Predicting the Outcomes of Impartial Games

Sergio Abraham Sum Munoz

Wofford College  
429 North Church Street  
Spartanburg, SC 29303-3663  
+1 (864) 597-4000

summunozsa@email.wofford.edu

**ABSTRACT**

in this paper we describe an algorithm based on game theory that can be used to predict the winner of AN IMPARTIAL GRAPH GAME, OR A VARIATION OF NIM.

**Keywords**

Impartial combinatory game; game state; optimal play; terminal node; playable soldier; complex node;

# Introduction

On HackerRank, there is a problem called Play on Benders that presents a graphical variation of Nim [3]. Here, two generals, Bumi and Iroh, are trying to find the best way to move soldiers around a city during battle. The city is separated into different locations with certain locations being connected by one-way paths. To strategize, they decide to play a game where soldiers start at random positions in the city and they both take turns moving one soldier from one location to another at a time. Since the city was built by some of the best architects in the world, soldiers will never be moving in circles. The first general to not be able to perform a legal move loses. In other words, the last general to make a valid move wins. It can be assumed that both generals will be making the best possible move each time and Bumi will always go first. Figure 1 represents an instance of what the battlefield will look like. The numbers inside the nodes are the labels for their respective node.

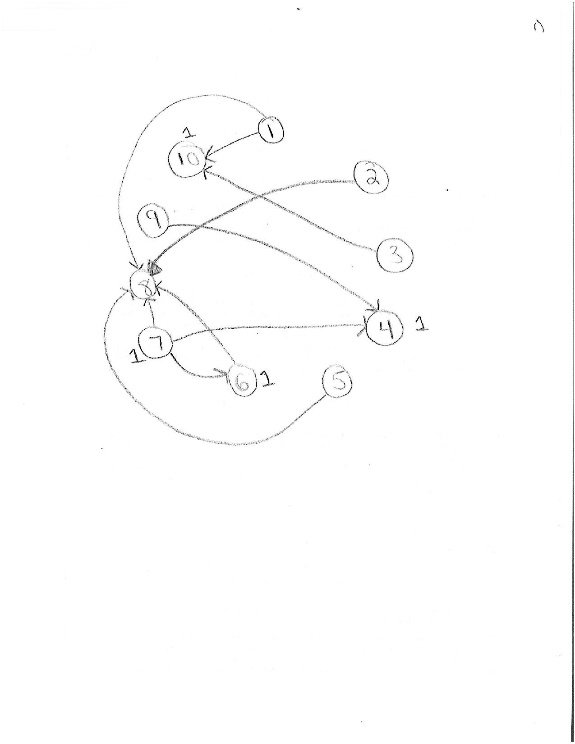


Figure . Problem Instance

On this example, we observe that there is a total of four soldiers with one soldier being at nodes 4, 6, 7, and 10. However, the only playable soldiers are at nodes 6 and 7 because the soldiers at nodes 4 and 10 cannot move anymore. In this example, Bumi will win the game. He does so by first moving the soldier on node 7 to node 6. Node 6 now contains two playable soldiers. Iroh then has no other option but to move one of the soldiers at node 6 to node 8, leaving a soldier behind at node 6. Bumi wins by moving the last playable soldier from node 6 to node 8.

# Background

Game theory is the study and application of decision making strategies that can be applied to behavioral relations and has evolved from studying the outcomes of games [1]. It is important to note that there are many game types that are studied, but we will focus on combinatorial games.

A game is considered **combinatorial** when only two players move alternatively, the game is not decided by chance, both players know the game state and all possible player moves, the game eventually ends, and the winner depends on who moves last with no ties allowed [1]. A specific type of combinatorial game we will predict the outcome of is the impartial game type. A game is considered **impartial combinatorial** when “the set of allowable moves depends only on the position of the game and not on which of the two players is moving” [2]. An example of an impartial combinatorial game is Nim.

The problem instance from Play on Benders can be represented as a directed acyclic graph using an adjacency list. Each location is a node and each directed edge represents the path from one location to the other. Figure 2 shows the adjacency list for Figure 1.

