Analyzing the Game of Cycles using Python

**Brian Freeman, Bushra Ibrahim,
Jayme Reed, and Dr. Emily Olson**

A thesis presented for the degree of
Bachelor of Science

MILLIKIN
UNIVERSITY

Department of Mathematics and Computational Sciences
Millikin University
May 21, 2023

Analyzing the Game of Cycles using Python

Brian Freeman, Bushra Ibrahim, Jayme Reed, and Dr. Emily Olson

Abstract

The Game of Cycles is a game that is played by two players on a simple, undirected, connected planar graph, called a board. Each player takes turns directing edges on a board with an arrow without allowing for a source or sink to occur. The game continues until a player creates a directed cycle cell or plays the last legal move. In this paper, we will examine optimal winning strategies for Players 1 and 2 on various boards, namely tree boards, using analytical methods and support our theories using our Python program to simulate the game.

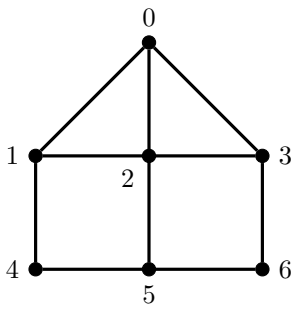


Figure 1: Sample Game Board

1 The Game of Cycles

The Game of Cycles is a game that is played by two players on a simple connected planar graph, called a game board (see Figure 1). The game with first defined by Francis Su in his 2020 book *Mathematics for Human Flourishing* [1]. Each player takes turns directing edges on a board with an arrow without allowing for a source or sink to occur. The game continues until a player creates a directed **cycle cell** – a section of the graph with no inner cycles that is surrounded by subsequent arrows – or makes the last possible move.

A game board consists of vertices and edges, vertices being points and edges being lines that connect them. Edges that form simple cycles in the graph are of particular interest because if all edges of a given cell are marked, then it becomes a cycle cell which signifies a victory. Graphs with no cells (and therefore, no simple cycles) are called **trees**, and victories on game boards that are trees are determined by the last played move. In this paper, we primarily explore winning strategies on tree game boards, and on some interesting non-tree boards, using manual and coded methods.

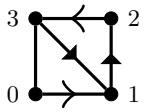
Figure 2 demonstrates the source/sink rule. The first graph violates the source rule because every edge connected to vertex A is pointing away from it. The second graph violates the sink rule because every edge connected to vertex B is pointing towards it. When we play the game, we do not apply the source/sink rule to vertices of degree 1, otherwise known as **leaves**.

Above we provide two simple game boards in Figure 3 to demonstrate valid victories and losses. In graph (a), there is a completed cell, so the player that made the last move won. In graph (b), there is no winner as the cycle contains other cells.

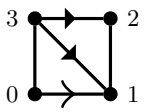
An **unmarkable edge** is an edge which can no longer be used because marking it in either direction would create a source or sink. In Figure 3, the edges (0,3) and (1,2) on (b) are unmarkable because marking them in either direction would create a source or sink at one vertex or another. For example, if we mark (0,3), vertex 0 will become a source and if you mark (2,1), vertex 1 will become a sink.

(a) Source **(b)** Sink

Figure 2: Sink or Source Graphs



(a) Winning Graph



(b) No-win Graph

Figure 3: Winning or No-Winning Graphs

2 Previously Known Results

In [2], Alvarado, et al. prove preliminary results, including that a full game board is guaranteed to have a winner as the board will contain a cycle cell. One of the results in this paper involves special cases with symmetry:

Corollary 1. *Let G be a board with $180°$ rotational symmetry, and non edge and its partner are part of the same cell. If there is no edge through the center of the board then Player 2 has a winning strategy. If there is such an edge, then Player 1 has a winning strategy.*

Corollary 2. *Let G be a board that is symmetric by reflection across some line, with no edges along that axis of symmetry and at most one edge crossing that axis of symmetry. On this board, Player 2 has a winning strategy if there is no edge crossing this axis of symmetry. If there is a single edge crossing this axis of symmetry. Player 1 has a winning strategy.*

There are known winning strategies for C_n boards [2].

Theorem 1. *The play on a C_n board is entirely determined by parity. If n is odd, Player 1 wins. If n is even, Player 2 wins.*

There are known winning strategies for path boards [3].

Theorem 2. *Assume we are playing the Game of Cycles on a path, and say player W is the player with the winning strategy. If the path has an odd number of edges Player 1 is player W, and if the path has an even number of edges Player 2 is player W.*

In [4], the authors discuss the hourglass graph. This is an example of a graph where the winning strategy changes depending on how the graph is drawn. The proof in [4] relies on Grundy numbers, and the proof we present in Section 3.1 outlines the winning strategies directly.

3 Our Results

3.1 The Hourglass Graph

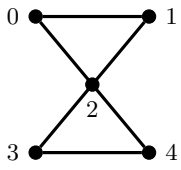
In Figure 4, we display (a), the *Hourglass* graph, and (b), the *Inverted Hourglass* graph. Both graphs are isomorphic and have the exact same adjacency matrix. However, the difference lies in their respective plane diagrams, that is, the way they are drawn. The way (a) is drawn, the two valid cycle cells are $\langle 0, 1, 2\rangle$ and $\langle 2, 3, 4\rangle$. In (b), the valid cycle cells are $\langle 0, 1, 2\rangle$ and $\langle 2, 3, 4, 2, 1, 0\rangle$. The latter cannot be completed without completion of the first cycle, so for (b), the only valuable cycle cell is $\langle 0, 1, 2\rangle$. We have found that due to this anomaly, it changes the optimal winning strategy.

Proposition 1. *If players are playing optimally, Player 2 wins on the Hourglass graph and Player 1 wins on the Inverted Hourglass graph.*

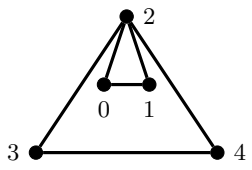
Proof. In optimal game play on the Inverted Hourglass, Player 1 will lead with edge $(0, 2)$.

↪ If Player 2 plays $(1, 0)$ or $(2, 1)$, Player 1 will complete the cell and win.

↪ If Player 2 plays $(1, 2)$, then edge $(0, 1)$ is unmarkable. Following this, Player 1 will play $(3, 4)$. Player 2 will have to play either $(4, 2)$ or $(2, 3)$ after that. Player 1 will play the final move and win.



(a) Hourglass Graph



(b) Inverted Hourglass

Figure 4: Hourglass Graphs

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

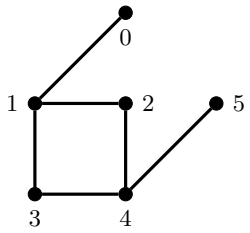


Figure 5: A graph with its corresponding adjacency matrix

↪ If instead, Player 2's first move is $(2,3)$, $(4,2)$, $(3,2)$, or $(2,4)$, Player 1 will play $(3,4)$ for the former two moves or $(4,3)$ for the latter two.

- If Player 2 plays the edge that would complete the non-cell cycle $\langle 2,3,4\rangle$ in either direction, Player 1 will play $(1,2)$, which becomes the last move as $(0,1)$ is now unmarkable.
- If instead of the previous move, Player 2 plays $(1,0)$, Player 1 will play $(2,1)$ and complete a cycle cell.

As all options for Player 2 have been exhausted leading to a Player 1 win, and Player 1 has the winning strategy for the Inverted Hourglass graph.

In an optimal game play on the Hourglass, Player 2 will win as outlined in Corollary 2. The axis of symmetry is through vertex 2. Since there is no edge that crosses that axis of symmetry, Player 2 has the winning strategy. □

3.2 Using Python to simulate the game

As a major part of our research, we developed Python code to simulate the Game of Cycles. Here is the link to reach our GitHub, where we have our Python program and supplemental files available for the public. The main file we use for simulating the game on trees and graphs alike is the `GameOfCycles.py`, and this is essentially the master program. The other file, `GoC_Objects.py`, has some graphs pre-typed so that we could just copy and paste into the master program and run it instead of re-entering the graph every single time.

