Domineering games with minimal number of moves

Major Project-II

Submitted to Delhi Technological University in partial fulfillment of the requirements for the award of the degree of
bachelor of technology
in
Mathematics and Computing

(2016-2020)

Submitted by:
Rohan Karthikeyan (2k16/MC/065)
Siddharth Sinha (2k16/MC/077)

Under the supervision of:
Dr. C. P. Singh
Associate Professor

Department of Applied Mathematics

Delhi Technological University
(Formerly Delhi College of Engineering)
Bawana Road, Delhi - 110042, India

May 2020

Domineering games with minimal number of moves

Rohan Karthikeyan, Siddharth Sinha *

May 25, 2020

*Department of Applied Mathematics, Delhi Technological University, Delhi, India

Contents

Declaration by the Authors

We, Rohan Karthikeyan (2k16/MC/065) and Siddharth Sinha (2k16/MC/077), students of B. Tech Mathematics and Computing, hereby declare that the project dissertation entitled "**Domineering games with minimal number of moves**" which is submitted by us to the Department of Applied Mathematics, Delhi Technological University, Delhi in partial fulfilment of the award of the degree of Bachelor of Technology, is original and not copied from any source without proper citation. This work has not previously formed the basis for the award of any Degree, Diploma Associateship, Fellowship or other similar title or recognition.

Rohan Karthikeyan
2k16/MC/065

Siddharth Sinha
2k16/MC/077

Bonafide Certificate

I hereby certify that this project dissertation entitled "**Domineering games with minimal number of moves**" submitted by Rohan Karthikeyan (2k16/MC/065) and Siddharth Sinha (2k16/MC/077) in partial fulfilment of the requirement for the award of the degree of Bachelor of technology in Mathematics and Computing to the Department of Applied Mathematics is a bonafide record of work done by the students under my supervision during the 8th semester. To the best of my knowledge this work has not been submitted in part or full for any degree or diploma in this university or elsewhere.

Dr. C. P. Singh
Associate Professor
Department of Applied Mathematics
Delhi Technological University

Acknowledgments

Our sincere thanks go first and foremost to Dr. Chandra Prakash Singh, our thesis advisor, who constantly encouraged us by his valuable advice on every aspect of the preparation of this thesis. Another person most deserving of our thanks is Dr. Samrith Ram of IIIT Delhi who encouraged us to visit him at the start of the year introducing us to the game of Domineering and from it, combinatorial game theory. Last but not the least, we sincerely thank our parents who understood the challenges that were before us and provided moral support throughout.

Abstract

Domineering is a 2-player game played on a complete $m \times n$ rectangular board, where one player places a domino vertically, and the second player horizontally. In this thesis, we obtain the minimum number of moves for a Domineering game to end on several rectangular $m \times n$ boards. This follows as a natural extension of counting domineering positions with certain special properties.

We also show how a forerunner to the sensitivity conjecture inspired this work on Domineering. In addition, we formulate two conjectures pertaining to $2 \times n$ (similarly $m \times 2$) and on $3 \times n$ (similarly, $m \times 3$) boards.

List of Figures

List of Tables

1 Introduction

Combinatorial games are 2-player games with perfect information and no chance such as chess or Go. Many combinatorial games have, for a fixed starting position, a finite number of options and the game is guaranteed to end in a finite number of moves; in theory we could determine by computer which player would win if both players play perfectly. In practice game theorists and computer scientists haven't determined the outcomes of games under perfect play because of the complexity of the required search.

In this paper, we consider the game **Domineering**. Also called Dominoes or Crosscram in older work, this game was created by Göran Andersson in the 1970s and then introduced to the public by Martin Gardner [11]. Both scientists first considered Domineering on the 8×8 board.

Play proceeds by two players taking turns placing a 1×2 tile (called domino) on adjacent empty squares; as a convention, Left places the tile vertically and Right places the tile horizontally. The game finishes when the player who cannot place a tile loses - this is the *normal play* condition. Since the board is gradually filled, Domineering is a converging game, the game always ends, and ties are impossible.

We will continue this section with an overview of our previous work on the sensitivity conjecture [1] and how we were led to analyse Domineering games. After delving into the basics of Combinatorial game theory, we'll then look at Huntemann et al.'s [2] result of counting the Domineering positions satisfying certain properties. We then use their results to obtain values for the minimum number of moves for a $m \times n$ Domineering game to end, before finishing with lines of future attack and our program code.

1.1 On Domineering and the sensitivity conjecture

In this section, we will look at how we proceeded to study Domineering games using minimum moves from a graph-theoretic beginning provided to us by our work on the sensitivity conjecture. We will first look at the necessary background before proceeding to describe a combinatorial game that follows naturally.

1.1.1 On the sensitivity conjecture

Boolean functions map a sequence of n bits to a single bit vector 0 or 1, represented as False and True respectively. Some of the simplest Boolean functions are the AND function $x \cdot y$, the OR function (non-exclusive) $x+y$, and the NOT function $\bar{x} = 1-x$.

There are several useful measures to describe the complexity of a Boolean function that can be stated mathematically. Two such measures are sensitivity and block sensitivity.

Definition 1. The **sensitivity** $s(f)$ of a Boolean function $f : \{0,1\}^n \to \{0,1\}$ on n variables is defined to be the largest number for which there is an x such that

there are at least $s(f)$ values of $i = 1, \cdots, n$ with $f(x + e_i) \neq f(x)$. Here e_i is the Boolean vector with exactly one 1 in the i-th position.

Definition 2. The **block sensitivity**, $bs(f)$ is defined to be the maximum number of disjoint subsets of $B_1, \cdots, B_t$ of $[n] = \{1, \cdots, n\}$ such that for all $1 \leq j \leq t, f(x) \neq f(x^{B_j})$ where x^{B_j} is the Boolean string which differs from x on exactly the bits of B_j.

Block sensitivity is known to be polynomially related to a number of other complexity measures of f, including the decision-tree complexity, and the certificate complexity. A long-standing open question that existed was whether sensitivity also belonged to this equivalence class. A positive answer to this question is commonly known as the *Sensitivity Conjecture* first proposed by Nisan and Szegedy.

Conjecture 1.1 (Sensitivity Conjecture)**.** *There exists an absolute constant $C > 0$, such that for every Boolean function f,*

$$bs(f) \leq s(f)^C$$

It may be useful to think of the sensitivity of a Boolean function in the following way: Take an array of n switches, with some wiring for a single light bulb. For different configurations of switches flipped on or off, the light bulb is either on, or off. Then the circuit is said to be sensitive, with respect to the i-th switch if for some configuration of the states of the switches, flipping the i-th switch changes the state of the light bulb from on to off, or vice versa. For that configuration, there may be more than one switch for which the circuit is sensitive. If we count the number of such switches at which the circuit is sensitive, and do so for every configuration of the switches, then the greatest number of switches for which the circuit was sensitive for some configuration, is the sensitivity $s(f)$, where f is the Boolean function (ON/OFF) expressed by the circuit.

