Project 1: Search Algorithms

Alex Totah

CS170: Introduction to Artificial Intelligence

April 20, 2025

Contents

1 Introduction

This report analyzes four different search algorithms (BFS, DFS, UCS, A*) and their applications to path-finding. We take a visual approach to analyzing the outputs of these algorithms and demonstrate how their underlying mechanisms affect search efficiency and solution quality. By comparing their performance across different metrics, we provide insights into which algorithms are best suited for specific problems. Additionally, we explore practical implementations and optimizations that can significantly improve performance in real-world applications.

1.1 Background

Breadth first search (BFS): A search algorithm that expands the shallowest nodes first.

- Implemented using a queue using a first in first out (FIFO) structure (queue).
- A complete search that finds the least amount of steps.
- Let d be the depth of the solution and b be the number of nodes in each layer. The time complexity of the search is $O(b^d)$
- When the search algorithm reaches depth d, the space complexity will be $O(b^d)$ since it will need to expand and store all the expandable nodes in the open list
- Will find the optimal solution if there is uniform cost for each action.

Depth First Search (DFS): A search algorithm that expands the deepest nodes first.

- Implemented using a queue using a last in first out (LIFO) structure (stack).

2 Results (20 points)

This section presents the results of the four pathfinding algorithms on the default map and two randomly generated maps using the GUI.

3 Depth First Search (DFS)

3.0.1 Default Map

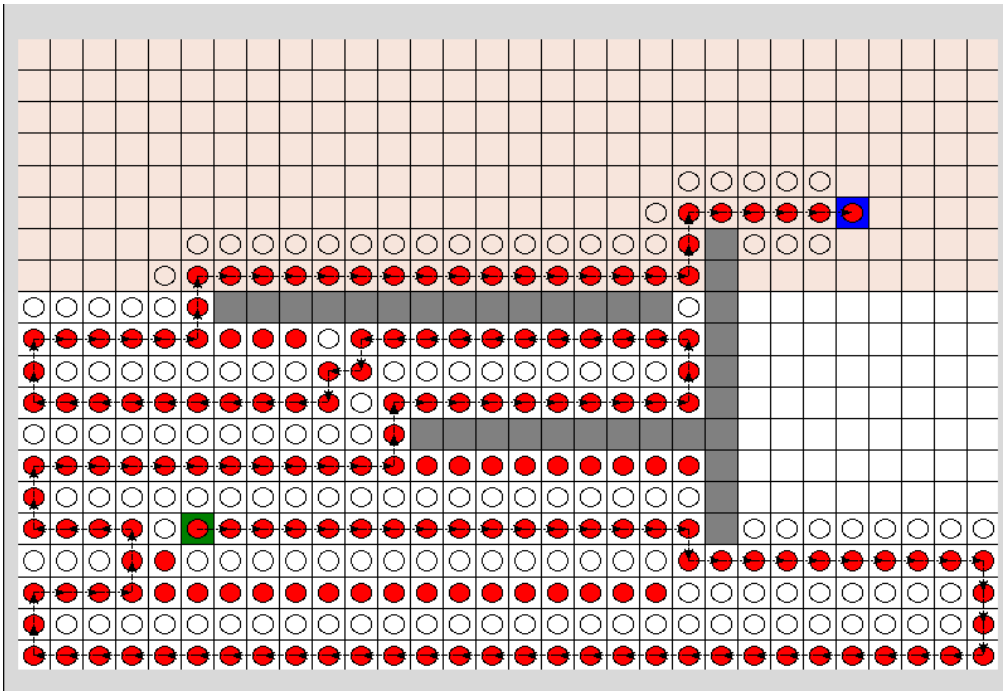

Figure 1: DFS on Default map. Total path cost: 192

Observations:

- The algorithm searches outward nodes first.
- It did not find the optimal path
- Searched only one path (Can have low memory usage)