To begin with, we chose to represent graphs as adjacency matrices, which in Python are translated into a list of lists. For example, Figure 5 shows a graph and its adjacency matrix, which would be represented as the following list of lists:

$$[0,1,0,0,0,0],[1,0,1,0,0,0],[0,1,0,0,1,0],[0,0,0,0,1,0],[0,0,1,1,0,1],[0,0,0,0,1,0]$$

Our `GameofCycles()` object creates a game board based on the number of vertices, n, of the input graph. This creates an $n \times n$ matrix filled with 0s. There is an `addEdge` function that is used to add edges between two vertices, entered as a pair of vertices. As the user adds edges to the graph, the adjacency matrix populates with 1s. The order in which you enter the pair of vertices to establish an edge does NOT matter. Our `GameofCycles()` object also has a couple of supplementary functions that can be used as needed, such as

`showMatrix()`, which prints out the adjacency matrix of a given graph, `returnMatrix()`, which returns the matrix object, and `isPlanar()`, which can be used to check if a given graph is planar or not. Here is an example of how a user can build a graph using our code:

```
test = GameofCycles(4)
test.addEdge(0,1)
test.addEdge(1,2)
test.addEdge(2,3)
test.addEdge(3,0)
```

This will create a graph resembling a square with four vertices and four edges. We have two additional functions to ease the creation of simple cycle or path graphs. The `makeCycle()` function can take the `GameofCycles()` object and add edges automatically to create a singular cycle graph with the given number of vertices. The `makePath()` function creates a singular path in the same manner. Below is how we can use both functions if need be:

```
test = GameofCycles(5)
test.makeCycle()
```

This will create the graph C_5, the cycle with 5 vertices.

```
test = GameofCycles(3)
test.makePath()
```

And this will create P_3, the path graph with 3 vertices.

There is an `addCycle()` function for the user to manually enter all cycle cells present in a given graph. This function stores all the cycle cells in a list that can be referred to later. For example, for our earlier `test = GameofCycles(4)` object, this is how C_4 can be added; `test.addCycle([1,2,3,0])`.

There is an `addDirection` function that is used to mark an input edge which is entered as an ordered pair. The order in which you enter the pair of vertices DOES matter. For example, `addDirection(0,1)` will direct the edge *from* 0 pointing *towards* 1, while `addDirection(1,0)` will direct the edge *from* 1 pointing *towards* 0. This way of entering edges allows us to direct them in both ways easily.

Marked edges are represented in the adjacency matrix with 2s and 3s, which are updated when a move is made. A "2" represents when an edge is directed away from a certain vertex, and a "3" represents when that edge is directed towards the vertex. For example, on a graph with only two vertices, labelled 0 and 1 with a single edge between them, the initial adjacency matrix would look like this: $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$

If we direct an edge using `addDirection(0,1)`, which directs the edge away from 0 and towards 1, the matrix will look like this: $\begin{bmatrix} 0 & 2 \\ 3 & 0 \end{bmatrix}$

If we direct an edge using `addDirection(1,0)`, which directs the edge away from 1 and towards 0, the matrix will look like this: $\begin{bmatrix} 0 & 3 \\ 2 & 0 \end{bmatrix}$

This stores the edges' direction in the matrix which can later be used to determine if a cycle cell has been completed or if the final move has been made, which defines a victory and is verified by our `checkWin()` function. It is also used to identify unmarkable edges in our `checkUnmarkable()` function.

The `checkWin()` function is crucial to our code because it must check if victory has been achieved after every single move a player makes, taking into account if a cycle has been completed or not, or if a player has made the final move.

Following that, we have our `playGame()` function which takes an input of the number of players, which is typically 2, and allows the user(s) to play one game on a given `GameofCycles` object. It will not allow

illegal moves to be made and prompts the user to enter moves until a valid move is made. After each move, `checkWin()` is used to determine if either player has won, and until then, it will continue to ask for moves. At the end of a game, it will tell the user which player won. This function can be used to play singular games with either one or more users dictating the moves of two players.

The instrumental function in our code is the `simulateGame()` function. It involves most of the prior functions to effectively simulate the game. The inputs for `simulateGame()` are player number and "True," only if the user wants a more thorough analysis of the moves made. If that analysis is not wanted, then just the player number will suffice.

This is how `simulateGame()` works;

- ↪ Using our previous `GameofCycles` object "test," to call `simulateGame()`: `test.simulateGame(2)`
- ↪ If requesting a thorough analysis of the moves, then call the function: `test.simulateGame(2, True)`
- ↪ First, a copy of the initial game board matrix is created. This copy will allow us to modify as each game is played and store it for outputting later.
- ↪ A list of all markable edges is made. If the user has unput a graph with one or more marked edges, those edges will NOT be included in this list.
- ↪ Random combinations of ways to mark edges are made, both toward and away from the involved vertices. For example, for an edge between vertices 1 and 2, $(1, 2)$ and $(2, 1)$ account for both ways to direct that edge.
- ↪ These combinations of edges for all markable edges in the graph are stored in a list of lists, each sublist containing a different combo, and the order of edges played are chosen randomly for the next step. This makes sure that when the game is played, the function will not play the same game twice by checking the order of the moves.
- ↪ Then, those edges are played in that order by making the appropriate moves. This continues until an illegal move is made, and at that point the function will check if someone has won.
- ↪ If someone HAS NOT won, it scraps the moves and starts a new game with the next set of ordered edges.
- ↪ If someone HAS won, it saves the played moves, the final matrix of the game board, and identifies whether Player 1 or Player 2 is the winner.
- ↪ An outfile is created, and to it, the function prints the initial board, the matrix endstates after each game, the winner of each game, and keeps record of the number of games played and the percentage of those games that Player 1 and Player 2 have won.
- ↪ This is repeated for every distinct set of ordered edge combinations until all combos have been run through, the number of games and percentages being updated after each game. The resulting outfile will contain all of this information once `simulateGame()` has finished running.
- ↪ If the user wants the thorough moves analysis, then that code is activated. This takes Player 1's first move played and checks how many of the winning boards that first move was played on, and how many losing boards that move was played on. If that move was on more winning boards than losing ones, that means it's an optimal first move.
- ↪ The winrate is how many endstates Player 1 wins with that optimal first move. Player 1 is typically the one that makes the first move on the game board, but if a board is given with one or more moves already made, "Player 1" is the player whose move it is next. This is important to remember when entering boards with an odd number of edges already marked, because that means the next player is actually Player 2, but the outfile will still address them as Player 1.

Figure 6: Trees with Exactly Two Leaves

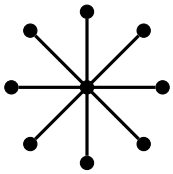


Figure 7: Star Trees

This is why `simulateGame()` is the most essential part of our code. It produces these conclusive results and can be run multiple times and play hundreds of games that we could not feasibly do by hand. This allows us to peer deeper into the strategies for various game boards, be those trees or non-tree graphs, and supports our theories on what moves are optimal for Player 1 and Player 2.

3.3 Trees, up to 9 vertices

Trees are simpler graphs to analyze due to their lack of cycles, and as such, they are the type of graphs we primarily focused on in our research. We have tested trees of interest all the way up to size 9 (size being determined by the number of vertices), and of those, we have recorded optimal strategies for the non-symmetric ones. See the strategies in Appendix A. Below we discuss each interesting tree with respect to its size and explain our proposed strategy. By playing on leaves, our results let us play on more trees than adhering to the source/sink rule.

Parity

Definition 1. *Odd trees are trees with an odd number of edges and an even number of vertices.*

Definition 2. *Even trees are trees with an even number of edges and an odd number of vertices.*

Theorem 3. *The optimal strategy for trees is based on the parity of its edges. In particular,*

↪ *Player 1 has the optimal strategy on odd trees.*

↪ *Player 2 has the optimal strategy on even trees.*

Uninteresting Cases

Exactly Two Leaves

The result for paths in Barua [3] matches ours in Theorem 3.

Star Trees

Trees where all edges come from one vertex (an example is in Figure 7) are star trees and will always follow parity, as either player can play an edge facing the opposite direction of a previous edge.

Interesting Cases

Semi-star Trees

Definition 3. *Semi-star trees are like star trees, but with a variation. When a star tree of size $m \geq 1$ and a star tree of size $n \geq 1$ are connected at their respective central vertices with one edge, we call this a semi-star tree. Notice a semit-star tree is a tree in which exactly two adjacent vertices have degree > 1.*

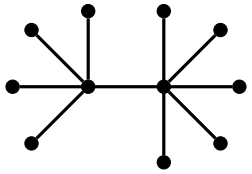


Figure 8: Semi-star Trees

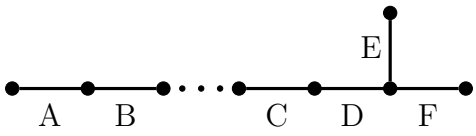


Figure 9: Y-Graph Trees

On semi-star trees, (example in Figure 8) the game is over in one move. The player with the optimal strategy will choose the edge that connects the two central vertices. Since there will be no way to make any other edge unmarkable, parity decides the winner.