The Sensitivity Conjecture can also be seen as concerning whether sensitivity is polynomially related to n, the number of input variables to the Boolean function. To resolve this, Gotsman and Linial proved the equivalence between the following two problems:

1. Denote the n-dimensional cube by $Q_n = \{-1, 1\}^n$ and the maximal degree of any graph G by $\Delta(G)$. For an induced subgraph G of Q_n with strictly greater than half the number of vertices, i.e., greater than 2^{n-1} vertices, find an effective lower bound for $\Delta(G)$ in terms of n.
2. Let $f : Q_n \to \{-1, 1\}$ be a Boolean function, with sensitivity $s(f)$. Denoting the degree of the multilinear polynomial representation of $f(x)$ by $\deg(f)$, find an effective upper bound for $\deg(f)$ in terms of $s(f)$.

It was previously established by Chung, Furedi, Graham and Seymour [10] that for an induced sub-graph G of Q_n with strictly more than half its vertices, the degree of some vertex v in G is bounded above logarithmically in n. Hao [3] recently improved this logarithmic bound to a polynomial relation.

Theorem 1.2 (Hao)**.** *For every integer $n \geq 1$, let G be an arbitrary $(2^{n-1}+1)$-vertex induced subgraph of Q_n, then*

$$\Delta(G) \geq \sqrt{n}$$

This result proves that the sensitivity and degree of a Boolean function are polynomially related, thereby confirming the Sensitivity Conjecture.

1.1.2 A combinatorial game

Now, Chung et al. [10] also constructed a $(2^{n-1}+1)$-vertex induced subgraph whose maximum degree is $\lceil \sqrt{n} \rceil$. This result essentially tells us that when we choose a selection of vertices from a hypercube that is greater than half of the total number of vertices, then we cannot avoid choosing vertices in a way that minimises the degree below $\lceil \sqrt{n} \rceil$. To demonstrate this principle, let us formulate this problem in the form of a game among two players.

Consider the vertices of the hypercube $Q_n = \{0,1\}^n$. There are two players who play a game alternately with the following rules:

1. Each player chooses a vertex on their turn. That vertex is then colored red.
2. Players can only choose vertices that aren't adjacent to any red vertex.
3. The game ends when a player is unable to choose a vertex with no adjacent vertices not colored red. The player on whose turn the game ends loses.

To illustrate, when the game is played on a standard 3-cube, we observe the game will always end after 4 moves, (or) 2 turns by each player, and the first player to move always wins. The number of moves also happens to be exactly half of the vertices of the cube. Looking at the induced subgraph of all the vertices colored red by the end of the game, we notice that the maximum degree of any vertex is zero.

If we color even one more vertex of the 3-cube, then we would get a vertex with a degree of 3 in the induced subgraph. This is because at the end of the game, we're left with only the vertices which are adjacent to at least one other vertex, or else the game would not be over, and the next player could make a move.

On close observation, we also notice that since the number of choices for a player to select a vertex is gradually reducing, this game is a converging game, the game always ends, and ties are impossible. With these properties, the game belongs to the category of *combinatorial games*, for which a whole theory (combinatorial game theory) has been developed.

2 Combinatorial Game Theory

2.1 Introduction

This section is adapted from [4]. Most of us enjoy playing combinatorial games, albeit without stopping to think of the mathematics behind them. Some such games we commonly play include Chess, Checkers and Go. But, it's interesting to note that despite these games having existed for millennia, the math that helps us determine the winner of a combinatorial game, or in general, develop a winning strategy for any player is very recent. This question was settled by Berlekamp, Conway and Guy only in the late 70s ([8], [9]). In this section, we outline some highlights of this theory.

A very important point to note first of all is that *combinatorial game theory* (CGT) is quite different from the "classical" game theory introduced by Von Neumann and Morgenstern. For the reader further interested in combinatorial game theory techniques, we recommend Albert et al.'s Lessons in Play [7] and Siegel's Combinatorial Game Theory [6].

Definition 3. (Combinatorial Game) In a combinatorial game, these limitations are set in place:

1. The game requires two players and two players only, called "Left" and "Right", who alternate moves. Also, nobody can miss his [1] turn.
2. Both players have knowledge of all possible moves in the game, that is, there is no hidden information.
3. The game should involve no chance, such as rolling dice or shuffling cards.
4. We define the rules of the game so that play will always complete.
5. The winner is decided by the game's final move: in **normal play convention**, the first player unable to move loses. In the **misère play convention**, the last player to move loses.

An unorthodox example of such games includes Nim [12]. In this game, positions are tuples of non-negative integers $(a_1, a_2, \cdots, a_n)$. A move consists in strictly decreasing exactly one of the a_i, $1 \leq i \leq n$, provided the resulting position remains valid. The first player who reaches the position $(0, 0, \cdots, 0)$ wins the game.

2.2 Questions in CGT

For any combinatorial game, we're usually on the hunt to find answers to these three questions:

1. Who wins the game?
2. What value does the game have (in Conway's notation)?
3. Is it possible to furnish a winning strategy for any player, i.e, a sequence of optimal moves for the winner, regardless of the other player's moves?

[1] For brevity, we use 'he' and 'his' where 'he or she' and 'his or her' is meant.

Let's briefly touch upon these problems and if possible, provide details specific to Domineering. We mention, that there are many important properties for Domineering other than these as well, such as finding temperatures (essentially, a measure of the importance of the next move).

Problem 1 (Outcome). Given a game $\mathcal{G}$ with a starting position S, find out whether $(\mathcal{G}, S)$ is $\mathcal{P}, \mathcal{N}, \mathcal{L}, \mathcal{R}$.

This question deals with determining the game's winner, also called *outcome*. One important point to note is that we are usually concerned with who wins if both players *play perfectly*. What does this term mean? One feature of playing perfectly is visible: if a player can force a win, then he makes a move that allows him to win. However, what if there's no such move? In practice, it's then good play to tie the opponent in ropes. We formalize this in the following theorem:

Theorem 2.1. *(Fundamental Theorem of Combinatorial Games) Fix a game G played between players Left and Right, with Left moving first. Either Left can force a win moving first, or Right can force a win moving second, but not both.*

Proof. Every move of Left is to a position which, by induction, is either a first-player Right win or a second-player Left win. Left can win by choosing any move (if there's one) in the latter category. If there's no such move available, then Right wins by using his winning strategy. □

Now, this result accommodates four possibilities, helping us to categorize positions into one of four outcome classes:

1. $\mathcal{L}$ if Left can win (has a winning strategy) regardless of who moves first,
2. $\mathcal{R}$ if Right can win regardless of who moves first,
3. $\mathcal{N}$ if the first player (the *N*ext player) can force a win,
4. $\mathcal{P}$ if the second player (the *P*revious player) can force a win.

