NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

SEM2020/21 Year 3

CZ3005 Lab 1 Report

Group Member	Matriculation Number
Chua Zi Jian	U2022354J
Min Kabar Kyaw	U2021858K

Task 1 : Shortest Path of relaxed version NYC instance

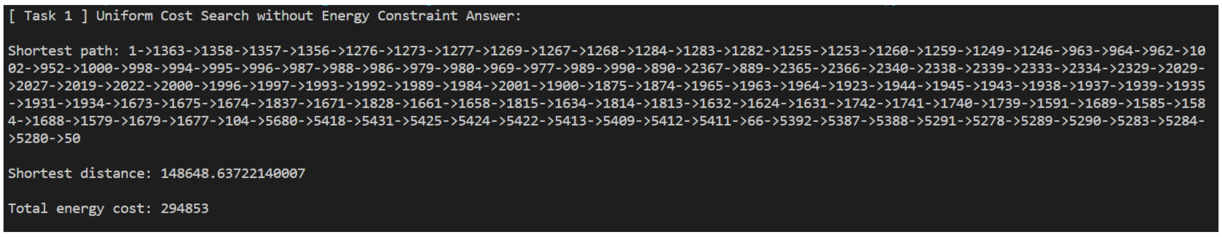
Initial Thoughts:

The shortest path problem is a well-known and familiar problem, and one of the solutions we know from our experience in previous modules is Dijkstra's Algorithm. However, Dijkstra's Algorithm for shortest path will attempt to find the shortest path from the source to **all other nodes**, which is not necessary for this task. An alternative which is very similar to Dijkstra's Algorithm is the Uniform Cost Search.

Approach:

Since the problem required to get the shortest path which is the optimal solution, the choice of the search algorithms are limited to Breadth-First Search and Uniform Cost Search. Uniform Cost Search will run faster than Breadth-First Search as it does not have to visit every node in each level until it reaches the goal. So Uniform Cost Search is a better approach.

Output:



```
[ Task 1 ] Uniform Cost Search without Energy Constraint Answer:

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->10
02->952->1000->998->994->995->996->987->988->986->979->980->969->977->989->990->890->2367->889->2365->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->1943->1938->1937->1939->1935
->1931->1934->1673->1675->1674->1837->1671->1828->1661->1658->1815->1634->1814->1813->1632->1624->1631->1742->1741->1740->1739->1591->1689->1585->158
4->1688->1579->1679->1677->104->5680->5418->5431->5425->5424->5422->5413->5409->5412->5411->66->5392->5387->5388->5291->5278->5289->5290->5283->5284-
>5280->50

Shortest distance: 148648.63722140007

Total energy cost: 294853
```

Task 2: Uninformed Search Algorithm for NYC

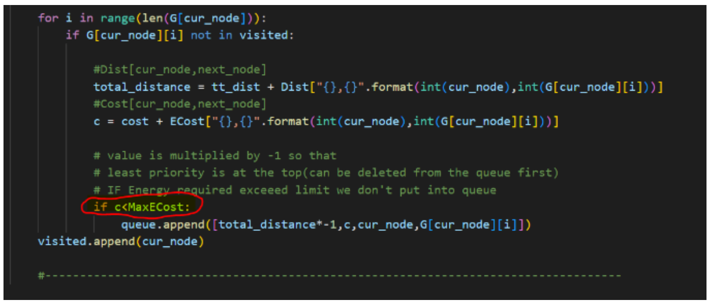
Initial Thoughts:

This problem is known as the **Constrained Shortest Path,** and it is NP-Hard [1].

Approach:

For this problem, we decided to use Uniform Cost Search since we have used it in Task 1. But in this problem, there is an additional constraint which is the energy capacity which has to be accounted for by making modifications to the algorithm.

At first, we decided to implement the same UCS algorithm but with an additional check of energy cost to ensure total energy capacity is not exceeded before we add the current node to the queue.



