1	a , w	c , y
	x_2	b , x	d , z

Figure 13: A general 2x2 ordinal game

Here the values a,b,c,d and w,x,y,z represent the unknown preferences of Player 1 and Player 2, respectively. Without knowing any of these values, what can we determine about this game? If we allow the expression $P(s,t)$ to represent the number of possible permutations of s objects in t places we can determine the total number of different ways the general 2x2 ordinal game of Figure 13 can be played. Since Player 1 can assign any of four preference values (1, 2, 3, or 4) to the variables a,b,c,d , he has a total of

$$P(4,4) = 4! = 24 \qquad (3.1)$$

possible strategies. Likewise Player 2 has 24 possible strategies. Since each player can assign his preferences independently of the other player, the game of Figure 13 can be played in $24^2 = 576$ different ways. In other words, only 576 possible 2x2 ordinal games exist. So, any of the infinite number of cardinal games can be converted into one of the 576 possible ordinal games.

3.2 MINIMAL ORDINAL GAME SIMPLIFICATION

The conversion from cardinal games to ordinal games has proven fruitful. However can we perform the same analysis for the conversion from ordinal games to minimal ordinal games? Suppose we have the matrices J_1 for Player 1 and J_2 for Player 2 as follows

$$J_1 = \begin{bmatrix} a & c \\ b & d \end{bmatrix} \text{ and } J_2 = \begin{bmatrix} w & y \\ x & z \end{bmatrix}.$$

If we find the column-rank ordered matrix J_1^{co} and the row rank-ordered matrix J_2^{ro}, we have effectively created a general minimal ordinal game, based on the ordinal game in Figure 13. It is easy to list all of the possible ways the matrices J_1^{co} and J_2^{ro} can occur.

$$J_{1A}^{co} = \begin{bmatrix} 1 & 1 \\ 2 & 2 \end{bmatrix} \text{ or } J_{1B}^{co} = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix} \text{ or } J_{1C}^{co} = \begin{bmatrix} 2 & 2 \\ 1 & 1 \end{bmatrix} \text{ or } J_{1D}^{co} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$$

$$J_{2A}^{ro} = \begin{bmatrix} 1 & 2 \\ 1 & 2 \end{bmatrix} \text{ or } J_{2B}^{ro} = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix} \text{ or } J_{2C}^{ro} = \begin{bmatrix} 2 & 1 \\ 2 & 1 \end{bmatrix} \text{ or } J_{2D}^{ro} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}$$

So Player 1 and Player 2 each have only four strategies, not twenty-four strategies, meaning the total number of possible minimal ordinal games is only $4^2 = 16$. Since we showed in Chapter 2 that the simplification from ordinal games to minimal ordinal games preserves all Nash solutions, we can be confident that throughout this simplification we have lost no information about the Nash solution characterization of the game of Figure 13. Now instead of dealing with 576 possible 2x2 ordinal games, we really have to consider only 16 2x2 minimal ordinal games.

We can easily list all 16 of those minimal ordinal games. The 16 minimal ordinal games have been labeled according to their relationship to the column and row rank-ordered matrices J_1^{co} and J_2^{ro}. So the game labeled 1_B,2_A corresponds to the game where the players use the strategies J_{1B}^{co} and J_{2A}^{ro}. Note that the Nash solutions have been highlighted in bold type.

$1_A,2_A$		P_2	
		y_1	y_2
P_1	x_1	1,1	1,2
	x_2	2,1	2,2

$1_A,2_B$		P_2	
		y_1	y_2
P_1	x_1	1,1	1,2
	x_2	2,2	2,1

$1_A,2_C$		P_2	
		y_1	y_2
P_1	x_1	1,2	1,1
	x_2	2,2	2,1

$1_A,2_D$		P_2	
		y_1	y_2
P_1	x_1	1,2	1,1
	x_2	2,1	2,2

$1_B,2_A$		P_2	
		y_1	y_2
P_1	x_1	1,1	2,2
	x_2	2,1	1,2

$1_B,2_B$		P_2	
		y_1	y_2
P_1	x_1	1,1	2,2
	x_2	2,2	1,1

$1_B,2_C$		P_2	
		y_1	y_2
P_1	x_1	1,2	2,1
	x_2	2,2	1,1

$1_B,2_D$		P_2	
		y_1	y_2
P_1	x_1	1,2	2,1
	x_2	2,1	1,2

$1_C,2_A$		P_2	
		y_1	y_2
P_1	x_1	2,1	2,2
	x_2	1,1	1,2

$1_C,2_B$		P_2	
		y_1	y_2
P_1	x_1	2,1	2,2
	x_2	1,2	1,1

$1_C,2_C$		P_2	
		y_1	y_2
P_1	x_1	2,2	2,1
	x_2	1,2	1,1

$1_C,2_D$		P_2	
		y_1	y_2
P_1	x_1	2,2	2,1
	x_2	1,1	1,2

$1_D,2_A$		P_2	
		y_1	y_2
P_1	x_1	2,1	1,2
	x_2	1,1	2,2

$1_D,2_B$		P_2	
		y_1	y_2
P_1	x_1	2,1	1,2
	x_2	1,2	2,1

$1_D,2_C$		P_2	
		y_1	y_2
P_1	x_1	2,2	1,1
	x_2	1,2	2,1

$1_D,2_D$		P_2	
		y_1	y_2
P_1	x_1	2,2	1,1
	x_2	1,1	2,2

Figure 14: All 16 possible 2x2 minimal ordinal games.

Note that we can now easily characterize the previously discussed 2x2 games by matching each with one of the sixteen games of Figure 14. Both the prisoners' dilemma and the Cold War arms race are the same as the minimal ordinal game $1_A,2_A$. The dating game is the same as the minimal ordinal game $1_B,2_B$. Likewise, any other 2x2 game can be reduced to one of these sixteen minimal ordinal games.

This sort of simplification can be very appealing when one player has incomplete knowledge about the strategy of the other player. In this case, the player must consider all possible outcomes of the game. This simplification eliminates unnecessary effort, and in some cases can direct a player toward the best course of action.

3.3 CHARACTERIZATION OF NASH SOLUTIONS FOR 2X2 GAMES

Since we have listed all of the possible 2x2 minimal ordinal games, we can categorize these games based the existence of zero, one, or more Nash solutions. By inspection we can determine that twelve 2x2 minimal ordinal games have a unique Nash solution, two 2x2 minimal ordinal games have multiple, non-unique Nash solutions, and two 2x2 minimal ordinal games have no Nash solution.

Furthermore, we note that the unique Nash solutions are distributed equally among the four combinations of the control variables. The following table shows the relationship between the 2x2 games listed in Figure 14 and the location of unique Nash solutions.

Table 2: Distribution of the unique Nash solutions in 2x2 minimal ordinal games

Unique Nash Solution Location	**2x2 Games**
[1,1]	$1_A,2_A$; $1_A,2_B$; $1_B,2_A$
[1,2]	$1_A,2_C$; $1_A,2_D$; $1_D,2_C$
[2,1]	$1_C,2_A$; $1_C,2_D$; $1_D,2_A$
[2,2]	$1_B,2_C$; $1_C,2_B$; $1_C,2_C$

On the other hand, non-unique Nash solutions occur in only two games, $1_B,2_B$ and $1_D,2_D$ and only on the diagonal and the off diagonal.

Since the simplification from cardinal games to ordinal games and from ordinal games to minimal ordinal games resulted in no loss of Nash solutions, we can apply these results to both ordinal and cardinal games. If twelve out of sixteen, or three-fourths of all 576 2x2 minimal ordinal games have unique Nash solutions, then three-fourths of all 2x2 ordinal games, or 432 games, have unique Nash solutions. Likewise 72 2x2 ordinal games have no Nash solution and 72 2x2 ordinal games have two non-unique Nash solutions.

So if we know nothing about the objective functions in a 2x2 cardinal game, we can predict the game has a 75% chance of having a unique Nash solution, a 12.5% chance of having no Nash solution, and a 12.5% chance of having more than one Nash solution.

3.4 GENERALIZATION OF THE MINIMAL ORDINAL GAME SIMPLIFICATION

Next, we have to wonder if these results can be generalized to minimal ordinal games larger than 2x2. In the definition of the rank-ordered matrix M^o from the $n \times m$ matrix M (Definition 2.1), we stated that entries in M which have equal values should be ranked with the same preference. As such we would expect the number of preferences n_p to adhere to

$$n_p \le nm . \tag{3.2}$$

For the rest of the discussion, we will assume instead

$$n_p = nm . \tag{3.3}$$

In other words, no repeated entries exist in M^o. With this assumption in mind, we can begin to generalize the results from the 2x2 case. Suppose we have an ordinal game defined by the $n \times m$ matrices J_1 and J_2. We already know the total number of possible ordinal $n \times m$ games is given by $((nm)!)^2$.

If we instead consider $n \times m$ minimal ordinal games, we find that, as expected, we can significantly reduce the number of possible games. In an $n \times m$ game, the decision space for Player 1 contains n choices, and the decision space for Player 2 contains m choices. For any given choice made by Player 2, Player 1 can arrange his n choices into n locations. This means he can play

$$P(n,n) = n! \tag{3.4}$$

strategies for each choice made by Player 2. Since Player 1 can choose each strategy independently, he has a total of $(n!)^m$ possible strategies.

Example 3.3: Suppose we have the following 3x5 ordinal game. Note that the preference rankings of each player are unimportant in this example, so they are not shown.

		P_2				
		y_1	y_2	y_3	y_4	y_5
P_1	x_1					
	x_2					
	x_3					
Number of strategies:		3!	3!	3!	3!	3!

Figure 15: Number of P_1 strategies for each choice by P_2

If Player 2 uses y_3, Player 1 can respond with any of $3! = 6$ possible strategies. Likewise if Player 2 uses y_5, Player 1 can respond in $3! = 6$ ways, all of which are independent of his to y_3. As such, Player 1 can use $(n!)^m = (3!)^5 = 6^5 = 7,776$ possible strategies to play the game.

The same logic can be applied to the situation faced by Player 2. He can play a total of $(m!)^n = (5!)^3 = 120^3 = 1,728,000$ strategies. Since both players are free to choose their strategies independently, a total of

$$(n!)^m (m!)^n = (7,776)(1,728,000) = 13,436,928,000$$

minimal ordinal games exist. This is a tremendous number of possibilities, but the simplification from ordinal to minimal ordinal games is even more startling. Table 3 shows some samples of the number of possible ordinal games and possible minimal ordinal games for games of various sizes.

Table 3: Comparison between ordinal games and minimal ordinal games

Game Size	Number of Ordinal Games	Number of Minimal Ordinal Games
2x2	576	16
2x3	720	288
3x3	362,880	46,656
3x4	479,001,600	17,915,904
4x4	2.09×10^{13}	1.10×10^{11}
8x8	1.27×10^{89}	4.88×10^{73}
10x10	9.33×10^{157}	1.57×10^{131}

Even for games as small as 3x3, the simplification from ordinal games to minimal ordinal games is significant.

Given an $n \times m$ minimal ordinal game, we would ideally like to have a general formula for the number of games that have a unique Nash solution, the number that have multiple non-unique Nash solutions, and the number that have no Nash solution. However this generalization has proven very difficult to formulate. Instead, we can gain more insight by exploring another special case, 3x3 ordinal games.

4.0 3X3 ORDINAL GAMES

4.1 EXPLORATION OF 3X3 MINIMAL ORDINAL GAMES

Suppose we have the following two 3x3 minimal ordinal games.

		Player 2		
		y_1	y_2	y_3
Player 1	x_1	1 , 1	3 , 3	2 , 2
	x_2	3 , 2	2 , 3	3 , 1
	x_3	2 , 1	1 , 3	1 , 2

		Player 2		
		y_1	y_2	y_3
Player 1	x_1	1 , 1	3 , 2	2 , 3
	x_2	3 , 3	2 , 2	3 , 1
	x_3	2 , 1	1 , 2	1 , 3

Figure 16: Two similar 3x3 minimal ordinal games

These two 3x3 minimal ordinal games are similar, but they are clearly different games. Note however that $\{x_1, y_1\}$ is a unique Nash solution for both games. In fact, the flow of game play will commence exactly the same for both games. If Player 2 begins by choosing y_3, the game play will be as follows

$$y_3 \to x_3 \to y_1 \to x_1.$$

Likewise if Player 1 begins with strategy x_2, the game play for both games will be

$$x_2 \to y_3 \to x_3 \to y_1 \to x_1.$$

Since each player will always choose his best strategy in the given situation, we can easily follow the flow of the game by noticing where the top-rated preferences (the 1's) are located. So in truth, these two minimal ordinal games will always have the same outcome, the Nash solution $\{x_1, y_1\}$. Also, given an arbitrary starting point, both games will commence in the same manner.

Since the top-rated preferences of each player in a given situation are the only pieces of information necessary to determine the Nash solution and the flow of the game, we can ignore the other preferences and concentrate on only where the 1's will occur.

4.2 TOP-RATED PREFERENCE MINIMAL ORDINAL GAME SIMPLIFICATION

Suppose we have a 3x3 matrix J_1^{co}, the matrix for Player 1 in a minimal ordinal game.

$$J_1^{co} = \begin{bmatrix} 1 & 1 & 1 \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix}$$

Here the $\times$ characters represent don't care conditions in six of the entries in the matrix. These don't care conditions could be any preference ranking. If we fix the 1's at locations [1,1] and [1,2], and allow the 1 in column 3 to change position, we have the following three possible strategies.

$$\begin{bmatrix} 1 & 1 & 1 \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix} \text{ or } \begin{bmatrix} 1 & 1 & \times \\ \times & \times & 1 \\ \times & \times & \times \end{bmatrix} \text{ or } \begin{bmatrix} 1 & 1 & \times \\ \times & \times & \times \\ \times & \times & 1 \end{bmatrix}$$

Likewise if we fix the 1's in J_1^{co} in locations [1,1] and [2,2], we can generate three more possibilities.

$$\begin{bmatrix} 1 & \times & 1 \\ \times & 1 & \times \\ \times & \times & \times \end{bmatrix} \text{ or } \begin{bmatrix} 1 & \times & \times \\ \times & 1 & 1 \\ \times & \times & \times \end{bmatrix} \text{ or } \begin{bmatrix} 1 & \times & \times \\ \times & 1 & \times \\ \times & \times & 1 \end{bmatrix}$$

Finally, if we fix the 1's at locations [1,1] and [3,2], we come up with three other possibilities.

$$\begin{bmatrix} 1 & \times & 1 \\ \times & \times & \times \\ \times & 1 & \times \end{bmatrix} \text{ or } \begin{bmatrix} 1 & \times & \times \\ \times & \times & 1 \\ \times & 1 & \times \end{bmatrix} \text{ or } \begin{bmatrix} 1 & \times & \times \\ \times & \times & \times \\ \times & 1 & 1 \end{bmatrix}.$$

Each of these nine cases can be duplicated for the three possible locations of the 1 in the first column, giving us a total of 27 ways Player 1 can arrange his top-rated preferences. Likewise Player 2 can arrange his top-rated preferences in 27 ways. So in order to find the number of games that have a unique Nash solution, the number that have multiple non-unique Nash solutions, and the number that have no Nash solutions, we need to consider only $27^2 = 729$ games. In Table 3, we determined that we would need to consider 46,656 3x3 minimal ordinal games. Clearly this simplification greatly minimizes the number of possibilities, while still preserving the Nash solution characteristics of each game.

