Intelligent Agents

Practical 4

PART 1 – Search

These questions are designed to ensure that you understand the core concepts of informed search. Please attempt the questions before the practical so that they can be discussed there.

1. A Wumpus World agent has explored a part of a 10x10 map and survived long enough to locate the gold. The goal is to now return to the start (1,1). The map is given below – white squares are explored, dark squares are not, and a 'B' means that a breeze was sensed in that square. The agent is in location (10,7).
    a. Draw up a decision graph/tree for the agent to return to the goal. You don't need to do all of it but want to be sure that you're comfortable with how this works.
    b. The problem is pathfinding but, depending on the ordering, Depth First search may work quite well. Consider how your ordering of notes in the graph would affect a DFS.
    c. Because we can estimate the distance to goal, we should really use informed search. How might you calculate a heuristic for this problem?
    d. Choose an appropriate heuristic and use that for the agent to plan their path back to the goal. Again, you don't need to do the complete search but want to do enough that you're comfortable with how it works.
    e. Would the agent be moving while this search is happening? Explain?

	1	2	3	4	5	6	7	8	9	10
1			B	▨	B	B	▨	B		
2				B	B		B			
3	B			B	▨			B	B	
4	▨	B		B	▨		B	▨	▨	B
5	▨	B		▨	▨		B	▨	B	
6	▨	B						B		
7	▨	▨	B							
8	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
9	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
10	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨

2. Consider again the graph from the previous worksheet with heuristic measures attached.
    a. Examine the heuristic measures given. Are these admissible? Consistent?
    b. Run an A* search on the graph, using the heuristic given.
    c. Compare the A* search to the DFS and BFS results from last week.

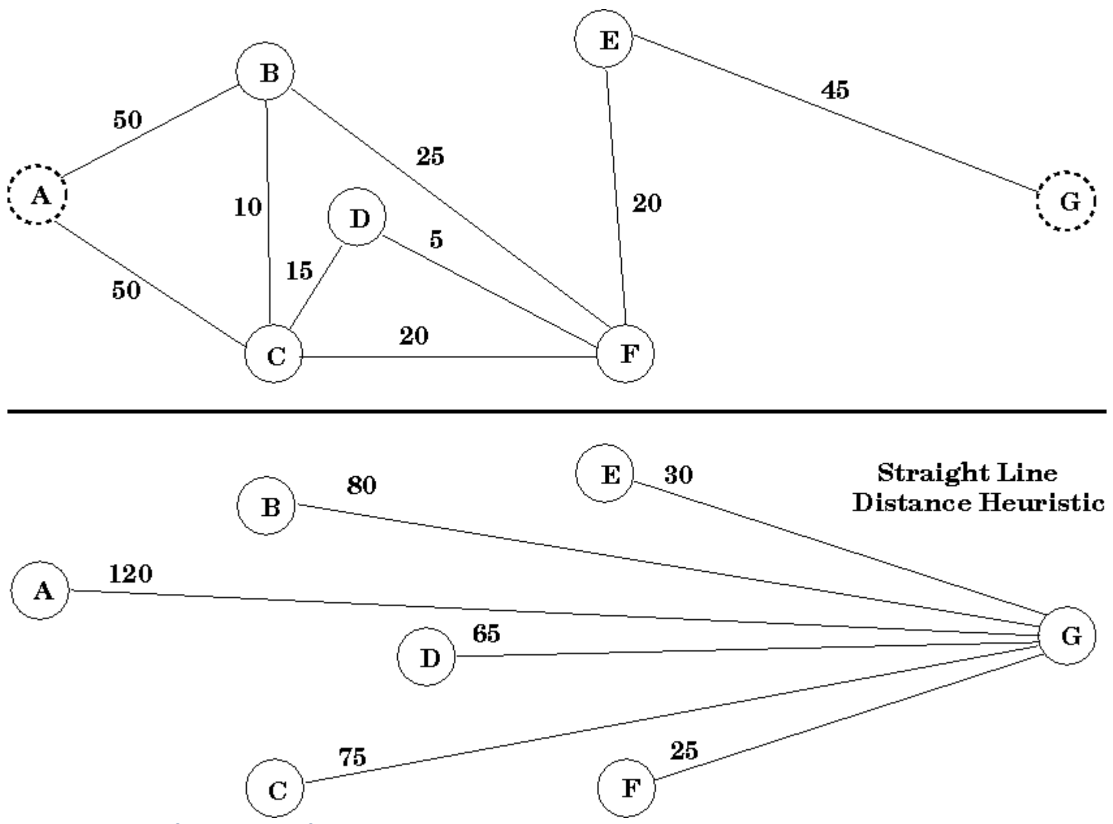

Figure 1: Graph for Q2, taken from COMP3006 material

PART 2 – Coding

Your task (which should actually be done after Part 3) is to code an A* search. Your program should do the following:

- Read in a graph file located in the same directory as the program (a sample file, G1.txt, can be found in the same folder as this worksheet).
- Read in a heuristic file located in the same directory as the program (a sample file, H1.txt, can be found in the same folder as this worksheet).
- Run an A* search.
    - Report each node added to the priority queue (by node name and f-cost) and each node expanded.
    - When you find the goal, report the path taken to get to the goal and the path cost.

PART 3 – Coding Study

Part of the assignment is going to be implementing search algorithms. That means that you need to be able to ingest a search graph to run the algorithms on (which means loading it and storing it in an appropriate structure). Last week you wrote some code that ingested a graph file. For informed search, you will also need a heuristic file.

The format of the heuristic file will be that each line of the file gives the details of one vertex of the graph. Each line has two positive integers, separated by spaces. The is the vertex and the second is the value of the heuristic at that vertex. Your program should verify that there is exactly one vertex with a heuristic value of zero and that the vertices in the heuristic file match those in the graph file.

An example of this is the file H1.txt, which contains the heuristics for Graph 1. This can be found in the same folder as this worksheet.

There is no restriction on what languages that you use and what libraries you use except that the result must work in the lab computers in 314.219. If you use code from another source, please use appropriate code referencing.

Bonus Coding Competition:

There is no practical next week, but here's a challenge for you. Find a way to count the number of different acyclic paths from a source to a goal in a path. In other words, given a graph, a source (usually 0) and a goal (usually the node with the highest label), find all possible sequences v_0 v_1 v_2 …. v_{n-1} v_n where v_0 is the source node, v_n is the goal node and each pair of adjacent vertices in the sequence (v_x and v_{x+1}) are connected by an edge in the graph and no vertex appears twice in the sequence.

For example, graph 1 has the following valid sequences:

- A B F E G
- A C F E G
- A B C F E G
- A C B F E G
- … and so on

The aim is to write some code that will ingest a network file and return a number equal to the number of acyclic paths. Graph 1 has six such paths.

On the Thursday during the study break I will release a graph file at 10am that will have around 20 vertices and enough connections to ensure that the problem can't be easily solved by hand. The first e-mail I receive with the correct number of paths in the file will get a small (chocolate-based) reward and my congratulations. If I haven't received a correct answer by the start of the next lecture, I will take the closest attempt. If I have received no attempts I'll keep the chocolate.

Note – this may sound like an easy problem, but it has its complications. Consider how you can use searches that we've covered already to solve it.