Y-Graph Trees

Definition 4. *Y-graphs are trees with at least 4 edges that have exactly three leaves, where two of the leaves are adjacent to the same vertex. Those leaves will be referred to as Y-leaves.*

(An example of a Y-graph can be found in Figure 9)

Strategy for odd Y-graphs: (Player 1 has the optimal strategy.)

↪ Player 1's optimal first move is to find the edge between edges A and D that creates a symmetry. Player 1 can mark the edge in either direction

- If Player 2 continues to mark on the path, Player 1 will mirror that move by marking the symmetric edge in the opposite direction on the other side of the symmetry edge
- If Player 2 marks either edges E or F, Player 1 will mark the other Y-leaf in the opposite direction of that vertex.

Since every move Player 2 plays has a corresponding move Player 1 plays in response, Player 1 has the optimal strategy.

Strategy for even Y-graphs: (Player 2 has the optimal strategy.)

↪ If the Y-graph has 4 edges, Player 2 has the optimal strategy. We can see this because the only edge that could be unmarkable is the single non-leaf edge. Player 2 can ensure this edge is marked by adding a direction after Player 1's first move. Then there are two leaves left to be marked, which ensures a victory for Player 2.

↪ Next, we consider Y-graphs with more than 4 edges. We will describe the moves Player 2 should do after each of Player 1's moves. Keep in mind that Player 2 desires an even number of unmarkable edges.

- If Player 1 starts on either edge E or edge F either facing the vertex of degree three or away, Player 2 will play the other Y-leaf in the opposite direction.
- Notice that since the path from A to D has an even number of edges, the path has a central vertex and pairs of edges that are symmetric to each other. So, A is symmetric to D, B is symmetric to C, etc. If Player 1 then plays any edge along the path from A to D, Player 2 should mark

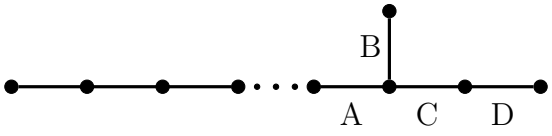


Figure 10: L-Head Trees

the edge that is symmetric to it along the path and mark it in the same direction as Player 1. This will guarantee that if Player 1 tries to make an unmarkable edge, Player 2 will also create an unmarkable edge, thus ensuring that the number of unmarkable edges at the end is even.

Since every move Player 1 plays has a corresponding move for Player 2 to play in response, Player 2 has the optimal strategy.

L-Heads

Definition 5. *L-heads are trees that have 6 or more vertices. The shape of L-heads is a path with a leaf coming out of the third to last vertex, thus making the shape of an L at the end of a tree. The L-center is the vertex that has degree three.*

(An example of an *L*-head can be found in Figure 10)

Strategy for odd *L*-head:

↪ Player 1 will play edge D, which is the *L*-head leaf not connected to the *L*-center.

↪ If Player 2 plays edge A, which is the edge directly adjacent to the *L*-head, Player 1 will play the edge next to their move. This will prevent Player 2 from making an odd number of unmarkable edges.

↪ If Player 2 instead plays either edge B or edge C, the two edges remaining on the *L*-head, Player 1 will play the edge not played by Player 2. This makes it so no matter what Player 2 does next, Player 1 will make an even number of unmarkable edges.

↪ If Player 2 instead plays off the *L*-head, Player 1 will follow the strategy for an odd path up to the vertex of degree 2 for edge A.

Strategy for an even *L*-head:

↪ The goal for Player 2, regardless of where Player 1 starts, is to prevent Player 1 from making an unmarkable edge, or, if Player 1 does make an unmarkable edge, Player 2 needs to make a second edge unmarkable in order to win.

↪ If Player 1 plays edge A or edge D as their first move, Player 2 should mark the other edge in the same direction. This will prevent edge C from becoming unmarkable.

↪ If Player 1 plays an edge that is not on the *L*-head as their first move, player two should apply the strategy for an even path up to the vertex of degree 2 for edge A.

Trees with U-Edges

Definition 6. *A U-edge is an edge whose incident vertices both have exactly degree* 2.

A *U*-edge is more likely to become an unmarkable edge any other edge, such as edge A in Figure 11. We will describe the strategy for trees with *U*-edges but we do not include *L*-head graphs or *Y*-graphs as they have their own strategies.
Strategy for an odd tree:

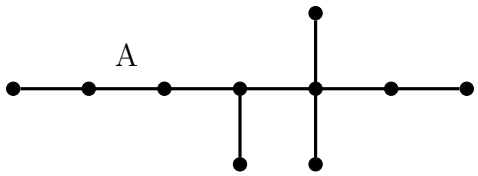


Figure 11: Trees with U-Edges

↪ Player 1 wants no unmarkable edges. As such, the first move they make will be to claim a U-edge.

↪ Player 1 will want to ensure every U-edge is marked, regardless of Player 2's move.

↪ Player 1, then, will prevent any unmarkable edges

Strategy for an even tree:

↪ Player 1 will attempt to make an unmarkable edge in order to win. Player 2 wants no unmarkable edges. As such, regardless of where Player 1's first move was, Player 2 will mark the U-edge.

↪ Player 2 wants to ensure U-edge is marked so this strategy will continue until all U-edges are marked.

↪ Player 2, then, will prevent any unmarkable edges.

3.4 Other planar graphs

Non-tree graphs are those that contain one or more cycles in them. With the inclusion of cycles, the number of different ways to win increase as victory can be achieved either through completing a cycle, or by making the last move, while with trees you could only win by doing the latter.

3.5 3 player Game

In 3-player game on C_n, winner is determined by the number of marked edges in C_n. This is similar to the result in Theorem 1. Let n be the number of edges. Now, to determine the winner, we will take the k from the set of $\mathbb{N}$ such that $\{0, 2, 4, ..., \lfloor n/2 \rfloor\}$. We will do $n - k \pmod 3$ up to $k = \lfloor n/2 \rfloor$. For a Player 3 win, $n - k \equiv 0 \pmod 3$. For a Player 2 win, $n - k \equiv 2 \bmod(3)$. For a Player 1 win, $n - k \equiv 1 \pmod 3$.

Let us look at C_8 for an example, where n = 8 and k = 4

↪ $8 - 0 \equiv 2 \pmod 3$. Thus, when all edges are marked, Player 2 wins.

↪ $8 - 2 \equiv 0 \pmod 3$. Thus, when six edges are marked, Player 3 wins.

↪ $8 - 4 \equiv 1 \pmod 3$. Thus, when four edges are marked, Player 1 wins.

4 Conclusion

The Game of Cycles continues to intrigue us despite the hundreds of game boards we studied within the past year. The simple and effective strategies that we confirmed for paths, the more complicated but parity-associated strategies we developed for trees, and the possible strategies that we hope others confirm for graphs with cycles – they all stem from a deceptively straightforward game that, in actuality, has a myriad of possible routes for victory. Our Python code, if anything, only further supported this notion, having produced several end-states for various graphs. This just goes to show how vast a potential lays in graph theory, even within the moves of players on a mathematical game. We hope these findings help add to the Game of Cycles conversation on a larger scale and assist in the research of this game, inching closer to completeness.

References

[1] F. Su, *Mathematics for Human Flourishing*. Yale University Press, 2020.

[2] R. Alvarado, M. Averett, B. Gaines, C. Jackson, M. L. Karker, M. A. Marciniak, F. Su, and S. Walker, "The game of cycles," *The American Mathematical Monthly*, vol. 128, no. 10, pp. 868–887, 2021.

[3] C. Barua, E. Burkholder, G. Fragoso, and Z. Szaniszlo, "Analyzing a graph theory game," 2022.

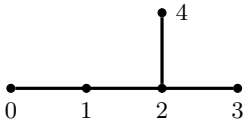
[4] R. Fokkink and J. Zandee, "Some remarks on the game of cycles," 2022.

A Strategies for Trees, up to 9 vertices

This appendix contains all the strategies for trees with up to 9 vertices. Some of the trees have additional strategies detailed earlier in this paper (such as L-heads or Y-graphs). Others have symmetry properties and so their optimal strategies were proven by [2] in Corollaries 1 and 2.

A.1 Four Edges / Five Vertices (Player 2 optimal strategy)

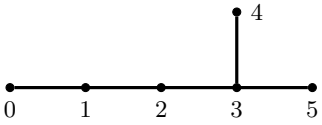
Graph 1



Optimal strategy: Player 1 can only win if the (1,2) edge is unmarkable. Player 2 thus must mark said edge on their turn to guarantee a win. Player 1 can do nothing to stop this.