Note that each of the 27 possible strategies which Player 1 can use represents a multiplicity of minimal ordinal games. In each column, Player 1 has two preferences to rank, besides his top-rated preference. He can order those two preferences in two locations, meaning for each of the three columns, Player 1 has $P(2,2) = 2! = 2$ independent choices for his other preferences. Thus each of the 27 strategies Player 1 can use represent his strategies in

$2^3 = 8$ minimal ordinal games. The situation is identical for Player 2. Using this multiplicity, we can arrive at the total number of 3x3 minimal ordinal games in Table 3.

$$27 * 8 * 27 * 8 = 46,656 .$$

4.3 GENERALIZATION OF THE TOP-RATED PREFERENCE MINIMAL ORDINAL GAME SIMPLIFICATION

Given an $n \times m$ minimal ordinal game, Player 1 can place a 1 in each column of his matrix J_1^{co} in any of the n rows. Since J_1^{co} has m columns, Player 1 can position his 1's in n^m ways. Likewise Player 2 can position his 1's in m^n ways.

Once Player 1 has positioned his 1's in a given manner, he can arrange the remaining $n-1$ preferences in $n-1$ locations, meaning he has a total of

$$P(n-1, n-1) = (n-1)! \tag{4.1}$$

options in each column. Since the matrix J_1^{co} has m columns, Player 1 can arrange all of these other preferences in $((n-1)!)^m$ ways. Thus the n^m possible strategies Player 1 can use each represent a multiplicity of $((n-1)!)^m$ minimal ordinal games. Likewise the m^n possible strategies Player 2 can use each represent a multiplicity of $((m-1)!)^n$ minimal ordinal games.

So given an $n \times m$ minimal ordinal game, if we consider only the top-rated preferences for each player, a total of $(n^m)(m^n)$ different games exist.

In Chapter 3 we found that using minimal ordinal games, Player 1 has $(n!)^m$ possible strategies. Now we have decreased that number even more, showing that Player 1 really has only n^m possible strategies. Note that using the multiplicity of each of those strategies, we have

$$n^m((n-1)!)^m = (n(n-1)!)^m = (n!)^m \,. \tag{4.2}$$

Thus we have again lost no Nash solutions by focusing only on the top-rated preferences for each player. Table 4 shows some comparisons between the number of minimal ordinal games and the number of top-rated preference minimal ordinal games.

Table 4: The number of minimal and top-rated preference minimal ordinal games

Game Size	**Number of Minimal Ordinal Games**	**Number of Top-rated Preference Minimal Ordinal Games**
2x2	16	16
2x3	288	72
3x3	46,656	729
3x4	17,915,904	5,184
4x4	1.10×10^{11}	65,536
8x8	4.88×10^{73}	2.81×10^{14}
10x10	1.57×10^{131}	1.00×10^{20}

Without a general formula for the number of higher order games having unique, non-unique, and no Nash solutions, this simplification offers tremendous computational advantages. Using this simplification to top-rated preference minimal ordinal games, we can devise an algorithm to determine the number of games that have unique Nash solutions, the number that have multiple non-unique Nash solutions, and the number that have no Nash solution.

All 729 top-rated preference minimal ordinal games of size 3x3 are listed in Appendix A. As in Figure 14, the Nash solutions have been highlighted in bold type.

Example 4.1: Suppose a dairy must determine each day which one of the following types of milk to produce: whole milk, 2% milk, or skim milk. On a given day, the company profit per unit of milk depends on the type of cheese a certain cheese producer is making that day. The dairy supplies milk to the cheese producer, but the dairy does not know which type of cheese will be produced on any given day. Viewed as a two player game, we can determine the following bimatrix. The values in the bimatrix represent the dairy's profit per unit of milk, in dollars. The question marks represent the unknown preferences of the cheese producer.

		Cheese Producer		
		Colby	Cheddar	Swiss
Dairy	Whole	10 , ?	9 , ?	6 , ?
	2%	7 , ?	5 , ?	12 , ?
	Skim	2 , ?	3 , ?	8 , ?

Figure 17: Dairy profits

If the dairy is attempting to maximize its profit, and considers only at its top-rated preferences, we have the following top-rated preference minimal ordinal game, where the x's represent don't care conditions.

		Cheese Producer		
		Colby	Cheddar	Swiss
Dairy	Whole	1 , ?	1 , ?	x , ?
	2%	x , ?	x , ?	1 , ?
	Skim	x , ?	x , ?	x , ?

Figure 18: Dairy top-rated preferences

Given these prefences for the dairy, we know the cheese producer can play one of only 27 strategies. Furthermore, we can be certain that unique Nash solutions can occur only in locations [1,1], [1,2], and [2,3]. The dairy can easily examine all 27 possible strategies of the cheese producer, and determine where the unique Nash solutions are most likely to occur. The 27 possible games have been highlighted in the list of all 729 3x3 top-rated preference minimal ordinal games in Appendix A. We can determine by inspection that twelve of the 27 strategies the cheese producer can use will result in games that have either no Nash solution or multiple Nash solutions. Of the fifteen remaining games, six of the games will have a unique Nash solution at location [1,1], six will have a unique Nash solution at [1,2], and only three will have a unique Nash solution at location [2,3]. If the game is played on 27 consecutive days, six days will end up with the Nash solution at [1,1], six days at [1,2], and three days at [2,3]. If the dairy executives examine the profits in Figure 17, they should recognize that whole milk and Colby or whole milk and Cheddar will occur more often than 2% milk and Swiss. As Figure 17 shows, the dairy's highest profits occur in the least likely of the unique Nash solutions. The dairy should then modify its process to increase profits in the two situations that will occur most often. If the dairy does increase its profits in these two cases, its top-rated preferences will not change, meaning the Nash solutions of the games will not be affected.

So without any knowledge of the strategy of the other player involved in the game, our results allow the dairy to quickly determine where they need to optimize their process to maximize their profits.

5.0 NASH SOLUTION SEARCH ALGORITHM

5.1 EXPLANATION OF TOP-RATED PREFERENCE MINIMAL ORDINAL GAME SEARCH ALGORITHM

In the absence of a general formula for the number of top-rated preference minimal ordinal games which have a unique Nash solution, multiple non-unique Nash solutions, and no Nash solution, we can describe and implement an algorithm to search all possible top-rated preference minimal ordinal games of a given size. When this algorithm is devised as a recursive algorithm, the result is surprisingly simple.

In general, a recursive algorithm must have two cases, a recursive case where the algorithm calls itself, and a base case where the algorithm does not call itself and ends the recursion. Since our goal is to compare each strategy of Player 1 with each strategy of Player 2, it makes sense to pick a given strategy for Player 1, and compare that to all possible strategies for Player 2. Next the algorithm should pick another, different strategy for Player 1 and compare that to all strategies for Player 2. This process should repeat until all strategies for Player 1 have been exhausted.

So the algorithm should have one recursive function that loops through all strategies for Player 1. For each strategy of Player 1, that function should call another recursive function that loops through all strategies for Player 2. Pseudo code describing the algorithm used to find all strategies for Player 1 is given below.

```
player1_strategies(p1_game,p2_game,num_rows,num_columns,column)
{
      row = 1
      while (row <= num_rows)
      {
            p1_game[row,column] = 1
            if (column == num_columns)    // Base case
            {
                  player2_strategies(p1_game,p2_game,num_rows,
                                     num_columns,1)
            }
            else                          // Recursive case
            {
                  player1_strategies(p1_game,p2_game,num_rows,
                                     num_columns,column+1)
            }
            p1_game[row,column] = 0
            row = row+1
      }
}
```

The `player1_stategies` function is called with the `num_rows` and `num_columns` parameters equal to the number of rows and columns, respectively, in the game. The `column` parameter is initially passed with a value of 1, for the first column. The pseudo code for the algorithm to find all of the strategies for Player 2 is given below.

```
player2_strategies(p1_game,p2_game,num_rows,num_columns,row)
{
      column = 1
      while (column <= num_columns)
      {
            p2_game[row,column] = 1
            if (row == num_rows)    // Base case
            {
                  game_type = categorize_game(p1_game,p2_game)
                  if (game_type == unique_nash)
                  {
                        num_unique_nash = num_unique_nash+1
                  }
                  else if (game_type == non_unique_nash)
                  {
                        num_nonunique_nash = num_nonunique_nash+1
                  }
                  else
                  {
                        num_no_nash = num_no_nash+1
                  }
            }
            else                    // Recursive case
            {
                  player2_strategies(p1_game,p2_game,num_rows,
                                   num_columns,row+1)
            }
            p2_game[row,column] = 0
            column = column+1
      }
}
```

To better understand this algorithm, consider the simplest case, that of a 2x2 ordinal game.

Example 5.1: Suppose we want to use this algorithm to categorize all of the 2x2 top-rated preference minimal ordinal games. In the first call to `player1_strategies`, we'll have the following values for the input parameters:

```
p1_game = [0 0]
          [0 0]
p2_game = [0 0]
          [0 0]
num_rows = 2
num_columns = 2
column = 1
```

After we put a 1 in position [1,1] of the `p1_game`, we will make the recursive call to `player1_strategies` with the following input parameters:

```
p1_game = [1 0]
          [0 0]
p2_game = [0 0]
          [0 0]
num_rows = 2
num_columns = 2
column = 2
```

We will then add another 1 to `p1_game`, this time in position [1,2]. Since we have reached the last column (column 2), we're at the base case, and will call p2_strategies with the following parameters:

```
p1_game = [1 1]
          [0 0]
p2_game = [0 0]
          [0 0]
num_rows = 2
num_columns = 2
row = 1
```

Next we will set a 1 in position [1,1] of `p2_game`, and call `player2_strategies` again, with the following parameters:

```
p1_game = [1 1]
          [0 0]
p2_game = [1 0]
          [0 0]
num_rows = 2
num_columns = 2
row = 2
```

Finally, we set a 1 in position [2,1] of `p2_game` and hit the base case, where we call `categorize_game` with the following parameters:

```
p1_game = [1 1]
          [0 0]
p2_game = [1 0]
          [1 0]
```

This game has a unique Nash solution at location [1,1], so we increment the number of unique Nash solutions. Next we set a 0 in position [2,1] of `p2_game`, go to the top of the loop, and set a 1 in `p2_game` at position [2,2]. We're again at the base case, so we call `categorize_game` with the following parameters:

```
p1_game = [1 1]
          [0 0]
p2_game = [1 0]
```

```
          [0 1]
```

Now return from the second call to `player2_strategies`, and are back in the first call to `player2_strategies` with

```
row = 1
column = 1
```

We set a 0 in position [1,1] of `p2_game` and go to the top of the while loop. There we set a 1 in position [1,2] of `p2_game`, and make a call again to `p2_strategies` with the following parameters:

```
p1_game = [1 1]
          [0 0]
p2_game = [0 1]
          [0 0]
num_rows = 2
num_columns = 2
row = 2
```

After examining the two possibilities for row 2 of this `p2_game` parameter, we have finished comparing all four strategies of Player 2 to the initial strategy of Player 1. We then return from the first call to `player2_strategies`, get the next strategy for Player 1, and start the process again.

The complete program listing implementing the Nash solution search algorithm can be found in Appendix B.

5.2 RESULTS OBTAINED FROM THE ALGORITHM

The following table lists the number of games examined, the number of games with a unique Nash solution, the number of games with multiple non-unique Nash solutions, and the number of games with no Nash solution for games of varying sizes.

Table 5: Results of Nash solution search algorithm for games of varying sizes

Game Size	Number of Top-rated Preference Minimal Ordinal Games	Number of Games with a Unique Nash Solution	Number of Games with Multiple Nash Solutions	Number of Games with No Nash Solution
2x2	16	12	2	2
2x3	72	48	12	12
3x3	729	423	150	156
3x4	5,184	2,808	1152	1224
4x4	65,535	33,184	15,432	16,920
5x5	9,765,625	4,581,225	2,419,520	2,764,880
6x6	2,176,782,336	973,830,816	552,255,120	650,696,400

Note that the number of games with multiple Nash solutions and the number of games with no Nash solution is the same only for the 2x2 and 2x3 cases. Also, note how quickly the number of games grows, even with the top-rated preference minimal ordinal game simplification.

It may be more useful to view these results as percentages of the total number of games for a given size.

Table 6: Percentage results of the Nash solution search algorithm

Game Size	Percent of Total Games with a Unique Nash Solution	Percent of Total Games with Multiple Nash Solutions	Percent of Total Games with No Nash Solution
2x2	75.00%	12.50%	12.50%
2x3	66.67%	16.67%	16.67%
3x3	58.02%	20.58%	21.40%
3x4	54.17%	22.22%	23.61%
4x4	50.64%	23.55%	25.82%
5x5	46.91%	24.78%	28.31%
6x6	44.74%	25.37%	29.89%

Note that as the size of the game increases, the number of games with unique Nash solutions, as a percentage of the total number of games, decreases. These percentages seem to be approaching a point of stability where an increase in game size no longer has a noticeable impact on the percentages. The figure below shows the values from Table 6 in a graph.

