CZ3005 Artificial Intelligence Assignment 1 Final Report

Group Member	Matric Number	Contribution
Tan Yue Jun	U2021253G	Equal
Dion Toh Siyong	U2021674D	Equal
Hariharan Dhruv	U2023933G	Equal

1. Task 1

1.1 Approach

Task 1 requires us to find the shortest path from the starting node (1) to the terminal node (50). Since there are no resource constraints for this task, the problem is effectively reduced to a shortest-path problem.

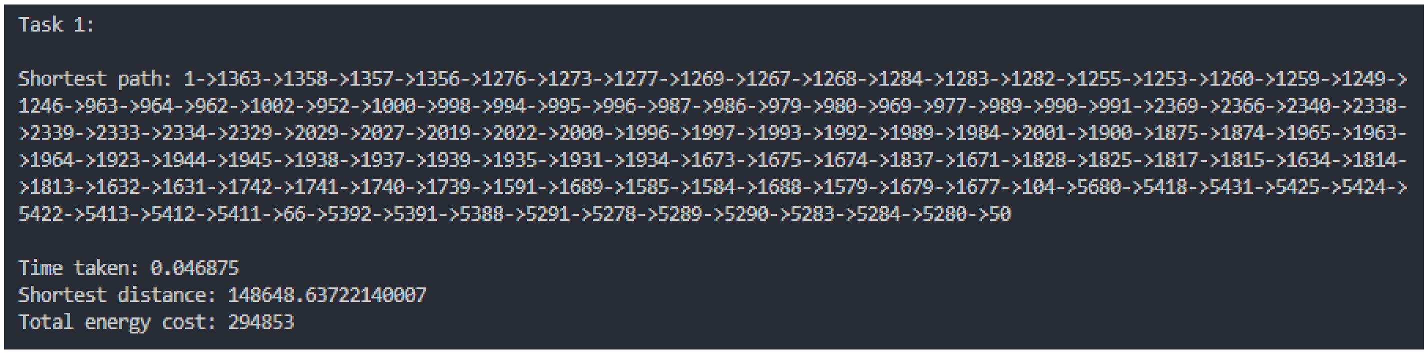
A shortest-path problem is best solved through algorithms like Dijkstra or Uniform-Cost Search (UCS), as it is complete and therefore it is optimal (ScienceDirect Topics). We implemented it by using a set to keep track of visited nodes, as well as a priority queue to help decide which node to expand to next.

Algorithm Approach:

1. Insert the starting node into the priority queue
2. Repeat while the priority queue is not empty:
    a. Remove the node with the lowest total distance
    b. If the removed node is in the visited set, we continue to the next loop
    c. If the removed node is the terminal node, calculate and print the total distance as well as the path. The algorithm ends here
    d. Otherwise, enqueue all the neighbouring nodes of the current node to the priority queue, with their cumulative distance from the starting node

1.2 Output

Uniform Cost Search	
Shortest Distance	148648.63722140007
Total Energy Cost	294853
Time Taken	0.046875



```
Task 1:

Shortest path: 1->1363->1358->1357->1356->1276->1273->1277->1269->1267->1268->1284->1283->1282->1255->1253->1260->1259->1249->
1246->963->964->962->1002->952->1000->998->994->995->996->987->986->979->980->969->977->989->990->991->2369->2366->2340->2338-
>2339->2333->2334->2329->2029->2027->2019->2022->2000->1996->1997->1993->1992->1989->1984->2001->1900->1875->1874->1965->1963-
>1964->1923->1944->1945->1938->1937->1939->1935->1931->1934->1673->1675->1674->1837->1671->1828->1825->1817->1815->1634->1814-
>1813->1632->1631->1742->1741->1740->1739->1591->1689->1585->1584->1688->1579->1679->1677->104->5680->5418->5431->5425->5424->
5422->5413->5412->5411->66->5392->5391->5388->5291->5278->5289->5290->5283->5284->5280->50

Time taken: 0.046875
Shortest distance: 148648.63722140007
Total energy cost: 294853
```

2. Task 2

2.1 Approach

Task 2 requires us to find the most optimal path, shortest if possible, with provided resource constraints (Distance & Energy Costs).