The following table can serve as a useful guide to view these outcome classes:

Outcome classes		*When Right moves first*	
		Right wins	Left wins
When Left moves first	Left wins	$\mathcal{N}$	$\mathcal{L}$
	Right wins	$\mathcal{R}$	$\mathcal{P}$

Example 2.2. *Consider Domineering played on $3 \times n$ grids. If $n = 1$, because Right has no (horizontal) moves, the grid is clearly $\mathcal{L}$. If $n = 2$ or 3, we can quickly check that the first player has a winning strategy, thus giving $\mathcal{N}$-positions. When $n > 3$, we can prove that $3 \times n$ grids are $\mathcal{R}$, since placing an horizontal domino in the middle row allows two free moves for Right, whereas a vertical move doesn't contain constrain further moves of Left.*

Problem 2 (Value). Given a game $\mathcal{G}$ with a starting position S, compute its Conway's value.

It was Conway [9] who first defined the concept of a game. In this theory, every game position is assigned a numeric value among the set of *surreal numbers*. We can characterize this value as the number of moves ahead that Left has towards his opponent. A more formal definition can be found in [6]. Now, as a game of Domineering progresses, it often naturally breaks into smaller components (Figure 2.1). Consequently, we ask how we can exploit the decomposition. This leads us to:

Definition 4. (Game Sum) A sum of two or more game positions is the position obtained by placing the game positions side by side. When it is your move, you can make a single move in a summand of your choice. As usual, the last person to move wins.

A player on their turn has to decide which component they want to play in and make their move there. This is called the **disjunctive sum** of these components.

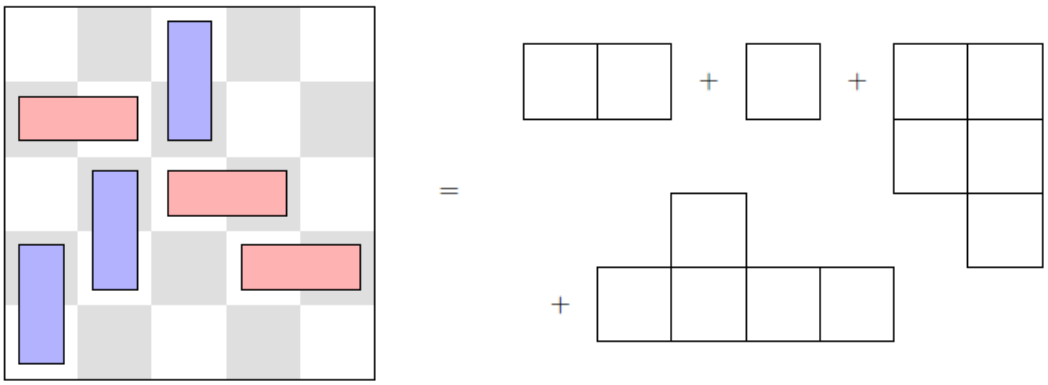

Figure 2.1: Sum of Domineering positions

Example 2.3. *Rather surprisingly, there are no formulas to compute Conway's values for even small Domineering boards.*

Problem 3 (Winning Strategy). Given a game $\mathcal{G}$ and a starting position S, give a winning move from S for the player having a winning strategy.

Once we're able to ascertain the game's outcome class, we look to map out the corresponding winning strategy.

3 Counting Domineering positions

This section is adapted from [2]. We define the following terms helping us to enumerate positions at the end of a game:

Definition 5. A **Right end** position is a position in which Left potentially has moves available but Right has no moves; a **Left end** position is defined similarly.

Definition 6. A **maximal end** position is a position where no player can place a domino, i.e., it's both a left end and right end.

3.1 Maximal End Positions

To count the maximal Domineering positions on a $m \times n$ board, we find the generating function

$$F_{m,n}(x, y) = \sum f(a, b)x^a y^b$$

where $f(a, b)$ is the number of maximal Domineering positions with a vertical (placed by Left) and b horizontal (placed by Right) dominoes.

To accomplish this, we tile rectangles using tiles with edge labels, then count the tilings that correspond to positions. We use the tiles in Figure 3.1.

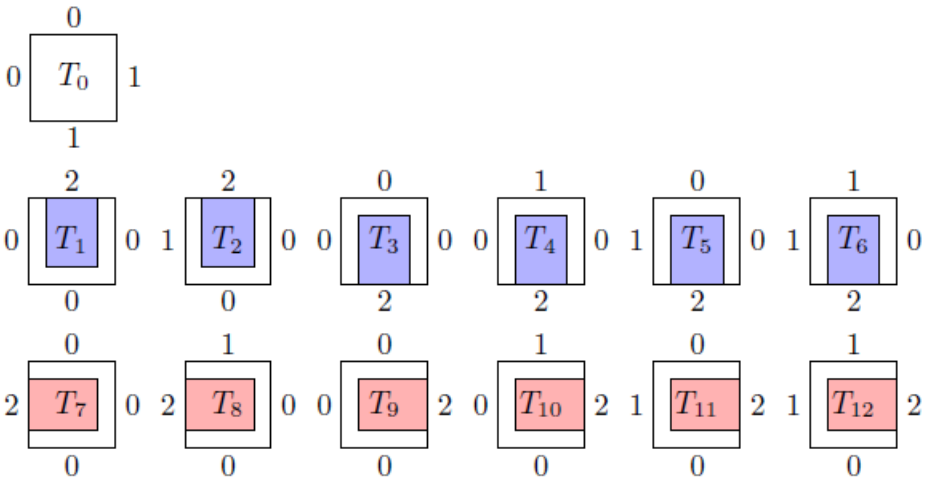

Figure 3.1: Tiles used to count maximal Domineering positions

Next, two conditions are required to determine when a rectangular tiling forms a maximal Domineering position:

1. *Adjacency condition*: All shared edges of adjacent tiles have the same label.
2. *Boundary condition*: The left and top boundary edges of the tiling have label 0; the right and bottom boundary edges have label 0 or 1.

Any maximal Domineering position can be uniquely represented using such a tiling. Figure 3.2 shows a maximal Domineering position and its equivalent tiling.

