CSC520 Fall 2020 Homework 1
Due August 26th at 11:59pm EST

This assignment includes both conceptual and code questions. It must be completed individually. You may not collaborate with other students, share code, or exchange partial answers. Questions involving answers or code must be emailed to the instructor or TAs directly or discussed during office hours. Your answers to the conceptual questions must be uploaded to Moodle as a pdf file titled `Assign1-<unityid>.pdf`. Your source code must be submitted as a self-contained zip called `Assign1-<unityid>.zip` All code must be clear, readable, and well-commented. You may only use libraries to provide standard data structures. All third party library use must be checked with Dr. Lynch or the TA *before* submission.

Note: The code will be tested on the NCSU VCL system. You are advised to test your code there before submission.

Question 1 (15 points)

The Kathmandu Durbar Square is one of the UNESCO World Heritage sites. Suppose we have an agent that works for the Kathmandu Durbar Square museum. The agent is assigned to check the state of the museum's hallways and artifacts every six hours. The hallways have different stationary objects, some of which are fragile and irreplaceable. Contact should be avoided. Answer the following questions and *justify your answers*:

(a) Define PEAS specification for the agent with environment features specified.

(b) Is it sufficient for the agent to be simple reflex? Why or why not? And if not what level is necessary?

(c) Would the ability to move randomly improve the agent's performance or not? Identify possible disadvantages.

Question 2 (25 points)

Locate a real-world AI application not previously discussed in class. This must be a specific commercial product or research tool that has been *developed and deployed* for use. Answer the questions below. In each case you must *cite* published materials describing the application when justifying your answers.

1. Provide a PEAS specification for the application.
2. Specify the agent's hardware.
3. What are the percepts and actions used by the agent?
4. Is it necessary for the agent to be a learning agent?
5. What is one improvement that you can make to the published design? Justify your improvement.

Question 3 (60 points)

You have been given a file that represents a road map of the United States (and some other areas). Nodes represent locales, and the numbers on each edge indicate driving distances between connected cities. The distances and lat/long coordinates may be found in the `Cities.txt` file. In Java, or Python, implement Depth-First Search, Breadth-First Search, and A* using the heuristic of straight-line distance between the cities.

- Ties must be broken in alphabetical order.
- You may use the formula for lat/long shown in the header file.
- Your code should be capable of tracking the expanded nodes as well as finding the solution path.
- For the questions below you are advised to trace the algorithms yourself to verify that they are correct.
- Submit your code along with answers to the questions below. Your grade will be based on both your answers and the quality of your code.

Name your source file and main function as `SearchAssign1`. Java files must include instructions on how to compile the code on the VCL to produce a self-contained jar. Python code must be built into a single package. The code will be executed as follows:

```
java -jar SearchAssign1.jar <searchtype> <inputfile> <startcity> <endcity> <outfile>


python SearchAssign1/SearchMain.py <searchtype> <inputfile> <startcity> <endcity>
<outputfile>
```

- `Searchtype` can be `bfs, dfs,` or `astar`.
- `inputfile` is the Cities.txt file.
- City names are the same as they appeared in the roads.txt file.
- `outputfile` is the name of an output file that will collect a log of the search process. The output file should include a record every time a node is expanded and added to the queue. Once the search is complete write the final search path and the remaining contents of the queue to the file in a human-readable form.

Experiment by executing your code and answer the following questions:

1. Consider the path from vancouver to newOrleans. Run your algorithms and show the nodes expanded by each one as well as the paths returned by each of the algorithms.
2. Is there a case where DFS outperforms BFS in terms of number of nodes expanded? If yes, what is the case? If not, explain why?
3. Is there a case where DFS outperforms BFS in terms of number of nodes on the solution path? If yes, what is the case? If not, explain why?
4. Are there any pairs of cities (A,B) for which the A* algorithm finds a different path from B to A than from A to B? If yes, what is the case? If not, explain why?
5. Is there a pair of cities for which DFS outperforms A* in terms of the number of nodes expanded? If yes, what is the case? If not, explain why?

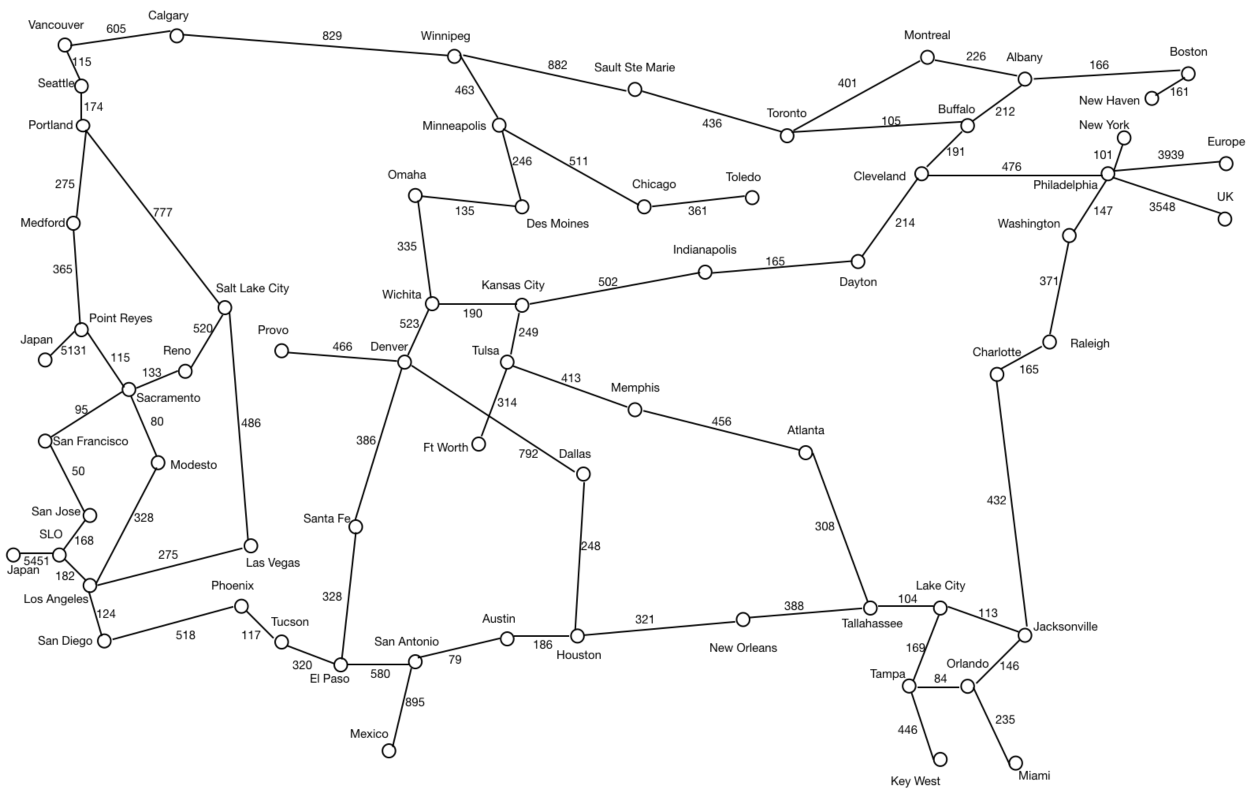

Figure 1: US roads map