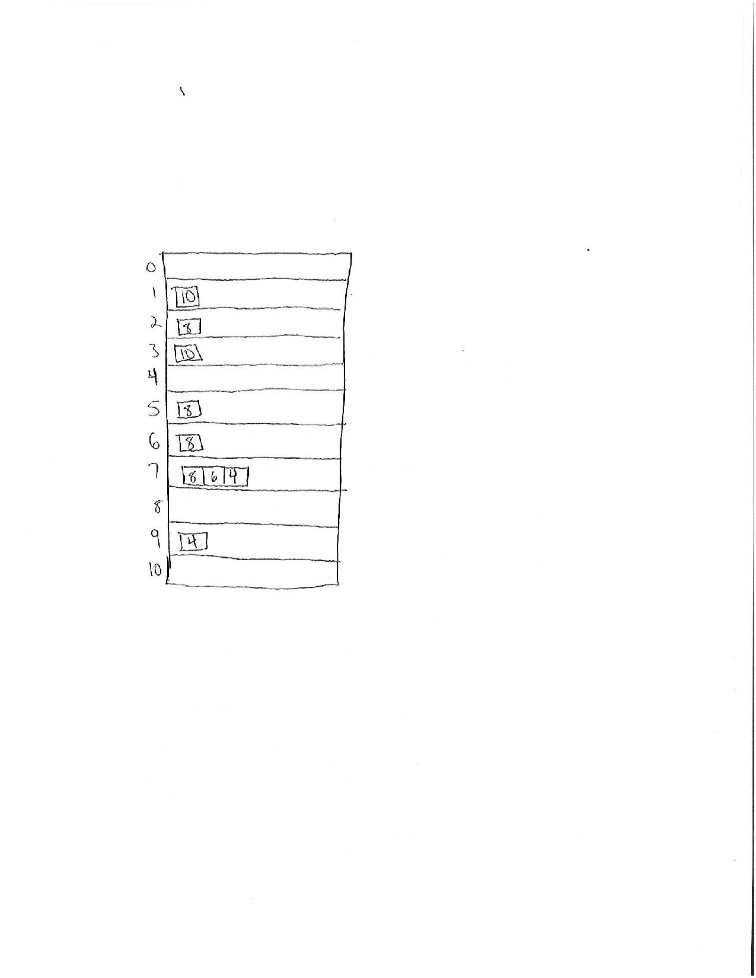


Figure . Adjacency List of Graph in Figure 1

A location that does not have a path to another location will be considered **terminal** and will have an empty list at its respective index in the adjacency list. Playable soldiers will not start off at terminal nodes.

# Algorithm

We first determined which nodes are terminal. Soldiers that start at **terminal nodes** do not need to be considered in the algorithm since they cannot be moved. The remaining **playable soldiers** are at nodes that have at least one neighbor. We also determined if there are situations where a node has more than one neighbor and if any of those neighbors is not terminal [Pech, personal communication]. A node that falls under this situation is considered a **complex node.** See Figure 3, which shows an instance of this **complex node**.