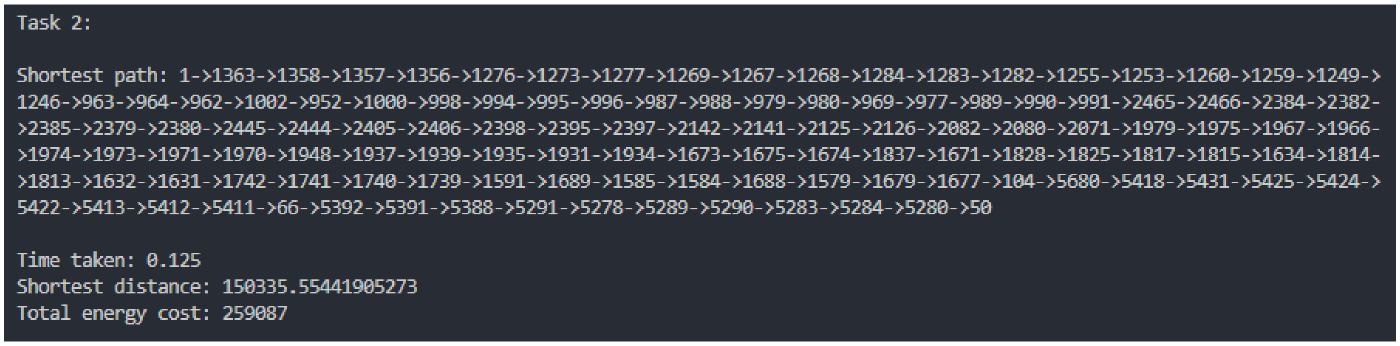
This means that we should not maintain a record of visited nodes, as we will need to revisit these nodes to search for a more cost-effective path. Instead we have 2 hashmaps, distHash and energyHash to keep track of the respective lowest cumulative values for that node from the starting node.

Algorithm Approach:

1. Insert the starting node into the priority queue
2. Repeat while the priority queue is not empty:
    a. Remove the node with the lowest total distance
    b. If the removed node is the terminal node, calculate and print the total distance, total energy as well as the path. The algorithm ends here
    c. Otherwise, we check for all of the neighbouring nodes to see if the total energy required to get to that neighbouring node from starting node exceeds our energy budget
        i. If the total energy required exceeds, we move on to the next neighbouring node
    d. If the neighbouring node's total distance or total energy from starting node is less than the value in distHash or energyHash (default value = MAXINT), we update the corresponding hashmaps, and enqueue it back to the priority queue, with their cumulative distance from the starting node

2.2 Output

Uniform Cost Search	
Shortest Distance	150335.55441905273
Total Energy Cost	259087
Time Taken	0.140625



```
Task 2:

Shortest path: 1->1363->1358->1357->1356->1276->1273->1277->1269->1267->1268->1284->1283->1282->1255->1253->1260->1259->1249->
1246->963->964->962->1002->952->1000->998->994->995->996->987->988->979->980->969->977->989->990->991->2465->2466->2384->2382-
>2385->2379->2380->2445->2444->2405->2406->2398->2395->2397->2142->2141->2125->2126->2082->2080->2071->1979->1975->1967->1966-
>1974->1973->1971->1970->1948->1937->1939->1935->1931->1934->1673->1675->1674->1837->1671->1828->1825->1817->1815->1634->1814-
>1813->1632->1631->1742->1741->1740->1739->1591->1689->1585->1584->1688->1579->1679->1677->104->5680->5418->5431->5425->5424->
5422->5413->5412->5411->66->5392->5391->5388->5291->5278->5289->5290->5283->5284->5280->50

Time taken: 0.125
Shortest distance: 150335.55441905273
Total energy cost: 259087
```

3. Task 3

3.1 Approach

Since our implementation of an uninformed search algorithm for Task 2 was based on UCS, our implementation for this task would largely be the same, differing only by the new heuristic function given A* search is an informed search algorithm (UCS is a special case of A* search, where h(n) = 0). The evaluation function of A* is f(n) = g(n) + h(n). The implementation of h(n) is explained in section 3.2.

Algorithm Approach:

1. Insert the starting node into the priority queue
2. Repeat while the priority queue is not empty:
    a. Remove the node with the lowest score
    b. If the removed node is the terminal node, calculate and print the total distance, total energy as well as the path. The algorithm ends here
    c. Otherwise, we check for all of the neighbouring nodes to see if the total energy required to get to that neighbouring node from starting node exceeds our energy budget
        i. If the total energy required exceeds, we move on to the next neighbouring node
    d. If the neighbouring node's total distance or total energy from starting node is less than the value in distHash or energyHash (default value = MAXINT), we update the corresponding hashmaps, calculate the new score for the neighbouring node and enqueue it back to the priority queue, with the calculated score

3.2 Heuristics Function

For our A* search's heuristic, we needed to implement the greedy algorithmic paradigm. This means that we want to choose the next path that offers the closest and immediate benefit. Translating that to the NYC problem, it means that our heuristic is estimated by the Euclidean distance between the current node and the end node. In other words, we wish to find the shortest distance between the two nodes.