3.0.2 Random 1

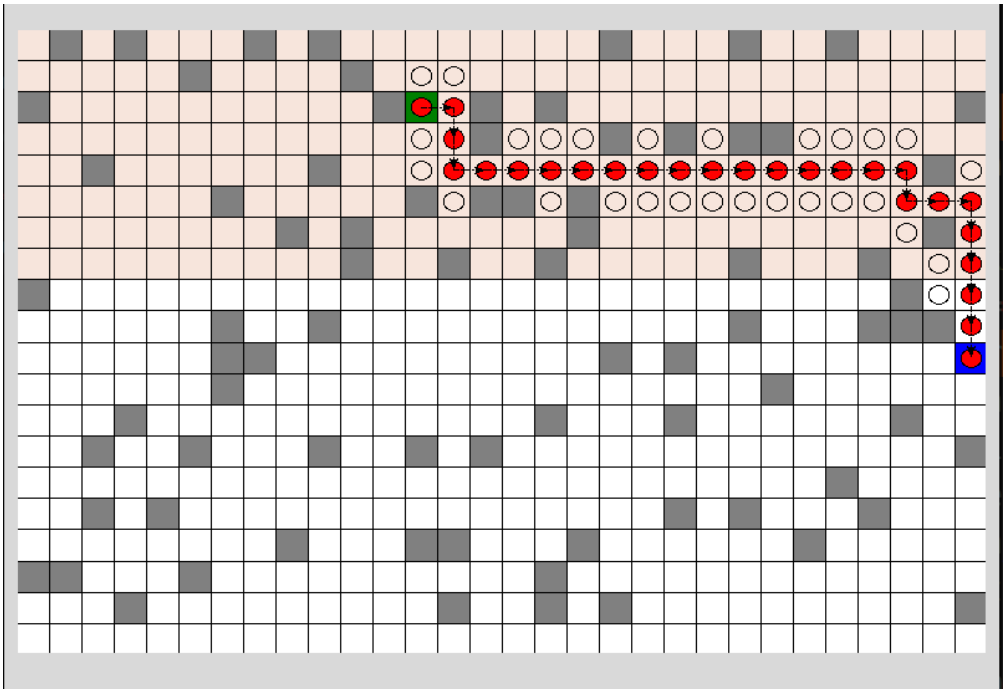

Figure 2: DFS on Random 1 map. Total path cost: 69

Observations:

- Found the path very easily by chance.
- Very low memory usage
- Searched only one path