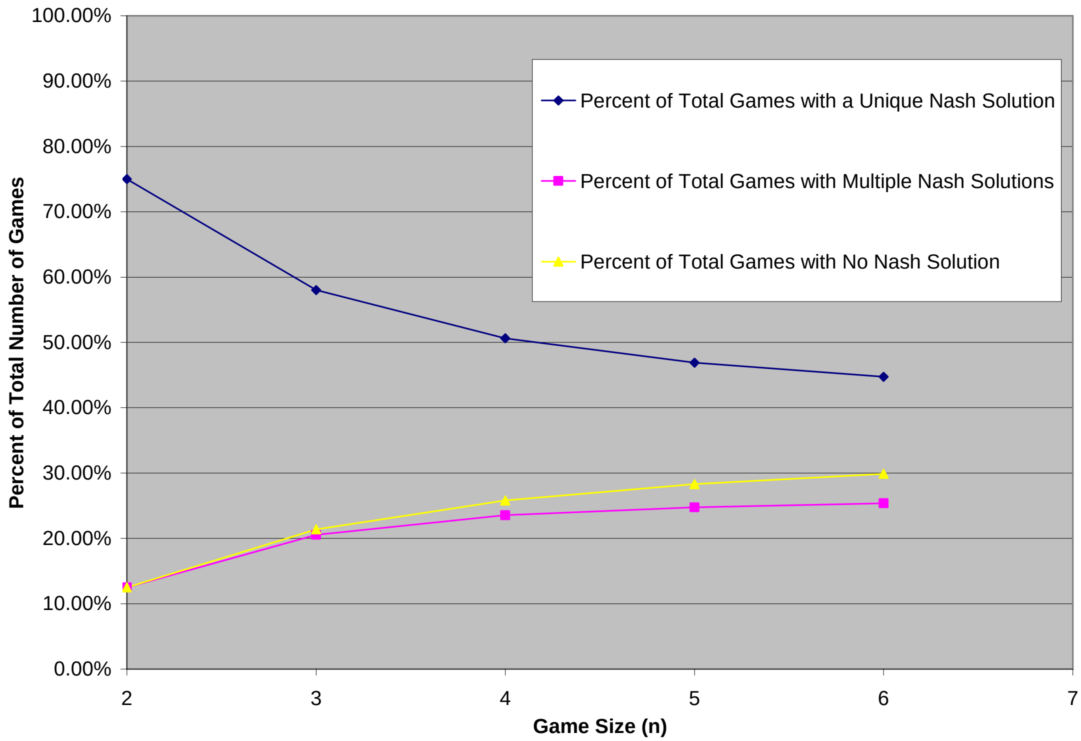

Figure 19: Nash solution distribution for nxn Games

Even with data for games only up to size 6x6 we can observe an asymptotic trend, and conjecture that the number of games with a unique Nash solution won't make up less than 40% of the total number of games. Likewise the number of games with either multiple Nash solutions, or no Nash solution, won't exceed about 30% of the total number of games, respectively.

The time to examine larger and larger games can quickly grow beyond the realm of possibility. The following table lists the times[2] required to run the algorithm.

---

[2] These times were obtained using the GNU time command on a computer with an Intel Pentium II 333 MHz processor running Linux kernel version 2.6.10.

Table 7: Execution times of the Nash solution search algorithm

Game Size	Time to Complete Algorithm
2x2	0.001 seconds
3x3	0.002 seconds
4x4	0.068 seconds
5x5	12.780 seconds
6x6	59 minutes 10.175 seconds

In the 6x6 case, the algorithm required about 3,550 seconds to examine 2,176,782,336 games, which corresponds to a rate of about 613,178 games per second. At this rate, without the top-rated preference minimal ordinal game simplification or the minimal ordinal game simplification, it would take 9.37×10^{27} years to come to the same conclusion! Even using the minimal ordinal game simplification, a comparable computer would require 1.00×10^{21} years of processing time. The following table shows the length of time required to consider games of higher order at this rate.

Table 8: Time to consider all possible games at a rate of 613,178 games per second

Game Size	Time to Examine all Minimal Ordinal Games	Time to Examine all Top-rated Preference Minimal Ordinal Games
7x7	3.53×10^{31} years	12.8 days
8x8	2.52×10^{60} years	14.6 years
9x9	6.16×10^{86} years	7,762 years
10x10	8.10×10^{117} years	5.17×10^{3} years

So despite the advantages of the top-rated preference minimal ordinal game simplification, computational barriers still exist.

6.0 APPLICATIONS AND CONCLUSIONS

6.1 INCOMPLETE KNOWLEDGE

Even with these computational barriers, the top-rated preference minimal ordinal game simplification has some appeal. Our initial assumptions about the Nash solution stated that both players must have knowledge of the strategy of their opponent. Certainly as the game is played out, this is the case. However before the game begins, each player most likely has limited, if any knowledge of the strategy of the opposing player. Example 4.1 (the dairy and cheese companies) began to explore this concept of incomplete knowledge. If a given player can examine all of the possible responses by his opponent to a given strategy, the player can in some sense score that strategy. By scoring a few feasible strategies, the player can determine which strategy is best or optimal.

The process described above can occur without the simplifications discussed, but these simplifications decrease the number of possible opposing strategies to consider, and therefore give the player using the simplifications an advantage. If no cost is incurred by acting first, it is always in a player's best interest to act first, since that often allows the player to dictate which Nash solution those games with multiple Nash solutions will settle on. If Player 1 acts first, he can often choose a strategy that will allow him to be sure that all of the games with a unique

Nash solution and those with multiple Nash solutions will work to his benefit. The following example, taken in part from [12], will illustrate this concept.

Example 6.1: Suppose we have a cardinal game formulated as follows. Two competing firms are attempting to determine which of three products to advertise. The options for Firm 1 are products $x_1, x_2,$ or x_3. The options for Firm 2 are products $y_1, y_2,$ or y_3. The values of the objective functions J_{F_1} and J_{F_2} for each combination of choices are given in the following figure.

		Firm 2		
		y_1	y_2	y_3
Firm 1	x_1	8 , 10	5 , 8	8 , 7
	x_2	7 , 5	8 , 6	11 , 7
	x_3	5 , 6	9 , 9	7 , 7

Figure 20: The product advertisement cardinal game

In this example, Firm 1 is attempting to maximize J_{F_1} and Firm 2 is attempting to maximize J_{F_2}. If we simplify this game to a top-rated preference minimal ordinal game, we can quickly determine the Nash solutions.

		Firm 2		
		y_1	y_2	y_3
Firm 1	x_1	1 , 1	x , x	x ,x
	x_2	x ,x	x , x	1 , 1
	x_3	x , x	1 , 1	x , x

Figure 21: The product advertisement top-rated preference minimal ordinal game

If Firm 2 acts as the leader (in the Stackelberg sense [12]), then the rational reaction set for Firm 1 is $D = \{\{x_1, y_1\}, \{x_2, y_3\}, \{x_3, y_2\}\}$. Thus the Stackelberg strategy for Firm 2 as leader would be the pair in D that maximizes J_{F_2}, $\{x_1, y_1\}$[3]. This leaves the objective function of Firm 1, J_{F_1}, valued at 8. From the perspective of Firm 1, this is the worst of all the Nash solutions. It would be in the firm's advantage to act first. In this case, the rational reaction set for Firm 2 is also D, but now Firm 1 can advertise x_2, forcing Firm 2 into the Nash solution at $\{x_2, y_3\}$, the best Nash solution for Firm 1. If both the firms would like to compromise, each could reach a middle ground by selecting the Nash solution at $\{x_3, y_2\}$. Firm 1 can gain the upper hand by acting

[3] Note that in this example, the members of the rational reaction set D correspond to the three Nash solutions of the game. In general this is not the case, and in general, the method of this example would not apply. The Stackelberg strategy is used to show which of the Nash solutions Firm 2 would likely prefer, if it could act as first. Since we assume that neither firm can forcibly exert its will upon the other, neither can actually act as leader.

first. At worst, Firm 1 can minimize its losses in the compromise case. If however, Firm 1 waits to act second, it is likely to end up in its worst case scenario.

In a more realistic case, Firm 1 may be aware of the values of its own objective function, but most likely it knows little if any information about the objective function of Firm 2. Using the top-rated preference minimal ordinal game simplification, Firm 1 needs to consider only 27 possible strategies of Firm 2. If Firm 1 were to consider all ordinal games ($(3^2)! = 362,880$ Firm 2 strategies) or even all minimal ordinal games ($(3!)^3 = 216$ Firm 2 strategies) it would be wasting time and resources, and would not be able to react as quickly. If Firm 2 chooses the following strategy,

$$J_{F_2} = \begin{bmatrix} 1 & \times & \times \\ \times & 1 & \times \\ \times & 1 & \times \end{bmatrix}$$

Firm 1 must act quickly to begin its advertising first and focus on advertising x_3, so it can avoid the Nash solution at (x_1, y_1), like the situation initially explored. Likewise if Firm 2 uses another strategy, say

$$J_{F_2} = \begin{bmatrix} 1 & \times & \times \\ \times & 1 & \times \\ 1 & \times & \times \end{bmatrix}$$

Firm 1 can be assured that the situation will come to rest at the only Nash solution, $\{x_1, y_1\}$, and Firm 1 need not waste resources by speeding up its advertising since acting first offers it no advantage. Firm 1 may instead attempt to negotiate with Firm 2 and share the market. In this way, Firm 1 can quickly examine all of the possible outcomes of the game and formulate a course of action for each possible strategy of Firm 2. So before the game begins, Firm 1 can know not only what its course of action will be, but even more so, it can know the likelihood that each course of action must be taken.

6.2 CATEGORIZATION OF SOLUTION TYPE

In [13] Basar outlines three different types of solutions for games involving Stackelberg strategies. His solutions (viewed from the perspective of two player games) can be classified as follows.

Type A: Concurrent Solution.

Although they act non-cooperatively, both players realize that the optimal solution for each occurs when one of the two is the leader. As such, the other player cooperates in becoming the follower.

Type B: Nonconcurrent Solution.

In this case, both players would benefit from becoming the leader. As Basar points out, the player who can process data more quickly will become the leader and gain an advantage.

Type C: Stalemate Solution.

Here both players will actually benefit from being the follower. As such, neither player desires to act first. Basar explains that a reasonable solution can be reached if both players negotiate or bargain in some way.

Although Basar explored Stackelberg strategies, these three solution types can be likened to the three Nash solution types discussed here. From the Nash solution perspective, Basar's Type A solution is similar to an ordinal game with a unique Nash solution. Since the equilibrium point of the game is determined, both players can non-cooperatively agree on the solution. Likewise Type B solutions correspond to ordinal games with multiple Nash solutions, where the outcome is uncertain until play begins, and each player can gain an advantage by acting first. The original game in Example 6.1 has a Type B solution. Finally, Type C solutions correspond to ordinal games with no Nash solution. In this case, an equilibrium state will not be reached, and the players must negotiate or modify the parameters of the game.

Using the top-rated preference minimal ordinal game simplification, one player can more quickly examine the strategies of the opposing player and can categorize each possible game into one of the above types. If the game has a Type A solution, the player need not waste valuable resources by attempting to act first. If the game has a Type B solution, the player can begin processing data in an attempt to act first and control the game. If the game has a Type C solution, the player can approach his opponent at the bargaining table, or attempt to modify his own strategy.

Assuming that the player can determine the cost of implementing any of his own strategies, he can then formulate a cost/benefit analysis for each possible strategy employed by his opponent.

Example 6.2: Suppose Company 1 formulates strategies for production in two different sectors of the market. Although Company 1 cannot determine the strategy of its competitor, Company 2, in either sector, it can eliminate a few possible strategies which are not available to Company 2. After analyzing all possible games for competition in sector X, Company 1 finds that 67% of the games have a Type A solution, 10% of the games have a Type B solution and 23% of the games have a Type C solution. So in sector X, Company 1 will benefit by acting first in only 10% of the possibilities (those games with a Type B solution). However in sector Y, Company 1 finds that 53% of the games have a Type B solution, whereas only 16% of the games have a Type A solution and 31% of the games have a Type C solution. In sector Y, Company 1 will benefit in 53% of the games by acting first. Thus Company 1 should devote its resources to sector Y, in an attempt to act first and control the game. If Company 1 can make this decision faster than Company 2, it will likely waste fewer resources in sector X. So the top-rated preference minimal ordinal game simplification can allow Company 1 to more quickly determine its best course of action.

Based on the numerical results presented in Chapter 5, we can see that as the size of the game grows, the likelihood of a Type A solution decreases. As discussed, the number of games of each solution type seems to stabilize. However if the game is larger than 4x4, fewer than 50% of the solutions will be of Type A. So without knowing anything about the game, a player can immediately realize that in more than half the games he can likely gain an advantage by acting first, either by controlling a Type B solution or coming to the bargaining table quickly without wasting resources in a Type C solution.

6.3 CONCLUSIONS

Throughout this thesis, we have explored an important subset of game theory known as ordinal game theory. Since the objective functions in cardinal games are often difficult or impossible to determine mathematically, ordinal game theory offers an attractive alternative to cardinal game theory. Every cardinal game can be converted into a corresponding ordinal game.

Since only a finite number of ordinal games of a given size exist, it is possible to list each ordinal game and characterize it according to the number of Nash solutions for the game.

Based on this characterization, three different types of ordinal games exist: those with only one unique Nash solution, those with multiple non-unique Nash solutions, and those with no Nash solution.

Despite the finiteness of the set of $n \times m$ ordinal games, for even small values of n and m any attempt to analyze the total number of possible games can reach beyond the realm of computational possibility. However, we have shown that using the minimal ordinal game simplification and the top-rated preference minimal ordinal game simplification, games as large as 6x6 can be brought into the realm of computational possibility.

For games of arbitrary size, we have described and implemented a recursive algorithm to compute the number of top-rated preference minimal ordinal games which have a unique Nash solution, the number which have multiple non-unique Nash solutions, and the number which have no Nash solution. Finally, we have presented applications of this simplification, showing that decreasing the computational complexity allows players to formulate strategies and courses of action more quickly when they have little or no knowledge of their opponent's strategy.

APPENDIX A

ALL 3X3 TOP-RATED PREFERENCE MINIMAL ORDINAL GAMES

The following pages list all 729 3x3 top-rated preference minimal ordinal games. Those highlighted in red are used with Example 4.1.

