**CSCI 6511 Artificial Intelligence**

**Ruocheng Shan**

**Project 1 Report – Uninformed and Informed Search**

## 1. User Guide

### 1.1 Environment

- Have Python 3.5 + installed in your device

- Install dependences by the commend

**pip install -r requirements.txt**

```
➜  search_alg git:(master) ✗ pip install -r requirements.txt
Requirement already satisfied: matplotlib==3.1.3 in /Users/wawang250/.pyenv/versions/3.7.4/lib/python3.7/
Requirement already satisfied: numpy==1.18.1 in /Users/wawang250/.pyenv/versions/3.7.4/lib/python3.7/site
Requirement already satisfied: cycler>=0.10 in /Users/wawang250/.pyenv/versions/3.7.4/lib/python3.7/site-
Requirement already satisfied: python-dateutil>=2.1 in /Users/wawang250/.pyenv/versions/3.7.4/lib/python3
Requirement already satisfied: kiwisolver>=1.0.1 in /Users/wawang250/.pyenv/versions/3.7.4/lib/python3.7/
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /Users/wawang250/.pyenv/versio
Requirement already satisfied: six in /Users/wawang250/.pyenv/versions/3.7.4/lib/python3.7/site-packages
Requirement already satisfied: setuptools in /Users/wawang250/.pyenv/versions/3.7.4/lib/python3.7/site-pa
```

### 1.2 Run single case test on terminal

- Run program by commend

**python main.py**

- Follow instructions to input

- Get the results in your terminal

```
➜  search_alg git:(master) ✗ python main.py
----------------------Program Start-----------------------
Enter Graph Code (100_0.1, 2000_0.4, etc): 100_0.3
Enter source point: 1
Enter destination point: 99
---------------------------------
[INFO] Running dijkstra algorithm
start node:  1
goal node:  99
----------Result for dijkstra-----------
-Total searching time is: 0.0008370876312255859 s
-Shortest distance is  58
-The path from start to end is ['1', '78', '99']
-Total searching steps: 138
---------------------------------
[INFO] Running A* algorithm
start node:  1
goal node:  99
----------Result for A*-----------
-Total searching time is: 0.0005772113800048828 s
-Shortest distance is  58
-The path from start to end is ['1', '78', '99']
-Total searching steps: 114
----------------------Program Start-----------------------
Enter Graph Code (100_0.1, 2000_0.4, etc):
```

Note:

a. some end node is not reachable from the start node

b. if not specify start and end, a random pair will be generated

### 1.2 Run all cases test on terminal

- Run test script by commend

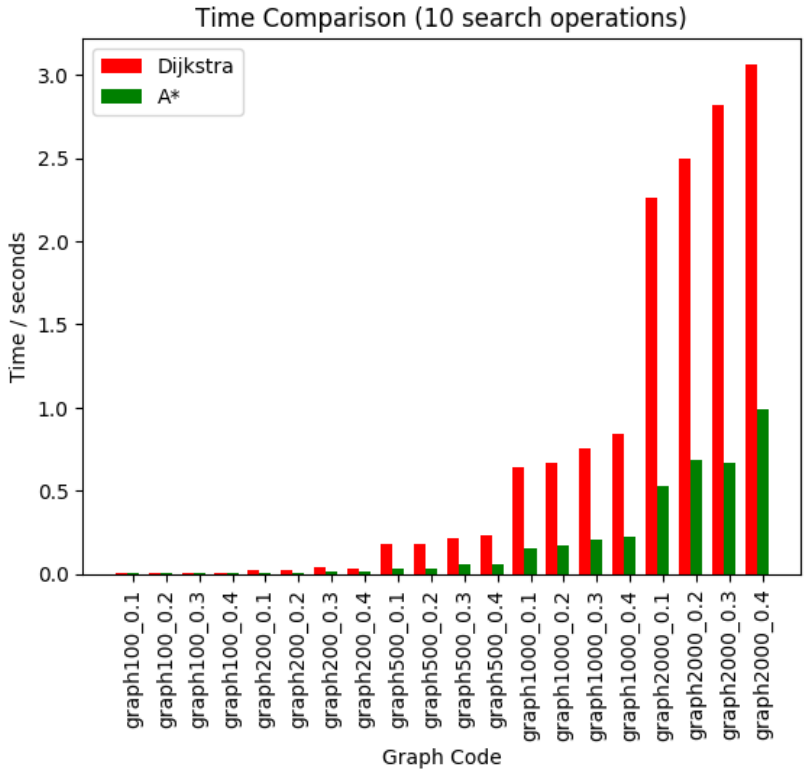
**python test.py**

- The program will generate 10 pairs of start and end nodes for each graph
- It takes a little while since randomly generated pair sometimes not reachable and will be re-generated
- A **.png** file will be create in the **results** folder
- The .png file is an analysis of comparison in actual runtime of each algorithm
- Logs will be printed in your terminal

```
→  search_alg git:(master) ✗ python test.py
```

## 2. Result Analysis

### 2.1 Time comparision



### 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*

## 3. Algorithm Choice

I chose Dijkstra for uninformed search and A* for informed search. Since A* has a heuristic function for each node, and we can consider the heuristic function for Dijkstra is all 0. Hence, these two algorithms are comparable.

For the heuristic calculation in A*, I used Euclidian distance to measure. But for the up-to 8 neighbor nodes of a node, the h values is set to be 0 by definition.

## 4. Project Structure

Find the structure below for the project structure of source code: