

Assignment 2

AI2002-Artificial Intelligence

Name: Mirza Humayun Masood

Section :CY-A

*Submitted to: Sir Shoaib Saleem Khattak*

Department of Cyber Security BS(CY)

FAST-NUCES Islamabad

**Introduction**

In this assignment, We used search algorithms to find a path which completes a challenge within an enchanted forest. The forest is represented as a weighted graph, where Arin, the adventurer, must find a path through obstacles, portals, and varied terrains to reach a hidden treasure(The Goal).

In this we compare uninformed search methods (BFS, DFS,UCS) with informed searches (A\*, Greedy Best-First Search) to analyze efficiency in finding the optimal path. By evaluating their strengths and limitations, we gain insights into their effectiveness in solving real-world navigation problems.

**Experimentation Explained**

So first in this Assignment, we were given a grid, in which there were multiple nodes with multiple terrains, each having its own Marker and Cost. So for that i made a function GVN, which gets the neighbours with their respective costs, which was useful to construct the graph.So after that i applied BFS,DFS,UCS and GBFS,A\*, and saw their results.

So as the visualization function was given, I researched and used generative AI to Display the path on the graph, so let's see the Path generated by every Searching algorithm we studied.

**BFS**

These are the evaluation results of the BFS

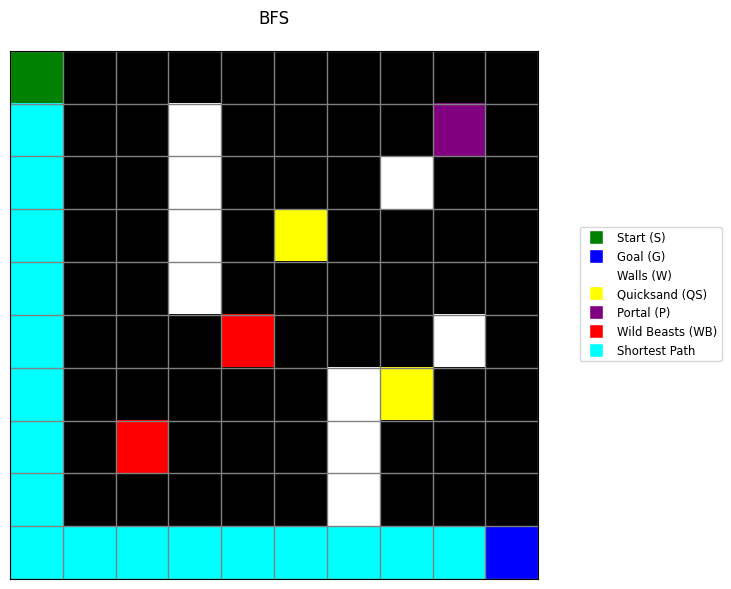
Path Cost: 16

Expanded Nodes: 90

Execution Time: 0.000313 seconds

Memory Used: 16.18 KB

Obstacles: [(1, 3), (2, 3), (2, 7), (3, 3), (4, 3), (5, 8), (6, 6), (7, 6), (8, 6)]



It is the path found by BFS.

**DFS**

Path Cost: 42

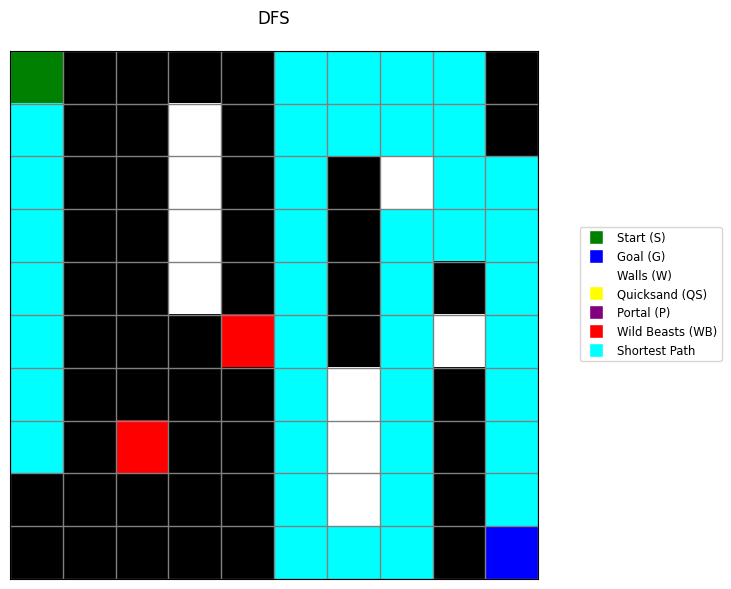
Expanded Nodes: 90

Execution Time: 0.000171 seconds

Memory Used: 15.33 KB

Obstacles: [(1, 3), (2, 3), (2, 7), (3, 3), (4, 3), (5, 8), (6, 6), (7, 6), (8, 6)]

As you can see the path is made by DFS, as it is too spread.