1,1	1,x	1,x
x,1	x,x	x,x
x,1	x,x	x,x

1,1	1,x	1,x
x,1	x,x	x,x
x,x	x,1	x,x

1,1	1,x	1,x
x,1	X,x	x,x
x,x	X,x	x,1

1,1	1,x	1,x
x,x	x,1	x,x
x,1	x,x	x,x

1,1	1,x	1,x
x,x	x,1	x,x
x,x	x,1	x,x

1,1	1,x	1,x
x,x	x,1	x,x
x,x	x,x	x,1

1,1	1,x	1,x
x,x	x,x	x,1
x,1	x,x	x,x

1,1	1,x	1,x
x,x	x,x	x,1
x,x	x,1	x,x

1,1	1,x	1,x
x,x	x,x	x,1
x,x	x,x	x,1

1,x	**1,1**	1,x
x,1	x,x	x,x
x,1	x,x	x,x

1,x	**1,1**	1,x
x,1	x,x	x,x
x,x	x,1	x,x

1,x	**1,1**	1,x
x,1	x,x	x,x
x,x	x,x	x,1

1,x	**1,1**	1,x
x,x	x,1	x,x
x,1	x,x	x,x

1,x	**1,1**	1,x
x,x	x,1	x,x
x,x	x,1	x,x

1,x	**1,1**	1,x
x,x	x,1	x,x
x,x	x,x	x,1

1,x	**1,1**	1,x
x,x	x,x	x,1
x,1	x,x	x,x

1,x	**1,1**	1,x
x,x	x,x	x,1
x,x	x,1	x,x

1,x	**1,1**	1,x
x,x	x,x	x,1
x,x	x,x	x,1

1,x	1,x	**1,1**
x,1	x,x	x,x
x,1	x,x	x,x

1,x	1,x	**1,1**
x,1	x,x	x,x
x,x	x,1	x,x

1,x	1,x	**1,1**
x,1	x,x	x,x
x,x	x,x	x,1

1,x	1,x	**1,1**
x,x	x,1	x,x
x,1	x,x	x,x

1,x	1,x	**1,1**
x,x	x,1	x,x
x,x	x,1	x,x

1,x	1,x	**1,1**
x,x	x,1	x,x
x,x	x,x	x,1

1,x	1,x	**1,1**
x,x	x,x	x,1
x,1	x,x	x,x

1,x	1,x	**1,1**
x,x	x,x	x,1
x,x	x,1	x,x

1,x	1,x	**1,1**
x,x	x,x	x,1
x,x	x,x	x,1

1,1	1,x	x,x
x,1	x,x	1,x
x,1	x,x	x,x

1,1	1,x	x,x
x,1	x,x	1,x
x,x	x,1	x,x

1,1	1,x	x,x
x,1	x,x	1,x
x,x	x,x	x,1

1,1	1,x	x,x
x,x	x,1	1,x
x,1	x,x	x,x

1,1	1,x	x,x
x,x	x,1	1,x
x,x	x,1	x,x

1,1	1,x	x,x
x,x	x,1	1,x
x,x	x,x	x,1

1,1	1,x	x,x
x,x	x,x	**1,1**
x,1	x,x	x,x

1,1	1,x	x,x
x,x	x,x	**1,1**
x,x	x,1	x,x

1,1	1,x	x,x
x,x	x,x	**1,1**
x,x	x,x	x,1

1,x	**1,1**	x,x
x,1	x,x	1,x
x,1	x,x	x,x

1,x	**1,1**	x,x
x,1	x,x	1,x
x,x	x,1	x,x

1,x	**1,1**	x,x
x,1	x,x	1,x
x,x	x,x	x,1

1,x	**1,1**	x,x
x,x	x,1	1,x
x,1	x,x	x,x

1,x	**1,1**	x,x
x,x	x,1	1,x
x,x	x,1	x,x

1,x	**1,1**	x,x
x,x	x,1	1,x
x,x	x,x	x,1

1,x	**1,1**	x,x
x,x	x,x	**1,1**
x,1	x,x	x,x

1,x	**1,1**	x,x
x,x	x,x	**1,1**
x,x	x,1	x,x

1,x	**1,1**	x,x
x,x	x,x	**1,1**
x,x	x,x	x,1

1,x	1,x	x,1
x,1	x,x	1,x
x,1	x,x	x,x

1,x	1,x	x,1
x,1	x,x	1,x
x,x	x,1	x,x

1,x	1,x	x,1
x,1	x,x	1,x
x,x	x,x	x,1

1,x	1,x	x,1
x,x	x,1	1,x
x,1	x,x	x,x

1,x	1,x	x,1
x,x	x,1	1,x
x,x	x,1	x,x

1,x	1,x	x,1
x,x	x,1	1,x
x,x	x,x	x,1

1,x	1,x	x,1
x,x	x,x	**1,1**
x,1	x,x	x,x

1,x	1,x	x,1
x,x	x,x	**1,1**
x,x	x,1	x,x

1,x	1,x	x,1
x,x	x,x	**1,1**
x,x	x,x	x,1

1,1	1,x	x,x
x,1	x,x	x,x
x,1	x,x	1,x

1,1	1,x	x,x
x,1	x,x	x,x
x,x	x,1	1,x

1,1	1,x	x,x
x,1	x,x	x,x
x,x	x,x	**1,1**

1,1	1,x	x,x
x,x	x,1	x,x
x,1	x,x	1,x

1,1	1,x	x,x
x,x	x,1	x,x
x,x	x,1	1,x

1,1	1,x	x,x
x,x	x,1	x,x
x,x	x,x	**1,1**

1,1	1,x	x,x
x,x	x,x	x,1
x,1	x,x	1,x

1,1	1,x	x,x
x,x	x,x	x,1
x,x	x,1	1,x

1,1	1,x	x,x
x,x	x,x	x,1
x,x	x,x	**1,1**

1,x	**1,1**	x,x
x,1	x,x	x,x
x,1	x,x	1,x

1,x	**1,1**	x,x
x,1	x,x	x,x
x,x	x,1	1,x

1,x	**1,1**	x,x
x,1	x,x	x,x
x,x	x,x	**1,1**

1,x	**1,1**	x,x
x,x	x,1	x,x
x,1	x,x	1,x

1,x	**1,1**	x,x
x,x	x,1	x,x
x,x	x,1	1,x

1,x	**1,1**	x,x
x,x	x,1	x,x
x,x	x,x	**1,1**

1,x	**1,1**	x,x
x,x	x,x	x,1
x,1	x,x	1,x

1,x	**1,1**	x,x
x,x	x,x	x,1
x,x	x,1	1,x

1,x	**1,1**	x,x
x,x	x,x	x,1
x,x	x,x	**1,1**

1,x	1,x	x,1
x,1	x,x	x,x
x,1	x,x	1,x

1,x	1,x	x,1
x,1	x,x	x,x
x,x	x,1	1,x

1,x	1,x	x,1
x,1	x,x	x,x
x,x	x,x	**1,1**

1,x	1,x	x,1
x,x	x,1	x,x
x,1	x,x	1,x

1,x	1,x	x,1
x,x	x,1	x,x
x,x	x,1	1,x

1,x	1,x	x,1
x,x	x,1	x,x
x,x	x,x	**1,1**

1,x	1,x	x,1
x,x	x,x	x,1
x,1	x,x	1,x

1,x	1,x	x,1
x,x	x,x	x,1
x,x	x,1	1,x

1,x	1,x	x,1
x,x	x,x	x,1
x,x	x,x	**1,1**

1,1	x,x	1,x
x,1	1,x	x,x
x,1	x,x	x,x

1,1	x,x	1,x
x,1	1,x	x,x
x,x	x,1	x,x

1,1	x,x	1,x
x,1	1,x	x,x
x,x	x,x	x,1

1,1	x,x	1,x
x,x	**1,1**	x,x
x,1	x,x	x,x

1,1	x,x	1,x
x,x	**1,1**	x,x
x,x	x,1	x,x

1,1	x,x	1,x
x,x	**1,1**	x,x
x,x	x,x	x,1

1,1	x,x	1,x
x,x	1,x	x,1
x,1	x,x	x,x

1,1	x,x	1,x
x,x	1,x	x,1
x,x	x,1	x,x

1,1	x,x	1,x
x,x	1,x	x,1
x,x	x,x	x,1

1,x	x,1	1,x
x,1	1,x	x,x
x,1	x,x	x,x

1,x	x,1	1,x
x,1	1,x	x,x
x,x	x,1	x,x

1,x	x,1	1,x
x,1	1,x	x,x
x,x	x,x	x,1

1,x	x,1	1,x
x,x	**1,1**	x,x
x,1	x,x	x,x

1,x	x,1	1,x
x,x	**1,1**	x,x
x,x	x,1	x,x

1,x	x,1	1,x
x,x	**1,1**	x,x
x,x	x,x	x,1

1,x	x,1	1,x
x,x	1,x	x,1
x,1	x,x	x,x

1,x	x,1	1,x
x,x	1,x	x,1
x,x	x,1	x,x

1,x	x,1	1,x
x,x	1,x	x,1
x,x	x,x	x,1

1,x	x,x	**1,1**
x,1	1,x	x,x
x,1	x,x	x,x

1,x	x,x	**1,1**
x,1	1,x	x,x
x,x	x,1	x,x

1,x	x,x	**1,1**
x,1	1,x	x,x
x,x	x,x	x,1

1,x	x,x	**1,1**
x,x	**1,1**	x,x
x,1	x,x	x,x

1,x	x,x	**1,1**
x,x	**1,1**	x,x
x,x	x,1	x,x

1,x	x,x	**1,1**
x,x	**1,1**	x,x
x,x	x,x	x,1

1,x	x,x	**1,1**
x,x	1,x	x,1
x,1	x,x	x,x

1,x	x,x	**1,1**
x,x	1,x	x,1
x,x	x,1	x,x

1,x	x,x	**1,1**
x,x	1,x	x,1
x,x	x,x	x,1

1,1	x,x	x,x
x,1	1,x	1,x
x,1	x,x	x,x

1,1	x,x	x,x
x,1	1,x	1,x
x,x	x,1	x,x

1,1	x,x	x,x
x,1	1,x	1,x
x,x	x,x	x,1

1,1	x,x	x,x
x,x	**1,1**	1,x
x,1	x,x	x,x

1,1	x,x	x,x
x,x	**1,1**	1,x
x,x	x,1	x,x

1,1	x,x	x,x
x,x	**1,1**	1,x
x,x	x,x	x,1

1,1	x,x	x,x
x,x	1,x	**1,1**
x,1	x,x	x,x

1,1	x,x	x,x
x,x	1,x	**1,1**
x,x	x,1	x,x

1,1	x,x	x,x
x,x	1,x	**1,1**
x,x	x,x	x,1

1,x	x,1	x,x
x,1	1,x	1,x
x,1	x,x	x,x

1,x	x,1	x,x
x,1	1,x	1,x
x,x	x,1	x,x

1,x	x,1	x,x
x,1	1,x	1,x
x,x	x,x	x,1

1,x	x,1	x,x
x,x	**1,1**	1,x
x,1	x,x	x,x

1,x	x,1	x,x
x,x	**1,1**	1,x
x,x	x,1	x,x

1,x	x,1	x,x
x,x	**1,1**	1,x
x,x	x,x	x,1

1,x	x,1	x,x
x,x	1,x	**1,1**
x,1	x,x	x,x

1,x	x,1	x,x
x,x	1,x	**1,1**
x,x	x,1	x,x

1,x	x,1	x,x
x,x	1,x	**1,1**
x,x	x,x	x,1

1,x	x,x	x,1
x,1	1,x	1,x
x,1	x,x	x,x

1,x	x,x	x,1
x,1	1,x	1,x
x,x	x,1	x,x

1,x	x,x	x,1
x,1	1,x	1,x
x,x	x,x	x,1

1,x	x,x	x,1
x,x	**1,1**	1,x
x,1	x,x	x,x

1,x	x,x	x,1
x,x	**1,1**	1,x
x,x	x,1	x,x

1,x	x,x	x,1
x,x	**1,1**	1,x
x,x	x,x	x,1

1,x	x,x	x,1
x,x	1,x	**1,1**
x,1	x,x	x,x

1,x	x,x	x,1
x,x	1,x	**1,1**
x,x	x,1	x,x

1,x	x,x	x,1
x,x	1,x	**1,1**
x,x	x,x	x,1

1,1	x,x	x,x
x,1	1,x	x,x
x,1	x,x	1,x

1,1	x,x	x,x
x,1	1,x	x,x
x,x	x,1	1,x

1,1	x,x	x,x
x,1	1,x	x,x
x,x	x,x	**1,1**

1,1	x,x	x,x
x,x	**1,1**	x,x
x,1	x,x	1,x

1,1	x,x	x,x
x,x	**1,1**	x,x
x,x	x,1	1,x

