COM4103 Software Development Treasure Hunt Report

Contents

Introduction

This treasure hunt game is python based. The aim is for players to find treasure before their opponent by navigating a grid, collecting powerups, avoiding traps and using search algorithms.

It will show you how the use of Binary Search, Breadth-First Search (BFS) and Depth-First Search (DFS) algorithms can make the process of finding the treasure faster. Random grid elements, health tracking, and turn-based gameplay all make the game both difficult and enjoyable.

This report explains how the game was developed, problems encountered along the way, and how they were solved. It also demonstrates application of programming skills in creating a game.

Project Overview

This treasure hunt is a grid-based game where two players are competing against each other to see who can find the treasure first. The grid contains powerups that boost health, traps that reduce health, and treasure, all of which are places randomly for each game. Both players must take it in turns to move whilst avoiding losing all of their health and being eliminated.

There are three search algorithms that players can use to help them find the treasure faster:

1. Binary Search - Quickly narrows down rows or columns to locate key items.
2. Breadth-First Search (BFS) - Finds the shortest and safest path to the treasure.
3. Depth-First Search (DFS) - Explores alternate routes to avoid obstacles or find hidden power-ups.

The game is turn based which makes it more fair for both players. The health tracking makes the game more challenging as players want to avoid the traps and collect the powerup so that they do not die. The element of randomisation makes the game a lot more exciting as it is certain that every game will be different to the previous one.

Challenges and Solutions

One challenge I faced along the way was not knowing how to incorporate the search algorithms into the game. I was unsure of the structure and how to allow players to use them. After many different trials and wondering where to place them, I decided to create a list of options for the players to choose their next move. This list would include all three algorithms and if selected, they would then help narrow down the search for the treasure.

Another problem I faced was not knowing where the players were on the grid. When I first created the grid, I made it so that I could see where the powerups, traps and treasure was. When I tested the game, I quickly came to the realisation that it was going to be very confusing for players if they could not see where they were / how far they were from the treasure. I overcame this problem by adding the players onto the map (P1 & P2) so that they could track where they were.

Search Algorithms

Three search algorithms have been used to create this game.

1. "Binary search is an efficient algorithm for finding an item from a sorted list of items. It works by repeatedly dividing in half the portion of the list that could contain the item,

until you've narrowed down the possible locations to just one" (Khan Academy, n.d.). This has been implemented in this game to help players narrow down their search for the treasure, traps and powerups in a specific row. It will half the row and then search to see if the item your looking for is in that row.

2. "Breadth-First Search (BFS) is one of the simplest and most widely used algorithms. It explores all aspects of something to find a solution, starting from a given starting point and moving outward layer by layer. The algorithm does not make assumptions about the possible location of the solution but explores all nodes equally, ensuring that it checks every possibility until it finds the target or exhausts all options" (CelerData, 2024). In this game, BFS starts where the player is and checks all the nearby cells. If the treasure is found it returns the path, if not it keeps going until all moves have been checked.
3. "DFS (Depth initial search) is an algorithm that starts with the first node in the network and continues to explore until it discovers the desired node or even a node with no children" (unacademy, n.d.). In the game, DFS starts at the players position and moves to one of the next cells and marks it as visited. If the treasure is found, if finishes. If not, it explores a different path.

Flowchart

I have created a flowchart using figma to show how to play the game:

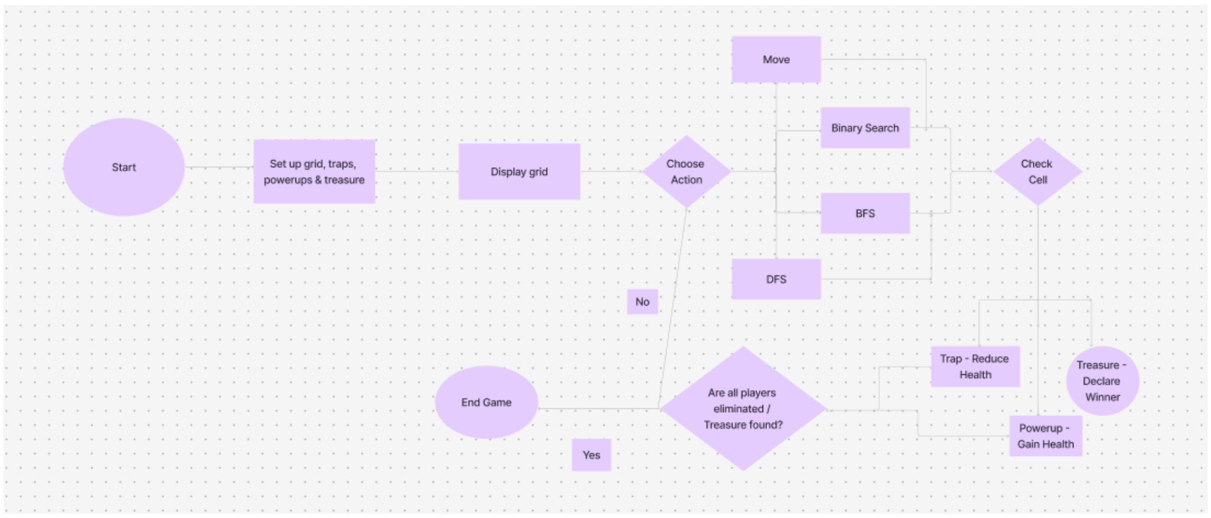
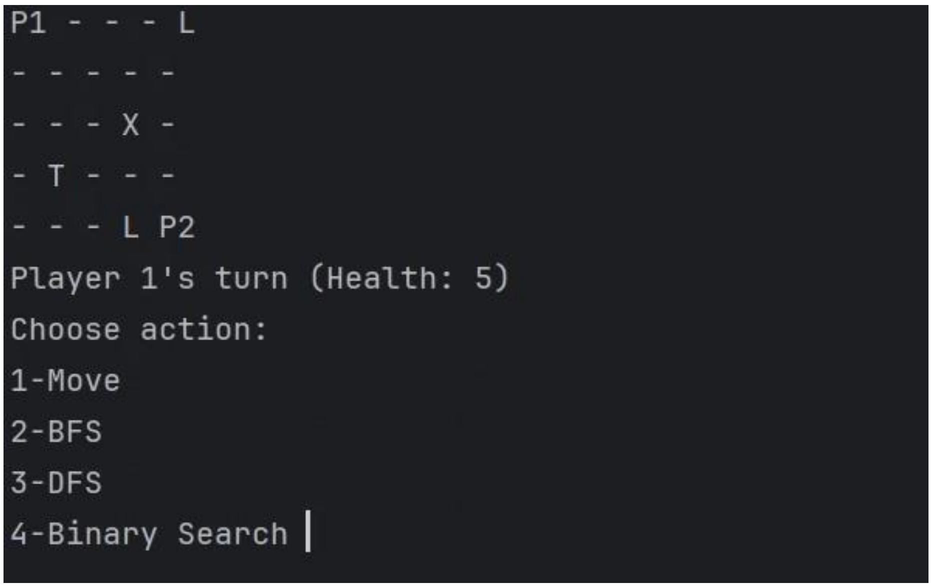


Figure 1: Image of Flowchart

Test Evidence

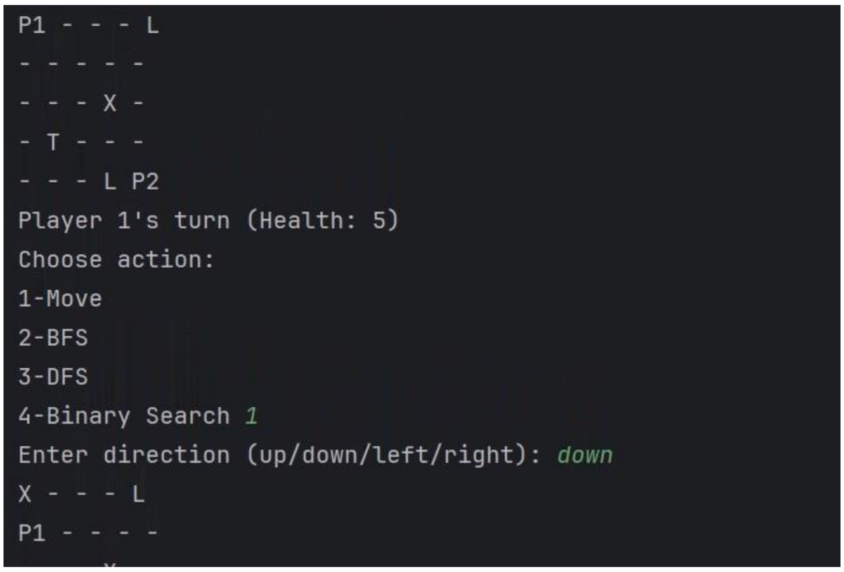
Here is some test evidence of me testing the game and its many features:



```
P1 - - - L
- - - - -
- - - X -
- T - - -
- - - L P2
Player 1's turn (Health: 5)
Choose action:
1-Move
2-BFS
3-DFS
4-Binary Search
```

Figure 2: First glimpse of game

When you first run the game, you are presented with the above (figure 2). The grid shows the two players, traps(L), powerups(X) and treasure(T). It says the which players turn it is their current health and a choice of actions which you must choose from.