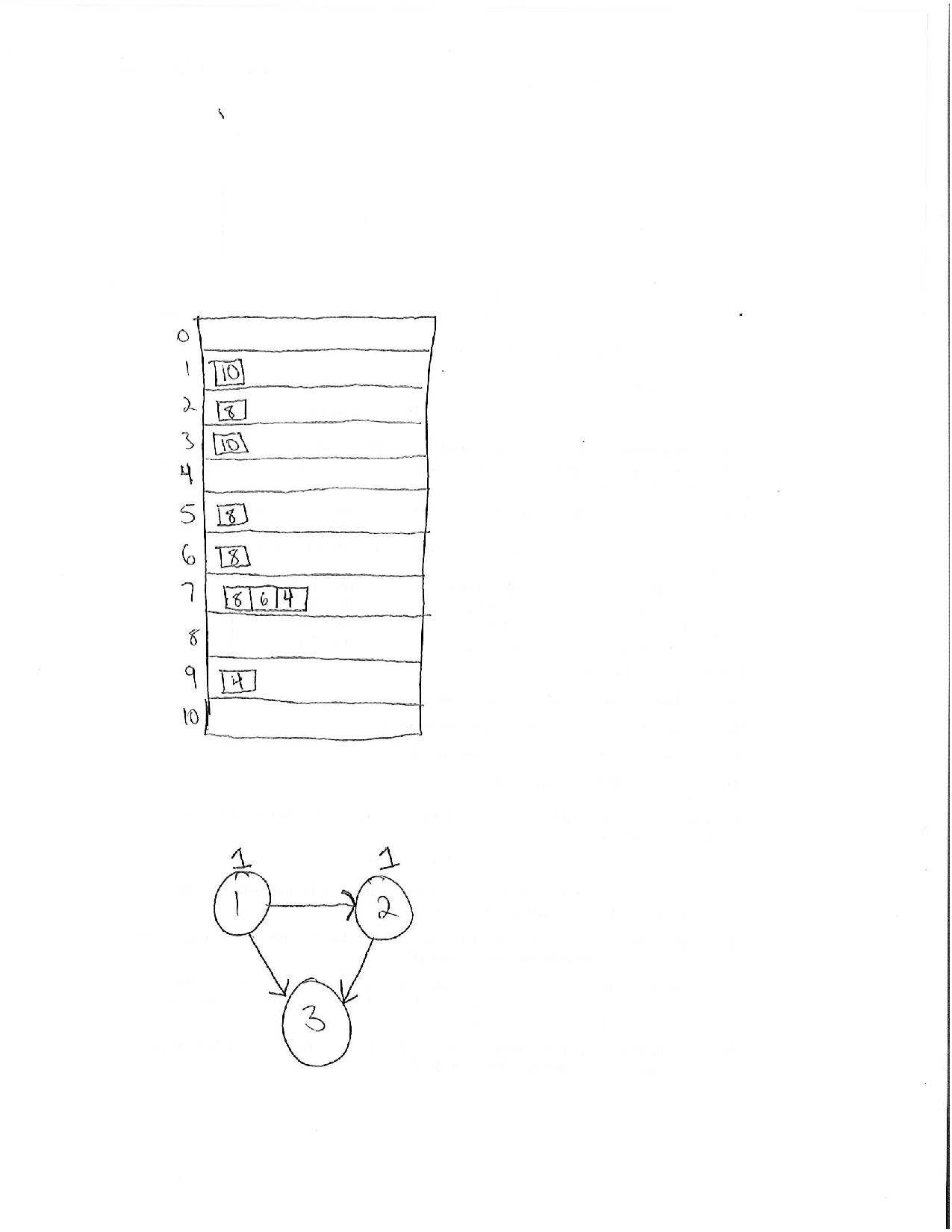


Figure . Complex Node

Node 1 in Figure 3 is the complex node since it has more than one neighbor and one of them is not terminal (node 2).

We determined that the following mathematical expression can be used when analyzing each node that has at least a **playable soldier** at the start of the game [Pech, personal communication]:

*S* represents the number of soldiers that started at the node and *h* represents the *maximum* number of hops each soldier must take until landing at a terminal node. Since each node that is analyzed will have its own expression of the form , all the expressions are added together to produce a final sum. Essentially, the final sum represents the total number of moves that will be performed in the game. If the final sum is odd, player 1 wins. This is true because player 1 will have the last move on an odd number of total moves. If the final sum is even, then player 2 wins since player 2 gets to go last on an even number of total moves.

## Pseudocode

*Bumi\_or\_Iroh(g, query, n*)

// Input: A graph *g,* a list of ints representing starting location of

// each soldier *query,* and a positive int

// Output: Bumi or Iroh

non\_terminals = [] //list of nodes that aren't terminal of size n+1

For each node in g:

If node not terminal:

Non\_terminals[node] = True

Else:

Non\_terminals[node] = False

Final\_sum = 0

// find max value of nodes to travel until terminal per soldier

For each soldier in query:

Determine if its starting node is terminal

If starting node is not terminal:

*hops* = maximum hops until reaching terminal node

Final\_sum += *s* \* *hops // s =* # of soldiers at start node

if final\_sum is even:

return Bumi

else:

return Iroh

## An Example

We will now refer to the example given in the introduction. There are 4 soldiers on the battlefield. There is a soldier at nodes 10, 7, 6, and 4 at the start of the game. Figure 4 shows what this graph looks like.