1,1	x,x	x,x
x,x	**1,1**	x,x
x,x	x,x	**1,1**

1,1	x,x	x,x
x,x	1,x	x,1
x,1	x,x	1,x

1,1	x,x	x,x
x,x	1,x	x,1
x,x	x,1	1,x

1,1	x,x	x,x
x,x	1,x	x,1
x,x	x,x	**1,1**

1,x	x,1	x,x
x,1	1,x	x,x
x,1	x,x	1,x

1,x	x,1	x,x
x,1	1,x	x,x
x,x	x,1	1,x

1,x	x,1	x,x
x,1	1,x	x,x
x,x	x,x	**1,1**

1,x	x,1	x,x
x,x	**1,1**	x,x
x,1	x,x	1,x

1,x	x,1	x,x
x,x	**1,1**	x,x
x,x	x,1	1,x

1,x	x,1	x,x
x,x	**1,1**	x,x
x,x	x,x	**1,1**

1,x	x,1	x,x
x,x	1,x	x,1
x,1	x,x	1,x

1,x	x,1	x,x
x,x	1,x	x,1
x,x	x,1	1,x

1,x	x,1	x,x
x,x	1,x	x,1
x,x	x,x	**1,1**

1,x	x,x	x,1
x,1	1,x	x,x
x,1	x,x	1,x

1,x	x,x	x,1
x,1	1,x	x,x
x,x	x,1	1,x

1,x	x,x	x,1
x,1	1,x	x,x
x,x	x,x	**1,1**

1,x	x,x	x,1
x,x	**1,1**	x,x
x,1	x,x	1,x

1,x	x,x	x,1
x,x	**1,1**	x,x
x,x	x,1	1,x

1,x	x,x	x,1
x,x	**1,1**	x,x
x,x	x,x	**1,1**

1,x	x,x	x,1
x,x	1,x	x,1
x,1	x,x	1,x

1,x	x,x	x,1
x,x	1,x	x,1
x,x	x,1	1,x

1,x	x,x	x,1
x,x	1,x	x,1
x,x	x,x	**1,1**

1,1	x,x	1,x
x,1	x,x	x,x
x,1	1,x	x,x

1,1	x,x	1,x
x,1	x,x	x,x
x,x	**1,1**	x,x

1,1	x,x	1,x
x,1	x,x	x,x
x,x	1,x	x,1

1,1	x,x	1,x
x,x	x,1	x,x
x,1	1,x	x,x

1,1	x,x	1,x
x,x	x,1	x,x
x,x	**1,1**	x,x

1,1	x,x	1,x
x,x	x,1	x,x
x,x	1,x	x,1

1,1	x,x	1,x
x,x	x,x	x,1
x,1	1,x	x,x

1,1	x,x	1,x
x,x	x,x	x,1
x,x	**1,1**	x,x

1,1	x,x	1,x
x,x	x,x	x,1
x,x	1,x	x,1

1,x	x,1	1,x
x,1	x,x	x,x
x,1	1,x	x,x

1,x	x,1	1,x
x,1	x,x	x,x
x,x	**1,1**	x,x

1,x	x,1	1,x
x,1	x,x	x,x
x,x	1,x	x,1

1,x	x,1	1,x
x,x	x,1	x,x
x,1	1,x	x,x

1,x	x,1	1,x
x,x	x,1	x,x
x,x	**1,1**	x,x

1,x	x,1	1,x
x,x	x,1	x,x
x,x	1,x	x,1

1,x	x,1	1,x
x,x	x,x	x,1
x,1	1,x	x,x

1,x	x,1	1,x
x,x	x,x	x,1
x,x	**1,1**	x,x

1,x	x,1	1,x
x,x	x,x	x,1
x,x	1,x	x,1

1,x	x,x	**1,1**
x,1	x,x	x,x
x,1	1,x	x,x

1,x	x,x	**1,1**
x,1	x,x	x,x
x,x	**1,1**	x,x

1,x	x,x	**1,1**
x,1	x,x	x,x
x,x	1,x	x,1

1,x	x,x	**1,1**
x,x	x,1	x,x
x,1	1,x	x,x

1,x	x,x	**1,1**
x,x	x,1	x,x
x,x	**1,1**	x,x

1,x	x,x	**1,1**
x,x	x,1	x,x
x,x	1,x	x,1

1,x	x,x	**1,1**
x,x	x,x	x,1
x,1	1,x	x,x

1,x	x,x	**1,1**
x,x	x,x	x,1
x,x	**1,1**	x,x

1,x	x,x	**1,1**
x,x	x,x	x,1
x,x	1,x	x,1

1,1	x,x	x,x
x,1	x,x	1,x
x,1	1,x	x,x

1,1	x,x	x,x
x,1	x,x	1,x
x,x	**1,1**	x,x

1,1	x,x	x,x
x,1	x,x	1,x
x,x	1,x	x,1

1,1	x,x	x,x
x,x	x,1	1,x
x,1	1,x	x,x

1,1	x,x	x,x
x,x	x,1	1,x
x,x	**1,1**	x,x

1,1	x,x	x,x
x,x	x,1	1,x
x,x	1,x	x,1

1,1	x,x	x,x
x,x	x,x	**1,1**
x,1	1,x	x,x

1,1	x,x	x,x
x,x	x,x	**1,1**
x,x	**1,1**	x,x

1,1	x,x	x,x
x,x	x,x	**1,1**
x,x	1,x	x,1

1,x	x,1	x,x
x,1	x,x	1,x
x,1	1,x	x,x

1,x	x,1	x,x
x,1	x,x	1,x
x,x	**1,1**	x,x

1,x	x,1	x,x
x,1	x,x	1,x
x,x	1,x	x,1

1,x	x,1	x,x
x,x	x,1	1,x
x,1	1,x	x,x

1,x	x,1	x,x
x,x	x,1	1,x
x,x	**1,1**	x,x

1,x	x,1	x,x
x,x	x,1	1,x
x,x	1,x	x,1

1,x	x,1	x,x
x,x	x,x	**1,1**
x,1	1,x	x,x

1,x	x,1	x,x
x,x	x,x	**1,1**
x,x	**1,1**	x,x

1,x	x,1	x,x
x,x	x,x	**1,1**
x,x	1,x	x,1

1,x	x,x	x,1
x,1	x,x	1,x
x,1	1,x	x,x

1,x	x,x	x,1
x,1	x,x	1,x
x,x	**1,1**	x,x

1,x	x,x	x,1
x,1	x,x	1,x
x,x	1,x	x,1

1,x	x,x	x,1
x,x	x,1	1,x
x,1	1,x	x,x

1,x	x,x	x,1
x,x	x,1	1,x
x,x	**1,1**	x,x

1,x	x,x	x,1
x,x	x,1	1,x
x,x	1,x	x,1

1,x	x,x	x,1
x,x	x,x	**1,1**
x,1	1,x	x,x

1,x	x,x	x,1
x,x	x,x	**1,1**
x,x	**1,1**	x,x

1,x	x,x	x,1
x,x	x,x	**1,1**
x,x	1,x	x,1

1,1	x,x	x,x
x,1	x,x	x,x
x,1	1,x	1,x

1,1	x,x	x,x
x,1	x,x	x,x
x,x	**1,1**	1,x

1,1	x,x	x,x
x,1	x,x	x,x
x,x	1,x	**1,1**

1,1	x,x	x,x
x,x	x,1	x,x
x,1	1,x	1,x

1,1	x,x	x,x
x,x	x,1	x,x
x,x	**1,1**	1,x

1,1	x,x	x,x
x,x	x,1	x,x
x,x	1,x	**1,1**

1,1	x,x	x,x
x,x	x,x	x,1
x,1	1,x	1,x

1,1	x,x	x,x
x,x	x,x	x,1
x,x	**1,1**	1,x

1,1	x,x	x,x
x,x	x,x	x,1
x,x	1,x	**1,1**

1,x	x,1	x,x
x,1	x,x	x,x
x,1	1,x	1,x

1,x	x,1	x,x
x,1	x,x	x,x
x,x	**1,1**	1,x

1,x	x,1	x,x
x,1	x,x	x,x
x,x	1,x	**1,1**

1,x	x,1	x,x
x,x	x,1	x,x
x,1	1,x	1,x

1,x	x,1	x,x
x,x	x,1	x,x
x,x	**1,1**	1,x

1,x	x,1	x,x
x,x	x,1	x,x
x,x	1,x	**1,1**

1,x	x,1	x,x
x,x	x,x	x,1
x,1	1,x	1,x

1,x	x,1	x,x
x,x	x,x	x,1
x,x	**1,1**	1,x

1,x	x,1	x,x
x,x	x,x	x,1
x,x	1,x	**1,1**

1,x	x,x	x,1
x,1	x,x	x,x
x,1	1,x	1,x

1,x	x,x	x,1
x,1	x,x	x,x
x,x	**1,1**	1,x

1,x	x,x	x,1
x,1	x,x	x,x
x,x	1,x	**1,1**

1,x	x,x	x,1
x,x	x,1	x,x
x,1	1,x	1,x

1,x	x,x	x,1
x,x	x,1	x,x
x,x	**1,1**	1,x

1,x	x,x	x,1
x,x	x,1	x,x
x,x	1,x	**1,1**

1,x	x,x	x,1
x,x	x,x	x,1
x,1	1,x	1,x

1,x	x,x	x,1
x,x	x,x	x,1
x,x	**1,1**	1,x

1,x	x,x	x,1
x,x	x,x	x,1
x,x	1,x	**1,1**

x,1	1,x	1,x
1,1	x,x	x,x
x,1	x,x	x,x

x,1	1,x	1,x
1,1	x,x	x,x
x,x	x,1	x,x

x,1	1,x	1,x
1,1	x,x	x,x
x,x	x,x	x,1

x,1	1,x	1,x
1,x	x,1	x,x
x,1	x,x	x,x

x,1	1,x	1,x
1,x	x,1	x,x
x,x	x,1	x,x

x,1	1,x	1,x
1,x	x,1	x,x
x,x	x,x	x,1

x,1	1,x	1,x
1,x	x,x	x,1
x,1	x,x	x,x

x,1	1,x	1,x
1,x	x,x	x,1
x,x	x,1	x,x

x,1	1,x	1,x
1,x	x,x	x,1
x,x	x,x	x,1

x,x	**1,1**	1,x
1,1	x,x	x,x
x,1	x,x	x,x

x,x	**1,1**	1,x
1,1	x,x	x,x
x,x	x,1	x,x

x,x	**1,1**	1,x
1,1	x,x	x,x
x,x	x,x	x,1

x,x	**1,1**	1,x
1,x	x,1	x,x
x,1	x,x	x,x

x,x	**1,1**	1,x
1,x	x,1	x,x
x,x	x,1	x,x

x,x	**1,1**	1,x
1,x	x,1	x,x
x,x	x,x	x,1

x,x	**1,1**	1,x
1,x	x,x	x,1
x,1	x,x	x,x

x,x	**1,1**	1,x
1,x	x,x	x,1
x,x	x,1	x,x

x,x	**1,1**	1,x
1,x	x,x	x,1
x,x	x,x	x,1

x,x	1,x	**1,1**
1,1	x,x	x,x
x,1	x,x	x,x

x,x	1,x	**1,1**
1,1	x,x	x,x
x,x	x,1	x,x

x,x	1,x	**1,1**
1,1	x,x	x,x
x,x	x,x	x,1

x,x	1,x	**1,1**
1,x	x,1	x,x
x,1	x,x	x,x

x,x	1,x	**1,1**
1,x	x,1	x,x
x,x	x,1	x,x

x,x	1,x	**1,1**
1,x	x,1	x,x
x,x	x,x	x,1

x,x	1,x	**1,1**
1,x	x,x	x,1
x,1	x,x	x,x

x,x	1,x	**1,1**
1,x	x,x	x,1
x,x	x,1	x,x

x,x	1,x	**1,1**
1,x	x,x	x,1
x,x	x,x	x,1

x,1	1,x	x,x
1,1	x,x	1,x
x,1	x,x	x,x

x,1	1,x	x,x
1,1	x,x	1,x
x,x	x,1	x,x

x,1	1,x	x,x
1,1	x,x	1,x
x,x	x,x	x,1

x,1	1,x	x,x
1,x	x,1	1,x
x,1	x,x	x,x

x,1	1,x	x,x
1,x	x,1	1,x
x,x	x,1	x,x

x,1	1,x	x,x
1,x	x,1	1,x
x,x	x,x	x,1

x,1	1,x	x,x
1,x	x,x	**1,1**
x,1	x,x	x,x

x,1	1,x	x,x
1,x	x,x	**1,1**
x,x	x,1	x,x

x,1	1,x	x,x
1,x	x,x	**1,1**
x,x	x,x	x,1

x,x	**1,1**	x,x
1,1	x,x	1,x
x,1	x,x	x,x

x,x	**1,1**	x,x