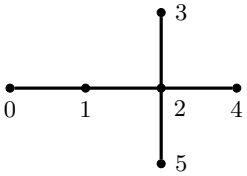
A.2 Five Edges / Six Vertices (Player 1 optimal strategy)

Graph 1



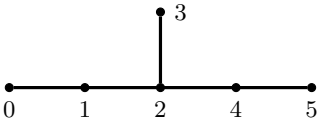
This is an Y-graph. As described in Section 3.3, Player 1 has the optimal strategy.

Graph 2



Optimal strategy: Player 2 can only win if the (1,2) edge is unmarkable. Player 1 thus must mark said edge on their turn to guarantee a win. Player 2 can do nothing to stop this.

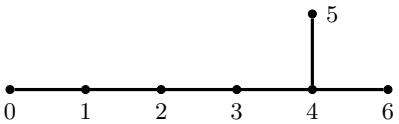
Graph 3



Optimal strategy: Player 2 can only win if either the (1,2) or (2,4) edge remain unmarkable. Player 1 thus must start on the (1,2) edge. No matter what player 2 does afterward, player 1 will ensure (2,4) is marked to ensure the win.

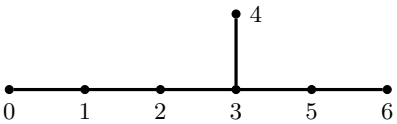
A.3 Six Edges / Seven Vertices (Player 2 optimal strategy)

Graph 1



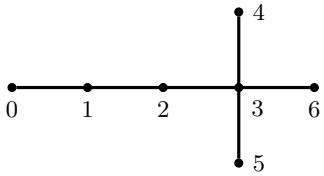
This is an Y-graph. As described in Section 3.3, Player 2 has the optimal strategy.

Graph 2



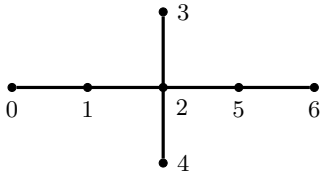
This is an L-head graph. As described in Section 3.3, Player 2 has the optimal strategy.

Graph 3



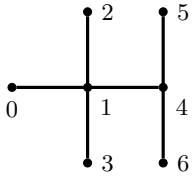
Optimal Strategy: Player 1 can only win if (1,2) or (2,3) remains unmarkable. Player 2 will mark (1,2) facing the same way as player 1's first move. Player 1 then will either mark (2,3) and lose or a different move. Then player 2 will play (2,3) and win. Instead, if player 1 plays (1,2) as their first move, player 2 will play (2,3) and win the game. Player 2 will win.

Graph 4



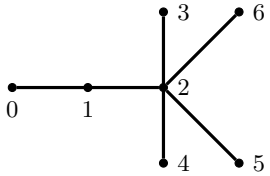
Optimal Strategy: Player 1 can only win if (1,2) or (2,5) remains unmarkable. Player 2 will play (1,2) and (2,5) as their first two moves. Since no matter what player 1 does can stop player 2 from making these moves, player 2 wins. Player 1 playing either of the two moves means player 2 will play the opposite and win.

Graph 5



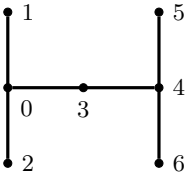
Optimal strategy: Player 1 can only win if edge (1,4) remains unmarkable. Player 2 will play (1,4) as their first move. Since there is nothing player 1 can do to stop them, player 2 wins.

Graph 6



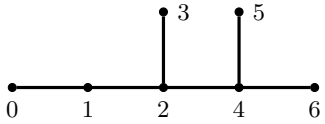
Optimal strategy: Player 1 can only win if edge (1,2) remains unmarkable. Thus, player 2 will play (1,2) on their turn. Since there is nothing player 1 can do to stop player 2 from making this move, player 2 wins.

Graph 7



Optimal strategy: Player 1 can only win if either (0,3) or (3,4) remain unmarkable. If player 1 starts on a leaf, player 2 will play either (0,3) or (3,4) where the non-3 vertex is the same as the corresponding value from player 1's move. Player 2 will then play the other edge on their next turn, as nothing player 1 does can stop them from playing it. Therefore, player 2 wins.

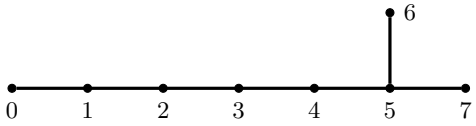
Graph 8



Optimal strategy: Player 1 can only win if either (1,2) or (2,4) remain unmarkable. If player 1 plays (0,1), player 2 plays (1,2). No matter what player 1 does next, player 2 plays (2,4) and wins. Instead, if player 1 does not start with (0,1), player 2 plays (2,4) pointing away from the vertex of degree 3 that is connected to player 1's first move. Since player 1 cannot stop (1,2) from being marked by player 2's next turn, player 2 wins.

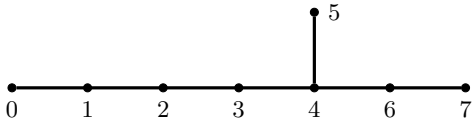
A.4 Seven Edges / Eight Vertices (Player 1 optimal strategy)

Graph 1



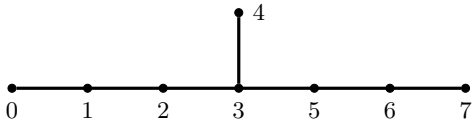
This is an Y-graph. As described in Section 3.3, Player 1 has the optimal strategy.

Graph 2



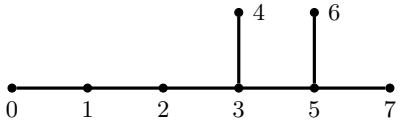
This is an L-head graph. As described in Section 3.3, Player 1 has the optimal strategy.

Graph 3



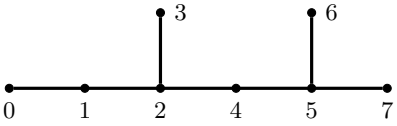
Optimal Strategy: Player 2 can only win if only one of (1,2), (2,3), (3,5), or (5,6) become unmarkable. Player 1 starts on (3,4). Whatever move player 2 plays the rest of the game, player 1 will play the mirrored move in the same direction. For example, if player 2 plays (2,3) going from 2 to 3, player 1 will play (3,5) going from 3 to 5. This ensures that either 0 or 2 edges become unmarkable, so player 1 wins.

Graph 4



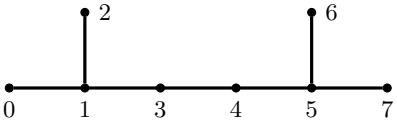
Optimal Strategy: Player 2 can only win if only one of (1,2), (2,3), or (3,5) become unmarkable. Player 1 starts with (3,5). If player 2 plays (2,3), player 1 plays (1,2) in the same direction to ensure no unmarkable edges. If instead player 2 plays (0,1), player 1 plays (2,3) in the same direction to make no edges unmarkable. Otherwise, player 1 plays (1,2) in the same direction as their (3,5) move to prevent any unmarkable edges. In all cases, since no edges are unmarkable, player 1 wins.

Graph 5



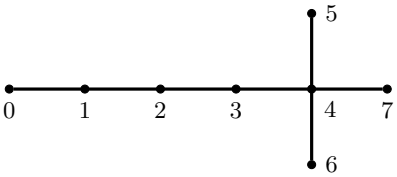
Optimal Strategy: Optimal Strategy: Player 2 can only win if only one of (1,2), (2,4), or (4,5) become unmarkable. Player 1 will play (1,2). If player 2 plays (2,4) or (4,5), player 1 plays the opposite and no edges can be unmarkable. Instead, if player 2 plays any other move, player 1 will play (4,5) in the same direction as the (1,2) move. This makes sure (2,4) cannot be unmarkable. Since no edges become unmarkable, player 1 wins.

Graph 6



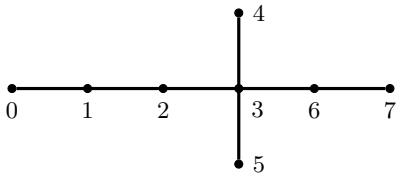
Optimal Strategy: Player 2 can only win if only one of (1,3), (3,4), or (4,5) become unmarkable. Player 1 starts on (3,4). If player 2 plays (0,1), (1,2), or (4,5), player 1 plays (1,3). If player 2 plays (1,3), (5,7), or (5,6), player 1 plays (4,5). Since player 1 prevents any edges from becoming unmarkable, player 1 wins.

Graph 7



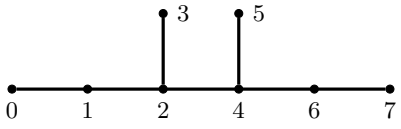
Optimal Strategy: Player 2 can only win if only one of (1,2), (2,3), or (3,4) become unmarkable. Player 1 will play (4,7). If player 2 plays (4,5) or (4,6), player 1 plays the opposite move facing the opposite direction from vertex 4 of the previous move. If player 2 plays (0,1), (1,2), (2,3), or (3,4), player 1 plays the other edge two away in the same direction. Since these moves prevent any edges from becoming unmarkable, player 1 wins.

