

# CSCI 6511 Artificial Intelligence

Ruocheng Shan

## Project 1 Report – Uninformed and Informed Search

### 1. User Guide

#### 1.1 Run single case test on terminal

```
python main.py v.txt e.txt [start] [end]
```

Note:

- v.txt and e.txt should be the **absolute path** of the file
- DO NOT** contain SPACE in the path
- if not specify start and end, a random pair will be generated

Example:

```
PS D:\GWU\2020Spring\6511Artificial_Intelligence-AmrinderArora\Project_1> python main.py D:\GWU\2020Spring\6511Artificial_Intelligence-AmrinderArora\Project_1\graphs\graph100_0.1\v.txt D:\GWU\2020Spring\6511Artificial_Intelligence-AmrinderArora\Project_1\graphs\graph100_0.1\e.txt 1 99
Start-----
[INFO] Reading vertex file from:
D:\GWU\2020Spring\6511Artificial_Intelligence-AmrinderArora\Project_1\graphs\graph100_0.1\v.txt
[INFO] Reading edge file from:
D:\GWU\2020Spring\6511Artificial_Intelligence-AmrinderArora\Project_1\graphs\graph100_0.1\e.txt
-----
[INFO] Running dijkstra algorithm
start node: 1
goal node: 99
-----Result for dijkstra-----
-Total searching time in second is: 0 seconds
-Total searching time in stamp is: 0.0009975433349609375
-Shortest distance is 105
-The path from start to end is ['1', '32', '56', '99']
-----
[INFO] Running astar algorithm
start node: 1
goal node: 99
-----Result for A*-----
-Total searching time in second is: 0 seconds
-Total searching time in stamp is: 0.0009970664978027344
-Shortest distance is 105
-The path from start to end is ['1', '32', '56', '99']
-Visited nodes for astar are ['1', '10', '62', '148', '31', '58', '92', '76', '32', '59', '38', '53', '95', '57', '84', '63', '71', '88', '65', '93', '79', '74', '56', '85', '68', '77', '99']
-Number of nodes visited: 27
PS D:\GWU\2020Spring\6511Artificial_Intelligence-AmrinderArora\Project_1>
```

#### 1.2 Run all cases test on terminal

```
python testall.py
```

Note:

- all start and end nodes are generated randomly for each graph in testall.py

Example:

```
PS D:\GWU\2020Spring\6511Artificial_Intelligence-AmrinderArora\Project_1> python testall.py
Testing for graph1000_0.1
dijkstra
start nodes: 701
goal node: 844
-----Result for dijkstra-----
-Total searching time in second is: 0 seconds
-Total searching time in stamp is: 0.03593635559082031
-Shortest distance is 60
-The path from start to end is ['701', '844']
-----
A*
start nodes: 701
goal node: 844
-----Result for A*-----
-Total searching time in second is: 0 seconds
-Total searching time in stamp is: 0.0020189285278320312
-Shortest distance is 60
-The path from start to end is ['701', '844']
-Visited nodes for astar are ['701', '738', '793', '996', '741', '986', '782', '892', '849', '743', '971', '899', '879', '981', '779', '975', '720', '778', '844']
-Number of nodes visited: 19
```

## 2. Result Analysis

### 2.1 Testing Method

- Generate random starting node and destination node pair for Dijkstra and A\* search
- Note that sometimes random generated nodes are **not reachable**
- If two nodes are not reachable, information will be showed on log

```
_____Testing for graph1000_0.3_____
_____dijkstra_____
start node: 495
goal node: 72
end node 72 is not reachable from start node 495
Algorithm failed
_____A*_____
start node: 495
goal node: 72
Algorithm failed
```

- Test for the shortest distance, path and total time cost for each algorithm

```
_____Testing for graph1000_0.4_____
_____dijkstra_____
start node: 478
goal node: 630
-----Result for dijkstra-----
-Total searching time in second is: 0 seconds
-Total searching time in stamp is: 0.04784560203552246
-Shortest distance is 71
-The path from start to end is ['478', '541', '630']
_____A*_____
start node: 478
goal node: 630
-----Result for A*-----
-Total searching time in second is: 0 seconds
-Total searching time in stamp is: 0.0050089359283447266
-Shortest distance is 71
-The path from start to end is ['478', '541', '630']
-Visited nodes for astar are ['478', '864', '933', '701', '622', '677', '909', '783', '541', '819', '738', '881', '894', '912', '989', '569', '970', '574', '734', '861', '898', '915', '840', '826', '643', '743', '720', '790', '905', '896', '868', '610', '685', '703', '740', '766', '814', '930', '801', '547', '938', '976', '966', '482', '891', '952', '551', '916', '636', '573', '963', '759', '522', '980', '979', '561', '854', '592', '530', '829', '520', '714', '843', '964', '648', '943', '807', '709', '730', '899', '704', '568', '580', '488', '712', '498', '698', '627', '524', '615', '893', '736', '767', '684', '723', '857', '967', '741', '555', '994', '600', '508', '613', '655', '841', '705', '932', '948', '552', '630']
-Number of nodes visited: 100
```

### 2.2 Testing Result

- a. **Total time cost** of a same searching problem using Dijkstra is higher than using A\*
- b. **All nodes are visited** in Dijkstra; a relatively **small number of nodes** are visited in A\*