Technical report of 1000 words for the treasure hunt game project

Contents

[Introduction 1](#_Toc2100833553)

[Project Overview 1](#_Toc939142327)

[How to Play the Game 2](#_Toc615616030)

[Challenges and solutions 2](#_Toc339408720)

[Search Algorithms 2](#_Toc868762329)

[Breadth-First Search (BFS) 2](#_Toc664848911)

[Depth-First Search (DFS) 3](#_Toc944214866)

[Binary Search (BS) 3](#_Toc417884086)

[Algorithm Summary 4](#_Toc1435602452)

[Pseudocode/ Flowchart 4](#_Toc1817574857)

[Test Evidence 6](#_Toc1570846087)

[Conclusion 6](#_Toc1226064594)

# Introduction

Welcome to Sango Treasure Hunt! Get ready to dive into an exhilarating adventure on a mysterious island rumored to hold hidden treasure. Your team of skilled adventurers will compete fiercely to find this treasure before anyone else. Prepare yourselves for thrilling challenges along the way. The treasure map is divided into grids, each one packed with traps and obstacles that you will skillfully navigate. And don’t forget, there are hidden power-ups waiting to be discovered that will give your team a powerful edge. Be mindful of your health, as each team has a limit. You’ll face challenges that deplete your health, so it’s crucial to work together efficiently to reach the treasure before your health runs out. If it hits zero, your team will be eliminated. So, assemble your crew and gear up. Get ready to conquer Sango Island and claim that treasure with confidence!

# Project Overview

## How to Play the Game

In the Treasure Hunt Game, you play on a grid where the goal is to find hidden treasures, avoid traps, and collect power-ups. Each player starts at the top-left corner of the grid with a set amount of health. Players take turns moving around the grid or searching for items. You can move up**,** down**,** left**,** orright using simple commands like move up or right. If you land on a treasure, your score increases. If you hit a trap, your health goes down, and if you find a power-up, your health increases.

To make the game more strategic, you can use search algorithms like BFS, DFS, or BinarySearch to locate treasures, traps, or power-ups. For example, you can type “search bfs" to find the closest treasure using Breadth-First Search. The game ends when all treasures are found, or all players are out of health.

# Challenges and solutions

During the development of the Treasure Hunt game, several challenges were encountered and addressed effectively. One of the main challenges was ensuring that the player's movement and grid updates functioned correctly without causing errors, such as moving out of bounds or overlapping positions with other players. This was solved by adding conditions to check the player’s position before executing a move and displaying a message when an invalid move occurred. Another challenge was implementing the search algorithms (BFS, DFS, and Binary Search) correctly to locate specific items on the grid. Initially, the algorithms had difficulty handling the grid boundaries and visited positions, but this was resolved by using checks to ensure that only valid grid cells were explored. Debugging and testing each algorithm step-by-step helped verify their accuracy. Finally, managing multiple game elements, such as health, score, and inventory (keys), required careful organization to avoid conflicts in tracking player states. This was addressed by structuring the player class to keep all relevant attributes and updating them after every action. Overall, these solutions ensured smooth gameplay and made the program work as intended.

# Search Algorithms

### Breadth-First Search (BFS)

The Breadth-First Search (BFS) algorithm helps players in the game locate specific items such as treasures, traps, power-ups, keys, and locked treasures. By exploring all possible moves around the player's current position level by level, BFS efficiently finds the shortest path to an item. It starts at the player's location and checks nearby cells, guaranteeing the discovery of the target if it exists, B. Awerbuch (1987).

### Depth-First Search (DFS)

The Depth-First Search (DFS) algorithm offers another option by exploring as far as possible in one direction before backtracking. It uses a stack to track visited locations, allowing for quicker searches in specific areas, though it may not provide the shortest path, resulting in longer search times. Players may choose DFS for a less structured exploration.