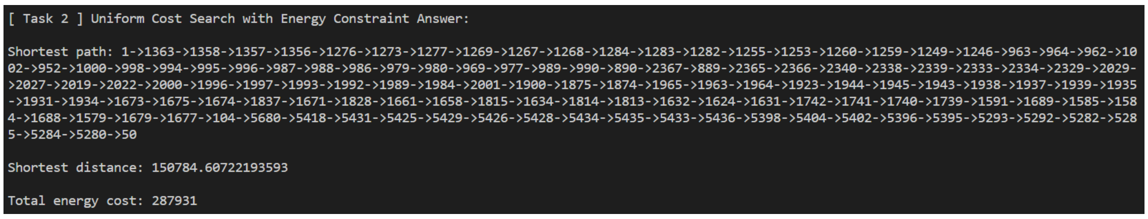
```python
for i in range(len(G[cur_node])):
    if G[cur_node][i] not in visited:

        #Dist[cur_node,next_node]
        total_distance = tt_dist + Dist["{},{}".format(int(cur_node),int(G[cur_node][i]))]
        #Cost[cur_node,next_node]
        c = cost + ECost["{},{}".format(int(cur_node),int(G[cur_node][i]))]

        # value is multiplied by -1 so that
        # least priority is at the top(can be deleted from the queue first)
        # IF Energy required exceeed limit we don't put into queue
        if c<MaxECost:
            queue.append([total_distance*-1,c,cur_node,G[cur_node][i]])
visited.append(cur_node)

#--------------------------------------------------------------------------------
```

And the result we get is:



```
[ Task 2 ] Uniform Cost Search with Energy Constraint Answer:

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->10
02->952->1000->998->994->995->996->987->988->986->979->980->969->977->989->990->890->2367->889->2365->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->1943->1938->1937->1939->1935
->1931->1934->1673->1675->1674->1837->1671->1828->1661->1658->1815->1634->1814->1813->1632->1624->1631->1742->1741->1740->1739->1591->1689->1585->158
4->1688->1579->1679->1677->104->5680->5418->5431->5425->5429->5426->5428->5434->5435->5433->5436->5398->5404->5402->5396->5395->5293->5292->5282->528
5->5284->5280->50

Shortest distance: 150784.60722193593

Total energy cost: 287931
```

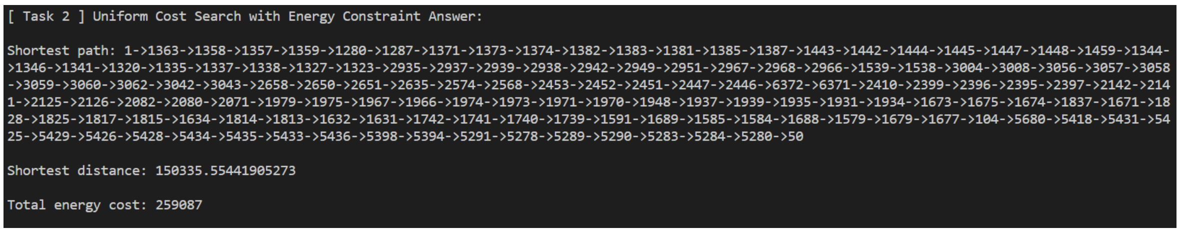
The result seems to be correct, and it does not exceed the energy constraint. However, the issue with using UCS as-is is that the final solution may not be the path with the least distance; this happens when UCS chooses a node that has the least total distance from source at a particular instance, but the shortest path through that node may exceed the energy capacity, leading to the next best alternative within that path that does not exceed the energy budget.

This means that UCS can miss out on paths that are not the most optimal but with low energy cost requirement, but would otherwise be the optimal path that adheres to the energy cost requirement as it visits the lowest cost nodes first.

Thus, we are required to revisit entries in the queue (even if the nodes have been visited by the current path) which may have been deemed to have a larger distance cost in the beginning, but may later lead to the optimal shortest path within the energy constraints.

The details of the algorithm can be found in the codes.