Graph 8



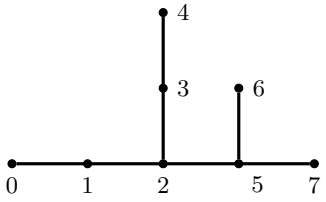
Optimal Strategy: Player 2 can only win if only one of (1,2), (2,3), or (3,6) are unmarkable. Player 1 will start with (3,5). If player 2 plays (3,6) or (6,7), player 1 plays (3,4) the same direction as they played (3,5). If player 2 plays (1,2) or (3,6), player 1 plays (3,4) the opposite direction from vertex 3 as they played (3,5). If player 2 plays (0,1) or (2,3), player 1 plays the opposite move in the same direction. These moves will ensure there are no unmarkable edges, meaning player 1 wins.

Graph 9



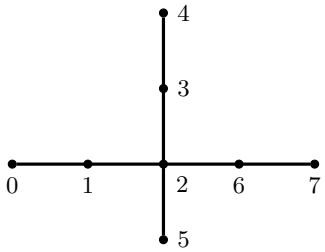
Optimal Strategy: Player 2 can only win if only one of (1,2), (2,4), or (4,6) are unmarkable. Player 1 will play (2,4). If player 2 plays (1,2), (4,5), or (6,7), player 1 plays (4,6). If player 2 plays (0,1), (2,3), or (4,), player 1 plays (1,2). Since player 1 prevents unmarkable edges in both cases, player 1 wins.

Graph 10



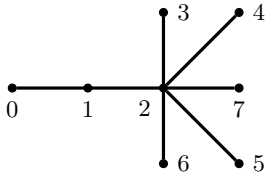
Optimal Strategy: Player 2 only wins if only one of (1,2), (2,3), or (2,5) become unmarkable. Player 1 will play (1,2). If player 2 plays (2,3) or (2,5), player 1 plays the opposite move. Otherwise, player 1 plays (2,5) facing the opposite direction from vertex 2 as their first move. In either case, no edges can be unmarkable, so player 1 wins.

Graph 11



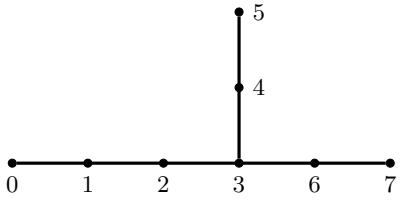
Optimal Strategy: Player 2 only wins if only one of (1,2), (2,6), or (2,3) become unmarkable. Player 1 will play (1,2). If player 2 plays (2,6) or (2,3), player 1 plays the opposite move. Otherwise, player 1 plays (2,6) facing the opposite direction from vertex 2 as their first move. In either case, no edges can be unmarkable, so player 1 wins

Graph 12



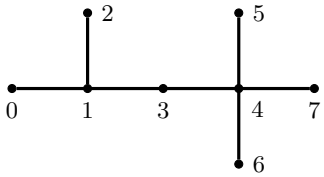
Optimal Strategy: Player 2 can only win if (1,2) is unmarkable. Player 1 plays (1,2) and wins.

Graph 13



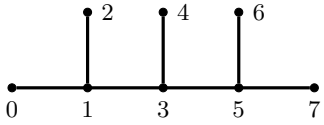
Optimal Strategy: Player 2 can only win if only one of (1,2), (2,3), (3,4), or (3,6) become unmarkable. Player 1 will play (1,2) from vertex 1 to vertex 2. If player 2 plays (0,1), (2,3), (3,6), or (6,7), player 1 will play (3,4) going from vertex 3 to vertex 4. If player 2 plays (3,4) or (4,5), player 1 plays (3,6) going from vertex 3 to vertex 6. In both cases, there are no unmarkable edges, so player 1 wins.

Graph 14



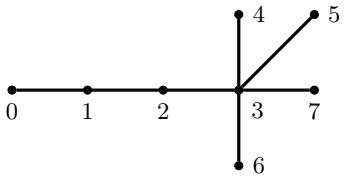
Optimal Strategy: Player 2 can only win if only one of (1,3) or (3,4) become unmarkable. Player 1 plays (1,3). No matter what player 2 does, player 1 will then play a move so (3,4) is marked and wins.

Graph 15



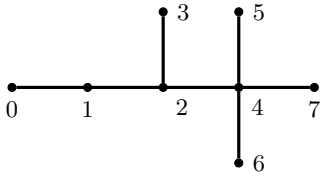
Optimal Strategy: Player 2 can only win if only one of (1,3) or (3,5) become unmarkable. Player 1 plays (1,3). No matter what player 2 does, player 1 will then play a move so (3,5) is marked and wins.

Graph 16



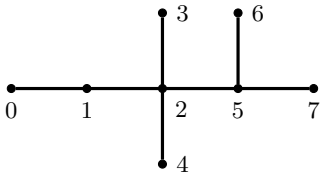
Optimal Strategy: Player 2 can only win if only one of (1,2) or (2,3) become unmarkable. Player 1 plays (1,2). No matter what player 2 does, player 1 will then play a move so (2,3) is marked and wins.

Graph 17



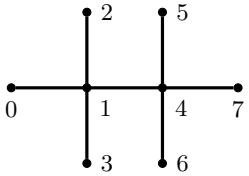
Optimal Strategy: Optimal Strategy: Player 2 can only win if only one of (1,2) or (2,4) become unmarkable. Player 1 plays (1,2). No matter what player 2 does, player 1 will then play a move so (2,4) is marked and wins.

Graph 18



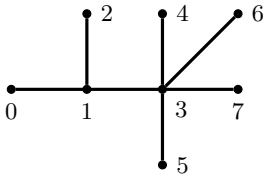
Optimal Strategy: Player 2 can only win if only one of (1,2) or (2,5) become unmarkable. Player 1 plays (1,2). No matter what player 2 does, player 1 will then play a move so (2,5) is marked and wins.

Graph 19



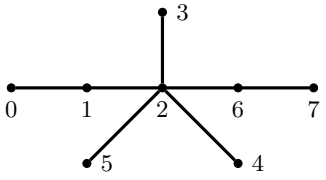
Optimal Strategy: Player 2 can only win if (1,4) becomes unmarkable. Player 1 plays (1,4) and wins.

Graph 20



Optimal Strategy: Player 2 can only win if (1,3) becomes unmarkable. Player 1 plays (1,3) and wins.

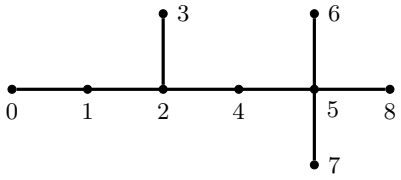
Graph 21



Optimal Strategy: Player 2 can only win if only one of (1,2) or (2,6) become unmarkable. Player 1 plays (1,2). No matter what player 2 does, player 1 will then play a move so (2,6) is marked and wins.

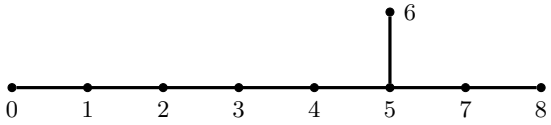
A.5 Eight Edges / Nine Vertices (Player 2 optimal strategy)

Graph 1



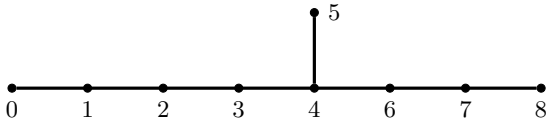
Optimal Strategy: Player 1 can only win if only one of either (1,2), (2,4), or (4,5) become unmarkable. If player 1 plays (1,2) first, player 2 will play (2,4). Then no matter what player 1 plays, player 2 can play (4,5) and have no unmarkable edges. If player 1 starts with any other edge first, player 2 will play (1,2) in the same direction as player 1's first move. After player 1's second move, if two of (5,6), (5,7), or (5,8) are marked, player 2 will play (4,5). Otherwise, player 2 will mark (2,4). Player 2 will play the opposite move next. Since no edges will be unmarkable, player 2 wins.

Graph 2



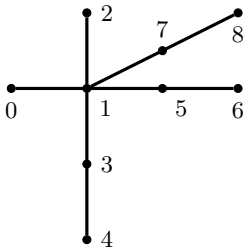
This is an L-head graph. As described in Section 3.3, Player 2 has the optimal strategy.