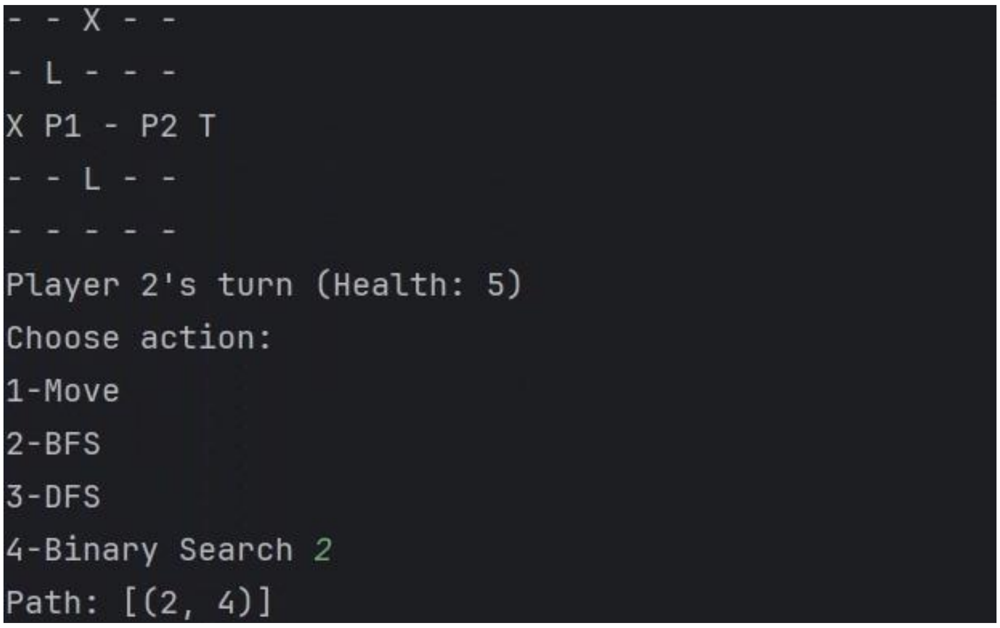


```
P1 - - - L
- - - - -
- - - X -
- T - - -
- - - L P2
Player 1's turn (Health: 5)
Choose action:
1-Move
2-BFS
3-DFS
4-Binary Search 1
Enter direction (up/down/left/right): down
X - - - L
P1 - - - -
```

Figure 3: Demonstration of what happens when you choose action 1

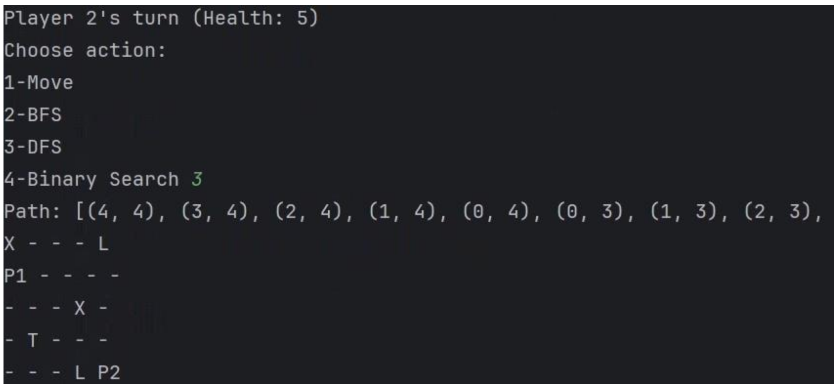
When choosing action 1, you are asked which direction you wish to move in. Once entered your counter then moves into the neighbouring cell. As seen in Figure 3, P1 has moved one space down.

Then it is automatically player 2’s turn, and the process repeats itself.



```
- - X - -
- L - - -
X P1 - P2 T
- - L - -
- - - - -
Player 2's turn (Health: 5)
Choose action:
1-Move
2-BFS
3-DFS
4-Binary Search 2
Path: [(2, 4)]
```

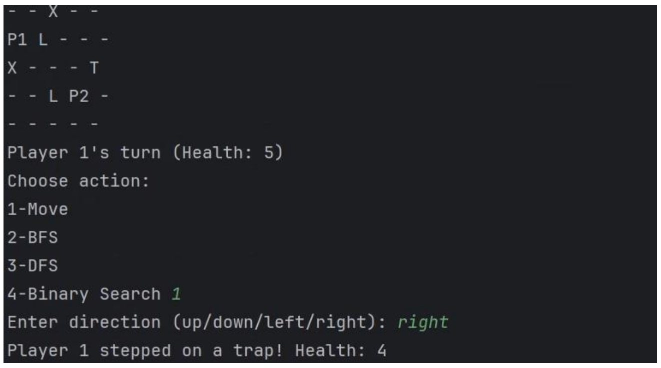
Figure 4: This shows what happens when you choose BFS



```
Player 2's turn (Health: 5)
Choose action:
1-Move
2-BFS
3-DFS
4-Binary Search 3
Path: [(4, 4), (3, 4), (2, 4), (1, 4), (0, 4), (0, 3), (1, 3), (2, 3),
X - - - L
P1 - - - -
- - - X -
- T - - -
- - - L P2
```