In the end, we get a better result but with the trade-off of longer execution time.



```
[ Task 2 ] Uniform Cost Search with Energy Constraint Answer:

Shortest path: 1->1363->1358->1357->1359->1280->1287->1371->1373->1374->1382->1383->1381->1385->1387->1443->1442->1444->1445->1447->1448->1459->1344-
>1346->1341->1320->1335->1337->1338->1327->1323->2935->2937->2939->2938->2942->2949->2951->2967->2968->2966->1539->1538->3004->3008->3056->3057->3058
->3059->3060->3062->3042->3043->2658->2650->2651->2635->2574->2568->2453->2452->2451->2447->2446->6372->6371->2410->2399->2396->2395->2397->2142->214
1->2125->2126->2082->2080->2071->1979->1975->1967->1966->1974->1973->1971->1970->1948->1937->1939->1935->1931->1934->1673->1675->1674->1837->1671->18
28->1825->1817->1815->1634->1814->1813->1632->1631->1742->1741->1740->1739->1591->1689->1585->1584->1688->1579->1679->1677->104->5680->5418->5431->54
25->5429->5426->5428->5434->5435->5433->5436->5398->5394->5291->5278->5289->5290->5283->5284->5280->50

Shortest distance: 150335.55441905273

Total energy cost: 259087
```

Task 3: A* Search Algorithm for NYC

Initial Thoughts:

A* Search algorithm uses a combination of uniform search cost and heuristics to make decisions.

In this problem, we are given the x-y coordinates of the nodes; this information can be used to derive a well-known heuristic for pathfinding problems, the Euclidean distance between two nodes.

The Euclidean distance, also known as the Pythagorean distance, refers to the length of a straight line segment between two points [2]. It is easily calculated for two points using their x-y coordinates using the following formula:

$$D = \sqrt{(x_1^2 - x_2^2) + (y_1^2 - y_2^2)}$$

Where (x_1, y_1) and (x_2, y_2) are coordinates of point 1 and point 2 respectively.

In the context of this problem, the euclidean distance that we are interested in is that of between any candidate nodes and the goal node; those with a shorter euclidean distance to the goal are deemed to be better choices for expansion. Thus, we can use the Euclidean distance function directly to be h(n), the heuristic cost function of the A* function.

Approach:

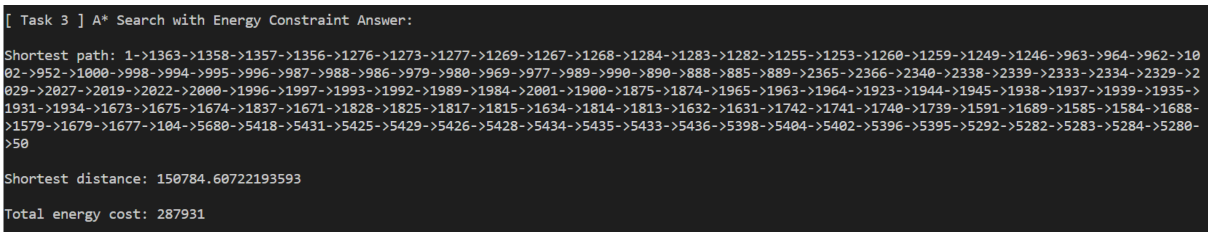
For A* search algorithm, we will sort the queue based on the evaluation function of

F(n)=g(n)+h(n), where g(n) is the Uniform Cost Search cost to the current nodes and h(n) is the heuristic function where it is the straight line distance between the current node and the end node.

And due to the energy constraint we will have the similar approach as Task 2 to get the optimal solution.

The details of the algorithm can be found in the codes.