3.0.3 Random 2

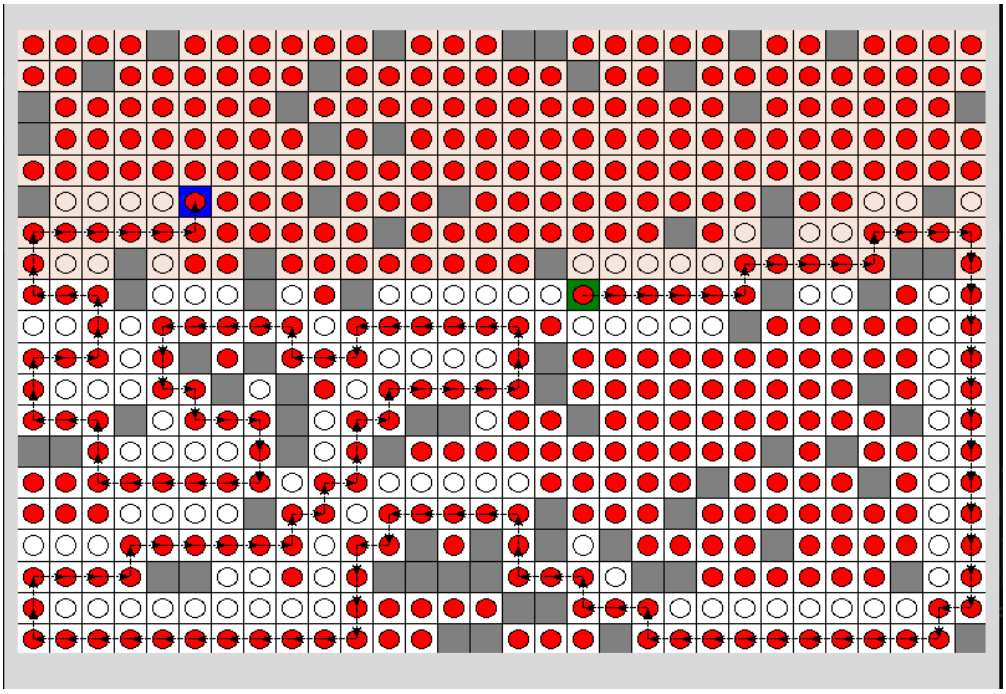

Figure 3: DFS on Random 2 map. Total path cost: 171

Observations:

- An instance where DFS not only doesn't find the shortest path
- Almost uses the entire grid in memory

4 Breadth First Search (BFS)

4.1 Default

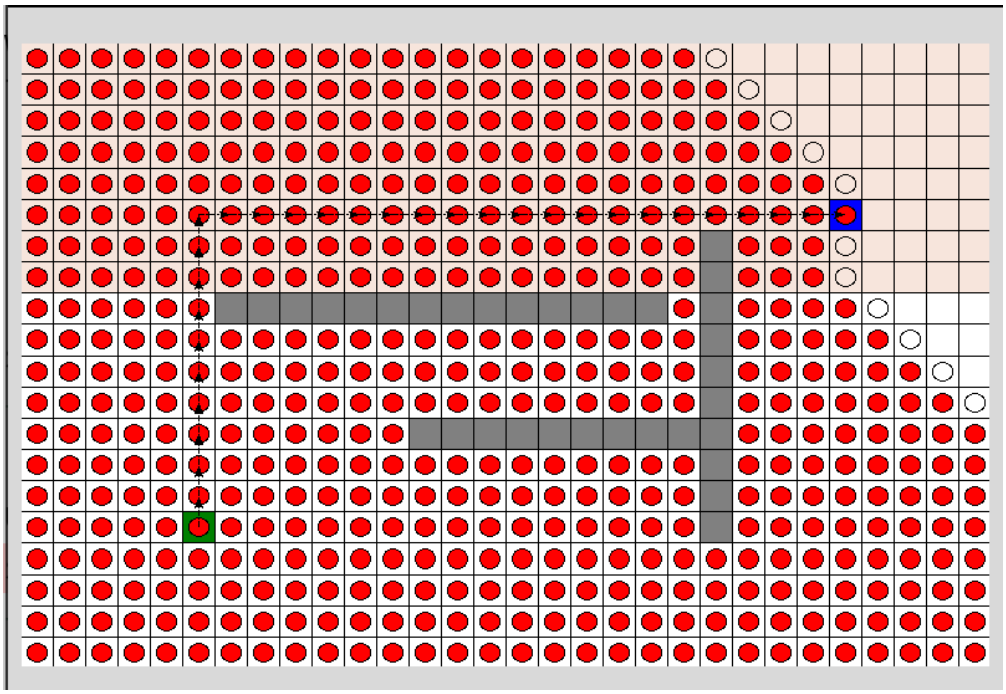

Figure 4: BFS on Default map. Total path cost: 76

Observations:

- Uses a significant amount of memory
- Finds an optimal path much.
- Searches the outer most nodes, but not necessarily closest to the goal

