Artificial Intelligence – Project 1 Report

Yibo Fu

G25190736

Algorithm Choice

I implemented the *Dijkstra Algorithm* for uninformed search to find the shortest path by expanding all nodes till reach the destination node which needed. I keep tracking the nodes I traveled and their iternations from each process.

```python
@timeit
def dijkstra(start, end, graph):
    shortest = {}
    cur = PriorityQueue(len(graph.nodes) ** 2)
    cur.put((0.0, (start, -1)))
    i = 0
    while not cur.empty():
        dist, (node, lastNode) = cur.get()
        if node in shortest:
            continue
        shortest[node] = (dist, lastNode)
        if node == end:
            break
        for _, neighbour, d in graph.edges[node]:
            if neighbour in shortest:
                continue
            cur.put((dist + d, (neighbour, node)))
            i += 1
    print(f"Iteration for {i} times")

    return parse_path(start, end, shortest)
```

Secondly, I implemented _A* Algorithm_ for informed search to find the shortest path. It has a Heuristic function to get estimation of cost to destination and estimate of cost to reach the destination then take that into consideration for path finding.

```python
@timeit
def a_star(start, end, graph):
    def heuristic(lhs, rhs):
        lhs = graph.get_node(lhs)
        rhs = graph.get_node(rhs)
        return abs(lhs.x - rhs.x) + abs(lhs.y - rhs.y)
    shortest = {}
    open_list = PriorityQueue(len(graph.nodes) ** 2)
    open_list.put((0.0, (start, -1, 0)))
    i = 0
    while not open_list.empty():
        f, (node, last_node, path_length) =
open_list.get()
        if node in shortest and path_length >=
shortest[node][0]:
            continue
        shortest[node] = (path_length, last_node)
        if node == end:
            break
        for _, neighbour, d in graph.edges[node]:
            if neighbour in shortest:
                continue
            g = path_length + d
            f = g + heuristic(neighbour, end)
            open_list.put((f, (neighbour, node, g)))
            i += 1
    print(f"Iteration for {i} times")

    return parse_path(start, end, shortest)
```

Both searching algorithms are wraped by timer which imported from time package and calculates run time for each search process.

Script Running

Follow the usage of this script as follows. Please note this script is based on Python 3.7.

```
$ python main.py graphs/[path] --start [] --end []
$ python main.py graphs/graph500_0.4 --start 20 --end 90
Start [a_star]:
Iteration for 21725 times
End [a_star].  Time elapsed: 0.065 sec.
Shortest path from 20 to 90:
[(20, 0), (372, 11.0), (282, 15.0), (349, 16.0), (90,
38.0)]

Start [dijkstra]:
Iteration for 28313 times
End [dijkstra].  Time elapsed: 0.081 sec.
Shortest path from 20 to 90:
[(20, 0.0), (372, 11.0), (282, 15.0), (349, 16.0), (90,
38.0)]
```

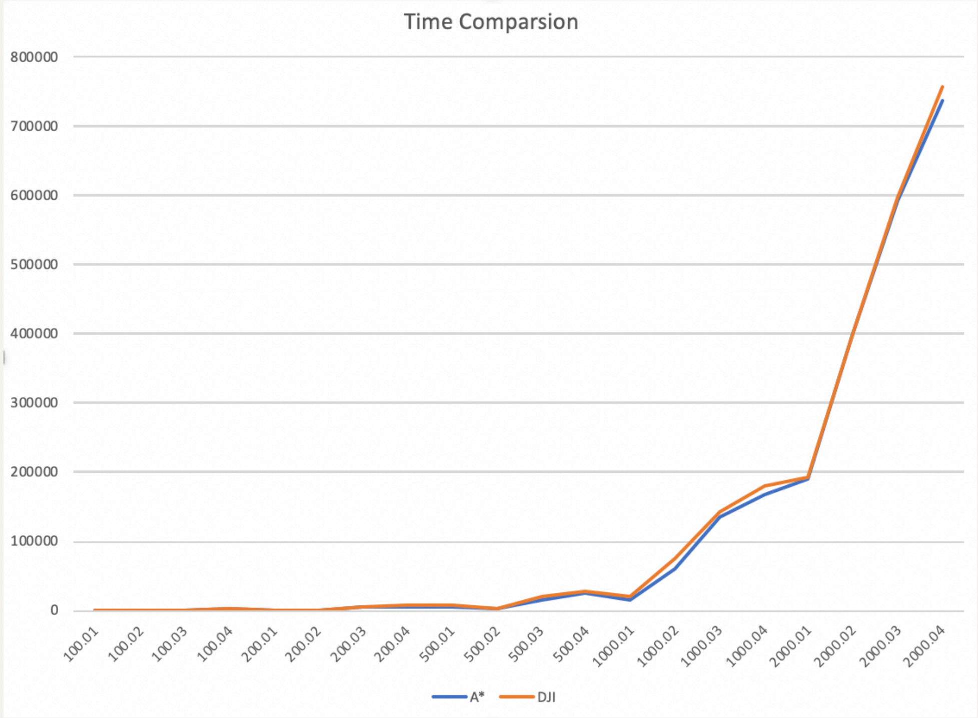
Where the [path] is the graph path and --start, --end are start and end point inputs. If no start or end point input, it will randomly genrate them.

```
$ python main.py graphs/graph500_0.4
Start [a_star]:
Iteration for 38786 times
End [a_star].  Time elapsed: 0.147 sec.
Shortest path from 448 to 424:
[(448, 0), (256, 2.0), (410, 13.0), (92, 17.0), (293,
18.0), (110, 20.0), (71, 26.0), (79, 32.0), (424, 48.0)]

Start [dijkstra]:
Iteration for 44337 times
End [dijkstra].  Time elapsed: 0.121 sec.
Shortest path from 448 to 424:
[(448, 0.0), (256, 2.0), (410, 13.0), (92, 17.0), (293,
18.0), (110, 20.0), (71, 26.0), (79, 32.0), (424, 48.0)]
```

Result Analysis

As discussed on previouse lectures, the A* algorithm should faster than Dijkasra algorithm. Also we can see that from above running results on shell since A* algorithm always has less iteration time than Dijkstra. For all graphs run time comparsion please refer to chart below where A* is always faster than Dijkasra algorithm.


Time Comparsion
800000
700000
600000
500000
400000
300000
200000
100000
0
100.01
100.02
100.03
100.04
200.01
200.02
200.03
200.04
500.01
500.02
500.03
500.04
1000.01
1000.02
1000.03
1000.04
2000.01
2000.02
2000.03
2000.04
A*
DJI