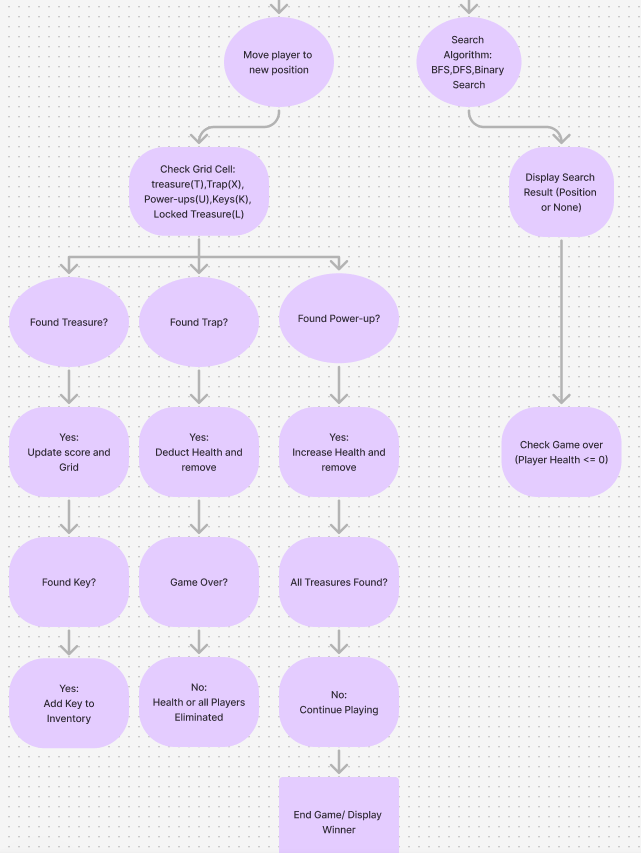
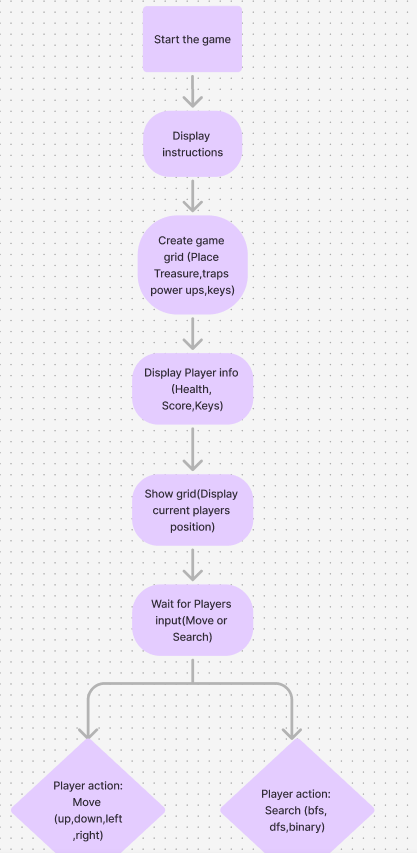
### Binary Search (BS)

The Binary Search (BS) algorithm is used for searching items in a single, sorted row of the grid .Lin ,A.(2019). By dividing the row into halves and checking the middle cell, BS significantly reduces the number of checks needed, making it faster than standard searches. It is useful when players know where the item might be.