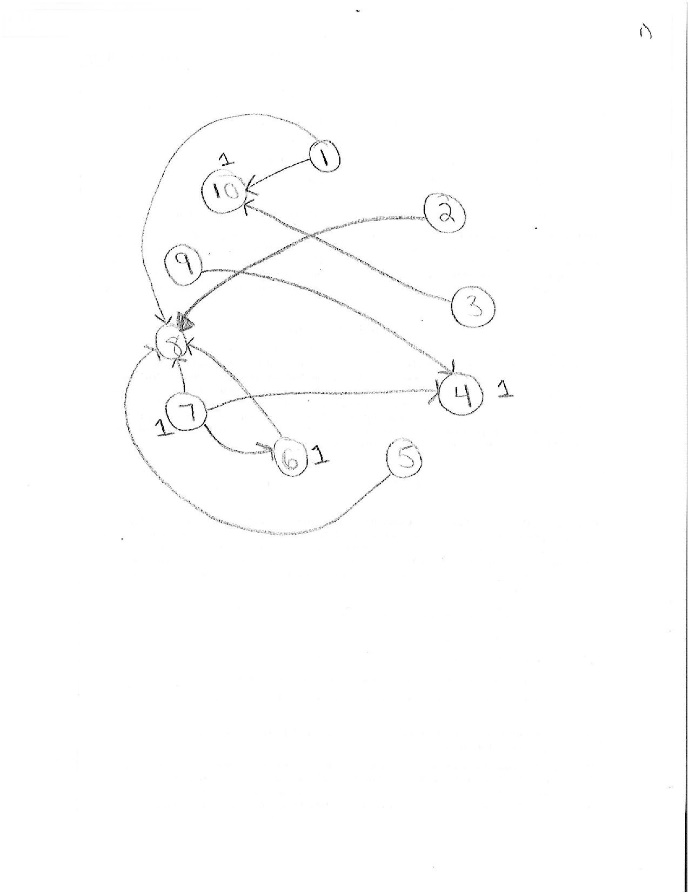


Figure . Graph of Problem Instance

Following the algorithm, the terminal nodes here are 4, 8, and 10. Immediately, we can discard the soldiers at node 10 and 4 since they are not playable. Figure 5 demonstrates this.

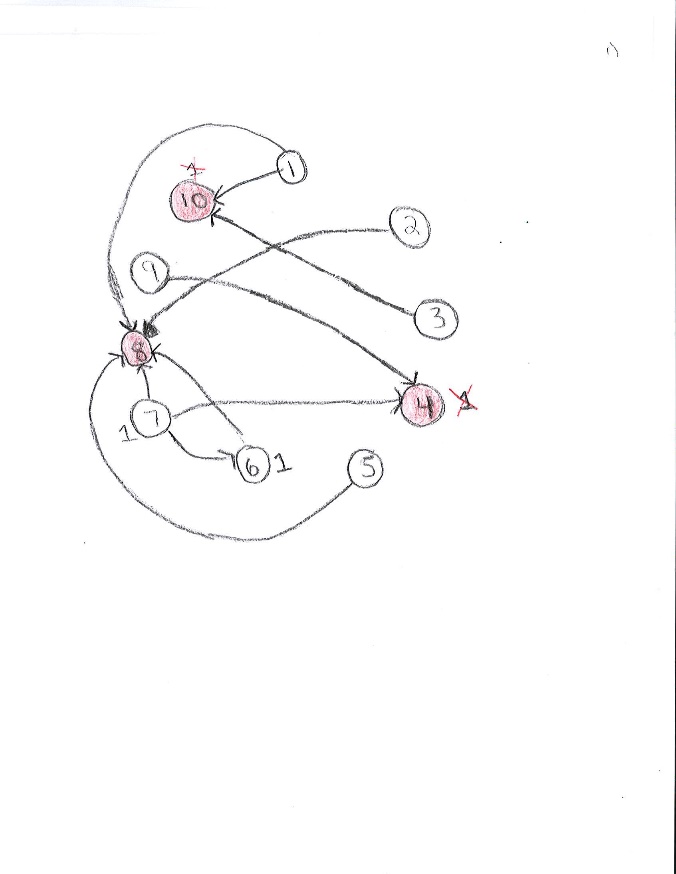


Figure . Updated Problem Instance showing terminal nodes

In Figure 5, the terminal nodes are shaded in red. We also crossed out the soldiers that started off at nodes 10 and 4 since they cannot be moved anymore.

We only care about the soldiers at nodes 7 and 6. The expression for the soldier at node 6 is simple: 1x1. This considers the number of soldiers at this node times the maximum number of paths until it reaches node 8, a terminal node.

The soldier at node 7 is located at a **complex node** where the node connects to two other locations and one of them is not terminal. Figure 6 shows a closer look at node 7.

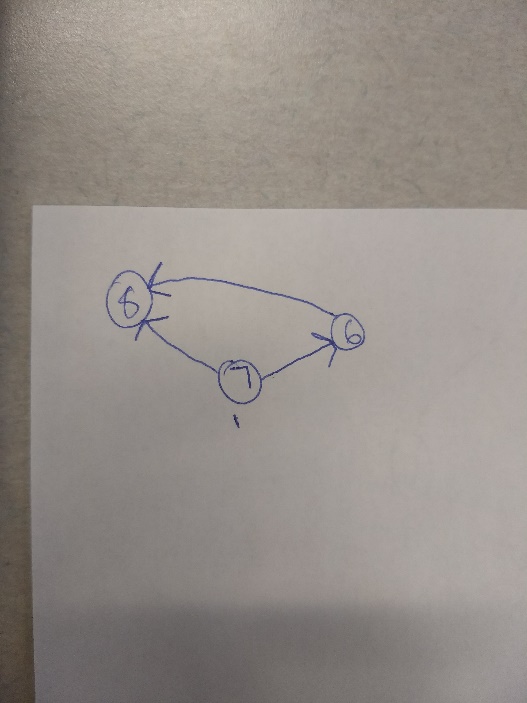


Figure 6. Node 7

We determined that the longest path for the soldier in node 7 to reach a terminal node is 2 hops. Therefore, its expression is 1 \* 2 where 1 is the number of soldiers at the node and 2 is the longest path until reaching a terminal node. Figure 7 demonstrates the expression for each node being analyzed.

