**Project 1 Path Finding**

David Bush

CS 470 Soule

May 24th, 2021

**Abstract:** For this project we are comparing different search algorithms on a map with varying movement costs for each cell. I implemented the bread-first, the least cost, the A\*, and the greedy best first search algorithms in Python to find a solution path. The results showed that the bread-first search found the shortest length of 27 and the least cost search found the lowest total cost path of 58. However, the greedy best first algorithm also found the same exact path as the bread-first. The A\* found a path with cost of 61 and length 29 and had a similar route to the least cost search.

**Introduction:** Pathfinding is a type of problem that uses search algorithms to find a path between two different points usually on some sort of grid or map. There can either be informed searches where the agent knows how close it is to the goal or uninformed searches where the agent does not have that knowledge. Depending on the search algorithm, it can also determine the shortest path (in actions) or the lowest cost path to find the desired goal. Pathfinding is important because its concepts can be applied to a variety of different problems. A couple examples could include finding the shortest path in a maze, finding the shortest distance for a map navigation system, or finding the fastest route for a race game. For my program, I was able to determine the lowest cost path using the least cost search algorithm and the shortest length path using the bread-first search. The greedy best first search also found the shortest length path and ended up having the same exact path as the bread-first. The A\* search had similar route to the least cost search but had a slightly higher cost.

**Algorithm and Heuristics Description:** In this project we explored four unique search algorithms. The first algorithm we implemented was a bread-first search which is an uninformed search that can find the shortest length path. The Bread-first algorithm works by exploring neighbor cells in all directions checking one level of the tree at a time. The first step is to place the neighbor cells that are not on the closed list onto an open list that acts as a queue. Then we remove the first item from the open list and check to see if it is the desired goal, stopping if true. Next, we add that cell to the closed list and repeat the previous steps until we eventually find the goal. For the second algorithm we used a least cost search which finds the lowest total cost path and is also an uninformed search. The least cost algorithm has a similar structure as the bread-first algorithm with how it explores neighbor cells, but the main difference is that it sorts the open list by the total cost it took to reach that cell before removing an item from the open list. This causes the least cost search to first check cells that have the lowest total cost path and ensures that it does not miss the lowest cost path. The third algorithm that we tested was the A\* search which is an informed search that uses some heuristic to find a possible lowest cost path. The heuristic is just an estimated cost to reach the goal at any given point, it can change the closer it gets to the goal. The heuristic I implemented was simply: H=max(distance1,distance2). The distances were calculated using the formulas: D1=|x-x’| + |y-y’| , D2=sqrt( (x-x’)^2 + (y-y’)^2 ), which takes the x, y of the goal cell and the x’, y’ of the current cell as variables. The A\* search is similar to the least cost search, but it sorts the open list by the heuristic value plus the total cost instead of just the total cost. If the heuristic doesn’t overestimate the cost to the goal, the A\* search should find a lowest cost path, but it can be hard to implement correctly. Finally, the last algorithm examined was the greedy best first which is an informed search close to the A\* search but it is not usually optimal. The greedy best first algorithm is related to the previous algorithms with how it explores the cells but when it sorts the open list it only looks at the heuristic instead of the heuristic plus total cost. The interesting part about the greedy best first search is that it can be tricked into taking a very long and expensive path because it goes only based on how close it believes it is to the goal. As we can see all these algorithms have similar steps, but the key difference is how each algorithm sorts the open list.

**Results:** Below there are tables and figures that contain the results from running each of the four search algorithms. The first table shows the length, cost, and open list count for each search. After that there is the base map with the dimensions, a start position and a goal position that was used for each test with a figure also showing how the costs were determined for each terrain. Then there are individual sections for each search algorithm that show the open list, the path found map, and the explored cells map demonstrating how the algorithms were searching.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Pathfinding Algorithms | | | | |
|  | **Breadth-First** | **Least Cost** | **A\*** | **Greedy** |
| Length | 27 | 31 | 29 | 27 |
| Cost | 70 | 58 | 61 | 70 |
| Open List Count  (at end) | 7 | 16 | 40 | 42 |

**Basic Setup:**

\*\*\* Start Position: (7, 0) ; Goal Position: (7, 18) ; Size: 15 x 20 \*\*\*

A picture containing background pattern

Description automatically generated

Base Map (used for each test)

|  |  |  |
| --- | --- | --- |
| Character | Meaning | Movement Cost |
| R | road | 1 |
| f | field | 2 |
| F | forest | 4 |
| h | hills | 5 |
| r | river | 7 |
| M | mountains | 10 |
| W | water | can't be entered |

**Bread-First Search:**

Graphical user interface, text

Description automatically generated with medium confidence

Open List Bread-First

A picture containing text

Description automatically generatedA picture containing text

Description automatically generated

Path Found Bread-First Explored Cells Bread-First

**Least Cost Search:**



Open List Least Cost

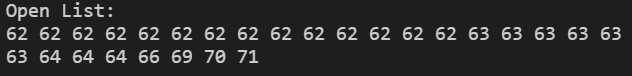
A picture containing text

Description automatically generatedText

Description automatically generated with medium confidence

Path Found Least Cost Explored Cells Least Cost

**A\* Search:**



Open List A\*

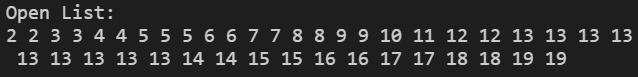
A picture containing calendar

Description automatically generatedA picture containing calendar

Description automatically generated

Path Found A\* Explored Cells A\*

**Greedy Best First Search:**



Open List Greedy

Text

Description automatically generated with medium confidenceA picture containing text

Description automatically generated

Found Path Greedy Explored Cells Greedy

**Conclusion/Comparison:** After completing tests on each algorithm, I was able to gather results for the length, total cost, path found, and explored cells. First if we examine the different lengths found by each search algorithm, its somewhat surprising that the bread-first and greedy best first both found the shortest length path of 27. This is mainly a coincidence and is due to the arrangement of this particular map. I would strongly expect that if given more maps to test, the length would end up being different for these algorithms. Now if we consider the various costs found, we see that the least cost search indeed has the lowest cost path at 58. The A\* search is close at a cost of 61 and both the bread-first and greedy found a cost of 70. The reason that the A\* search is not at the lowest cost is because the heuristic most likely is not a 100% optimized and needs to be changed slightly. Next, examining the solution paths we see each algorithm derived a slightly different route. This is because each algorithm has a unique approach to how it finds the goal. The least cost search took a somewhat longer route due to its nature to search the lowest cost paths. The A\* search was almost identical to the least cost’s path, but it was slightly different because of the heuristic. The bread-first and greedy best first searches had the same exact solution path which was somewhat unusual since both have very different approaches. If we observe the explored cells for each algorithm, we can see a clear difference between the uninformed and informed searches. The uniformed searches both explored almost every cell before finding the goal and the informed searches were quite the opposite exploring less cells and mainly in the direction of the goal. This shows the key difference between uninformed and informed searches being the number of cells explored and their direction. After comparing all the algorithms its not exactly clear which one is the best. Instead, we must consider the exact problem we are trying to solve and determine which algorithm would be the most suited. The informed searches have an advantage at using less memory and not having to explore as many cells. The least cost search is easy to implement and always find the lowest cost path. The breadth-first can be implemented simply and always find the shortest length path. To conclude, each algorithm that was tested has its own advantages and disadvantages and a unique method of searching.