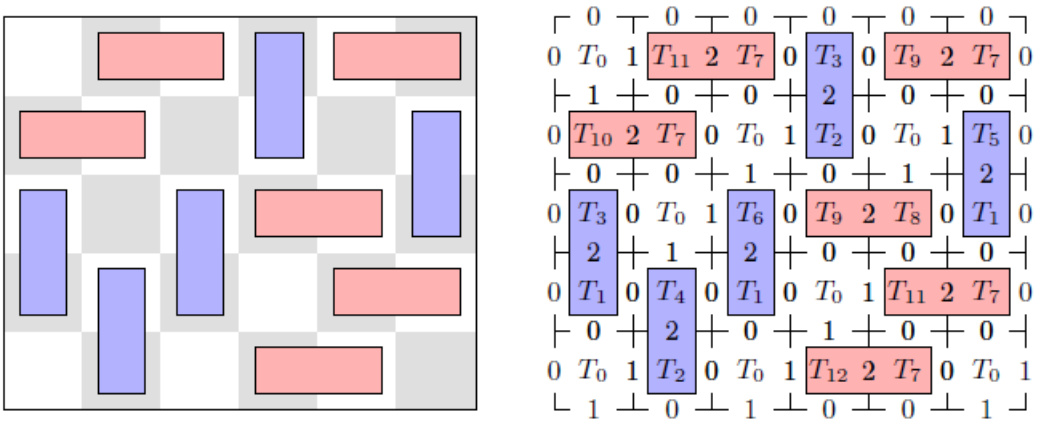

Figure 3.2: A maximal Domineering position and its equivalent tiling

Note that the number of Left dominoes equals the combined number of T_1 and T_2 tiles, while that of the Right dominoes equals the combined number of T_7 and T_8 tiles.

We define a **mosaic** to be a tiling that satisfies the adjacency condition, but not necessarily the boundary condition. A $1 \times q$ mosaic, i.e., a row of q tiles is called a **bar mosaic**.

We find generating polynomials by first counting bar mosaics. We then stack these bars to count rectangular mosaics; in this we consider the adjacency condition for vertically adjacent tiles. Finally, we restrict to those tilings that also satisfy the boundary condition. We will count mosaics instead of just Domineering tilings because a row or column of a Domineering rectangle isn't usually a Domineering position in itself.

The enumeration of bar mosaics will be done using the so-called bar-state matrices. We take care to construct these matrices so that we can count rectangular mosaics by matrix multiplication. The enumeration of the bar mosaics will be done recursively — we use the enumeration of bar mosaics to enumerate longer bar mosaics — so our matrices are recursively defined (using blocks).

Theorem 3.1. *The generating function for the maximal position of an $m \times n$ Domineering board is*

$$F_{m,n}(x,y) = \sum_{u \in \{0,1\}^n} \left(M_{0,n} + M'_{0,n} \right)^m (1 + \sum_{i=1}^{n} u_i 3^i, 1)$$

where $M_{0,0} = [1], M_{1,0} = [0], M_{2,0} = [0], M'_{0,0} = [0], M'_{1,0} = [1], M'_{2,0} = [0],$

$$M_{0,\,(q+1)} = \begin{bmatrix} M_{2,q} & M_{2,q} & xM_{0,q} \\ M_{1,q} & \mathbf{0} & \mathbf{0} \\ M_{0,q} & M_{0,q} & \mathbf{0} \end{bmatrix}, \quad M'_{0,\,(q+1)} = \begin{bmatrix} M'_{2,q} & M'_{2,q} & xM'_{0,q} \\ M'_{1,q} & \mathbf{0} & \mathbf{0} \\ M'_{0,q} & M'_{0,q} & \mathbf{0} \end{bmatrix}$$

$$M_{1,(q+1)} = \begin{bmatrix} M_{2,q} & M_{2,q} & xM_{0,q} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ M_{0,q} & M_{0,q} & \mathbf{0} \end{bmatrix}, \; M'_{1,(q+1)} = \begin{bmatrix} M'_{2,q} & M'_{2,q} & xM'_{0,q} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ M'_{0,q} & M'_{0,q} & \mathbf{0} \end{bmatrix}$$

$$M_{2,(q+1)} = \begin{bmatrix} yM_{0,q} & yM_{0,q} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}, \textit{and } M'_{2,(q+1)} = \begin{bmatrix} yM'_{0,q} & yM'_{0,q} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}$$

Proof. Note that the bar mosaics of length q are counted by the bar-state matrices. The matrix $M_{k,q}$ counts bars with starting label k and ending label 0, while the matrix $M'_{k,q}$ counts those with starting label k and ending label 1. Note the labels will be ternary strings, ordered lexicographically. Thus, the strings of length 2 are ordered as:

$$00, 01, 02, 10, 11, 12, 20, 21, 22$$

The column labels of the matrix correspond to the bar mosaic's top label, while the row label is its bottom label. Each bar is represented by a monomial $x^a y^b$ where a is the total number of T_1 and T_2 tiles in the bar and b is the total number of T_7 and T_8 tiles in the bar.

Each matrix has 9 blocks, labelled using roman numerals as shown in Figure 3.3, where the corresponding top and bottom labels for the leftmost tile are shown.

bottom \ top	0	1	2
0	I	II	III
1	IV	V	VI
2	VII	IIX	IX

Figure 3.3: The block construction of the state-matrices for maximal Domineering positions

In $M_{0,(q+1)}$, we have a left label of 0. The only tile meeting the conditions for block I (and $M_{0,(q+1)}$) is T_9. As T_9's right edge label is 2, the possible completions of our bar are given by $M_{2,q}$ and so our block I entry is $M_{2,q}$. The conditions for block II force T_{10} as the starting tile and the entry is again $M_{2,q}$. The conditions for block III force T_1 - giving the entry $xM_{0,q}$ as T_1 has right label of 0. For block IV, tile T_0 is the only possible starting tile, giving entry $M_{1,q}$. For blocks V and VI, we need a tile with bottom label 1 and top label 1 or 2 respectively, but no such tiles exist, so the entry is $\mathbf{0}$. The conditions for block VII are top label of 0 and a bottom label of 2; the only tile that works is T_3 and hence the entry is $M_{0,q}$. The conditions for block $VIII$ are similar but with top label 1, so T_4 is the only tile that works. The conditions for block IX are a top and a bottom label of 2 which never occurs and so the entry is $\mathbf{0}$.

In $M_{1,(q+1)}$, we have a starting (left) label of 1. This forces T_{11} in block I; T_{12} in block II; T_2 in block III; T_5 in block VII and T_6 in block $VIII$. The remaining

blocks have no possible tiles.

In $M_{2,(q+1)}$, we have a starting (left) label of 2. Thus, the only possible tiles are T_7 and T_8. In block I, the only possible tile is T_7 giving $yM_{0,q}$. In block II the tile is T_8 giving $yM_{0,q}$. All other blocks have no possible tiles, so are $\mathbf{0}$.

Note that the arguments for $M'_{0,(q+1)}, M'_{1,(q+1)}, M'_{2,(q+1)}$ are similar as the possible leftmost tile is identical and only the ending label changes.