1,1	x,x	1,x
x,x	x,1	x,x

x,x	**1,1**	x,x
1,1	x,x	1,x
x,x	x,x	x,1

x,x	**1,1**	x,x
1,x	x,1	1,x
x,1	x,x	x,x

x,x	**1,1**	x,x
1,x	x,1	1,x
x,x	x,1	x,x

x,x	**1,1**	x,x
1,x	x,1	1,x
x,x	x,x	x,1

x,x	**1,1**	x,x
1,x	x,x	**1,1**
x,1	x,x	x,x

x,x	**1,1**	x,x
1,x	x,x	**1,1**
x,x	x,1	x,x

x,x	**1,1**	x,x
1,x	x,x	**1,1**
x,x	x,x	x,1

x,x	1,x	x,1
1,1	x,x	1,x
x,1	x,x	x,x

x,x	1,x	x,1
1,1	x,x	1,x
x,x	x,1	x,x

x,x	1,x	x,1
1,1	x,x	1,x
x,x	x,x	x,1

x,x	1,x	x,1
1,x	x,1	1,x
x,1	x,x	x,x

x,x	1,x	x,1
1,x	x,1	1,x
x,x	x,1	x,x

x,x	1,x	x,1
1,x	x,1	1,x
x,x	x,x	x,1

x,x	1,x	x,1
1,x	x,x	**1,1**
x,1	x,x	x,x

x,x	1,x	x,1
1,x	x,x	**1,1**
x,x	x,1	x,x

x,x	1,x	x,1
1,x	x,x	**1,1**
x,x	x,x	x,1

x,1	1,x	x,x
1,1	x,x	x,x
x,1	x,x	1,x

x,1	1,x	x,x
1,1	x,x	x,x
x,x	x,1	1,x

x,1	1,x	x,x
1,1	x,x	x,x
x,x	x,x	**1,1**

x,1	1,x	x,x
1,x	x,1	x,x
x,1	x,x	1,x

x,1	1,x	x,x
1,x	x,1	x,x
x,x	x,1	1,x

x,1	1,x	x,x
1,x	x,1	x,x
x,x	x,x	**1,1**

x,1	1,x	x,x
1,x	x,x	x,1
x,1	x,x	1,x

x,1	1,x	x,x
1,x	x,x	x,1
x,x	x,1	1,x

x,1	1,x	x,x
1,x	x,x	x,1
x,x	x,x	**1,1**

x,x	**1,1**	x,x
1,1	x,x	x,x
x,1	x,x	1,x

x,x	**1,1**	x,x
1,1	x,x	x,x
x,x	x,1	1,x

x,x	**1,1**	x,x
1,1	x,x	x,x
x,x	x,x	**1,1**

x,x	**1,1**	x,x
1,x	x,1	x,x
x,1	x,x	1,x

x,x	**1,1**	x,x
1,x	x,1	x,x
x,x	x,1	1,x

x,x	**1,1**	x,x
1,x	x,1	x,x
x,x	x,x	**1,1**

x,x	**1,1**	x,x
1,x	x,x	x,1
x,1	x,x	1,x

x,x	**1,1**	x,x
1,x	x,x	x,1
x,x	x,1	1,x

x,x	**1,1**	x,x
1,x	x,x	x,1
x,x	x,x	**1,1**

x,x	1,x	x,1
1,1	x,x	x,x
x,1	x,x	1,x

x,x	1,x	x,1
1,1	x,x	x,x
x,x	x,1	1,x

x,x	1,x	x,1
1,1	x,x	x,x
x,x	x,x	**1,1**

x,x	1,x	x,1
1,x	x,1	x,x
x,1	x,x	1,x

x,x	1,x	x,1
1,x	x,1	x,x
x,x	x,1	1,x

x,x	1,x	x,1
1,x	x,1	x,x
x,x	x,x	**1,1**

x,x	1,x	x,1
1,x	x,x	x,1
x,1	x,x	1,x

x,x	1,x	x,1
1,x	x,x	x,1
x,x	x,1	1,x

x,x	1,x	x,1
1,x	x,x	x,1
x,x	x,x	**1,1**

x,1	x,x	1,x
1,1	1,x	x,x
x,1	x,x	x,x

x,1	x,x	1,x
1,1	1,x	x,x
x,x	x,1	x,x

x,1	x,x	1,x
1,1	1,x	x,x
x,x	x,x	x,1

x,1	x,x	1,x
1,x	**1,1**	x,x
x,1	x,x	x,x

x,1	x,x	1,x
1,x	**1,1**	x,x
x,x	x,1	x,x

x,1	x,x	1,x
1,x	**1,1**	x,x
x,x	x,x	x,1

x,1	x,x	1,x
1,x	1,x	x,1
x,1	x,x	x,x

x,1	x,x	1,x
1,x	1,x	x,1
x,x	x,1	x,x

x,1	x,x	1,x
1,x	1,x	x,1
x,x	x,x	x,1

x,x	x,1	1,x
1,1	1,x	x,x
x,1	x,x	x,x

x,x	x,1	1,x
1,1	1,x	x,x
x,x	x,1	x,x

x,x	x,1	1,x
1,1	1,x	x,x
x,x	x,x	x,1

x,x	x,1	1,x
1,x	**1,1**	x,x
x,1	x,x	x,x

x,x	x,1	1,x
1,x	**1,1**	x,x
x,x	x,1	x,x

x,x	x,1	1,x
1,x	**1,1**	x,x
x,x	x,x	x,1

x,x	x,1	1,x
1,x	1,x	x,1
x,1	x,x	x,x

x,x	x,1	1,x
1,x	1,x	x,1
x,x	x,1	x,x

x,x	x,1	1,x
1,x	1,x	x,1
x,x	x,x	x,1

x,x	x,x	**1,1**
1,1	1,x	x,x
x,1	x,x	x,x

x,x	x,x	**1,1**
1,1	1,x	x,x
x,x	x,1	x,x

x,x	x,x	**1,1**
1,1	1,x	x,x
x,x	x,x	x,1

x,x	x,x	**1,1**
1,x	**1,1**	x,x
x,1	x,x	x,x

x,x	x,x	**1,1**
1,x	**1,1**	x,x
x,x	x,1	x,x

x,x	x,x	**1,1**
1,x	**1,1**	x,x
x,x	x,x	x,1

x,x	x,x	**1,1**
1,x	1,x	x,1
x,1	x,x	x,x

x,x	x,x	**1,1**
1,x	1,x	x,1
x,x	x,1	x,x

x,x	x,x	**1,1**
1,x	1,x	x,1
x,x	x,x	x,1

x,1	x,x	x,x
1,1	1,x	1,x
x,1	x,x	x,x

x,1	x,x	x,x
1,1	1,x	1,x
x,x	x,1	x,x

x,1	x,x	x,x
1,1	1,x	1,x
x,x	x,x	x,1

x,1	x,x	x,x
1,x	**1,1**	1,x
x,1	x,x	x,x

x,1	x,x	x,x
1,x	**1,1**	1,x
x,x	x,1	x,x

x,1	x,x	x,x
1,x	**1,1**	1,x
x,x	x,x	x,1

x,1	x,x	x,x
1,x	1,x	**1,1**
x,1	x,x	x,x

x,1	x,x	x,x
1,x	1,x	**1,1**
x,x	x,1	x,x

x,1	x,x	x,x
1,x	1,x	**1,1**
x,x	x,x	x,1

x,x	x,1	x,x	x,x	x,1	x,x	x,x	x,1	x,x	x,x	x,1	x,x	x,x	x,1	x,x	x,x	x,1	x,x
1,1	1,x	1,x	**1,1**	1,x	1,x	**1,1**	1,x	1,x	1,x	**1,1**	1,x	1,x	**1,1**	1,x	1,x	**1,1**	1,x
x,1	x,x	x,x	x,x	x,1	x,x	x,x	x,x	x,1	x,1	x,x	x,x	x,x	x,1	x,x	x,x	x,x	x,1

x,x	x,1	x,x	x,x	x,1	x,x	x,x	x,1	x,x	x,x	x,x	x,1	x,x	x,x	x,1	x,x	x,x	x,1
1,x	1,x	**1,1**	1,x	1,x	**1,1**	1,x	1,x	**1,1**	**1,1**	1,x	1,x	**1,1**	1,x	1,x	**1,1**	1,x	1,x
x,1	x,x	x,x	x,x	x,1	x,x	x,x	x,x	x,1	x,1	x,x	x,x	x,x	x,1	x,x	x,x	x,x	x,1

x,x	x,x	x,1	x,x	x,x	x,1	x,x	x,x	x,1	x,x	x,x	x,1	x,x	x,x	x,1	x,x	x,x	x,1
1,x	**1,1**	1,x	1,x	**1,1**	1,x	1,x	**1,1**	1,x	1,x	1,x	**1,1**	1,x	1,x	**1,1**	1,x	1,x	**1,1**
x,1	x,x	x,x	x,x	x,1	x,x	x,x	x,x	x,1	x,1	x,x	x,x	x,x	x,1	x,x	x,x	x,x	x,1

x,1	x,x	x,x	x,1	x,x	x,x	x,1	x,x	x,x	x,1	x,x	x,x	x,1	x,x	x,x	x,1	x,x	x,x
1,1	1,x	x,x	**1,1**	1,x	x,x	**1,1**	1,x	x,x	1,x	**1,1**	x,x	1,x	**1,1**	x,x	1,x	**1,1**	x,x
x,1	x,x	1,x	x,x	x,1	1,x	x,x	x,x	**1,1**	x,1	x,x	1,x	x,x	x,1	1,x	x,x	x,x	**1,1**

x,1	x,x	x,x	x,1	x,x	x,x	x,1	x,x	x,x	x,x	x,1	x,x	x,x	x,1	x,x	x,x	x,1	x,x
1,x	1,x	x,1	1,x	1,x	x,1	1,x	1,x	x,1	**1,1**	1,x	x,x	**1,1**	1,x	x,x	**1,1**	1,x	x,x
x,1	x,x	1,x	x,x	x,1	1,x	x,x	x,x	**1,1**	x,1	x,x	1,x	x,x	x,1	1,x	x,x	x,x	**1,1**

x,x	x,1	x,x	x,x	x,1	x,x	x,x	x,1	x,x	x,x	x,1	x,x	x,x	x,1	x,x	x,x	x,1	x,x
1,x	**1,1**	x,x	1,x	**1,1**	x,x	1,x	**1,1**	x,x	1,x	1,x	x,1	1,x	1,x	x,1	1,x	1,x	x,1
x,1	x,x	1,x	x,x	x,1	1,x	x,x	x,x	**1,1**	x,1	x,x	1,x	x,x	x,1	1,x	x,x	x,x	**1,1**

x,x	x,x	x,1	x,x	x,x	x,1	x,x	x,x	x,1	x,x	x,x	x,1	x,x	x,x	x,1	x,x	x,x	x,1
1,1	1,x	x,x	**1,1**	1,x	x,x	**1,1**	1,x	x,x	1,x	**1,1**	x,x	1,x	**1,1**	x,x	1,x	**1,1**	x,x
x,1	x,x	1,x	x,x	x,1	1,x	x,x	x,x	**1,1**	x,1	x,x	1,x	x,x	x,1	1,x	x,x	x,x	**1,1**

x,x	x,x	x,1	x,x	x,x	x,1	x,x	x,x	x,1	x,1	x,x	1,x	x,1	x,x	1,x	x,1	x,x	1,x
1,x	1,x	x,1	1,x	1,x	x,1	1,x	1,x	x,1	**1,1**	x,x	x,x	**1,1**	x,x	x,x	**1,1**	x,x	x,x
x,1	x,x	1,x	x,x	x,1	1,x	x,x	x,x	**1,1**	x,1	1,x	x,x	x,x	**1,1**	x,x	x,x	1,x	x,1

x,1	x,x	1,x	x,1	x,x	1,x	x,1	x,x	1,x	x,1	x,x	1,x	x,1	x,x	1,x	x,1	x,x	1,x
1,x	x,1	x,x	1,x	x,1	x,x	1,x	x,1	x,x	1,x	x,x	x,1	1,x	x,x	x,1	1,x	x,x	x,1
x,1	1,x	x,x	x,x	**1,1**	x,x	x,x	1,x	x,1	x,1	1,x	x,x	x,x	**1,1**	x,x	x,x	1,x	x,1

x,x	x,1	1,x	x,x	x,1	1,x	x,x	x,1	1,x	x,x	x,1	1,x	x,x	x,1	1,x	x,x	x,1	1,x
1,1	x,x	x,x	**1,1**	x,x	x,x	**1,1**	x,x	x,x	1,x	x,1	x,x	1,x	x,1	x,x	1,x	x,1	x,x
x,1	1,x	x,x	x,x	**1,1**	x,x	x,x	1,x	x,1	x,1	1,x	x,x	x,x	**1,1**	x,x	x,x	1,x	x,1