Graph 3



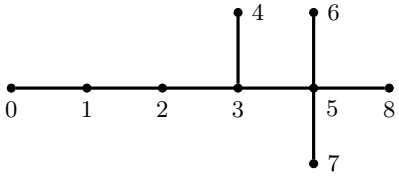
Optimal Strategy: Player 1 can only win if only one of (1,2), (2,3), (3,4), (4,6), or (6,7) become unmarkable. If player 1 plays (0,1) or (1,2) first, player 2 will play (2,3) in the same direction. Instead, if player 1 plays (2,3) or (3,4), player 2 will play (1,2) in the same direction. In either case, if player 1 plays (6,7) next, player 2 will play (4,6) to prevent any unmarkable edges. If player 1 plays any other move next instead, player 2 will play (6,7) second in the same direction as their first move, preventing any unmarkable edges. If player 1 starts with (6,7) or (4,5), player 2 will play the opposite move facing the same direction toward vertex 4 as player 1's move. Instead, if player 1 plays (4,6) or (7,8), player 2 will play the opposite move in the same direction. In either case, if player 1 plays (0,1) or (1,2) next, player 2 will play (2,3) in the same direction and prevent unmarkable edges. Instead, if player 1 plays (2,3) or (3,4), player 2 will play (1,2) in the same direction preventing unmarkable edges. There will be no unmarkable edges in all cases, causing player 2 to win.

Graph 4



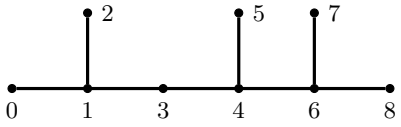
Optimal Strategy: Player 1 can only win if only one of (1,3), (1,5), or (1,7) become unmarkable. No matter what player 1 does, player 2 will play one of the edges attached to vertex 1 facing the opposite direction from vertex 1 as player 1's first move. This means no edges will be unmarkable, so player 2 wins.

Graph 5



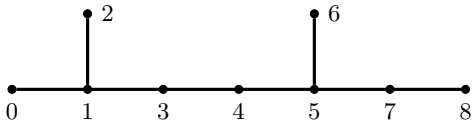
Optimal Strategy: Player 1 can only win if only one of (1,2), (2,3), or (3,5) become unmarkable. If player 1 plays (3,5), (5,6), (5,7), or (5,8), player 2 will play (1,2) in the same direction as player 1's move. Then player 2 will play (2,3) if available or (3,5) if it is already marked. If player 1 instead starts with (1,2) or (4,3), player 2 will play (3,5) facing the opposite direction from vertex 3 that player 1's move is pointing. If player 1 starts with (0,1) or (2,3) instead, player 2 will play the opposite move facing the same direction. Player 2 will make sure during their next move makes (3,5) marked. There are no unmarkable edges in all cases, so player 2 wins.

Graph 6



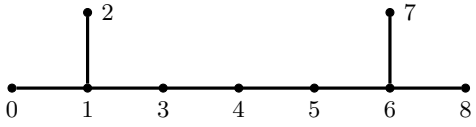
Optimal Strategy: Player 1 can only win if only one of (1,3), (3,4), or (4,6) become unmarkable. If player 1 plays (0,1) or (1,2), player 2 will play (1,3). If player 1 plays (1,3), (4,5), or (4,6), player 2 will play (3,4). If player 1 plays (3,4), (6,7), or (7,8), player 2 will play (4,6). Since player 1 cannot create an unmarkable edge if player 2 follows up with each of these moves, player 2 wins.

Graph 7



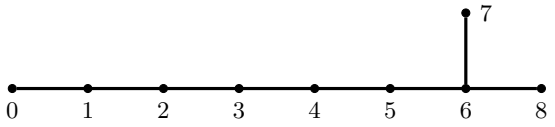
Optimal Strategy: Player 1 can only win if only one of (1,3), (3,4), (4,5), or (5,7) becomes unmarkable. If player 1 plays (0,1) or (1,2) first, player 2 will play (3,4) facing the opposite direction of vertex 1 as player 1's move. If player 1 plays (5,6), (5,7), or (7,8) first, player 2 will play (3,4) facing the opposite direction of vertex 5 as player 1's move. If player 1 plays (1,3) or (4,5) first, player 2 will play (3,4) in the same direction. After player 1's first move, if only one of (0,1) or (1,2) are marked and are facing the same way towards vertex 1 as (3,4), the opposite edge will be marked the opposite way towards vertex 1. Otherwise, if only one of (5,6), (5,7), or (7,8) are marked and are facing the same way towards vertex 5, either (5,6) if not marked or (5,7) if (5,6) is marked will be marked the opposite way towards vertex 5. In all cases, no edges will be unmarkable, so player 2 wins.

Graph 8



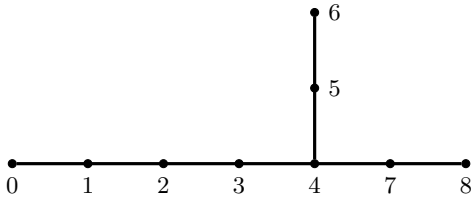
Optimal Strategy: Player 1 can only win if only one of (1,3), (3,4), (4,5), or (5,6) become unmarkable. If player 1 plays (1,3) or (3,4), player 2 will play (4,5) in the same direction and no edges can become unmarkable. If instead player 1 plays (4,5) or (5,6), player 2 will play (3,4) in the same direction and no edges can become unmarkable. In both cases, player 2 wins.

Graph 9



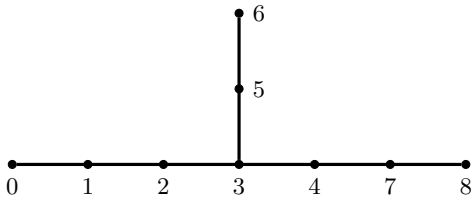
This is an Y-graph. As described in Section 3.3, Player 2 has the optimal strategy.

Graph 10



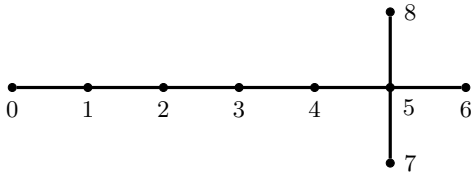
Optimal Strategy: Player 1 wins with an odd number of unmarkable edges. Playing (4,5) and (4,7) or (5,6) and (7,8) facing opposite directions from V4 will ensure (3,4) will be marked. The rest of the edges will play like an odd vertex path, which player 2 wins.

Graph 11



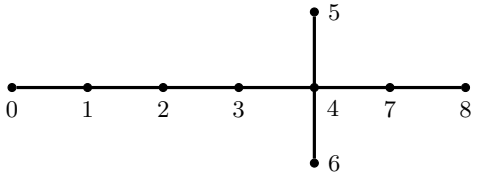
Optimal Strategy: Player 1 wins with an odd number of unmarkable edges. Playing (5,6) or (3,5) and (0,1) or (7,8) in the same way from V3 covers the leaves. Mark (2,3) or (3,4) to force an even number of unmarkable edges if the opposite is also played. Mark (1,2) or (4,7) to do the same. Thus, player 2 wins.

Graph 12



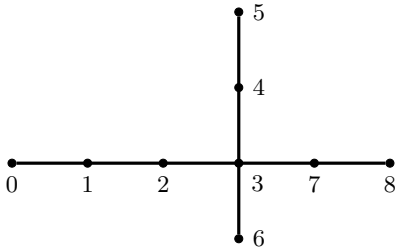
Optimal Strategy: Player 1 wins with an odd number of unmarkable edges. Playing (0,1) and (3,4) or (1,2) and (4,5) forces an even number of unmarkable edges. Playing the other edges will let player 2 play (5,6), (5,7), and (5,8) until one of the game ending edges is played. Thus, player 2 wins.

Graph 13



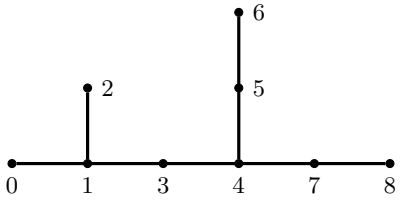
Optimal Strategy: Player 1 can only win if there is an odd number of unmarkable edges. Playing (4,5), (4,6), (4,7), and (7,8) prevents any edges from becoming unmarkable. When player 1 plays one of the other edges, player 2 will play another edge that guarantees that no edges are unmarkable, causing them to win.