Normal answer of A* Search with energy constraint:



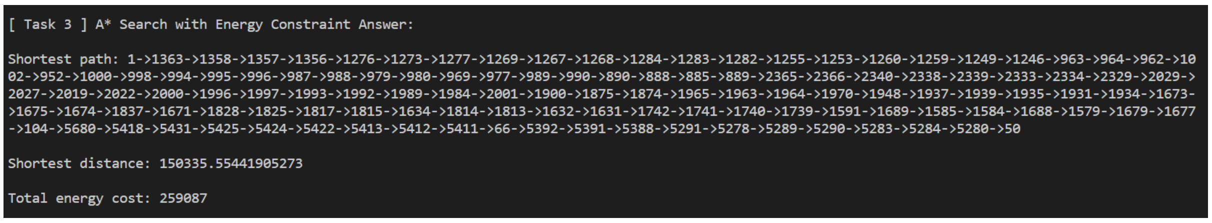
```
[ Task 3 ] A* Search with Energy Constraint Answer:

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->10
02->952->1000->998->994->995->996->987->988->986->979->980->969->977->989->990->890->888->885->889->2365->2366->2340->2338->2339->2333->2334->2329->2
029->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->5429->5426->5428->5434->5435->5433->5436->5398->5404->5402->5396->5395->5292->5282->5283->5284->5280-
>50

Shortest distance: 150784.60722193593

Total energy cost: 287931
```

Optimal solution of A* Search with energy constraint:



```
[ Task 3 ] A* Search with Energy Constraint Answer:

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->10
02->952->1000->998->994->995->996->987->988->979->980->969->977->989->990->890->888->885->889->2365->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->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
```

Performance Comparison

Initial Thoughts:

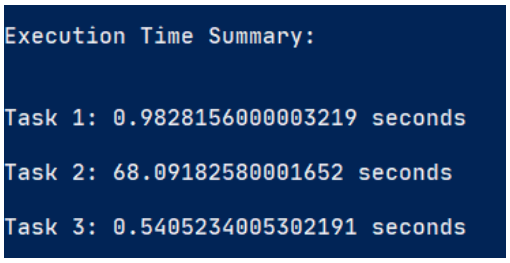
When obtaining benchmarks for comparison between different algorithms, it is important for us to consider our execution environment carefully to get the most accurate and unbiased results as some factors can affect results. For example, a simple way to measure performance could be using the system clock as a timer: getting the difference between the system clock value at the start and end of the function could yield us the time it took for the function to complete.

However, it should be noted that **time it took for the function to complete in our environment does not necessarily equate to time it took for the algorithm to perform meaningful work.**

One reason for such an occurrence is the Operating System's mechanism for scheduling processes; while our function is in the middle of running, the OS may attempt to perform multitasking by **context switching** out to other processes, effectively "stealing time" from our function. The system clock will continue to count down, even though our function has been effectively paused, awaiting processor resources to resume execution. This means that our experiment data time will have a positive offset from the true execution time.

Fortunately, there are already many benchmarking tools/libraries available for such purposes, and we have used one such python library to ensure there is sufficient accuracy to our results.

Experiment Data



```
Execution Time Summary:

Task 1: 0.9828156000003219 seconds

Task 2: 68.09182580001652 seconds

Task 3: 0.5405234005302191 seconds
```

There is a huge difference in execution time between Task 2 (UCS Optimal) and Task 3 (A* Optimal), by **more than a factor of 100!** This suggests the importance of integrating heuristics into our search algorithms as performance gains can be massive.

Conclusion

Both A* Search and Uniform Cost Search (UCS) are to solve the shortest path problem, both of them are able to get the optimal solution only when there is no energy constraint.

When there is energy constraint, a normal UCS or A* search with energy cost checking might not be able to obtain the optimal solution since exceeding the energy cost might cause the algorithm to miss out non-shortest paths that do not exceed the energy cost.

A* Search also performs much better than UCS at finding the optimal solution due to the heuristic function which is able to predict the possible shortest path instead of only checking the lowest path cost until the current node.

Contribution

Task 1, **Task 2,** **Task 3,** **Lab Report**	Chua Zi Jian (U2022354J) Min Kabar Kyaw (U2021858K)

References

[1] Garey, Michael R., and David S. Johnson.“Computers and intractability: a guide to NP-completeness.” (1979).

[2] Bogomolny, A., n.d. *The Distance Formula*. [online] Cut-the-knot.org. Available at: <https://www.cut-the-knot.org/pythagoras/DistanceFormula.shtml> [Accessed 7 September 2022].