Now restricting to tilings, we'll only consider bar mosaics of length n with left label 0 and right labels 0 or 1 due to the boundary condition. So, we only consider $M_{0,n} + M'_{0,n}$. Because stacking the bar mosaics corresponds to matrix multiplication, the mosaics with all left labels 0 and right labels 0 or 1 are enumerated in $(M_{0,n}+M'_{0,n})^m$. Finally, we need to restrict ourselves to top labels being all 0 and bottom labels 0 or 1. This means we need to sum the entries in the first column which are in rows numbered $1 + \sum_{i=1}^{n} u_i 3^i$ where the u_i are all 0 or 1 (because the ternary expansion of the row number -1 has no 2s), giving our result. □

3.2 Left and Right End Positions

For counting the Right end Domineering positions on a $m \times n$ board, we find the generating function

$$R_{m,n}(x, y) = \sum r(a, b)x^a y^b$$

where $r(a, b)$ is the number of Right end Domineering positions with a vertical (placed by Left) and b horizontal (placed by Right) dominoes. The generating polynomial for Left end positions on an $m \times n$ board, which we denote by $L_{m,n}(x, y)$, can be found by obtaining $R_{n,m}(x, y)$ and then switching x and y.

For Right ends, we may have two empty squares vertically adjacent as Left may potentially still place a domino. But we must not have two empty squares horizontally adjacent as Right must have no moves. We use the tiles in Figure 3.4.

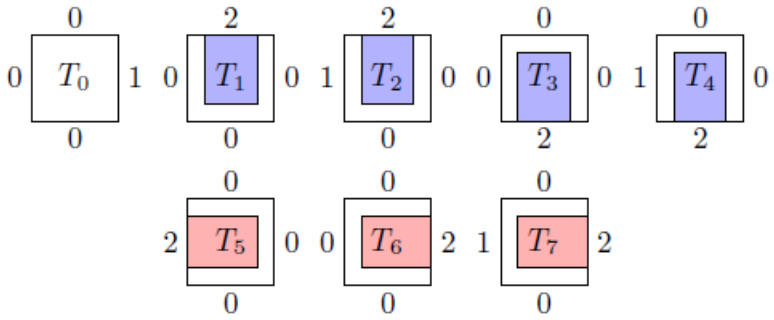

Figure 3.4: Tiles used to count Right Domineering positions

Next, two conditions are required to determine when a rectangular tiling forms a right end Domineering position:

1. *Adjacency condition*: All shared edges of adjacent tiles have the same label.
2. *Boundary condition*: The left, top and bottom edges of the tiling have label 0; the right edge has labels 0 or 1.

In turn, any right Domineering position can be uniquely represented using such a tiling. Figure 3.5 shows a right Domineering position and its equivalent tiling.

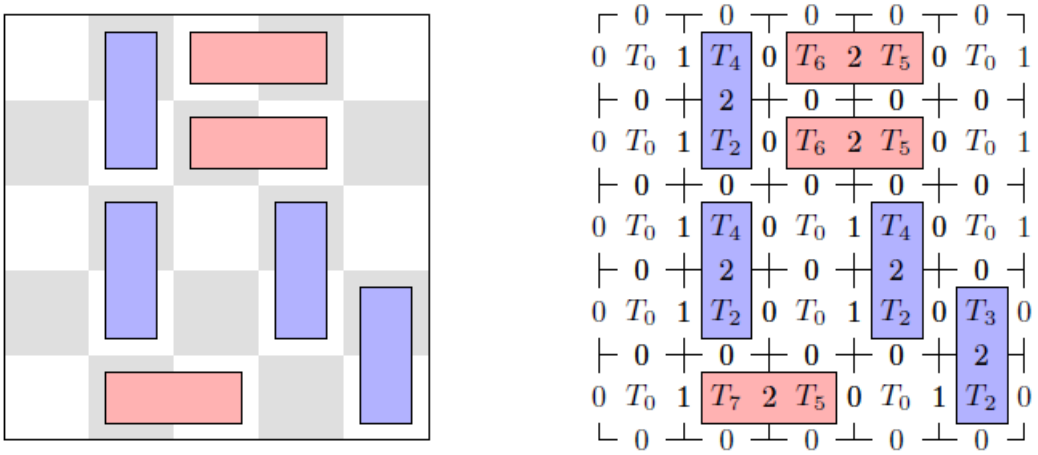

Figure 3.5: A Right Domineering position and its equivalent tiling

We will use the same technique as for counting maximal end Domineering positions of counting bar mosaics, multiplying the matrices, and then restricting to tilings, with only a few small differences. We use R for naming the bar-state matrices. The matrix $R_{k,q}$ counts those bar mosaics of length q with starting label k and ending label 0, while the matrix $R'_{k,q}$ counts those with starting label k and ending label 1.

Theorem 3.2. *The generating polynomial of Domineering Right ends on an $m \times n$ board is the $(1,1)$ entry of $(R_{0,n} + R'_{0,n})^m$, denoted by $R_{m,n}(x,y)$, where* $R_{0,0} = [1], R_{1,0} = [0], R_{2,0} = [0], R'_{0,0} = [0], R'_{1,0} = [1], R'_{2,0} = [0]$,

$$R_{0,\,(q+1)} = \begin{bmatrix} R_{1,q} + R_{2,q} & xR_{0,q} \\ R_{0,q} & \mathbf{0} \end{bmatrix}, \; R'_{0,\,(q+1)} = \begin{bmatrix} R'_{1,q} + R'_{2,q} & xR'_{0,q} \\ R'_{0,q} & \mathbf{0} \end{bmatrix}$$

$$R_{1,\,(q+1)} = \begin{bmatrix} R_{2,q} & xR_{0,q} \\ R_{0,q} & \mathbf{0} \end{bmatrix}, \; R'_{1,\,(q+1)} = \begin{bmatrix} R'_{2,q} & xR'_{0,q} \\ R'_{0,q} & \mathbf{0} \end{bmatrix}$$

$$R_{2,\,(q+1)} = \begin{bmatrix} yR_{0,q} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \textit{ and } R'_{2,\,(q+1)} = \begin{bmatrix} yR'_{0,q} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}$$

Proof. As the rows and the columns of a bar-state matrices are indexed by binary strings of length q, the lexical ordering of strings of length 3 is:

$$000, 001, 010, 011, 100, 101, 110, 111$$

The column labels of the matrix correspond to the top label of the bar mosaic, while the row label is the bottom label. Each matrix has 4 blocks, labelled using roman numerals as shown in Figure 3.6, where the corresponding top and bottom labels for the leftmost tile are shown.