****

**UCS**

Path Cost: 14

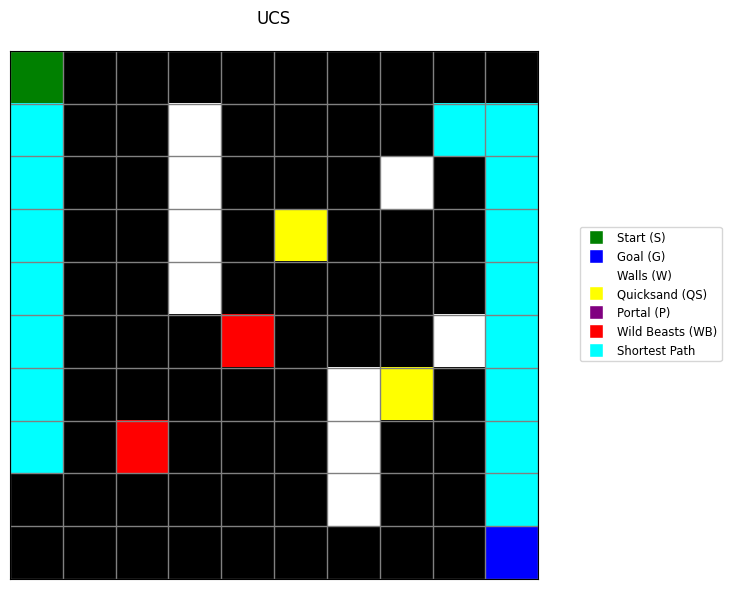
Expanded Nodes: 90

Execution Time: 0.000278 seconds

Memory Used: 12.53 KB

Obstacles: [(1, 3), (2, 3), (2, 7), (3, 3), (4, 3), (5, 8), (6, 6), (7, 6), (8, 6)]

The UCS was the first one to use the teleport, and cost 14.



**GBFS**

Path Cost: 17

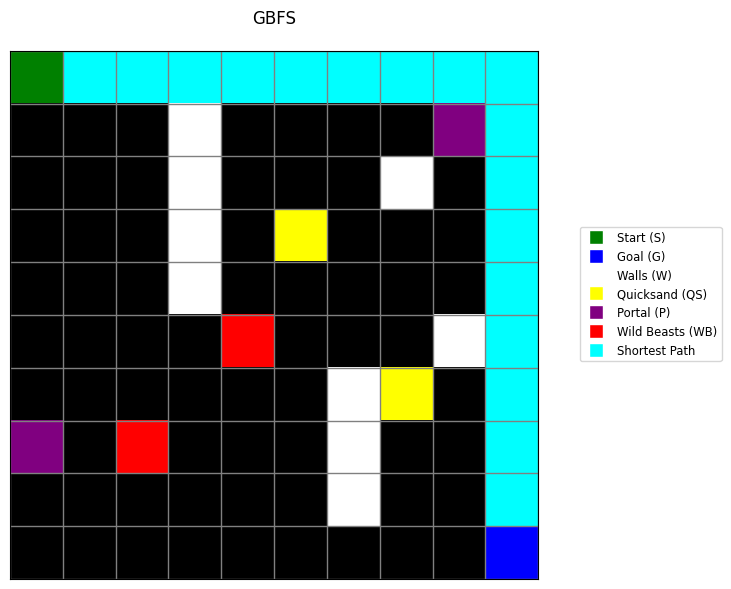
Expanded Nodes: 32

Execution Time: 0.000093 seconds

Memory Used: 2.52 KB

Obstacles: [(1, 3), (2, 3), (2, 7), (3, 3), (4, 3), (5, 8), (6, 6), (7, 6), (8, 6)]

GBFS used heuristic for its path finding, so I used Manhattan Distance. It was just on one side.



