**CSE 5360 AI-1**

**Assignment-1**

**Readme file**

**Name: Sagar Sharma**

**UTA ID: 1001626958**

Programming Language used: **Python 2**

**Note: All the files must be in the same directory**.

There are **4** files:

* find\_route.py
* Node.py
* input file
* heuristic values file

***Commands to run the program***:

For Uninformed search strategy i.e Uniform cost search:

**python find\_route.py uninf <input\_file.txt> <source> <destination>**

For Informed search strategy i.e A\* search:

**python find\_route.py inf <input\_file.txt> <source> <destination> <hueristic\_file.txt>**

Examples:

**python find\_route.py uninf input1.txt berlin Kassel**

**python find\_route.py inf input1.txt berlin Kassel h\_kassel.txt**

**Program Structure:**

**find\_route.py:**

This file contains the following functions:

1. **read\_file(input\_file)**:

This function reads the input file and returns the content of the file in a list.

1. **read\_hueristics(h\_file)**:

This function reads the heuristics file and return the content of the file in a list.

1. **expand\_node(fringe, paths, visited\_node)** for **Uniform cost search** or **a\_expand\_node(fringe, paths, heuristics, visited\_node)** for **A\* search**:

This function expands a node and returns all the nodes connected to that node.

It has parameters:

* ***fringe*** (has nodes to compare with the goal state, sorts the nodes based on ***cumulative path cost*** of the nodes for Uniform cost search or ***fvalues*** of the nodes for A\* search).
* ***paths*** (has step cost from one node to another node).
* ***visited\_node*** (contains record of visited node for graph search implementation).
* ***Heuristics*** (value associated with each node wrt the goal state, added with cumulative path cost to sort the fringe for A\* search).

This function just expands a node and adds all the generated nodes to the fringe if they are not visited yet and assigns all the values to node representing a state, for e.g. city name, parent node, fvalue, cumulative path cost, step cost.

1. **create\_fringe(paths, source, dest\_city)** or **a\_create\_fringe(paths, heuristics, source, dest\_city)**:

It has parameters:

* ***paths*** (has step cost from one node to another node).
* ***Source*** (it stores the start state).
* ***dest\_city*** (goal state city).
* ***Heuristics*** (value associated with each node wrt the goal state, added with cumulative path cost to sort the fringe for A\* search).

This function sorts the fringe based on the increasing **cumulative path cost for Uniform Cost search** or based on increasing **fvalue (gvalue+ heuristic value) for A\* search**.

1. **track\_path(des\_node)** for both **uniform cost search and A\* search**:

This function prints the **retraced path** from the destination node**(des\_node)** to the start node in reversed manner alongside the **associated step cost** as well as the **total cost** of the path.

1. **ucs(paths, source, des)**:

This function implements **Uninformed cost search** on the paths in the graph from the start state to the goal state. It makes use of create\_fringe(), expand\_node() and track\_path() functions to carry out the search strategy.

1. **a\_search(paths, heuristics, source, des)**:

This function implements **A\* search** on the paths in the graph from the start state to the goal state. It makes use of a\_create\_fringe(), a\_expand\_node() and track\_path() functions to carry out the search strategy.

1. **main()**:

Program starts from this function.

**References**:

* https://www.youtube.com/watch?v=5OJv6iHMtVw