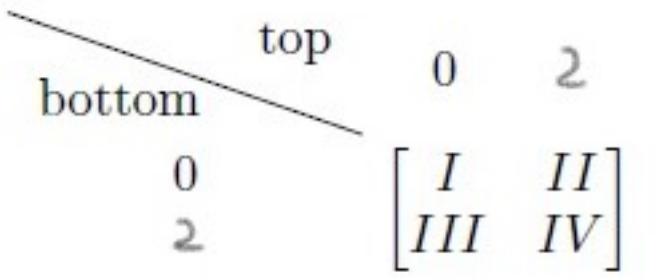

Figure 3.6: The block construction of the state-matrices for Right end Domineering positions

In $R_{0,(q+1)}$, we have a left label of 0. The only tile that meets the conditions for block I (and $R_{0,(q+1)}$) is T_0 and T_6 (with top and bottom labels 0). As the right edge label of T_0 is 1, the possible completions of our bar are given by $R_{1,q}$ and correspondingly, the possible completions of our bar using T_6 (with right edge label 2) is $R_{2,q}$ and so our block I entry is $R_{1,q} + R_{2,q}$. The conditions for block II (that the leftmost tile have top label 2 and bottom label 0) force T_1 as the starting tile and the entry is $xR_{0,q}$ because the right label of tile T_1 is 0 and $R_{0,q}$ counts the bar mosaics with left label 0 and right label 0. The conditions for block III force T_3 - giving the entry $R_{0,q}$ as T_3 has the right label of 0. The conditions for block IV are a top and a bottom label of 2 which never occurs and so the entry is $\mathbf{0}$.

In $R_{1,(q+1)}$, we have a starting (left) label of 1. This forces T_7 in block I; T_2 in block II; T_4 in block III. The conditions for block IV are a top and a bottom label of 2 which never occurs and so the entry is $\mathbf{0}$.

In $R_{2,(q+1)}$, we have a starting (left) label of 2. Thus, the only possible tile is T_5 giving $yR_{0,q}$. All other blocks, having no possible leftmost tiles, count no mosaics and thus their blocks are $\mathbf{0}$.

Note that the arguments for $R'_{0,(q+1)}, R'_{1,(q+1)}, R'_{2,(q+1)}$ are similar as the possible leftmost tile is identical and only the ending label changes.

Fixing the string of top labels and of bottom labels for the $2 \times n$ mosaic corresponds to specifying an entry in the matrix. To satisfy the boundary condition on the top and bottom, we need to restrict to all top and bottom labels 0. This means we take the entry in the top left corner, entry $(1, 1)$. □

4 Results

In this section, we state and discuss the results we've obtained and pose some problems not considered in the literature thus far.

One observes that the presence of a monomial of the form $x^a y^b$ in either the Left end or Right end or the Maximal end polynomials indicates the tiling of the $m \times n$ Domineering board with a vertical (placed by Left) tiles and b horizontal (placed by Right) tiles satisfying the required properties of the position.

It naturally follows that the minimum number of moves for a $m \times n$ Domineering game to end, which we denote by $\alpha_{m,n}$, equals the lowest sum of the degrees of x and y in the Left end, Right end and Maximal end polynomials. That is:

$$\alpha_{m,n} = \min\{a + b \, : x^a y^b \text{occurs in } L_{m,n}(x, y) \text{ or } R_{m,n}(x, y) \text{ or } F_{m,n}(x, y)\}$$

We have incorporated the matrix recurrence relations for the Left, Right and Maximal end positions defined previously into our program written in Maple. The program works by first determining the terms in the Left end, Right end and Maximal end polynomials where the difference between the powers of x and y is at most 1. That is because these polynomials actually count all legal Domineering positions satisfying the required properties. Then, we write another procedure to determine the polynomial(s) corresponding to $\alpha_{m,n}$. Additionally, the polynomials were analysed to check whether the term(s) occur in the Left and/or Right and/or Maximal end positions.

We have generated results up to all rectangular boards up to size 8×8 and some other rectangular boards. Beyond that point, the computations quickly grows too large to fit in the average home computer's main memory. An overview of the results is given in Table 4.1.

Some observations are in order:

1. The term with the least degree for all $m \times 1$ and $1 \times n$ boards with $m, n \geq 1$ is actually $x^0 y^0 = 1$. However, by the definition of a combinatorial game, someone must make a move. Hence, we take the next smallest term which is x for all $m \times 1$ boards with $m > 1$ and y for all $1 \times n$ boards with $n > 1$.
2. The left end polynomials $L_{m,n}(x, y)$ can be quickly determined for $m \ll n$ and small $m > 1$. Likewise, the right end polynomials $R_{m,n}(x, y)$ can be quickly determined for $n \ll m$ and small $n > 1$.
3. The minimum number of moves for a $m \times n$ Domineering game to end is the same as that for a $n \times m$ Domineering game to end, that is, $\alpha_{m,n} = \alpha_{n,m}$.
4. For large m, n, it seems that $\alpha_{m,n}$ occurs in both the Left end and the Right end polynomials.
5. It seems that the minimum number of moves is obtained by both players playing perfectly only for small m, n.

	n									
m	1	2	3	4	5	6	7	8	9	10
1	1^{LR}	1^{LR}	1^{LR}	1^{LR}	1^{LR}	1^{LR}	1^{LR}	1^{LR}	1^{LR}	1^{LR}
2	1^{LR}	1^{LR}	1^{R}	3^{LR}	3^{LR}	3^{R}	4^{R}	4^{R}	5^{R}	5^{R}
3	1^{LR}	1^{L}	2^{LR}	3^{LR}	3^{L}	4^{L}	5^{LR}	5^{R}	6^{L}	
4	1^{LR}	3^{LR}	3^{LR}	4^{LR}	5^{LR}	7^{LR}	7^{LR}	8^{LR}	9^{LR}	
5	1^{LR}	3^{LR}	3^{R}	5^{LR}	7^{LR}	8^{LR}	9^{LR}	11^{LR}		
6	1^{LR}	3^{L}	4^{R}	7^{LR}	8^{LR}	9^{LR}	11^{LR}	12^{LR}		
7	1^{LR}	4^{L}	5^{LR}	7^{LR}	9^{LR}	11^{LR}	12^{LR}	15^{LR}		
8	1^{LR}	4^{L}	5^{R}	8^{LR}	11^{LR}	12^{LR}	15^{LR}	16^{LR}		
9	1^{LR}	5^{L}	6^{R}	9^{LR}						
10	1^{LR}	5^{L}								

Table 4.1: The minimum number of moves for $m \times n$ Domineering games to end. The superscript over the numbers denote the polynomials (Left end and/or Right end and/or Maximal end) in which these numbers occur. For instance, the 2×4 game ends in 3 moves and this position occurs in both the Left end and Right end polynomials. Hence, the entry is denoted 3^{LR}.