**A\***

Path Cost: 14

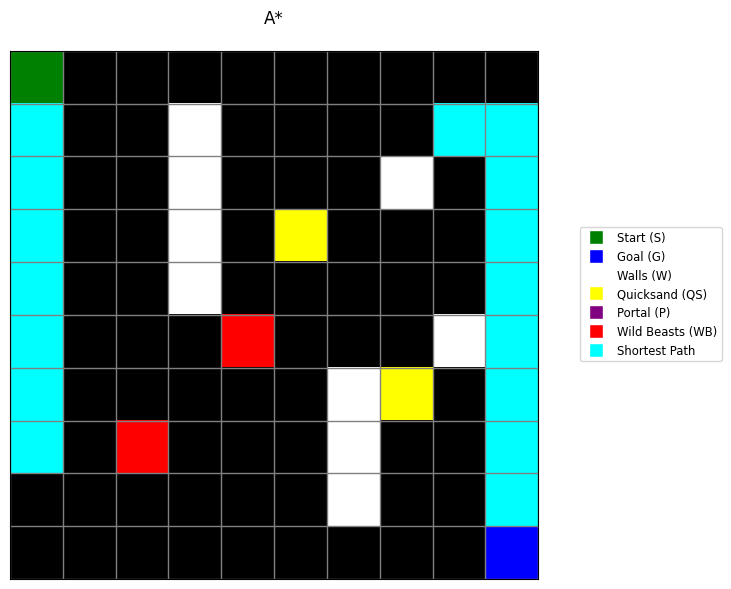
Expanded Nodes: 74

Execution Time: 0.000417 seconds

Memory Used: 9.17 KB

Obstacles: [(1, 3), (2, 3), (2, 7), (3, 3), (4, 3), (5, 8), (6, 6), (7, 6), (8, 6)]

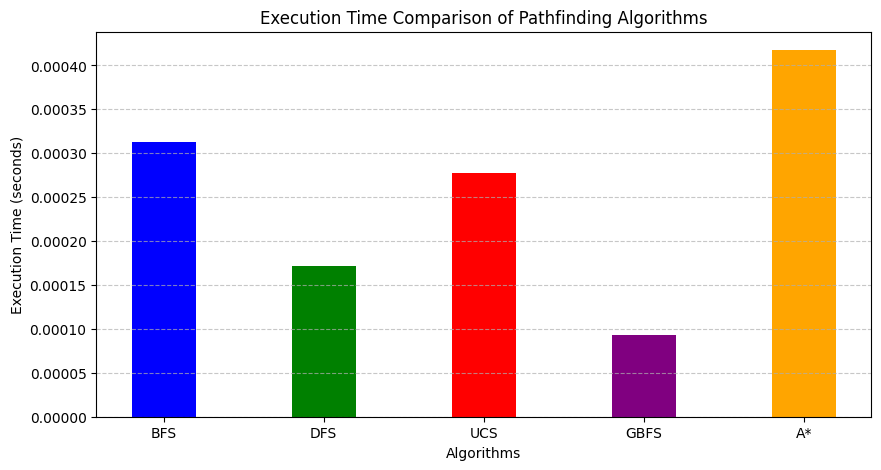
It gave the same result as UCS, so they both are alike, but it's more costly than UCS.



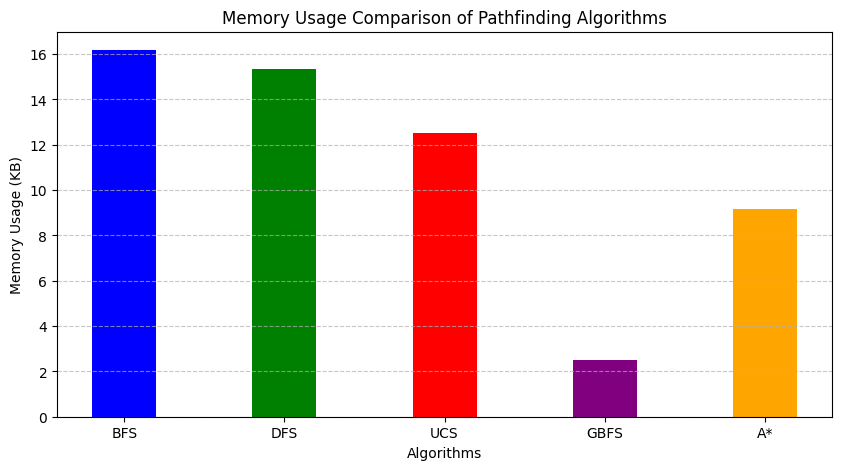
So let's compare Each attribute of every Searching Algorithm.