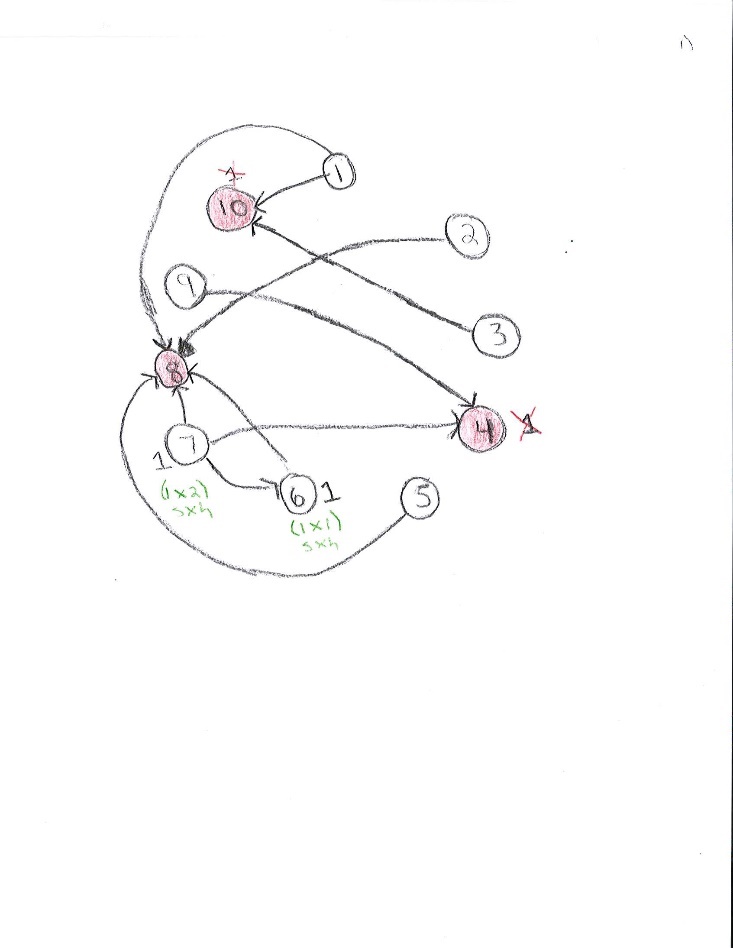


Figure 7. Expressions in Green Represent their Nodes

By adding both expressions from Figure 7, we get a final sum of 3. Since Bumi was player 1 in this instance, he wins.

## Time Complexity

Although the implementation of the algorithm for Play on Benders is not complete, finding the *maximum* number of hops until a soldier reaches a terminal node seems to be exponential. That is because the algorithm loops through the list of neighbors in the adjacency list. It gets worse as neighbors are not terminal and have neighbors that are not terminal as well.

# ConclusionS

By working on Play on Benders, we had the chance to learn about game theory. After talking to the creator of Play on Benders, Ray Valiente [Valiente, personal commuication], he pointed us towards an article that used the Sprague-Grundy Theorem to approach this problem. We wished we had more knowledge about using this theorem since it would need a dynamic programming approach to be implemented. We are interested in trying to solve other game theory problems. This problem also helped us to get more familiar with graphs. We gained more skills in trying to traverse through a graph.

# ACKNOWLEDGMENTS

Our thanks to ACM SIGCHI for allowing us to modify templates they had developed. We are also thankful to Dr. Wesley Pech for giving us a crash course in game theory. Thank you to Ray Valiente for posing this problem on HackerRank and sending us material to learn about game theory.

# REFERENCES

1. Casey, M., 2016*. Basic Game Theory*. SlidePlayer.com (2016), DOI= http://slideplayer.com/slide/7510284.
2. Ferguson, T. 2016. *Game Theory*. University of California Los Angeles, DOI= http://www.math.ucla.edu/~tom/Game\_Theory/comb.pdf.
3. Valiente, R. 2018. *Play on Benders.* HackerRank. DOI= https://www.hackerrank.com/contests/101apr14/challenges/benders-play/problem.