Project 1

CSCI-4511/6511

GitHub Link：https://github.com/lukelii/CSCI-4511

Overall Data Modification:

Since the data for both Edges and Vertices are stored in the same file, I have used Numpy package to divide them into two separate arrays for later use. And formed an adjacency matrix for Both searches.

Package used:

NumPy, sys, datetime, networkx, dijkstar

Part 1:

For this Part, I used Dijkstra algorithm which is really the fundamental part for A\*(Informed search).

For Dijkstra, I created a shortest path tree set sptSet, which keeps track of vertices in order to calculate the minimum shortest distance, initially empty. Read in the distance value from Adjacency matrix and initially assign it to 0 so it will be pick first (since all the other value are initialize as INF). While the shortest path from source to goal is not found, pick a temp vertex with minimum distance that is not in sptSet yet, do the calculations and include it in sptSet, repeat this step for all vertices and update its distance value.

Alternatively, by using package Dijkstar, we can get the result really simply by importing our 2D array.

Guide for Script:

Method 1:

Using Dijkstra.py, run py -3 Dijkstra.py in the folder with p1\_grapth.txt.

-To change source or goal: change code in line 141, g.dijkstra(0,99)

first element is source and second will be the goal

Method 2:

Using package Dijkstar, pip install dijkstar first

Using Dijtwo.py, run py -3 Dijtwo.py in the folder with p1\_grapth.txt.

-To change source or goal: This script read values from txt file, to manually change it , please modify code in line 46, print(find\_path(graph, int(P[0][1]), int(P[1][1]))), first underlined part is the source, second is the part

Part 2:

For informed search, I am using A\* search. To utilize informed search, A\* search is basically similar with Dijkstra besides having a heuristic function to ‘predict’ the distance from the chosen point and applies the heuristic value H with calculated distance G from source to chosen point (F= G + H), by iterating through the list, we can find the smallest F, which will reduce our run-time since there will be less un-used path calculated.

Guide for script:

-Using Networkx package, pip install networkx

- Using Astar.py, run py -3 Astar.py in the folder with p1\_grapth.txt.

-To modify source and goal, update line 124 and 125, modify 0,99 in both lines.

Performance comparison:

Since our data set is limited, the run-time differences are indistinguishable. Both scripts run under 0.5s with minor fluctuation.