x,x	x,1	1,x
1,x	x,x	x,1
x,1	1,x	x,x

x,x	x,1	1,x
1,x	x,x	x,1
x,x	**1,1**	x,x

x,x	x,1	1,x
1,x	x,x	x,1
x,x	1,x	x,1

x,x	x,x	**1,1**
1,1	x,x	x,x
x,1	1,x	x,x

x,x	x,x	**1,1**
1,1	x,x	x,x
x,x	**1,1**	x,x

x,x	x,x	**1,1**
1,1	x,x	x,x
x,x	1,x	x,1

x,x	x,x	**1,1**
1,x	x,1	x,x
x,1	1,x	x,x

x,x	x,x	**1,1**
1,x	x,1	x,x
x,x	**1,1**	x,x

x,x	x,x	**1,1**
1,x	x,1	x,x
x,x	1,x	x,1

x,x	x,x	**1,1**
1,x	x,x	x,1
x,1	1,x	x,x

x,x	x,x	**1,1**
1,x	x,x	x,1
x,x	**1,1**	x,x

x,x	x,x	**1,1**
1,x	x,x	x,1
x,x	1,x	x,1

x,1	x,x	x,x
1,1	x,x	1,x
x,1	1,x	x,x

x,1	x,x	x,x
1,1	x,x	1,x
x,x	**1,1**	x,x

x,1	x,x	x,x
1,1	x,x	1,x
x,x	1,x	x,1

x,1	x,x	x,x
1,x	x,1	1,x
x,1	1,x	x,x

x,1	x,x	x,x
1,x	x,1	1,x
x,x	**1,1**	x,x

x,1	x,x	x,x
1,x	x,1	1,x
x,x	1,x	x,1

x,1	x,x	x,x
1,x	x,x	**1,1**
x,1	1,x	x,x

x,1	x,x	x,x
1,x	x,x	**1,1**
x,x	**1,1**	x,x

x,1	x,x	x,x
1,x	x,x	**1,1**
x,x	1,x	x,1

x,x	x,1	x,x
1,1	x,x	1,x
x,1	1,x	x,x

x,x	x,1	x,x
1,1	x,x	1,x
x,x	**1,1**	x,x

x,x	x,1	x,x
1,1	x,x	1,x
x,x	1,x	x,1

x,x	x,1	x,x
1,x	x,1	1,x
x,1	1,x	x,x

x,x	x,1	x,x
1,x	x,1	1,x
x,x	**1,1**	x,x

x,x	x,1	x,x
1,x	x,1	1,x
x,x	1,x	x,1

x,x	x,1	x,x
1,x	x,x	**1,1**
x,1	1,x	x,x

x,x	x,1	x,x
1,x	x,x	**1,1**
x,x	**1,1**	x,x

x,x	x,1	x,x
1,x	x,x	**1,1**
x,x	1,x	x,1

x,x	x,x	x,1
1,1	x,x	1,x
x,1	1,x	x,x

x,x	x,x	x,1
1,1	x,x	1,x
x,x	**1,1**	x,x

x,x	x,x	x,1
1,1	x,x	1,x
x,x	1,x	x,1

x,x	x,x	x,1
1,x	x,1	1,x
x,1	1,x	x,x

x,x	x,x	x,1
1,x	x,1	1,x
x,x	**1,1**	x,x

x,x	x,x	x,1
1,x	x,1	1,x
x,x	1,x	x,1

x,x	x,x	x,1
1,x	x,x	**1,1**
x,1	1,x	x,x

x,x	x,x	x,1
1,x	x,x	**1,1**
x,x	**1,1**	x,x

x,x	x,x	x,1
1,x	x,x	**1,1**
x,x	1,x	x,1

x,1	x,x	x,x
1,1	x,x	x,x
x,1	1,x	1,x

x,1	x,x	x,x
1,1	x,x	x,x
x,x	**1,1**	1,x

x,1	x,x	x,x
1,1	x,x	x,x
x,x	1,x	**1,1**

x,1	x,x	x,x
1,x	x,1	x,x
x,1	1,x	1,x

x,1	x,x	x,x
1,x	x,1	x,x
x,x	**1,1**	1,x

x,1	x,x	x,x
1,x	x,1	x,x
x,x	1,x	**1,1**

x,1	x,x	x,x
1,x	x,x	x,1
x,1	1,x	1,x

x,1	x,x	x,x
1,x	x,x	x,1
x,x	**1,1**	1,x

x,1	x,x	x,x
1,x	x,x	x,1
x,x	1,x	**1,1**

x,x	x,1	x,x
1,1	x,x	x,x
x,1	1,x	1,x

x,x	x,1	x,x
1,1	x,x	x,x
x,x	**1,1**	1,x

x,x	x,1	x,x
1,1	x,x	x,x
x,x	1,x	**1,1**

x,x	x,1	x,x
1,x	x,1	x,x
x,1	1,x	1,x

x,x	x,1	x,x
1,x	x,1	x,x
x,x	**1,1**	1,x

x,x	x,1	x,x
1,x	x,1	x,x
x,x	1,x	**1,1**

x,x	x,1	x,x
1,x	x,x	x,1
x,1	1,x	1,x

x,x	x,1	x,x
1,x	x,x	x,1
x,x	**1,1**	1,x

x,x	x,1	x,x
1,x	x,x	x,1
x,x	1,x	**1,1**

x,x	x,x	x,1
1,1	x,x	x,x
x,1	1,x	1,x

x,x	x,x	x,1
1,1	x,x	x,x
x,x	**1,1**	1,x

x,x	x,x	x,1
1,1	x,x	x,x
x,x	1,x	**1,1**

x,x	x,x	x,1
1,x	x,1	x,x
x,1	1,x	1,x

x,x	x,x	x,1
1,x	x,1	x,x
x,x	**1,1**	1,x

x,x	x,x	x,1
1,x	x,1	x,x
x,x	1,x	**1,1**

x,x	x,x	x,1
1,x	x,x	x,1
x,1	1,x	1,x

x,x	x,x	x,1
1,x	x,x	x,1
x,x	**1,1**	1,x

x,x	x,x	x,1
1,x	x,x	x,1
x,x	1,x	**1,1**

x,1	1,x	1,x
x,1	x,x	x,x
1,1	x,x	x,x

x,1	1,x	1,x
x,1	x,x	x,x
1,x	x,1	x,x

x,1	1,x	1,x
x,1	x,x	x,x
1,x	x,x	x,1

x,1	1,x	1,x
x,x	x,1	x,x
1,1	x,x	x,x

x,1	1,x	1,x
x,x	x,1	x,x
1,x	x,1	x,x

x,1	1,x	1,x
x,x	x,1	x,x
1,x	x,x	x,1

x,1	1,x	1,x
x,x	x,x	x,1
1,1	x,x	x,x

x,1	1,x	1,x
x,x	x,x	x,1
1,x	x,1	x,x

x,1	1,x	1,x
x,x	x,x	x,1
1,x	x,x	x,1

x,x	**1,1**	1,x
x,1	x,x	x,x
1,1	x,x	x,x

x,x	**1,1**	1,x
x,1	x,x	x,x
1,x	x,1	x,x

x,x	**1,1**	1,x
x,1	x,x	x,x
1,x	x,x	x,1

x,x	**1,1**	1,x
x,x	x,1	x,x
1,1	x,x	x,x

x,x	**1,1**	1,x
x,x	x,1	x,x
1,x	x,1	x,x

x,x	**1,1**	1,x
x,x	x,1	x,x
1,x	x,x	x,1

x,x	**1,1**	1,x
x,x	x,x	x,1
1,1	x,x	x,x

x,x	**1,1**	1,x
x,x	x,x	x,1
1,x	x,1	x,x

x,x	**1,1**	1,x
x,x	x,x	x,1
1,x	x,x	x,1

x,x	1,x	**1,1**
x,1	x,x	x,x
1,1	x,x	x,x

x,x	1,x	**1,1**
x,1	x,x	x,x
1,x	x,1	x,x

x,x	1,x	**1,1**
x,1	x,x	x,x
1,x	x,x	x,1

x,x	1,x	**1,1**
x,x	x,1	x,x
1,1	x,x	x,x

x,x	1,x	**1,1**
x,x	x,1	x,x
1,x	x,1	x,x

x,x	1,x	**1,1**
x,x	x,1	x,x
1,x	x,x	x,1

x,x	1,x	**1,1**
x,x	x,x	x,1
1,1	x,x	x,x

x,x	1,x	**1,1**
x,x	x,x	x,1
1,x	x,1	x,x

x,x	1,x	**1,1**
x,x	x,x	x,1
1,x	x,x	x,1

x,1	1,x	x,x
x,1	x,x	1,x
1,1	x,x	x,x

x,1	1,x	x,x
x,1	x,x	1,x
1,x	x,1	x,x

x,1	1,x	x,x
x,1	x,x	1,x
1,x	x,x	x,1

x,1	1,x	x,x
x,x	x,1	1,x
1,1	x,x	x,x

x,1	1,x	x,x
x,x	x,1	1,x
1,x	x,1	x,x

x,1	1,x	x,x
x,x	x,1	1,x
1,x	x,x	x,1

x,1	1,x	x,x
x,x	x,x	**1,1**
1,1	x,x	x,x

x,1	1,x	x,x
x,x	x,x	**1,1**
1,x	x,1	x,x

x,1	1,x	x,x
x,x	x,x	**1,1**
1,x	x,x	x,1

x,x	**1,1**	x,x
x,1	x,x	1,x
1,1	x,x	x,x

x,x	**1,1**	x,x
x,1	x,x	1,x
1,x	x,1	x,x

x,x	**1,1**	x,x
x,1	x,x	1,x
1,x	x,x	x,1

x,x	**1,1**	x,x
x,x	x,1	1,x
1,1	x,x	x,x

x,x	**1,1**	x,x
x,x	x,1	1,x
1,x	x,1	x,x

x,x	**1,1**	x,x
x,x	x,1	1,x
1,x	x,x	x,1

x,x	**1,1**	x,x
x,x	x,x	**1,1**
1,1	x,x	x,x

x,x	**1,1**	x,x
x,x	x,x	**1,1**
1,x	x,1	x,x

x,x	**1,1**	x,x
x,x	x,x	**1,1**
1,x	x,x	x,1

x,x	1,x	x,1
x,1	x,x	1,x
1,1	x,x	x,x

x,x	1,x	x,1
x,1	x,x	1,x
1,x	x,1	x,x

x,x	1,x	x,1
x,1	x,x	1,x
1,x	x,x	x,1

x,x	1,x	x,1
x,x	x,1	1,x
1,1	x,x	x,x

x,x	1,x	x,1
x,x	x,1	1,x
1,x	x,1	x,x

x,x	1,x	x,1
x,x	x,1	1,x
1,x	x,x	x,1

x,x	1,x	x,1
x,x	x,x	**1,1**
1,1	x,x	x,x

x,x	1,x	x,1
x,x	x,x	**1,1**
1,x	x,1	x,x

x,x	1,x	x,1
x,x	x,x	**1,1**
1,x	x,x	x,1

x,1	1,x	x,x
x,1	x,x	x,x
1,1	x,x	1,x

x,1	1,x	x,x
x,1	x,x	x,x
1,x	x,1	1,x

x,1	1,x	x,x
x,1	x,x	x,x
1,x	x,x	**1,1**

x,1	1,x	x,x
x,x	x,1	x,x
1,1	x,x	1,x

x,1	1,x	x,x
x,x	x,1	x,x
1,x	x,1	1,x

x,1	1,x	x,x
x,x	x,1	x,x
1,x	x,x	**1,1**

x,1	1,x	x,x
x,x	x,x	x,1
1,1	x,x	1,x

x,1	1,x	x,x
x,x	x,x	x,1
1,x	x,1	1,x

x,1	1,x	x,x
x,x	x,x	x,1
1,x	x,x	**1,1**

x,x	**1,1**	x,x
x,1	x,x	x,x
1,1	x,x	1,x

x,x	**1,1**	x,x
x,1	x,x	x,x
1,x	x,1	1,x

x,x	**1,1**	x,x
x,1	x,x	x,x
1,x	x,x	**1,1**

x,x	**1,1**	x,x
x,x	x,1	x,x
1,1	x,x	1,x

x,x	**1,1**	x,x
x,x	x,1	x,x
1,x	x,1	1,x

x,x	**1,1**	x,x
x,x	x,1	x,x
1,x	x,x	**1,1**

x,x	**1,1**	x,x
x,x	x,x	x,1
1,1	x,x	1,x

x,x	**1,1**	x,x
x,x	x,x	x,1
1,x	x,1	1,x

x,x	**1,1**	x,x
x,x	x,x	x,1
1,x	x,x	**1,1**

x,x	1,x	x,1
x,1	x,x	x,x
1,1	x,x	1,x

x,x	1,x	x,1
x,1	x,x	x,x
1,x	x,1	1,x

x,x	1,x	x,1
x,1	x,x	x,x
1,x	x,x	**1,1**

x,x	1,x	x,1
x,x	x,1	x,x
1,1	x,x	1,x

x,x	1,x	x,1
x,x	x,1	x,x
1,x	x,1	1,x

x,x	1,x	x,1
x,x	x,1	x,x
1,x	x,x	**1,1**

x,x	1,x	x,1
x,x	x,x	x,1
1,1	x,x	1,x

x,x	1,x	x,1
x,x	x,x	x,1
1,x	x,1	1,x

x,x	1,x	x,1
x,x	x,x	x,1
1,x	x,x	**1,1**

x,1	x,x	1,x
x,1	1,x	x,x
1,1	x,x	x,x

x,1	x,x	1,x
x,1	1,x	x,x
1,x	x,1	x,x

x,1	x,x	1,x
x,1	1,x	x,x
1,x	x,x	x,1

x,1	x,x	1,x
x,x	**1,1**	x,x
1,1	x,x	x,x

x,1	x,x	1,x
x,x	**1,1**	x,x
1,x	x,1	x,x

x,1	x,x	1,x
x,x	**1,1**	x,x
1,x	x,x	x,1

x,1	x,x	1,x
x,x	1,x	x,1
1,1	x,x	x,x

x,1	x,x	1,x
x,x	1,x	x,1
1,x	x,1	x,x

x,1	x,x	1,x
x,x	1,x	x,1
1,x	x,x	x,1

x,x	x,1	1,x
x,1	1,x	x,x
1,1	x,x	x,x

x,x	x,1	1,x
x,1	1,x	x,x
1,x	x,1	x,x

x,x	x,1	1,x
x,1	1,x	x,x
1,x	x,x	x,1

x,x	x,1	1,x
x,x	**1,1**	x,x
1,1	x,x	x,x

x,x	x,1	1,x
x,x	**1,1**	x,x
1,x	x,1	x,x

x,x	x,1	1,x
x,x	**1,1**	x,x
1,x	x,x	x,1

x,x	x,1	1,x
x,x	1,x	x,1
1,1	x,x	x,x

x,x	x,1	1,x
x,x	1,x	x,1
1,x	x,1	x,x

x,x	x,1	1,x
x,x	1,x	x,1
1,x	x,x	x,1

x,x	x,x	**1,1**
x,1	1,x	x,x
1,1	x,x	x,x

x,x	x,x	**1,1**
x,1	1,x	x,x
1,x	x,1	x,x

x,x	x,x	**1,1**
x,1	1,x	x,x
1,x	x,x	x,1

x,x	x,x	**1,1**
x,x	**1,1**	x,x
1,1	x,x	x,x

x,x	x,x	**1,1**
x,x	**1,1**	x,x
1,x	x,1	x,x

x,x	x,x	**1,1**
x,x	**1,1**	x,x
1,x	x,x	x,1

x,x	x,x	**1,1**
x,x	1,x	x,1
1,1	x,x	x,x

x,x	x,x	**1,1**
x,x	1,x	x,1
1,x	x,1	x,x

x,x	x,x	**1,1**
x,x	1,x	x,1
1,x	x,x	x,1

x,1	x,x	x,x
x,1	1,x	1,x
1,1	x,x	x,x

x,1	x,x	x,x
x,1	1,x	1,x
1,x	x,1	x,x

x,1	x,x	x,x
x,1	1,x	1,x
1,x	x,x	x,1

x,1	x,x	x,x
x,x	**1,1**	1,x
1,1	x,x	x,x

x,1	x,x	x,x
x,x	**1,1**	1,x
1,x	x,1	x,x

x,1	x,x	x,x
x,x	**1,1**	1,x
1,x	x,x	x,1

x,1	x,x	x,x
x,x	1,x	**1,1**
1,1	x,x	x,x

x,1	x,x	x,x
x,x	1,x	**1,1**
1,x	x,1	x,x

x,1	x,x	x,x
x,x	1,x	**1,1**
1,x	x,x	x,1

x,x	x,1	x,x
x,1	1,x	1,x
1,1	x,x	x,x

x,x	x,1	x,x
x,1	1,x	1,x
1,x	x,1	x,x

x,x	x,1	x,x
x,1	1,x	1,x
1,x	x,x	x,1

x,x	x,1	x,x
x,x	**1,1**	1,x
1,1	x,x	x,x

x,x	x,1	x,x
x,x	**1,1**	1,x
1,x	x,1	x,x

x,x	x,1	x,x
x,x	**1,1**	1,x
1,x	x,x	x,1

x,x	x,1	x,x
x,x	1,x	**1,1**
1,1	x,x	x,x

x,x	x,1	x,x
x,x	1,x	**1,1**
1,x	x,1	x,x

x,x	x,1	x,x
x,x	1,x	**1,1**
1,x	x,x	x,1

x,x	x,x	x,1
x,1	1,x	1,x
1,1	x,x	x,x

x,x	x,x	x,1
x,1	1,x	1,x
1,x	x,1	x,x

x,x	x,x	x,1
x,1	1,x	1,x
1,x	x,x	x,1

x,x	x,x	x,1
x,x	**1,1**	1,x
1,1	x,x	x,x

x,x	x,x	x,1
x,x	**1,1**	1,x
1,x	x,1	x,x

x,x	x,x	x,1
x,x	**1,1**	1,x
1,x	x,x	x,1

x,x	x,x	x,1
x,x	1,x	**1,1**
1,1	x,x	x,x

x,x	x,x	x,1
x,x	1,x	**1,1**
1,x	x,1	x,x

x,x	x,x	x,1
x,x	1,x	**1,1**
1,x	x,x	x,1

x,1	x,x	x,x
x,1	1,x	x,x
1,1	x,x	1,x

x,1	x,x	x,x
x,1	1,x	x,x
1,x	x,1	1,x

x,1	x,x	x,x
x,1	1,x	x,x
1,x	x,x	**1,1**

x,1	x,x	x,x
x,x	**1,1**	x,x
1,1	x,x	1,x

x,1	x,x	x,x
x,x	**1,1**	x,x
1,x	x,1	1,x

x,1	x,x	x,x
x,x	**1,1**	x,x
1,x	x,x	**1,1**

x,1	x,x	x,x
x,x	1,x	x,1
1,1	x,x	1,x

x,1	x,x	x,x
x,x	1,x	x,1
1,x	x,1	1,x

x,1	x,x	x,x
x,x	1,x	x,1
1,x	x,x	**1,1**

x,x	x,1	x,x
x,1	1,x	x,x
1,1	x,x	1,x

x,x	x,1	x,x
x,1	1,x	x,x
1,x	x,1	1,x

x,x	x,1	x,x
x,1	1,x	x,x
1,x	x,x	**1,1**

x,x	x,1	x,x
x,x	**1,1**	x,x
1,1	x,x	1,x

x,x	x,1	x,x
x,x	**1,1**	x,x
1,x	x,1	1,x

x,x	x,1	x,x
x,x	**1,1**	x,x
1,x	x,x	**1,1**

x,x	x,1	x,x
x,x	1,x	x,1
1,1	x,x	1,x

x,x	x,1	x,x
x,x	1,x	x,1
1,x	x,1	1,x

x,x	x,1	x,x
x,x	1,x	x,1
1,x	x,x	**1,1**

x,x	x,x	x,1
x,1	1,x	x,x
1,1	x,x	1,x

x,x	x,x	x,1
x,1	1,x	x,x
1,x	x,1	1,x

x,x	x,x	x,1
x,1	1,x	x,x
1,x	x,x	**1,1**

x,x	x,x	x,1
x,x	**1,1**	x,x
1,1	x,x	1,x

x,x	x,x	x,1
x,x	**1,1**	x,x
1,x	x,1	1,x

x,x	x,x	x,1
x,x	**1,1**	x,x
1,x	x,x	**1,1**

x,x	x,x	x,1
x,x	1,x	x,1
1,1	x,x	1,x

x,x	x,x	x,1
x,x	1,x	x,1
1,x	x,1	1,x

x,x	x,x	x,1
x,x	1,x	x,1
1,x	x,x	**1,1**

x,1	x,x	1,x
x,1	x,x	x,x
1,1	1,x	x,x

x,1	x,x	1,x
x,1	x,x	x,x
1,x	**1,1**	x,x

x,1	x,x	1,x
x,1	x,x	x,x
1,x	1,x	x,1

x,1	x,x	1,x
x,x	x,1	x,x
1,1	1,x	x,x

x,1	x,x	1,x
x,x	x,1	x,x
1,x	**1,1**	x,x

x,1	x,x	1,x
x,x	x,1	x,x
1,x	1,x	x,1

x,1	x,x	1,x
x,x	x,x	x,1
1,1	1,x	x,x

x,1	x,x	1,x
x,x	x,x	x,1
1,x	**1,1**	x,x

x,1	x,x	1,x
x,x	x,x	x,1
1,x	1,x	x,1

x,x	x,1	1,x
x,1	x,x	x,x
1,1	1,x	x,x

x,x	x,1	1,x
x,1	x,x	x,x
1,x	**1,1**	x,x

x,x	x,1	1,x
x,1	x,x	x,x
1,x	1,x	X,1

x,x	x,1	1,x
x,x	x,1	x,x
1,1	1,x	x,x

x,x	x,1	1,x
x,x	x,1	x,x
1,x	**1,1**	x,x

x,x	x,1	1,x
x,x	x,1	x,x
1,x	1,x	x,1

