CSCI-561 - Fall 2016 - Foundations of Artificial Intelligence
Homework 1

Due September 21, 2016 23:59:59

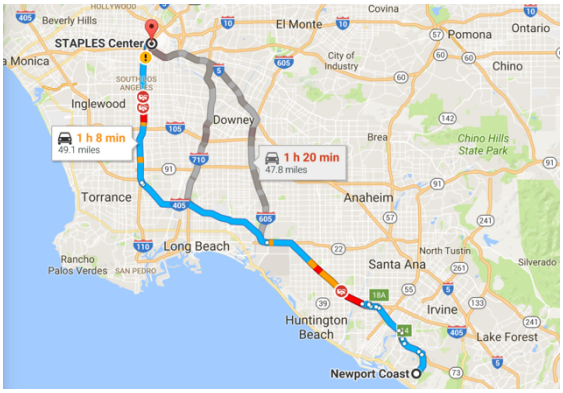

image from Google maps


photo from caranddriver.com

Guidelines

This is a programming assignment. You will be provided sample inputs and outputs (see below). Please understand that the goal of the samples is to check that you can correctly parse the problem definitions, and generate a correctly formatted output. The samples are very simple and it should not be assumed that if your program works on the samples it will work on all test cases. There will be more complex test cases and it is your task to make sure that your program will work correctly on any valid input. You are encouraged to try your own test cases to check how your program would behave in some complex special case that you might think of. Since **each homework is checked via an automated A.I. script**, your output should match the example format *exactly*. Failure to do so will most certainly cost some points. The output format is simple and examples are provided. You should upload and test your code on vocareum.com, and you will submit it there. You may use any of the programming languages provided by vocareum.com.

Grading

Your code will be tested as follows: Your program should not require any command-line argument. It should read a text file called “input.txt” in the current directory that contains a problem definition. It should write a file “output.txt” with your solution. Format for input.txt and output.txt is specified below. End-of-line convention is Unix (since vocareum is a Unix system).

The grading A.I. script will, 50 times:

- Create an input.txt file, delete any old output.txt file.
- Run your code.
- Compare output.txt created by your program with the correct one.

- If your outputs for all 50 test cases are correct, you get 100 points.
- If one or more test case fails, you get 50 – N points where N is the number of failed test cases.

Note that if your code does not compile, or somehow fails to load and parse input.txt, or writes an incorrectly formatted output.txt, or no output.txt at all, or OuTpUt.TxT, **you will get zero points**. Anything you write to stdout or stderr will be ignored and is ok to leave in the code you submit. Please test your program with the provided sample files to avoid this.

Academic Honesty and Integrity

All homework material is checked vigorously for dishonesty using several methods. All detected violations of academic honesty are forwarded to the Office of Student Judicial Affairs. To be safe you are urged to err on the side of caution. Do not copy work from another student or off the web. Keep in mind that sanctions for dishonesty are reflected in *your permanent record* and can negatively impact your future success. As a general guide:

Do not copy code or written material from another student. Even single lines of code should not be copied.
Do not collaborate on this assignment. The assignment is to be solved individually.
Do not copy code off the web. This is easier to detect than you may think.
Do not share any custom test cases you may create to check your program's behavior in more complex scenarios than the simplistic ones considered below.
Do not copy code from past students. We keep copies of past work to check for this.

Do ask the professor or TA if you are unsure about whether certain actions constitute dishonesty. It is better to be safe than sorry.

Project description

The Los Angeles Lakers are playing against their rivals the Boston Celtics tonight. Lakers star Jordan Clarkson wants to arrive earlier today to prepare himself for the game, and he is leaving from his mansion at Newport Coast to Staples Center. As everyone knows, Los Angeles is notorious for its traffic. Driving his 2016 Lamborghini Aventador, Jordan definitely does not want to be stuck in traffic. Please help Jordan find a route to get him to Staples Center as fast as possible.

To accomplish this, you will be given a list of freeway or road intersections (i.e., locations) and the time it would take to travel from there to other freeway or road intersections. You will be required to create a program that finds the fastest route Jordan must travel to get to Staples Center. Your program should run on vocareum.com using one of the languages and compiler supported by that platform. Your program will be graded on vocareum.

Your program will be given **live traffic** information in the input.txt file, which is an arbitrarily large list of current traveling times between intersections/locations. An example live traffic data would be a list of intersections to intersections with traveling time (in minutes), in the following format (see below for the full specification of input.txt):

```
JordanHome CA73/NewportCoastDr 5
CA73/NewportCoastDr I405/CA73 10
I405/CA73 I110/I405 25
I110/I405 I10/I110 20
I10/I110 I110/I405 30
I10/I110 I10/I405 9
I105/I110 I10/I110 7
I10/I110 StaplesCenter 3
```

Traveling time may not be the same for both directions. For example, in the above:

```
I110/I405 I10/I110 20
```

indicates that it takes 20 minutes to travel from `I110/I405` to `I10/I110` (northbound as you follow the 110 freeway), but

```
I10/I110 I110/I405 30
```

indicates that it takes 30 minutes in the other direction (southbound).