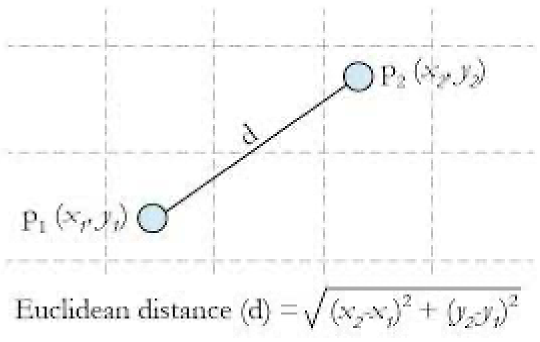
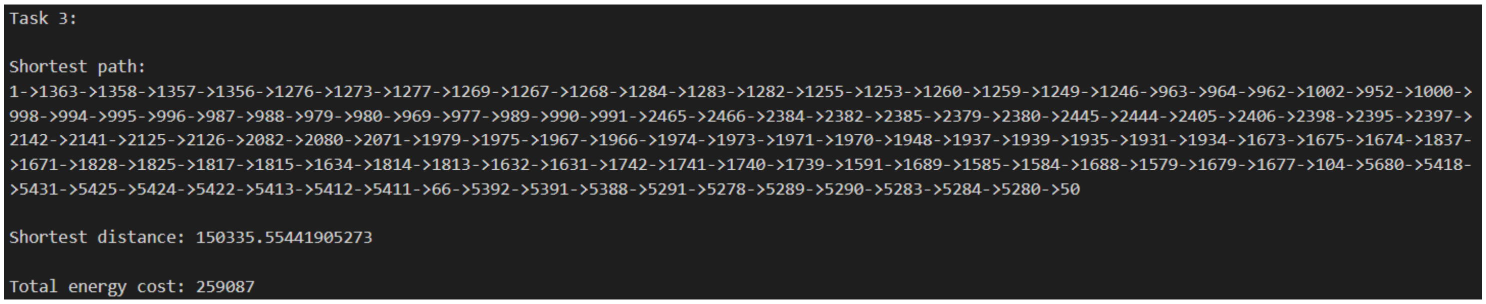


Figure 1: (SCIENTIFIQUE, C. A. F.É., 2016) Euclidean Distance

$$h(n) = \sqrt{(curNode.x - endNode.x)^2 + (curNode.y - endNode.y)^2}$$

3.3 Output

Uniform Cost Search	
Shortest Distance	150335.55441905273
Total Energy Cost	259087
Time Taken	0.03125



```
Task 3:

Shortest path:
1->1363->1358->1357->1356->1276->1273->1277->1269->1267->1268->1284->1283->1282->1255->1253->1260->1259->1249->1246->963->964->962->1002->952->1000->
998->994->995->996->987->988->979->980->969->977->989->990->991->2465->2466->2384->2382->2385->2379->2380->2445->2444->2405->2406->2398->2395->2397->
2142->2141->2125->2126->2082->2080->2071->1979->1975->1967->1966->1974->1973->1971->1970->1948->1937->1939->1935->1931->1934->1673->1675->1674->1837-
>1671->1828->1825->1817->1815->1634->1814->1813->1632->1631->1742->1741->1740->1739->1591->1689->1585->1584->1688->1579->1679->1677->104->5680->5418-
>5431->5425->5424->5422->5413->5412->5411->66->5392->5391->5388->5291->5278->5289->5290->5283->5284->5280->50

Shortest distance: 150335.55441905273

Total energy cost: 259087
```

4. Conclusion

Through this assignment, we came to the conclusion that there is no "best" algorithm when it comes to a search problem. The "best" algorithm that one can use is heavily dependent on the problem faced, whether we are looking for an optimal solution at the cost of a larger time complexity, or a "good enough" solution with a lower time complexity.

Additionally, the resource constraints played a part in the design of the search algorithm implemented. If we had kept track of the visited nodes for tasks 2 and 3, the result would not have been optimal. Moreover, favouring only one type of costs during the search, distance or energy cost, would have led to suboptimal results too, where the shortest path found is not the most energy efficient path amongst all the possible shortest paths.

Focusing on the overall results obtained, we can see that it is possible for improved time complexities when comparing between an informed and uninformed search. However the mileage of the improvements vary heavily based on the heuristic employed. A good heuristic can result in significant improvements, whereas a bad heuristic can have the opposite effect.

5. References

1. SCIENTIFIQUE, C. A. F. É. (2016, January 7). DISTANCES IN CLASSIFICATION.
2. *Shortest path problem*. Shortest Path Problem - an overview | ScienceDirect Topics. (n.d.). Retrieved October 6, 2022, from https://www.sciencedirect.com/topics/computer-science/shortest-path-problem