4.2 Random 1

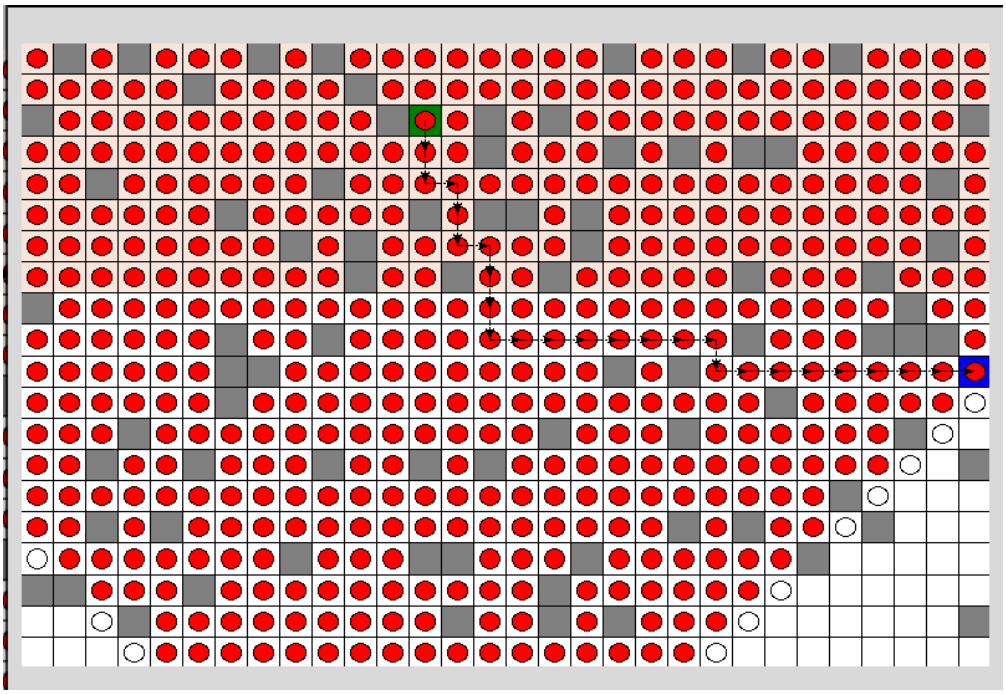

Figure 5: BFS on Random 1 map. Total path cost: 27

4.3 Random 2

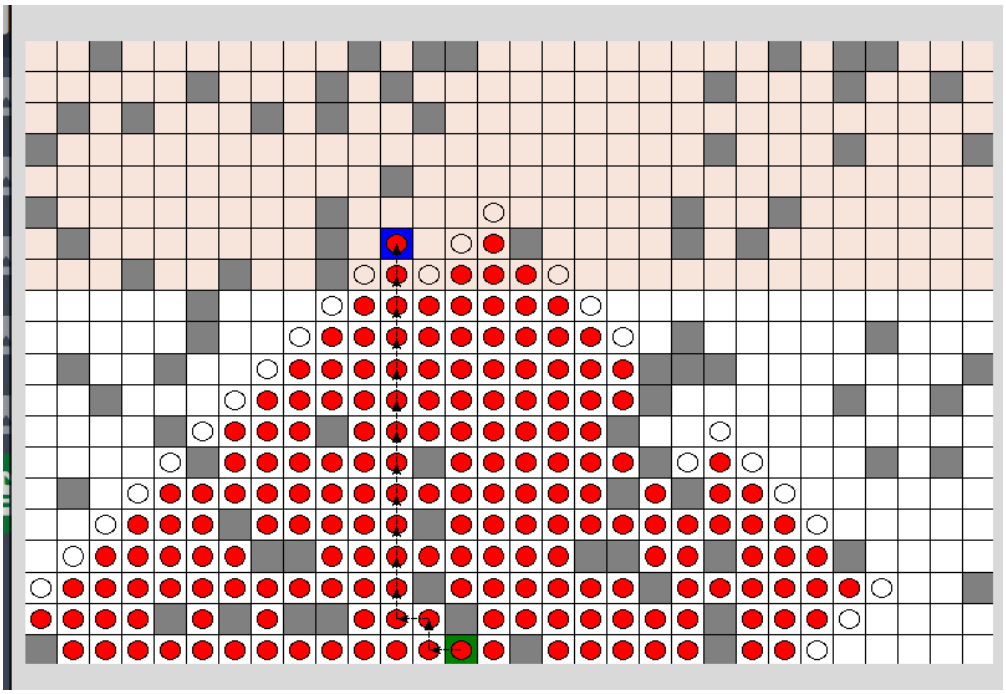

Figure 6: BFS on Random 2 map. Total path cost: 19

5 Uniform Cost Search (UCS)

5.0.1 Default

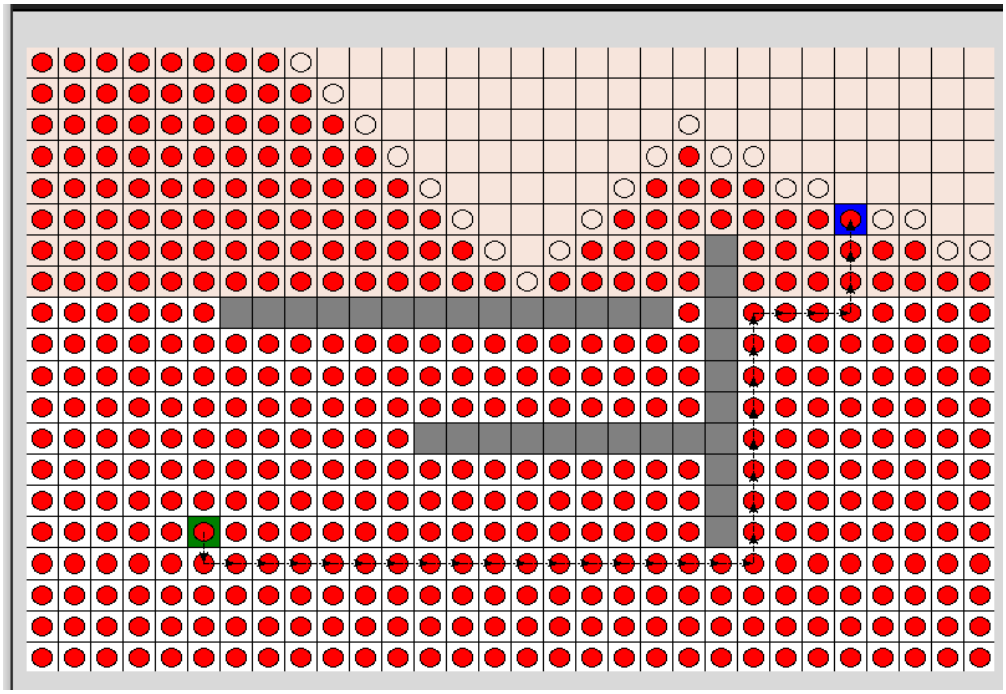

Figure 7: UCS on Default map. Total path cost: 38

5.1 Random 1