Graph 14



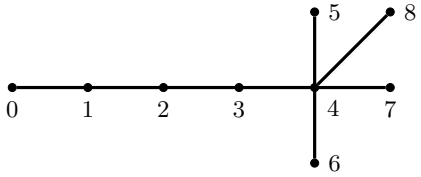
Optimal Strategy: Player 1 can only win if there are an odd number of unmarkable edges. Playing (0,1) and (2,3) prevents (1,2) from being unmarkable. Playing (3,4) and (3,7) prevents (2,3) from being unmarkable. For the rest of the edges, playing (3,6) prevents (3,4) and (3,7) from being unmarkable. Therefore, player 2 wins as all edges can be prevented from becoming unmarkable.

Graph 15



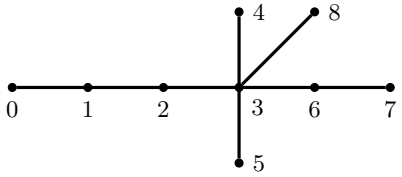
Optimal Strategy: Player 1 can only win if there are an odd number of unmarkable edges. Playing (4,5) and (4,7) or (5,6) and (7,8) facing the opposite directions from vertex 4 prevents (3,4) from becoming unmarkable. Playing (0,1) and (1,2) prevents (1,3) from becoming unmarkable. Playing (1,3) and (3,4) forces one of (4,5) or (4,7) to be playable by player 2, which prevents the other from becoming unmarkable. Therefore, player 2 wins.

Graph 16



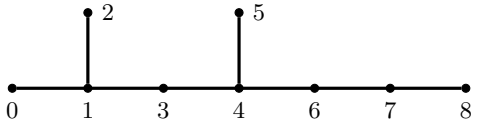
Optimal Strategy: Player 1 can only win if there are an even number of unmarkable edges. Playing (0,1) and (2,3) or (1,2) and (3,4) prevents any of the edges from becoming unmarkable. Because there are an even number of edges left, player 2 can keep playing those edges until player 1 plays one of the game ending moves. Therefore, player 2 wins.

Graph 17



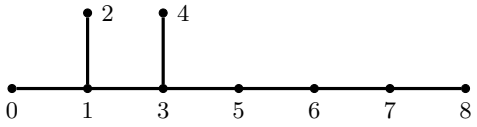
Optimal Strategy: Player 1 can only win if there are an even number of unmarkable edges. Playing (1,2) and (3,6) prevents any edges from becoming unmarkable. Playing (0,1) and (2,3) prevents (1,2) from becoming unmarkable, which can be followed up by (3,6) to make sure no edges are unmarkable. Playing other edges will cause player 2 to play (6,7) then (1,2) to win the game.

Graph 18



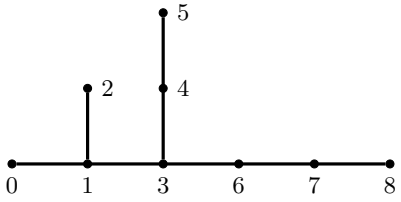
Optimal Strategy: Player 1 can only win if there are an odd number of unmarkable edges. Playing (1,3) or (3,4) and (6,7) ensure no edges are unmarkable. Playing (0,1) or (1,2) and (4,5) makes sure (1,3) and (3,4) are markable. Playing (4,6) and (7,8) makes sure (6,7) is markable. There is no way for player 1 to make an unmarkable edge, so player 2 wins.

Graph 19



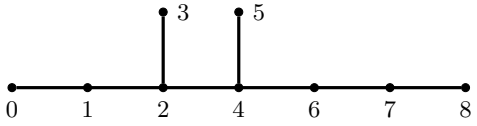
Optimal Strategy: Player 1 can only win if there are an odd number of unmarkable edges. Playing (3,5) and (6,7) or (5,6) and (7,8) prevent any of those edges to be unmarkable. If player 1 does not play any of those moves, player 2 can play (0,1), (1,2), (1,3), or (3,4) until they do. These moves prevent (1,2) from being unmarkable and force player 1 to make the losing move.

Graph 20



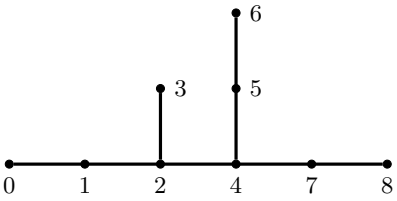
Optimal Strategy: Player 1 can only win if there are an odd number of unmarkable edges. Playing (1,3) and (6,7) prevents any edges from becoming unmarkable. Playing (3,7) and (7,8) prevents (6,7) from becoming unmarkable, which can be followed by (1,3) or (3,4) to make sure all edges are markable. Playing (0,1) and (1,2) or (3,4) and (4,5) forces one of the previous moves to be played. Thus, player 2 wins.

Graph 21



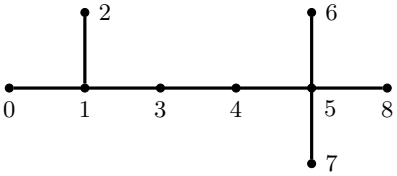
Optimal Strategy: Player 1 can only win if an odd number of edges are unmarkable. Playing (4,6) and (7,8) forces (6,7) to be markable. Playing (0,1) and (2,4) prevents (1,2) from becoming unmarkable. Playing (1,2) and (6,7) makes (2,4) and (4,6) markable. Playing (2,3) and (4,5) forces one of the previous moves to be made. This makes it so player 2 wins.

Graph 22



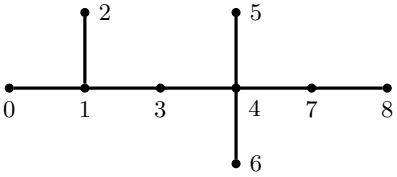
Optimal Strategy: Player 1 can only win if there are an odd number of unmarkable edges. Playing (1,2) and (4,7) or (2,3) and (4,5) prevents (2,4) from becoming unmarkable. Playing (0,1) and (2,4) prevents (1,2) from becoming unmarkable. Playing (5,6) and (7,8) prevents (4,5) and (4,7) from becoming unmarkable. Therefore, player 2 wins.

Graph 23



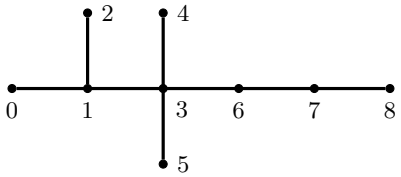
Optimal Strategy: Player 1 can only win if there are an odd number of unmarkable edges. Player 2 will play (3,4) followed by one of (5,6), (5,7), or (5,8) and win since there will be no unmarkable edges possible. If player 1 plays (3,4) first, player 2 will play (5,7) and win.

Graph 24



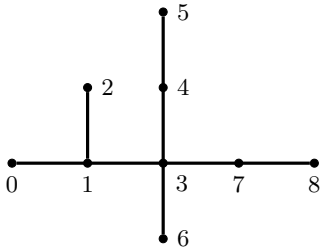
Optimal Strategy: Player 1 can only win if there are an odd number of unmarkable edges. Player 2 will play (1,3) then (4,7) to prevent any edges from becoming unmarkable. If player 1 plays (1,3) first, player 2 will play (4,7) so (3,4) becomes markable. Player 2 therefore will win.

Graph 25



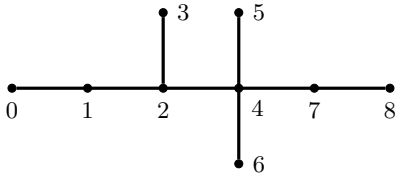
Optimal Strategy: Player 1 can only win if there are an odd number of unmarkable edges. Player 2 will play (6,7) then (1,3) to prevent any unmarkable edges. If player 1 plays (6,7) first, then player 2 will play (1,3) to prevent (3,6) from being unmarkable. Therefore, player 2 wins.

Graph 26



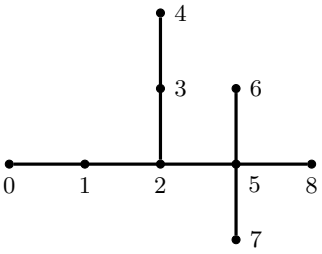
Optimal Strategy: Player 1 can only win if there are an odd number of unmarkable edges. Player 2 will play (3,7) then (3,4) to prevent any edges from being unmarkable. If player 1 starts with (3,7), then player 2 will play (3,4) to prevent (1,3) from becoming unmarkable. Therefore, player 2 wins.

Graph 27



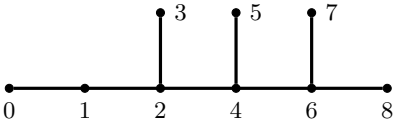
Optimal Strategy: Player 1 can only win if there are an odd number of unmarkable edges. Player 2 will play (4,7) then (1,2) to prevent any of the edges from becoming unmarkable. If player 1 leads with (4,7), player 2 will play (1,2) to prevent (2,4) from becoming unmarkable. Therefore, player 2 wins.