Figure 5: This shows what happens when you choose DFS

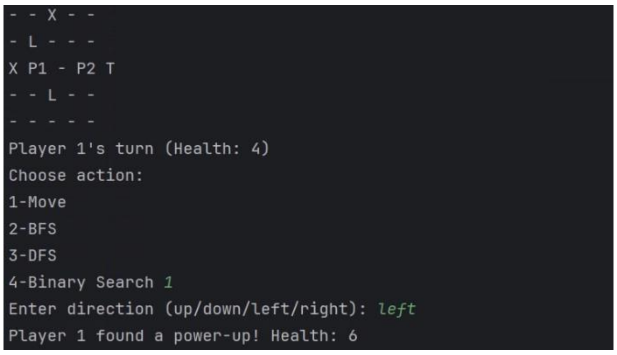
Figures 4 and 5 show what happens when you choose to use BFS or DFS. You are shown your path to get to treasure.



```
- - X - -
P1 L - - -
X - - - T
- - L P2 -
- - - - -
Player 1's turn (Health: 5)
Choose action:
1-Move
2-BFS
3-DFS
4-Binary Search 1
Enter direction (up/down/left/right): right
Player 1 stepped on a trap! Health: 4
```

Figure 6: This shows what happens when you land on a trap

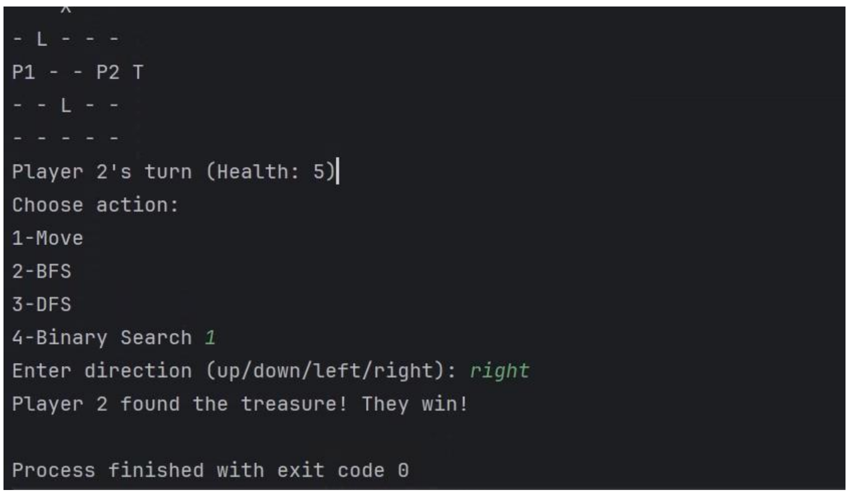
When you land on a trap, the original health of 5 falls down to 4. If this continues and all health is gone you are eliminated.



```
- - X - -
- L - - -
X P1 - P2 T
- - L - -
- - - - -
Player 1's turn (Health: 4)
Choose action:
1-Move
2-BFS
3-DFS
4-Binary Search 1
Enter direction (up/down/left/right): left
Player 1 found a power-up! Health: 6
```

Figure 7: This shows what happens when you land on a powerup

When landing on a powerup, health increases by 2.



```
- L - - -
P1 - - P2 T
- - L - -
- - - - -
Player 2's turn (Health: 5)
Choose action:
1-Move
2-BFS
3-DFS
4-Binary Search 1
Enter direction (up/down/left/right): right
Player 2 found the treasure! They win!

Process finished with exit code 0
```

Figure 8: Winning message

When the treasure has been found, a message is displayed to say which player found the treasure. The game stops here. However, if you wish to replay, you re run the code and a new grid will appear with all elements in different places to the previous game played.

Conclusions

This game shows how search algorithms have been used to create a treasure hunt game. Binary search, BFS and DFS were used to help players navigate their way to the treasure whilst avoiding traps and collecting powerups.

References

CelerData. (n.d.). *Breadt-First Search (BFS)*. https://celerdata.com/glossary/breadth-first-search-bfs

Khan Academy. (n.d.). *Computer Science Theory.* https://www.khanacademy.org/computing/computer-science/algorithms/binarysearch/a/binarysearch#:~:text=Binary%20search%20is%20an%20efficient,possible%20locations%20to%20just%20one.

Unacademy. (2024, August). *Difference Between BFS And DFS*. https://unacademy.com/content/gate-cse-it/difference-between-bfs-and-dfs/#:~:text=BFS%20(Breadth-first%20search),a%20node%20with%20no%20children.