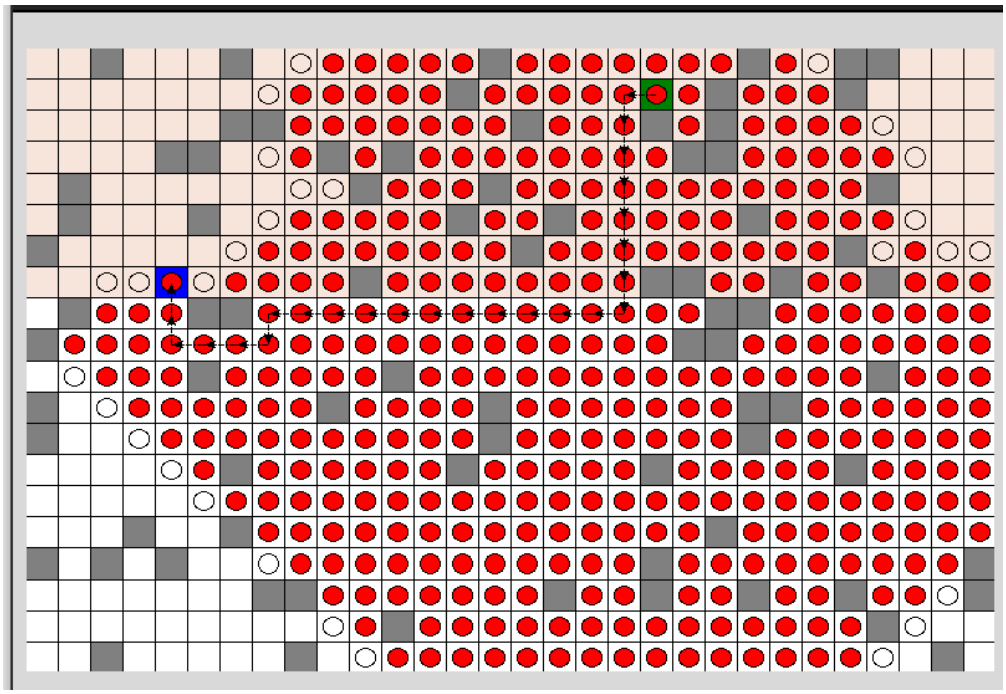

Figure 8: UCS on Random 1 map. Total path cost: 41

5.2 Random 2

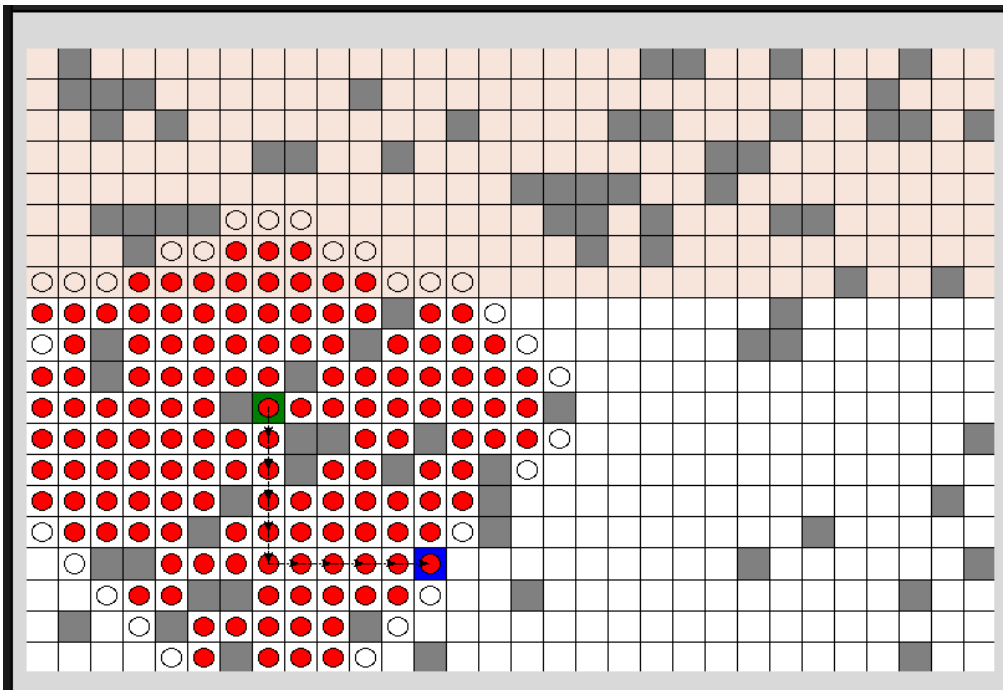

Figure 9: UCS on Random 2 map. Total path cost: 10

6 A-Star Search (A*)

6.0.1 Default

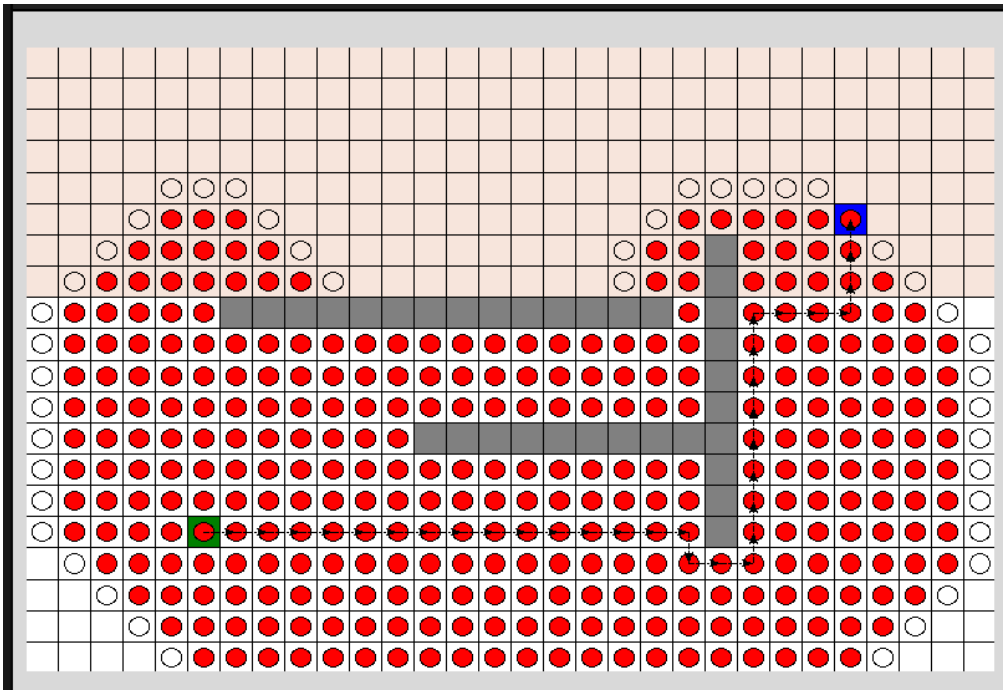

Figure 10: A* on Default map. Total path cost: 38

6.1 Random 1

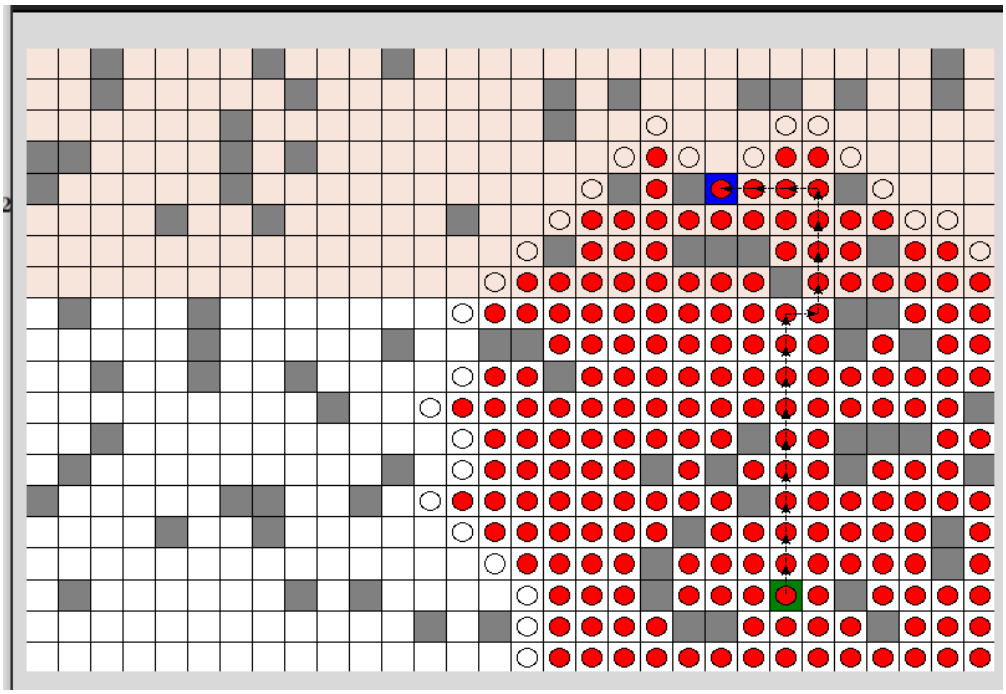

