1) For rules on collaboration and late policies, please see the course web page. The final problem involves programming.

a. On what sorts of graphs will DFS outperform BFS? On what sorts of graphs will BFS outperform DFS?

b. When would bi-directional BFS not be available as a search method? Give an example.

c. Consider a general search problem in which every node has:

- Depth $d(n)$: the length of the path (number of edges) to the node.
- $g(n)$: the cost of getting to the node.
- $h(n)$: the estimated cost of getting to the goal.

If you search using a function $f(n)$ to evaluate the "goodness" of n, and to decide which node to expand next, write expressions for $f(n)$ for the following types of search:

1. BFS
2. DFS
3. UCS
4. A*

2) Alice wants to run UCS on a graph where she knows some costs are negative or zero.

a. Why doesn't UCS work on Alice's graph?

b. Now assume that Alice knows the cost on any edge is bounded below by $\underline{c}$. She proposes running UCS on the graph, but modifying the algorithm to add $\underline{c} + 1$ to the cost of each edge, transforming the graph into one where all the edges are positive. Is this complete? Is it optimal? Why or why not?

c. Does your answer change if $\underline{c}$ is exactly equal to the minimum cost edge?

3) The London Underground ("Tube") is a subway and light rail system consisting of 306 stations and 13 different lines. In this problem, you will create a simplified subway planning system for the Tube that relies on the search techniques you learned about in lecture.

The cost of a route is calculated as follows: Each stop along a line has a cost of one, and each transfer has a cost of one. So a route that involves traveling through five stops and transferring at one of those stops has a total cost of 5+1=6.

There are three comma-separated values files that contain the data, which you can find at `/afs/andrew/course/15/381-f09/hwone`.

- `stations.txt` has a list of stations, with a unique ID, latitude, longitude, and station name.
- `links.txt` has a list of two station IDs with the line ID that connects them.
- `lines.txt` maps line IDs to line names.

For informed search methods, try as a heuristic 100 times the Euclidean distance between the coordinates of the stops. It's worth noting that Euclidean distance is not a strictly accurate measure of real, earthly distance but works fine over small areas (like cities).

a. The file `partone.txt` contains a set of 3 trips. On each line, the IDs of the start and end station are separated by a space. For each trip and each search technique listed below, generate a table that describes the cost of the (first) route found and the number of nodes expanded.

Uninformed Search Techniques:

- Depth-first Search
- Breadth-first Search
- Iterative Deepening Search

Informed Search Techniques:

- Greedy Best-first Search
- A* Search

Interpret your findings. Which method produced the best results? How well did the informed search techniques do compared to uninformed search? Why do you think this is? Can you think of a better heuristic?

b. The file `parttwo.txt` contains a set of 50 trips in the same format as `partone.txt`. For each trip, calculate the optimal lowest-cost route and display it along with its cost in the following format:

```
Start at (start station)
Take (line name) to (intermediate station)*
Take (line name) to (final station)
(cost of optimal route)
```

As the following example should clarify:

```
EXAMPLE INPUT:
186 28
245 285
EXAMPLE OUTPUT:
Start at Notting Hill Gate
Take Central Line to Bond Street
4
Start at Stockwell
Take Northern Line to Waterloo
Take Jubilee Line to Westminster
5
```

Submit the following to
`/afs/andrew/course/15/381-f09/handin/(your andrew id)/hwone`:

- A copy of your answer, as `answer.txt`.
- The Andrew IDs of your team, as `team.txt`.