6. We were unable to analyse the maximal end polynomials mainly because the computations quickly grow large even for small m, n.

On close observation, one can notice the following patterns, which to the best of our knowledge haven't been widely studied:

Problem 4.1. *The minimum number of moves for a $2 \times n$ Domineering game to end is given by*

$$\alpha_{2,n} = \left\lceil \frac{n}{2} \right\rceil \ \forall\, n \geq 5.$$

A similar result also holds for $m \times 2$ boards. One can easily construct Domineering positions that use the minimum number of moves shown in Table 4.1 on $2 \times n$ boards up to $n = 5$. However, we're unable to explicitly construct Domineering positions on $2 \times n$ boards with $n > 5$ using the minimum number of moves for the game to end.

Problem 4.2. *The minimum number of moves for a $3 \times n$ Domineering game to end is given by*

$$\alpha_{3,n} = \left\lfloor \frac{2n+1}{3} \right\rfloor \ \forall\, n \geq 1.$$

A similar result also holds for $m \times 3$ boards. We have been able to construct Domineering positions that use the least number of moves shown in Table 4.1 on $3 \times n$ boards up to $n = 6$.

5 Conclusions and Future research

We have analysed the minimum number of moves for a $m \times n$ Domineering game to end and formulated two new problems. We find four avenues to expand upon the research presented in this paper.

First, one could continue to find values for unsolved Domineering boards using better programs and/or more computer memory. We've started working in SageMath to analyze this problem.

Secondly, we intend to find the minimum number of moves for a $m \times n$ Domineering game to end if both players play perfectly.

Thirdly, we could try to analyse connections between the outcome classes of Domineering boards [5] and the results obtained in this paper.

Lastly, we could count game positions to analyze particular game situations either in theory or practice to develop better play strategies.

References

[1] R. Karthikeyan, S. Sinha, V. Patil, *On the resolution of the sensitivity conjecture*, Bull. Amer. Math. Soc. (N. S.), electronically published on March 27, 2020, DOI: https://doi.org/10.1090/bull/1697 (to appear in print).

[2] S. Huntemann, N. A. McKay, *Counting Domineering positions*, arXiv preprint, arXiv:1909.12419, 2019.

[3] Hao Huang. *Induced subgraphs of hypercubes and a proof of the Sensitivity Conjecture*. Annals of Mathematics, vol. 190, pp. 949-955, 2019.

[4] Eric Duchêne, *Combinatorial Games: From theoretical solving to AI Algorithms*, S. Schockaert and P. Senellart (Eds.): Scalable Uncertainty Management 2016, Lecture Notes in Computer Science, vol. 9858, pp. 3–17, 2016.

[5] G. C. Drummond-Cole, *An update on domineering on rectangular boards*, Integers, vol. 14, pp. 1-13, 2014.

[6] A. N. Siegel, *Combinatorial Game Theory*, Graduate Studies in Mathematics, vol. 146. American Mathematical Society, 2013.

[7] M.K. Albert, R.J. Nowakowski, D. Wolfe, *Lessons in Play: An Introduction to Combinatorial Game Theory*. A K Peters, 2007.

[8] E. Berlekamp, J.H. Conway, R.K. Guy, *Winning Ways for your Mathematical Plays*, vol. 1 (2001), vols. 2, 3 (2003), vol. 4 (2004). A K Peters.

[9] J.H. Conway, *On Numbers and Games*. A K Peters, 2001.

[10] F. Chung, Z. Furedi, R. Graham, P. Seymour, *On induced subgraphs of the cube*. J. Comb. Theory, Ser. A, vol. 49 (1), pp. 180–187, 1988.

[11] M. Gardner, *Mathematical games: cram, crosscram and quadraphage: new games having elusive winning strategies*, Sci. Am., vol. 230, pp. 106-108, 1974.

[12] C.L. Bouton, *Nim, a game with a complete mathematical theory*, Ann. Math. vol. 3, pp. 35-39, 1905.

```
> restart;
  with(LinearAlgebra):
  Left := proc(m::posint, n::posint)

  local M0:= <<1>>, M1:= <<0>>, M2:= <<0>>, Z:= <<0>>, i, B;
  global T, G;
  local m0:= <<0>>, m1:= <<1>>, m2:= <<0>>, z:= <<0>>, A;

  for i to m do
  (M2, M1, M0, Z):= (<y*M0, Z; Z, Z>, <M2, x*M0; M0, Z>, <M1+M2, x*
  M0; M0, Z>, <Z, Z; Z, Z>)
  od;

  for i to m do
  (m2, m1, m0, z):= (<y*m0, z; z, z>, <m2, x*m0; m0, z>, <m1+m2, x*
  m0; m0, z>, <z, z; z, z>)
  od;

  A:= (M0+m0):
  B:= (A)^n:
  T:= simplify(subs({x=y,y=x},B[1, 1]))
  end proc;

  LeastDegree:= proc(P::polynom, var::list)
  sort(map(degree, [op(P)], var));
  select(p->degree(p)=%[1], P);
  end proc;
```

```
Left := proc(m::posint, n::posint)
  local M0, M1, M2, Z, i, B, m0, m1, m2, z, A;
  global T, G;
  M0 := < <1> >;
  M1 := < <0> >;
  M2 := < <0> >;
  Z := < <0> >;
  m0 := < <0> >;
  m1 := < <1> >;
  m2 := < <0> >;
  z := < <0> >;
  for i to m do
    M2, M1, M0, Z := < <y*M0|Z>, <Z|Z> >, < <M2|x*M0>, <M0
    |Z> >, < <M1+M2|x*M0>, <M0|Z> >, < <Z|Z>, <Z|Z> >
  end do;
  for i to m do
    m2, m1, m0, z := < <y*m0|z>, <z|z> >, < <m2|x*m0>, <m0
    |z> >, < <m1+m2|x*m0>, <m0|z> >, < <z|z>, <z|z> >
  end do;
  A := M0+m0;
  B := A^n;
  T := simplify(subs({x=y, y=x}, B[1, 1]))
```

```
end proc
LeastDegree := proc(P::polynom, var::list)                                    (1)
    sort(map(degree, [op(P)], var)); select(p->degree(p) = `%`[1], P)
end proc
```

```
> G := select(T -> abs(degree(T, x) - degree(T, y)) < 2, expand(Left(4, 4)));
  LeastDegree(G, [x, y]);
```

$$G := 16\,y^4\,x^4 + 130\,y^3\,x^4 + 66\,x^3\,y^4 + 132\,x^3\,y^3 + 38\,x^3\,y^2 + 132\,x^2\,y^3 + 9\,y^2\,x^2$$
$$9\,y^2\,x^2 \qquad (2)$$

```
> subs({x = 1, y = 1}, T)
```