Figure 11: A* on Random 1 map. Total path cost: 31

6.2 Random 2

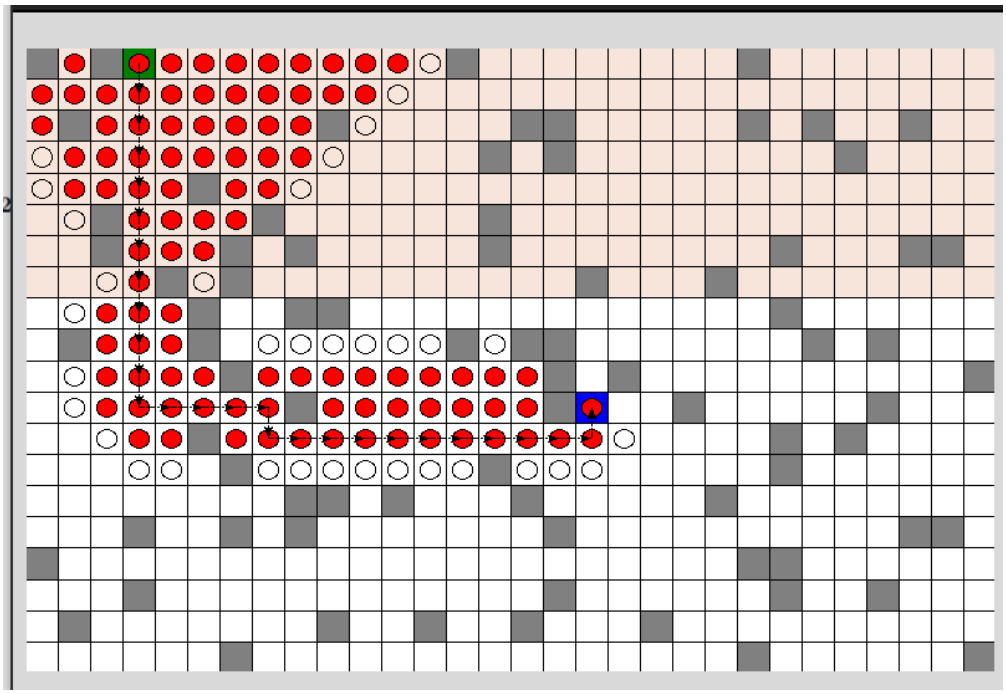

Figure 12: A* on Random 2 map. Total path cost: 41

7 Comparative Analysis

7.1 Data from experiments

Table 1: Summary of Path Costs for All Algorithms

Algorithm	**Default Map**	**Random Map 1**	**Random Map 2**
BFS	76	27	19
DFS	192	69	171
UCS	38	41	10
A*	38	31	41

7.2 A* on Uniform Cost Grid

A* search optimizes for the lowest total path cost using $f(n) = g(n) + h(n)$. On the default map, $g(n)$ accumulates potentially variable costs, leading A* to find the path minimizing this specific sum. On a uniform cost grid, $g(n)$ simply counts steps, so A* finds the path with the minimum number of steps.

7.3 BFS vs. UCS on Uniform Cost Grid

On a uniform cost grid, the paths found by BFS and Uniform Cost Search (UCS) will have the same cost. BFS finds the path with the fewest steps, while UCS finds the path with the lowest cumulative cost. Therefore, minimizing the number of steps (BFS's objective) is equivalent to minimizing the total cost (UCS's objective) in this scenario, resulting in paths with identical optimal costs.