Upon used time.

As you can see A\* algorithm took the most time, and GBFS took the shortest time, as on second num its DFS, so we can see the difference, And in this GBFS is better, but it's in only this scenario, it could change according to the graph.

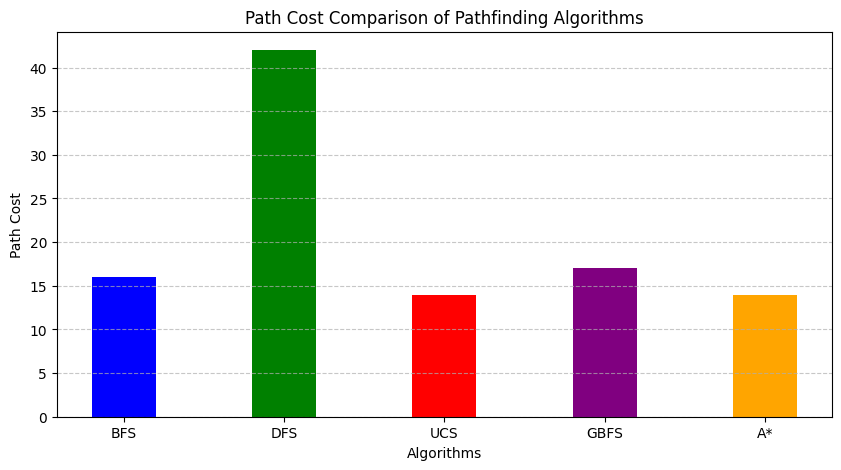


Memory Usage



As you can see, the GBFS used the lowest memory, and BFS used the most, and A\* used low memory than UCS.

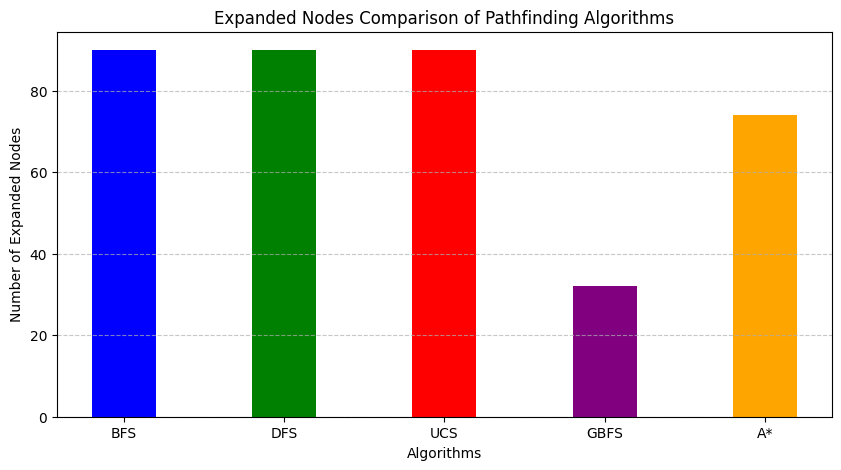
Path Cost



As you can see the path cost of UCS and A\* is equal and lowest, and then BFS, and then GBFS. DFS go with the highest cost.

Number of Nodes expanded

In this GBFS gets the lowest while DFS BFS and UCS remains the same. A\* is a bit smaller than UCS.



**Conclusion**

Among the given pathfinding algorithms, A\* and UCS (Uniform Cost Search) both achieve the optimal path cost of 14, meaning they find the shortest path while considering terrain costs. However, A\* is more efficient, expanding fewer nodes (74 vs. 90) and executing faster (0.000417s vs. 0.000278s) while using less memory (9.17 KB vs. 12.53 KB).

GBFS (Greedy Best-First Search) is the fastest and uses the least memory, but it results in a higher path cost (17), making it less optimal.

DFS (Depth-First Search) performs poorly, with a path cost of 42, expanding many nodes inefficiently.

Thus, A\* is the best choice, balancing optimal path cost, fewer expanded nodes, and reasonable execution time and memory usage.