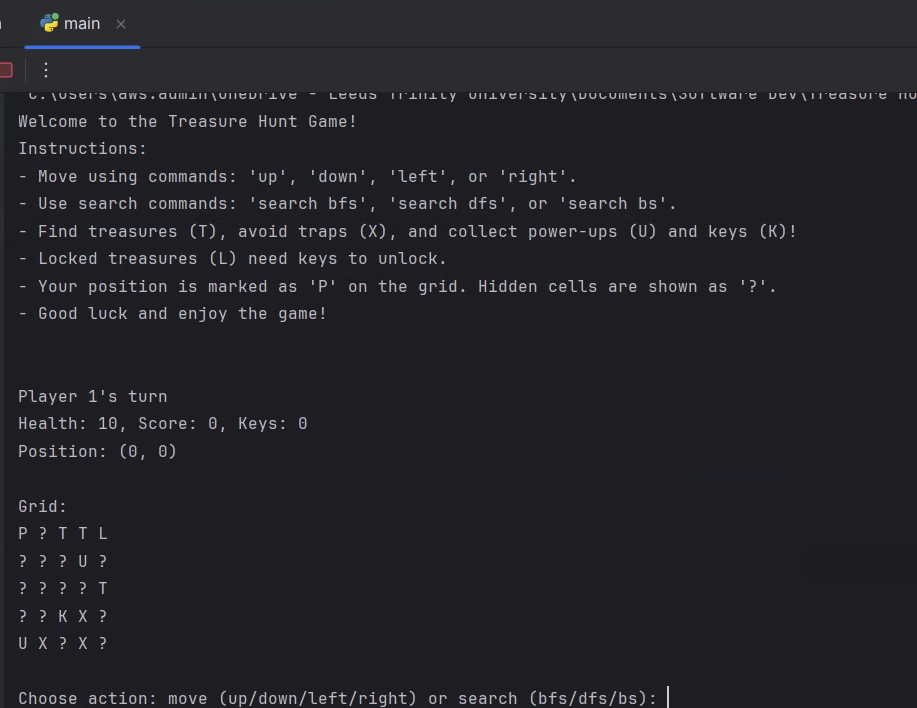
#### Algorithm Summary

BFS, DFS, and Binary Search give players strategic options for item searches based on their preferred playing style: BFS for guaranteed results, DFS for quick but less precise searches, and BS for targeted row searches.

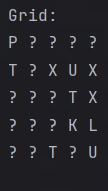
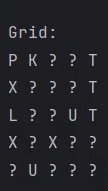
# Pseudocode/ Flowchart



# Test Evidence



The first player starts at the coordinates (0, 0), as this is the default starting position on the grid. The player's status, including health, score, and the number of keys, is displayed at the beginning of each turn to show their progress and current condition. This helps keep track of their performance throughout the game. Players can choose an action such as moving up, down, left, or right, or they can use one of the search algorithms (BFS, DFS, or Binary Search) to find hidden treasures, avoid traps, collect power-ups, or locate keys. The grid updates dynamically after every valid move, reflecting changes such as player position, collected items, or cleared traps. Invalid moves, such as hitting the grid boundary, are detected, and the game prevents the player from moving further in that direction.

The game has been tested to ensure that all mechanics work correctly. For example, collecting treasures increases the player’s score, traps decrease health, and keys allow locked treasures to be unlocked. Power-ups successfully restore health, and the search algorithms locate items as intended. The grid visually displays the player’s position, marked as 'P', and updates with hidden or collected items, maintaining player engagement. The game’s turn-based system ensures smooth gameplay, switching between players while preserving their individual progress.

Overall the game functions as designed, with all movement, interactions, and search algorithms working properly. Future improvements could include adding more features, such as additional types of obstacles or rewards, and implementing a graphical user interface (GUI) for a better user experience. A GUI would allow players to interact with the game visually instead of relying on text commands, making the game more intuitive and engaging.

# Conclusion

Sango Treasure Hunt is a fun and engaging game that combines strategy, exploration, and adventure in a 5x5 grid setup. Players explore the grid to find hidden treasures while avoiding traps and gathering power-ups. The turn-based mechanics, along with health tracking and scorekeeping, create a smooth experience for everyone involved.

By using search algorithms like Breadth-First Search (BFS), Depth-First Search (DFS), and Binary Search (BS), players can approach the game with a strategic mindset, adding to the overall challenge and enjoyment.

During the development process, I faced several challenges, including ensuring accurate player movement, updating the grid correctly, and making sure the algorithms worked as intended. These issues were resolved through careful debugging and thoughtful design. The game effectively combines exploration with decision-making, requiring players to manage their actions to succeed. Testing showed that all key features, like player interactions and grid updates, worked well.

Conclusively, Sango Treasure Hunt meets its goal of being an exciting and strategic treasure-hunting game. Future improvements, such as adding a graphical user interface (GUI) or more complex gameplay elements, could make it even more engaging. This project showcases my understanding of programming concepts and creative design, marking a successful step in my journey in game development.

References

Lin, A. (2019). Binary search algorithm. WikiJournal of Science, 2(1), 1–13. <https://search.informit.org/doi/10.3316/informit.573360863402659>

B. Awerbuch and R. Gallager, "A new distributed algorithm to find breadth first search trees," in IEEE Transactions on Information Theory, vol. 33, no. 3, pp. 315-322, May 1987, doi: 10.1109/TIT.1987.1057314.