x,x	x,1	1,x
x,x	x,x	x,1
1,1	1,x	x,x

x,x	x,1	1,x
x,x	x,x	x,1
1,x	**1,1**	x,x

x,x	x,1	1,x
x,x	x,x	x,1
1,x	1,x	x,1

x,x	x,x	**1,1**
x,1	x,x	x,x
1,1	1,x	x,x

x,x	x,x	**1,1**
x,1	x,x	x,x
1,x	**1,1**	x,x

x,x	x,x	**1,1**
x,1	x,x	x,x
1,x	1,x	x,1

x,x	x,x	**1,1**
x,x	x,1	x,x
1,1	1,x	x,x

x,x	x,x	**1,1**
x,x	x,1	x,x
1,x	**1,1**	x,x

x,x	x,x	**1,1**
x,x	x,1	x,x
1,x	1,x	x,1

x,x	x,x	**1,1**
x,x	x,x	x,1
1,1	1,x	x,x

x,x	x,x	**1,1**
x,x	x,x	x,1
1,x	**1,1**	x,x

x,x	x,x	**1,1**
x,x	x,x	x,1
1,x	1,x	x,1

x,1	x,x	x,x
x,1	x,x	1,x
1,1	1,x	x,x

x,1	x,x	x,x
x,1	x,x	1,x
1,x	**1,1**	x,x

x,1	x,x	x,x
x,1	x,x	1,x
1,x	1,x	x,1

x,1	x,x	x,x
x,x	x,1	1,x
1,1	1,x	x,x

x,1	x,x	x,x
x,x	x,1	1,x
1,x	**1,1**	x,x

x,1	x,x	x,x
x,x	x,1	1,x
1,x	1,x	x,1

x,1	x,x	x,x
x,x	x,x	**1,1**
1,1	1,x	x,x

x,1	x,x	x,x
x,x	x,x	**1,1**
1,x	**1,1**	x,x

x,1	x,x	x,x
x,x	x,x	**1,1**
1,x	1,x	x,1

x,x	x,1	x,x
x,1	x,x	1,x
1,1	1,x	x,x

x,x	x,1	x,x
x,1	x,x	1,x
1,x	**1,1**	x,x

x,x	x,1	x,x
x,1	x,x	1,x
1,x	1,x	x,1

x,x	x,1	x,x
x,x	x,1	1,x
1,1	1,x	x,x

x,x	x,1	x,x
x,x	x,1	1,x
1,x	**1,1**	x,x

x,x	x,1	x,x
x,x	x,1	1,x
1,x	1,x	x,1

x,x	x,1	x,x
x,x	x,x	**1,1**
1,1	1,x	x,x

x,x	x,1	x,x
x,x	x,x	**1,1**
1,x	**1,1**	x,x

x,x	x,1	x,x
x,x	x,x	**1,1**
1,x	1,x	x,1

x,x	x,x	x,1
x,1	x,x	1,x
1,1	1,x	x,x

x,x	x,x	x,1
x,1	x,x	1,x
1,x	**1,1**	x,x

x,x	x,x	x,1
x,1	x,x	1,x
1,x	1,x	x,1

x,x	x,x	x,1
x,x	x,1	1,x
1,1	1,x	x,x

x,x	x,x	x,1
x,x	x,1	1,x
1,x	**1,1**	x,x

x,x	x,x	x,1
x,x	x,1	1,x
1,x	1,x	x,1

x,x	x,x	x,1
x,x	x,x	**1,1**
1,1	1,x	x,x

x,x	x,x	x,1
x,x	x,x	**1,1**
1,x	**1,1**	x,x

x,x	x,x	x,1
x,x	x,x	**1,1**
1,x	1,x	x,1

x,1	x,x	x,x
x,1	x,x	x,x
1,1	1,x	1,x

x,1	x,x	x,x
x,1	x,x	x,x
1,x	**1,1**	1,x

x,1	x,x	x,x
x,1	x,x	x,x
1,x	1,x	**1,1**

x,1	x,x	x,x
x,x	x,1	x,x
1,1	1,x	1,x

x,1	x,x	x,x
x,x	x,1	x,x
1,x	**1,1**	1,x

x,1	x,x	x,x
x,x	x,1	x,x
1,x	1,x	**1,1**

x,1	x,x	x,x
x,x	x,x	x,1
1,1	1,x	1,x

x,1	x,x	x,x
x,x	x,x	x,1
1,x	**1,1**	1,x

x,1	x,x	x,x
x,x	x,x	x,1
1,x	1,x	**1,1**

x,x	x,1	x,x
x,1	x,x	x,x
1,1	1,x	1,x

x,x	x,1	x,x
x,1	x,x	x,x
1,x	**1,1**	1,x

x,x	x,1	x,x
x,1	x,x	x,x
1,x	1,x	**1,1**

x,x	x,1	x,x
x,x	x,1	x,x
1,1	1,x	1,x

x,x	x,1	x,x
x,x	x,1	x,x
1,x	**1,1**	1,x

x,x	x,1	x,x
x,x	x,1	x,x
1,x	1,x	**1,1**

x,x	x,1	x,x
x,x	x,x	x,1
1,1	1,x	1,x

x,x	x,1	x,x
x,x	x,x	x,1
1,x	**1,1**	1,x

x,x	x,1	x,x
x,x	x,x	x,1
1,x	1,x	**1,1**

x,x	x,x	x,1
x,1	x,x	x,x
1,1	1,x	1,x

x,x	x,x	x,1
x,1	x,x	x,x
1,x	**1,1**	1,x

x,x	x,x	x,1
x,1	x,x	x,x
1,x	1,x	**1,1**

x,x	x,x	x,1
x,x	x,1	x,x
1,1	1,x	1,x

x,x	x,x	x,1
x,x	x,1	x,x
1,x	**1,1**	1,x

x,x	x,x	x,1
x,x	x,1	x,x
1,x	1,x	**1,1**

x,x	x,x	x,1
x,x	x,x	x,1
1,1	1,x	1,x

x,x	x,x	x,1
x,x	x,x	x,1
1,x	**1,1**	1,x

x,x	x,x	x,1
x,x	x,x	x,1
1,x	1,x	**1,1**

APPENDIX B

NASH SOLUTION SEARCH ALGORITHM IMPLEMENTATION

The following is the program listing that implements the Nash solution search algorithm. The program is written in C.

```
#include <stdio.h>

#define unique_nash     0
#define non_unique_nash 1
#define no_nash         2

unsigned long num_unique_nash=0;
unsigned long num_non_unique_nash=0;
unsigned long num_no_nash=0;

static void p1_strategies(unsigned char *,unsigned char *,int,int,int);
static void p2_strategies(unsigned char *,unsigned char *,int,int,int);
static void set_location(unsigned char *,int,int,int,int);
static int categorize_game(unsigned char *,unsigned char *,int,int);
```

```
int main(   int         argc,
            char        *argv[])
{
      int idx;
      int num_rows, num_columns;
      unsigned long total_num_games;
      unsigned char *p1_game, *p2_game;

      if (argc != 3)
      {
            printf("Error, need two arguments\n");
            return(1);
      }

      num_rows = atoi(argv[1]);
      num_columns = atoi(argv[2]);

      p1_game = malloc(num_rows*num_columns*sizeof(unsigned char));
      p2_game = malloc(num_rows*num_columns*sizeof(unsigned char));

      p1_strategies(p1_game,p2_game,num_rows,num_columns,0);

      total_num_games = num_unique_nash + num_non_unique_nash +
                        num_no_nash;

      printf("Results for %dx%d top-rated preference minimal ordinal
                  games:\n",num_rows,num_columns);
      printf("-------------------------------------------------------
                  ---\n");
      printf("Total number of games: %d\n",total_num_games);
      printf("Number of games with a unique Nash solution:%d\n",
                  num_unique_nash);
      printf("Number of games with multiple non-unique Nash solutions:
                  %d\n",num_non_unique_nash);
      printf("Number of games with no Nash solution: %d\n",
                  num_no_nash);

      free(p1_game);
      free(p2_game);

      return(0);
}
```

```
static void p1_strategies(    unsigned char     *p1_game,
                              unsigned char     *p2_game,
                              int               num_rows,
                              int               num_columns,
                              int               column_idx)
{
      int row_idx;

      row_idx = 0;

      while (row_idx < num_rows)
      {
            set_location(p1_game,1,row_idx,column_idx,num_columns);
            if (column_idx == (num_columns-1))
            {
                  p2_strategies(p1_game,p2_game,num_rows,
                                    num_columns,0);
            }
            else
            {
                  p1_strategies(p1_game,p2_game,num_rows,num_columns,
                                    column_idx+1);
            }
            set_location(p1_game,0,row_idx,column_idx,num_columns);
            row_idx++;
      }
}
```

```
static void p2_strategies(    unsigned char     *p1_game,
                              unsigned char     *p2_game,
                              int               num_rows,
                              int               num_columns,
                              int               row_idx)
{
      int column_idx;
      int game_type;

      column_idx = 0;

      while (column_idx < num_columns)
      {
            set_location(p2_game,1,row_idx,column_idx,num_columns);
            if (row_idx == (num_rows-1))
            {
                  game_type = categorize_game(p1_game,p2_game,
                                                num_rows,num_columns);
                  if (game_type == unique_nash)
                  {
                        num_unique_nash++;
                  }
                  else if (game_type == non_unique_nash)
                  {
                        num_non_unique_nash++;
                  }
                  else
                  {
                        num_no_nash++;
                  }
            }
            else
            {
                  p2_strategies(p1_game,p2_game,num_rows,num_columns,
                                    row_idx+1);
            }
            set_location(p2_game,0,row_idx,column_idx,num_columns);
            column_idx++;
      }
}
```

```
static void set_location(     unsigned char     *game,
                              int               value,
                              int               row_idx,
                              int               column_idx,
                              int               num_columns)
{
      game[row_idx*num_columns+column_idx] = value;
}

static int categorize_game(   unsigned char     *p1_game,
                              unsigned char     *p2_game,
                              int               num_rows,
                              int               num_columns)
{
      int idx, num_nashes, game_size;

      game_size = num_rows*num_columns;
      num_nashes = 0;
      idx = 0;
      while (idx < game_size)
      {
            if ((p1_game[idx] == 1) && (p2_game[idx] == 1))
            {
                  num_nashes++;
            }
            idx++;
      }

      if (num_nashes > 1)
      {
            return(non_unique_nash);
      }
      else if (num_nashes == 1)
      {
            return(unique_nash);
      }
      else
      {
            return(no_nash);
      }
}
```

BIBLIOGRAPHY

1. Von Neumann, J., and Morgenstern, O., *The Theory of Games and Economic Behavior*, Princeton University Press, Princeton, New Jersey, 1947.

2. Issacs, R., *Differential Games*, John Wiley and Sons, Inc., New York, NY, 1965.

3. Starr, A. W., and Ho, Y. C., *Nonzero-Sum Differential Games*, Journal of Optimization Theory and Applications, Vol. 3, No. 3, pp.184 – 205, 1969.

4. Nash, J., *Noncooperative Games*, Annals of Mathematics, Vol. 54, pp. 286 – 295, 1951.

5. Basar, T., and Olsder, G. J., *Dynamic Noncooperative Game Theory*, Academic Press, London, 1995.

6. Ho, Y. C., Srinivas, R.S., and Vakili, P., Ordinal Optimization of DEDS, Discrete Events Dynamic Systems, Vol. 2, pp. 61-88, 1992.

7. Saaty, T. L., *The Analytic Hierarchy Process*, McGraw-Hill, New York, NY, 1980.

8. Saaty, T. L., and Vargas, L. G., *The logic of Priorities*, Kluwer Academic Publishers, Boston, Massachusetts, 1981.

9. Brams, S. J., *The Theory of Moves*, Cambridge University Press, New York, NY, 1994.

10. Cruz, J. B., Jr., and Simaan, M., *Ordinal Games and Generalized Nash and Stackelberg Solutions*, Journal of Optimization Theory and Applications, Vol 107, No. 2, pp. 205 – 222, 2000.

11. Rapoport, A., Guyer, M. J., and Gordon, D. G., *The 2 x 2 Game*, University of Michigan Press, Ann Arbor, MI, 1976.

12. Simaan, M., and Cruz, J. B., Jr., *On the Stackelberg Strategy in Nonzero-Sum Games*, Journal of Optimization Theory and Applications, Vol 11, No. 5, pp. 533 – 554, 1973.

13. Basar, T., *On the Relative Leadership Property of Stackelberg Strategies*, Journal of Optimization Theory and Applications, Vol. 11, No. 6, pp. 655 – 661, 1973.