Graph 28



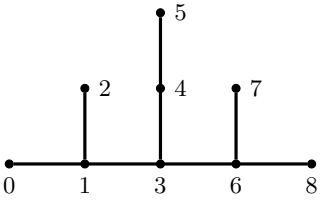
Optimal Strategy: Player 1 can only win if there are an odd number of unmarkable edges. Playing (0,1) and (2,3) or (1,2) and (3,4) prevents (2,5) from becoming unmarkable. Playing (2,5) can be followed up by either (1,2) or (2,3) to prevent the other from becoming unmarkable. Playing (5,6), (5,7), or (5,8) can be followed by (2,5) to force one of the first four moves mentioned to be played. This means that player 2 wins.

Graph 29



Optimal Strategy: Player 1 can only win if there are an odd number of edges that are unmarkable. Player 2 will play (1,2) then (4,6) to make sure all edges are markable. If player 1 plays (1,2) first, player 2 will play (4,6) to prevent (2,4) from becoming unmarkable. This means that player 2 will win.

Graph 30

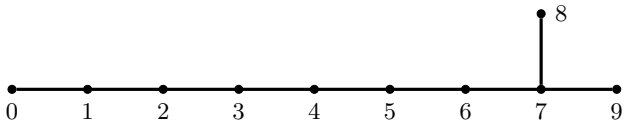


Optimal Strategy: Player 1 can only win if there are an odd number of edges that are unmarkable. Player 2 will play (1,3) then (3,6) to make sure all edges are markable. If player 1 plays (1,3) first, player 2 will play (3,6) to prevent (3,4) from becoming unmarkable. This means that player 2 will win.

A.6 Nine Edges / Ten Vertices (Player 1 optimal strategy)

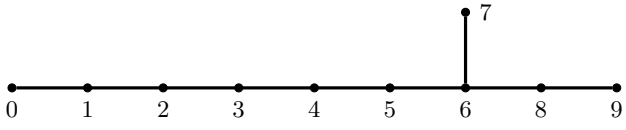
The following are a selection of trees on ten vertices and their optimal strategies.

Graph 1



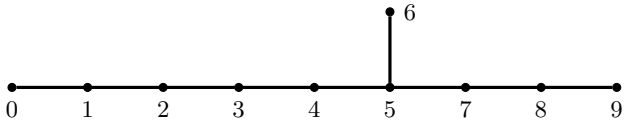
This is an Y-graph. As described in Section 3.3, Player 1 has the optimal strategy.

Graph 2



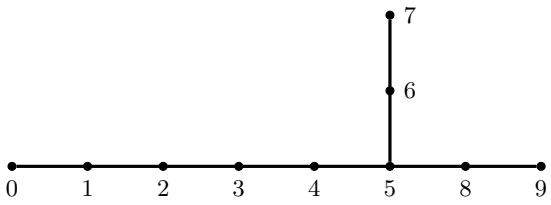
This is an L-head graph. As described in Section 3.3, Player 1 has the optimal strategy.

Graph 3



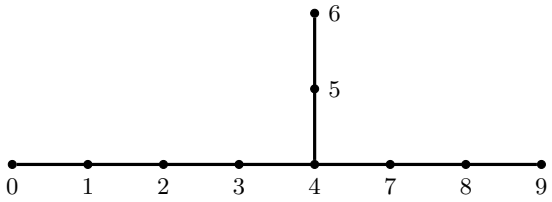
Optimal Strategy: Player 2 can only win if only one of (1,2), (2,3), (3,4), (4,5), (5,7), or (7,8) become unmarkable. Player 1 will play (2,3) first. If player 2 plays (5,6), (5,7), or (8,9), player 1 will play (7,8). If player 2 plays (7,8), player 1 will play (5,6) facing the opposite direction of vertex 5 from player 2's move. If player 2 plays (0,1) or (1,2), player 1 will play (4,5) where either 0 or 2 of the edges are unmarkable. If player 2 plays (3,4) or (4,5), player 1 will play (0,1) where either 0 or 2 of the edges are unmarkable. In all cases, there will either be 0 or 2 unmarkable edges, so player 1 wins.

Graph 4



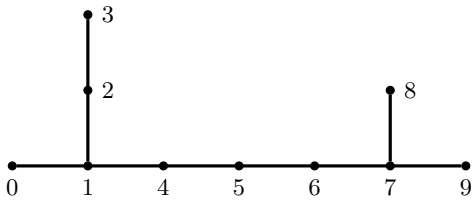
Optimal Strategy: Player 2 can only win if only one of (1,2), (2,3), (3,4), (4,5), (5,6), or (5,8) become unmarkable. Player 1 will play (2,3). For every move player 2 makes, player 1 will play the mirror move. The mirrors are (0,1) and (4,5), (1,2) and (3,4), (5,6) and (5,8), and (6,7) and (8,9). This ensures that an even number of edges are unmarkable, so player 1 wins.

Graph 5



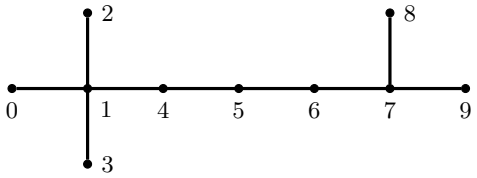
Optimal Strategy: Player 2 can only win if only one of (1,2), (2,3), (3,4), (4,5), (4,7), or (7,8) become unmarkable. Player 1 will start with (8,9). If player 2 plays (4,5) or (5,6), player 1 will play the opposite move. If player 2 plays (7,8), player 1 will play (4,7). If player 2 plays and makes (7,8) markable, player 1 will play (4,5) facing the opposite direction from vertex 4 as player 2's move. If player 2 plays (0,1) or (1,2), player 1 will play (3,4). If player 2 plays (2,3) or (3,4), player 1 will play (0,1). If player 2 plays (4,7) and makes (7,8) unmarkable, player 1 will play (4,5) facing the same direction from vertex 4 as player 2's move. Then, player 1 will play (0,1) facing the opposite direction as (4,7) on their turn, ensuring 2 edges are unmarkable. In all cases, either 0 or 2 edges are unmarkable, so player 1 wins.

Graph 6



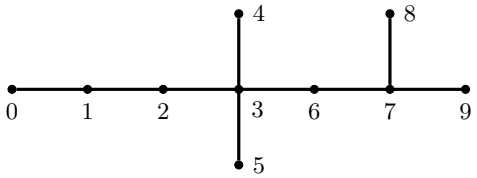
Optimal Strategy: Player 2 can only win if only one of (1,2), (1,4), (4,5), (5,6), or (6,7) become unmarkable. Player 1 will play (1,2). If player 2 plays (0,1), (2,3), (7,8), or (7,9), player 1 will play either (7,8) or (7,9) facing the opposite direction from vertex 7 as player 1's first move. This ensures that no matter what move player 2 plays next, player 1 can make 0 or 2 unmarkable edges, thus winning.

Graph 7



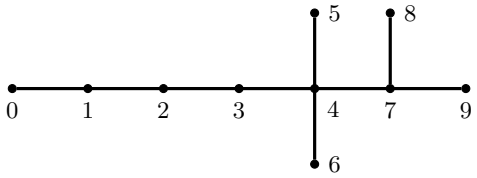
Optimal Strategy: Player 2 can only win if only one of (1,4), (4,5), (5,6), or (6,7) become unmarkable. Player 1 will play (0,1) first. Having the next two moves be (1,4) and (5,6) or (4,5) and (6,7) prevents any edges from becoming unmarkable. Having the next two moves be (1,2) and (1,3) or (7,8) and (7,9) ensures that player 1 does not accidentally make an unmarkable move. Therefore, player 1 wins.

Graph 8



Optimal Strategy: Player 2 can only win if there are an odd number of unmarkable edges. Player 1 will play (1,2) first. If player 2 plays an edge touching vertex 3, player 1 will play (3,6) facing the same direction as player 1's first move. If player 2 plays an edge touching vertex 7, player 1 will play (6,7) facing the same direction as player 1's first move. Otherwise, player 1 will play (2,3), (3,6), and (6,7) until all edges are markable, causing player 1 to win.

Graph 9



Optimal Strategy: Player 2 can only win if an odd number of edges are unmarkable. Player 1 will start with (4,7). Until player 2 plays (0,1), (1,2), (2,3), or (3,4), player 1 will play on the other 4 edges, with either (4,5) or (4,6) being played first to prevent (3,4) from becoming unmarkable. When one of the 4 previously mentioned edges are marked, player 1 will play a move that ensures no edges are unmarkable, making them the winner.