NANYANG TECHNOLOGICAL UNIVERSITY

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

Lab 1 Assignment: Search
CZ3005: Artificial Intelligence

Lab Group: SSP5

Group Name: Pikacent

Team Members:

Chai Wen Xuan[U2020350F]

2021/2022 Semester 2

1. Task
    1.1. Task 1
        1.1.1. Depth-First Search
            1.1.1.1. Result

Number of visited nodes: 20599

Execution Time: 0.02014780044555664s

Number of nodes in solution path: 5220

Shortest distance: 9038132.18922319

Total energy cost: 19402484

1.1.1.2. Explanation

In this Depth First Search(DFS) approach, “visited” set is used to avoid looping. Hence, completeness is achieved as it has finite search depth.

DFS uses stack(LIFO), hence it will deep dive to its first neighbour which is not visited until it finds the goal nodes.

Hence, DFS is not optimal. We can also notice that it has the highest number of visited nodes and number of nodes in solution path.

1.1.2. Breadth-First Search

1.1.2.1. Result

Number of visited nodes: 7790

Execution Time: 0.022806882858276367s

Number of nodes in solution path: 86

Shortest distance: 250857.31843462065

Total energy cost: 471515

1.1.2.2. Explanation

In this Breadth First Search(BFS) approach, a “visited” set is used to avoid looping.

BFS uses queue(FIFO), hence it will travel the nodes that visited earlier, layer by layer.

BFS, in this case, is also not optimal as all the distance/costs from nodes to nodes are not equal. However, you can notice that the number of nodes in the solution path in BFS is the smallest among all of the methods used.

1.1.3. Uniform-Cost Search (Distance)

1.1.3.1. Result

Number of visited nodes: 5647

Execution Time: 0.022510766983032227

Number of nodes in solution path: 121

Shortest distance: 148648.63722140013

Total energy cost: 294853

1.1.3.2. Explanation

In Uniform-Cost Search, the disadvantages of BFS is solved. In UCS, a priority queue is used to keep track of the smallest cost used to reach a node. It will prioritise to travel the node and hence, optimality is achieved.

In Task 1 we prioritise distance instead of cost.

1.2. Task 2

1.2.1. Uniform-Cost Search (Cost)

1.2.1.1. Result

Number of visited nodes: 5580

Execution Time: 0.022451877593994l4s

Number of nodes in solution path: 118

Shortest distance: 180253.51089652776

Total energy cost: 225102

1.2.1.2. Explanation

The solution met the energy budget which is 287932.

In Task 2, we prioritise cost to met the energy budget.

