Chris Alpuerto
Jonathan Quiroz
jquiroz44@csu.fullerton.edu
chrisalpuerto@csu.fullerton.edu

Project 2 Report

Algorithm 1: Walls of Maria and the Titans by Chris Alpuerto

In the Walls of Maria and the Titans problem, we are given an m x n 2D grid representing a city under siege. Each cell in the grid can either be a Titan-infested area represented by -1, a safe stronghold represented by 0, or an open city area represented by INF (represented by a string). The task is to update the grid so that each INF cell is replaced by the shortest distance to the nearest stronghold. If a city block cannot reach a stronghold, it should remain INF. Movement is only allowed up, down, left, or right, not diagonally. To solve this, I used a multi-source breadth-first search (BFS). I first added all the safe strongholds to a queue and then expanded outward, updating each open cell with its minimum distance from a stronghold. This strategy ensures that the algorithm processes all paths optimally and efficiently updates reachable areas while ignoring Titan-infested zones.

Algorithm 2: Vibe Check - Card Shuffle by Jonathan Quiroz

In the Vibe Check - Card Shuffle problem, the goal is to determine whether a list of cards can be grouped into sets of a specific size, where each group contains consecutive values. We are given an array of integers representing the hand of cards and an integer groupSize that defines the size of each group. The requirement is that each group must have exactly groupSize cards, and the cards within the group must be

Chris Alpuerto
Jonathan Quiroz
jquiroz44@csu.fullerton.edu
chrisalpuerto@csu.fullerton.edu

consecutive numbers. To solve this, I used a greedy algorithm with a frequency counter. I counted the occurrences of each card, then sorted the card values and tried to form groups starting from the smallest available card. For each starting point, I checked if enough consecutive cards existed to form a full group, reducing the count for each card used. If any group couldn't be completed, the function would return false. This method is efficient and works well for large input sizes.

Both of these problems required different algorithmic techniques but shared the common goal of building efficient and scalable solutions under specific constraints. The first focused on grid traversal and shortest paths, while the second centered around sequence grouping and greedy decision-making.