Beside live traffic information, Jordan also has an idea of how long it takes on a traffic-free Sunday from each intersection/location to StaplesCenter. Hence, the input.txt file will also contain Jordan's **Sunday traffic** estimate of traveling time from each location listed in the file to his destination, which is also an arbitrarily large list of intersections/locations with estimated traveling time (in minutes) from there to StaplesCenter on a traffic-free Sunday:

```
JordanHome 55
CA73/NewportCoastDr 50
I405/CA73 40
I110/I405 28
I10/I110 8
I10/I405 39
I105/I110 23
StaplesCenter 0
```

Your program should write in output.txt the list of intersections/locations traveled over in your solution path, including the starting and finishing locations and the **accumulated** time from start to that intersection/location, in order of travel, for example:

```
JordanHome 0
CA73/NewportCoastDr 5
I405/CA73 15
I110/I405 40
I10/I110 60
StaplesCenter 63
```

You must solve this problem using **Breadth-First search, Depth-First Search, Uniform-cost Search**, and **A* Search** separately.

Full specification for input.txt:

```
<ALGO>
<START STATE>
<GOAL STATE>
<NUMBER OF LIVE TRAFFIC LINES>
<… LIVE TRAFFIC LINES …>
<NUMBER OF SUNDAY TRAFFIC LINES>
<… SUNDAY TRAFFIC LINES …>
```

where:

<ALGO> is the algorithm to use and will be one of: “BFS”, “DFS”, “UCS”, “A*”.

<START STATE> is a string with the name of the start location (e.g., JordanHome).

<GOAL STATE> is a string with the name of the goal location (e.g., StaplesCenter).

<NUMBER OF LIVE TRAFFIC LINES> is the number of lines of live traffic information that follow.

<… LIVE TRAFFIC LINES …> are lines of live traffic information in the format described above, i.e., <STATE1> <STATE2> <TRAVEL TIME FROM STATE1 TO STATE2>

<NUMBER OF SUNDAY TRAFFIC LINES> is the number of lines of Sunday traffic estimates that follow.

<… SUNDAY TRAFFIC LINES …> are lines of sunday traffic information in the format described above, i.e., <STATE> <ESTIMATED TIME FROM STATE TO GOAL>

Full specification for output.txt:

Any number of lines with the following format for each:

```
<STATE> <ACCUMULATED TRAVEL TIME FROM START TO HERE>
```

Additional notes:

- Please name your program "**homework.xxx**" where 'xxx' is the extension for the programming language you choose. ("**py**" for python, "**cpp**" for C++, and "**java**" for Java). If you are using C++11, then the name of your file should be "**homework11.cpp**" and if you are using python3.4 then the name of your file should be "**homework3.py**".
- Times are positive or zero integers (32-bit ok, larger ok too).
- Capitalization of state names will be maintained.
- State names will not have any white spaces in them and will consist of ASCII letters, numbers and/or other non-special ASCII characters.
- There will not be any duplicate entries in the live traffic data.
- There will be exactly one entry in the Sunday traffic data for each state mentioned in the live traffic data.
- Intersections of 2 freeways are always given in the same order. For example, when referring to I110/I405, it is always I110/I405, rather than sometimes I405/I110 and sometimes I110/I405.
- In some test cases, you must be able to help Jordan find his way from/to other places as well (not only from JordanHome to StaplesCenter). After the game, you never know, he might want to go somewhere else to celebrate. Hence, make sure you read START STATE and GOAL STATE from input.txt and do not assume they will be the same as in the example above.
- In output.txt, the first line should always be "<START STATE> 0" and the last line's state should always be <GOAL STATE> (and the total accumulated travel time should follow).
- Hence, if the start state is the same as the goal state, you should output a single line "<START STATE> 0".
- The goal state will always be reachable from the start state, you do not need to worry about cases where no path can be found from start to goal.
- As studied in class, BFS and DFS by definition ignore step costs and always assume a unit step cost. The accumulated time reported in your output.txt should also use unit step costs (i.e., report the number of steps).
- If all else is equal while searching routes (ties), you should explore (enqueue) multiple paths from the same intersection **in the order** in which they are listed in the **live traffic** inputs.
- There are multiple definitions around the web and in various textbooks for the algorithms that you will implement. Different implementations may sometimes give different results (e.g., a recursive implementation of DFS sometimes gives slightly different results than the one studied in class, when ties are encountered and you follow the instructions to then follow paths in the order listed in the live traffic input). In this homework we ask you to ***use the algorithm definitions which we have used in the lecture slides***. The reference implementation used to create the problem solutions will use these algorithms. Hence, beware that using different algorithms may result in solutions that disagree with the grading script.
- The Sunday traffic information may or may not be admissible. We understand that when it is not admissible your solution may not be optimal. You should not worry about that.

Example 1: Consider this input.txt:

```
BFS
A
D
4
A B 5
A C 3
B D 1
C D 2
4
A 4
B 1
C 1
D 0
```

Would yield the following output.txt:

```
A 0
B 1
D 2
```

(remember that BFS assumes by definition a unit step cost, and see notes above). This path is optimal for BFS with a total cost of 2 (the other path through C is optimal as well with cost 2).

Example 2: Consider this input.txt:

```
A*
A
D
4
A B 3
A C 3
B D 2
C D 1
4
A 4
B 2
C 1
D 0
```

Would yield the following output.txt:

```
A 0
C 3
D 4
```