1.3. Task 3

1.3.1. A Star

1.3.1.1. Result

Number of visited nodes: 5720

Execution Time: 0.029345989227294922s

Number of nodes in solution path: 118

Shortest distance: 180253.51089652776

Total energy cost: 225102

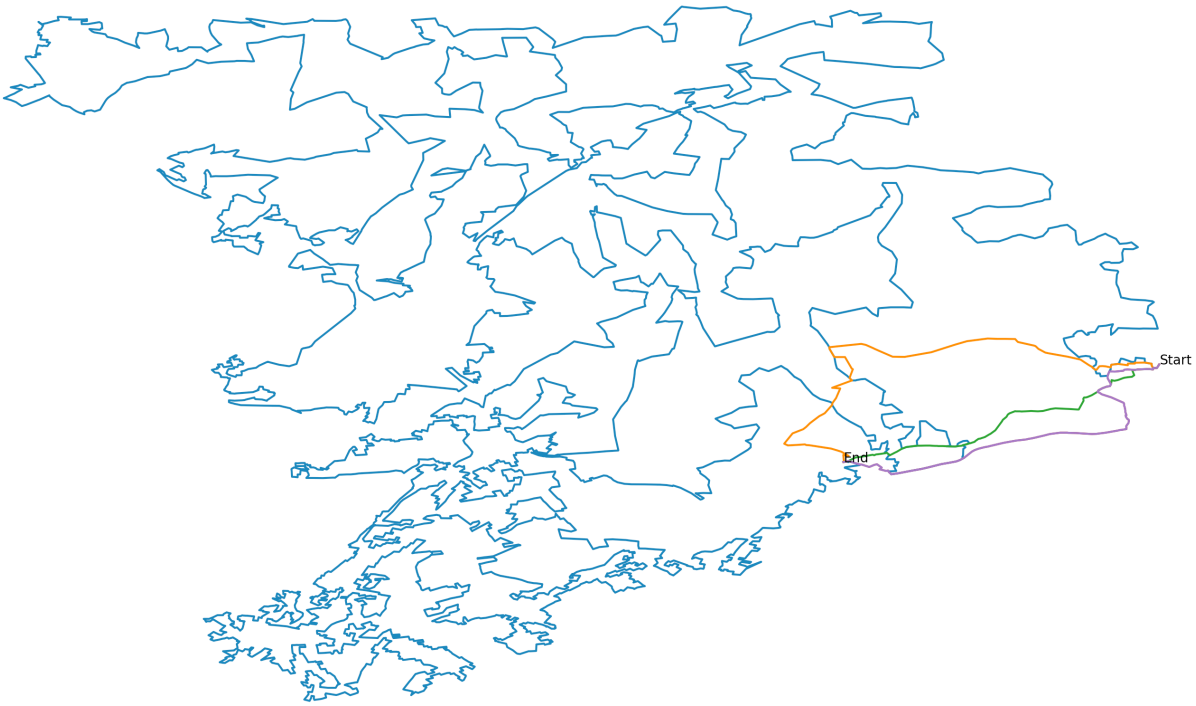
1.3.1.2. Explanation

The solution met the energy budget which is 287932.

In this A Star solution, an evaluation function with the sum of Depth[n] and H[n] is used, where Depth[n] is the cost to reach n and H[n] is the estimated cost to reach the goal node. The estimation is done by calculating the geographical distance between any nodes and goal node.

The algorithm will priorities smallest cost of Depth[n] + H[n] to travel first.

1.4. Visualization



Solution by different algorithms is plotted out for better visualization. The start and end nodes are also labelled out.

DFS: Blue, BFS: Orange, UCS(Distance): Purple , UCS(Cost)/A Star: Green

2. Contribution

All done by Chai Wen Xuan.

3. Conclusion

From DFS, BFS, UCS to A star. We can see the improvement in the algorithm by implementing different data structures and strategies, from stack, queue to priority queue.

We can see that UCS and A star has smallest number of nodes visited, comparing to DFS and BFS. Hence, they have better time complexity.

However, UCS and A star implements priority queue, making it spending more time complexity when inserting nodes into priority queue which cost O(log n). Besides, Depth[n], H[n] dictionary is also used, resulting in larger space complexity. Two of this factors increase the time and space complexity, making four algorithms showing similar execution time.

Hence, we can conclude that UCS and A Star is the most suitable algorithm to solve the shortest path problem. Considering their completeness, time complexity, space complexity and optimality.

However, we need to keep in mind that A star is not always optimal if our estimation for the heuristic function is not admissible and monotonic.

4. Reference

4.1. Python library, json — JSON encoder and decoder
   4.1.1. Usage: To load json file and store json object into python dictionary
   4.1.2. Link: https://docs.python.org/3/library/json.html
4.2. Python library, matplotlib.pyplot.plot
   4.2.1. Usage: To plot out coordinate of solution path for better visualization
   4.2.2. Link: https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.plot.html
4.3. Python library, queue — A synchronized queue class
   4.3.1. Usage: Use queue to implement BFS and priority queue to implement UCS
   4.3.2. Link: https://docs.python.org/3/library/queue.html
4.4. Python library, time — Time access and conversions
   4.4.1. Usage: To record execution time of each algorithm
   4.4.2. Link: https://docs.python.org/3/library/time.html
4.5. Python library, math — Mathematical functions
   4.5.1. Usage: Using sqrt() function to calculate distance between nodes
   4.5.2. Link: https://docs.python.org/3/library/math.html