$$1427 \qquad (3)$$

```
>
```

```
> restart;
  with(LinearAlgebra):
  Maximal := proc(m::posint, n::posint)

  local M0:= <<1>>, M1:= <<0>>, M2:= <<0>>, Z:= <<0>>, i, B, f;
  global T, G;
  local m0:= <<0>>, m1:= <<1>>, m2:= <<0>>, z:= <<0>>, t:= 3^n, d:=
  3^n, A;

  f := d -> seq(`if`(not member(2, convert(k, base, 3)), k+1,
  NULL), k = 0 .. d);

  for i to n do
  (M2, M1, M0, Z):= (<y*M0, y*M0, Z; Z, Z, Z; Z, Z, Z>, <M2, M2, x*
  M0; Z, Z, Z; M0, M0, Z>, <M2, M2, x*M0; M1, Z, Z; M0, M0, Z>, <Z,
  Z, Z; Z, Z, Z; Z, Z, Z>)
  od;

  for i to n do
  (m2, m1, m0, z):= (<y*m0, y*m0, z; z, z, z; z, z, z>, <m2, m2, x*
  m0; z, z, z; m0, m0, z>, <m2, m2, x*m0; m1, z, z; m0, m0, z>, <z,
  z, z; z, z, z; z, z, z>)
  od;

  A:= (M0+m0):
  B:= (A)^m:
  T:= simplify(add(B[i, 1], i in f(t - 1)))
  end proc;
```

$$Maximal := \mathbf{proc}(m::posint, n::posint) \tag{1}$$

$\mathbf{local}\ M0, M1, M2, Z, i, B, f, m0, m1, m2, z, t, d, A;$

$\mathbf{global}\ T, G;$

$M0 := \langle\langle 1\rangle\rangle;$

$M1 := \langle\langle 0\rangle\rangle;$

$M2 := \langle\langle 0\rangle\rangle;$

$Z := \langle\langle 0\rangle\rangle;$

$m0 := \langle\langle 0\rangle\rangle;$

$m1 := \langle\langle 1\rangle\rangle;$

$m2 := \langle\langle 0\rangle\rangle;$

$z := \langle\langle 0\rangle\rangle;$

$t := 3\hat{}n;$

$d := 3\hat{}n;$

$f := d \rightarrow seq(if(\mathbf{not}\ member(2, convert(k, base, 3))), k+1, NULL), k = 0..d);$

$\mathbf{for}\ i\ \mathbf{to}\ n\ \mathbf{do}$

$M2, M1, M0, Z := \langle\langle y*M0|y*M0|Z\rangle, \langle Z|Z|Z\rangle, \langle Z|Z|Z\rangle\rangle, \langle\langle M2|M2|x*M0\rangle, \langle Z|Z|Z\rangle, \langle M0|M0|Z\rangle\rangle, \langle\langle M2|M2|x*M0\rangle, \langle M1|Z|Z\rangle, \langle M0|M0|Z\rangle\rangle, \langle\langle Z|Z|Z\rangle, \langle Z|Z|Z\rangle, \langle Z|Z|Z\rangle\rangle$

$\mathbf{end\ do};$

$\mathbf{for}\ i\ \mathbf{to}\ n\ \mathbf{do}$

$m2, m1, m0, z := \langle\langle y*m0|y*m0|z\rangle, \langle z|z|z\rangle, \langle z|z|z\rangle\rangle, \langle\langle m2|m2|x$

```
        * m0 > , < z|z|z > , < m0|m0|z > > , < < m2|m2|x * m0 > , < m1|z|z > , < m0
        |m0|z > > , < < z|z|z > , < z|z|z > , < z|z|z > >
    end do;
    A := M0 + m0;
    B := A^m;
    T := simplify(add(B[i, 1], i in f(t - 1)))
end proc
```

```
> Maximal(2, 4);
  G := select(T → abs(degree(T, x) - degree(T, y)) < 2, expand(Maximal(2, 4)));
```

$$y^4 + 2\,y^3 + \left(3\,x^2 + 4\,x\right) y^2 + x^4$$

$$G := 3\,y^2\,x^2 + 4\,y^2\,x \qquad \textbf{(2)}$$

```
>
```

```
> restart;
  with(LinearAlgebra):
  Right := proc(m::posint, n::posint)

  local M0:= <<1>>, M1:= <<0>>, M2:= <<0>>, Z:= <<0>>, i, B;
  global T, G;
  local m0:= <<0>>, m1:= <<1>>, m2:= <<0>>, z:= <<0>>, A;

  for i to n do
  (M2, M1, M0, Z):= (<y*M0, Z; Z, Z>, <M2, x*M0; M0, Z>, <M1+M2, x*
  M0; M0, Z>, <Z, Z; Z, Z>)
  od;

  for i to n do
  (m2, m1, m0, z):= (<y*m0, z; z, z>, <m2, x*m0; m0, z>, <m1+m2, x*
  m0; m0, z>, <z, z; z, z>)
  od;

  A:= (M0+m0):
  B:= (A)^m:
  T:= simplify(B[1, 1])
  end proc;

  LeastDegree:= proc(P::polynom, var::list)
  sort(map(degree, [op(P)], var));
  select(p->degree(p)=%[1], P);
  end proc;
```

```
Right := proc(m::posint, n::posint)
    local M0, M1, M2, Z, i, B, m0, m1, m2, z, A;
    global T, G;
    M0 := < <1> >;
    M1 := < <0> >;
    M2 := < <0> >;
    Z := < <0> >;
    m0 := < <0> >;
    m1 := < <1> >;
    m2 := < <0> >;
    z := < <0> >;
    for i to n do
        M2, M1, M0, Z := < <y*M0|Z>, <Z|Z> >, < <M2|x*M0>, <M0
        |Z> >, < <M1+M2|x*M0>, <M0|Z> >, < <Z|Z>, <Z|Z> >
    end do;
    for i to n do
        m2, m1, m0, z := < <y*m0|z>, <z|z> >, < <m2|x*m0>, <m0
        |z> >, < <m1+m2|x*m0>, <m0|z> >, < <z|z>, <z|z> >
    end do;
    A := M0+m0;
    B := A^m;
    T := simplify(B[1, 1])
```

```
end proc
LeastDegree := proc(P::polynom, var::list)
    sort(map(degree, [op(P)], var)); select(p->degree(p) = `%`[1], P)
end proc
```

(1)

```
> G := select(T -> abs(degree(T, x) - degree(T, y)) < 2, expand(Right(2, 4)));
  LeastDegree(G, [x, y]);
```

$$G := 3\,y^2\,x^2 + 10\,y^2\,x$$

$$10\,y^2\,x$$

(2)

```
> subs({x = 1, y = 1}, T)
```

$$25$$

(3)

```
>
```