(364-1-1441) Foundations of Artificial Intelligence

Problem Set 1

Due: 26/11/2020

You need to submit both code and written answers. Problem 1 is a programmatic one, but there are things there you need to submit in writing. Problem 2 only requires written answers and not programming. You will submit one `q1.py` file containing your code, and one `answers.pdf` file containing your written answers (in English or Hebrew).

1 Problem 1: Surfin' USA

`https://www.youtube.com/watch?v=KcZn05qxVgg`

To celebrate the American 2020 elections, in this exercise you will build a function that checks how multiple voters in various US counties can reach their voting locations. You will receive a list of up to 5 starting places (expressed as US counties of the format "`<name>, <2 letter US state code>`"), and a series of the same number of ending locations. Your goal is to bring those voters to their destinations by going through adjacent counties (using a CSV file that tells you which are adjacent, see below).

You will get the starting and ending locations each as an array of strings.

For example:

Starting locations: {Washington County, UT ; Chicot County, AR ; Fairfield County, CT}

Ending locations: {San Diego County, CA ; Bienville Parish, LA ; Rensselaer County, NY}

The example below is how path output should look like. In each line, each county name represents a node in the path. Note that once a destination has been reached, it continues to appear, unchanged, in the output.

```
{Washington County, UT ; Chicot County, AR ; Fairfield County, CT}
{Mohave County, AZ ; Ashley County, AR ; Dutchess County, NY}
{San Bernardino County, CA ; Union Parish, LA ; Berkshire County, MA}
{Orange County, CA ; Lincoln Parish, LA ; Rensselaer County, NY}
{San Diego County, CA ; Bienville Parish, LA ; Rensselaer County, NY}
```

The different paths here are:

1. Washington County, UT → Mohave County, AZ → San Bernardino County, CA → Orange County, CA → San Diego County, CA

2. Chicot County, AR → Ashley County, AR → Union Parish, LA → Lincoln Parish, LA → Bienville Parish, LA

3. Fairfield County, CT → Dutchess County, NY → Berkshire County, MA → Rensselaer County, NY

In general, shorter paths are better.

In this exercise you will only implement A* algorithm to solve this. However, additional algorithms will be coming, so keep that in mind...

You will write a function in python called `find_path` that takes in 4 variables (in this order of input):

`starting_locations` This is the beginning of your search. You can assume these are all valid counties in the format they appear in the county DB (though you can write a function to check this, which will probably help in your debugging).

`goal_locations` These are the locations your voters need to reach from the starting locations. You can assume these are all valid locations (again, a function checking its legality will probably be helpful to you)

`search_method` This will be a number. It will have multiple possible values are, but for now you will only implement:

1. An A*-heuristic search. **You choose the heuristic**. In your submitted answers, containing the answers to the questions, you will detail your heuristic. It cannot be trivial (i.e., all 0). You need to explain in your submitted answers if your heuristic is admissible, consistent, or neither.

`detail_output` This is a binary variable. When it is false, your output is like the text above – you give the full chain of locations. The first one contains the *starting locations*, and every following line a list of counties that can be legally reached by a single step from the location line before it, until the last line contains the *goal locations*. If no path was found, the output is `No path found`.

If the binary variable is true, for the first transformation (from the first set of locations to your second set of locations) you need to print out your work process, so for search method:

A*-heuristic search Print out the heuristic value of the the state of your locations after the first transformation.

(again, for the additional algorithms to come, additional details will be needed here)

In order to know which counties are adjacent, you can use the file `adjacency.csv` downloadable from the course website. Each line contains a pair of adjacent counties (be careful, the file include each county as adjacent to itself!).

2 Problem 2: Search

Describe a state space in which iterative deepening search performs much worst than DFS (for example, time complexity of $O(n^2)$ vs. $O(n)$).