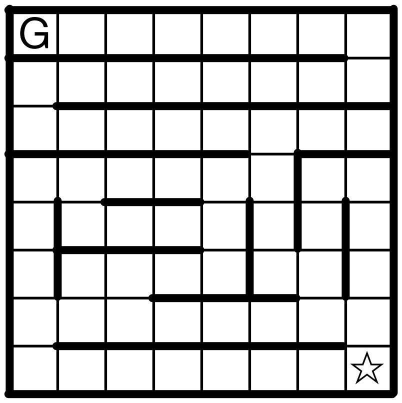
**Artificial Intelligence Homework 1**

If you have not already done so, read textbook sections 3.1-3.6.

1. Imagine an agent is stuck in a maze like this one:



The star represents the agent's starting location, and the G represents the goal location the agent is trying to reach. Assume the agent starts in the maze with a certain amount of *energy*, given by the variable $E_{start}$.

At any time step, the agent has two movement options. The agent may choose to ***walk*** one square in any direction (up, down, left, right; not diagonally) that is not blocked by a wall (this takes one unit of time). This does not use up any of the agent's energy. The agent may also choose to ***jump*** two squares in any single direction (up, down, left, right; not diagonally) that is not blocked by a wall (also takes one unit of time). However, when the agent jumps, it uses up one unit of energy (which it can never get back). If the agent's energy ever drops to zero, it may not jump anymore; it must walk the rest of the way. Furthermore, the agent cannot jump twice in a row (it may jump, then walk, then jump, but never jump then jump).

The agent's goal is to find a sequence of actions that ends at the goal location, while minimizing the total time it takes. The final amount of energy the agent has is irrelevant.

*Note that this is problem formulation is different from the regular navigation problem!*

All of these questions pertain to a generic maze of size *m* by *n*, not specifically the maze above, which is only shown to illustrate how the problem is set up.

(a) Formulate what a state looks like for this problem. Define it using specific data types.

(b) If the grid is *m* by *n*, what is the (maximum) size of the state space? In other words, come up with the tightest upper bound you can for how many states there are. Justify your answer. You should assume that all possible states are reachable from the start state.

(c) What is the maximum branching factor of this problem? Briefly justify your answer.

(d) The *Manhattan distance* between two points is defined to be the sum of the total vertical and horizontal differences between the points. In other words, if you imagine a city laid out using a grid (like Manhattan in New York City), the Manhattan distance is the total distance you would have to walk to get from one intersection to another (because no streets run diagonally).

For this problem, is the Manhattan distance from the agent's location to the goal location an admissible heuristic? Why or why not?

(e) Describe and justify a non-trivial admissible heuristic for this problem that is not the Manhattan distance to the goal.

(f) If we used an inadmissible heuristic to solve this problem, could it change the completeness of the search? Why or why not?

(g) If we used an inadmissible heuristic to solve this problem, could it change the optimality of the search? Why or why not?

2. Assume you are given a graph like the one we used in class for A*, with vertices representing locations and the edge weights representing travel time between locations. (That graph is supplied for you at the end of this homework to refresh your memory.) The *travelling salesperson problem* asks, "What is the fastest route that begins at a specific starting location, visits all the other locations in the graph in any order, and returns back to the starting location?" Suppose we want to solve this problem using A*.

   (a) Formulate how a state could be represented for this problem. Define this representation using specific data types.

   (b) What does the initial state look like, and what does the goal state(s) look like? (If it helps to use specific examples from the in-class graph, you may.) Hint: You will need to represent/store more than just the current location of the salesperson.

   (c) Suppose I define a heuristic for this problem to be the sum of all the straight-line distances from the salesperson to all the locations the salesperson hasn't visited yet. Explain why this heuristic is not admissible.

   (d) Define a non-trivial admissible heuristic for this problem. (Non-trivial means you must make a reasonably intelligent heuristic; you can't just pick *h(n)* = 0 or something similar.)

**3.** In the 3-disc Tower of Hanoi puzzle, you are given three pegs, labeled A, B, and C, and three discs, labeled D1, D2, and D3. D1 is larger than D2, and D2 is larger than D3. The puzzle starts with all three discs on peg A, with D1 on the bottom, D2 resting on top of D1, and D3 resting on top of D2. The goal is to move all the discs onto peg C, with the caveats being you can only ever move the top disc on any peg, and a larger disc can never be placed on top of a smaller disc. In other words, D3 may rest directly on top of D2 or D1, but D2 can only rest directly on top of D1, and D1 can never rest on top of any other disc.

   We will represent our actions using the following notation: MOVE(x, y, z) where x is the name of a disc, y is the peg where disc x is located (on top), and z is the peg where disc x is being moved to.

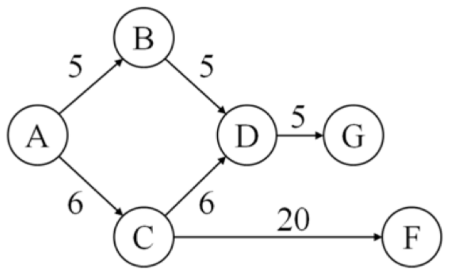
   (a) Define a state representation using specific data types.

   (b) What does the initial state look like? What does the goal state(s) look like?

   (c) Draw the state space as a directed graph, showing which states are reachable from which other states. Only include states that are reachable in at most three moves from the initial state. The vertices of the graph are the states, and the edges of the graph are the moves. Label each edge with the name of the move (as in MOVE(x, y, z)). Because each move is reversible, each edge should have a corresponding edge in the other direction, labeled with the "opposite" move. Note this is a *graph* of the search space, not a *search tree*.

(d) Invent a non-trivial admissible heuristic for this problem.

4. Consider the following graph, where we are trying to find the shortest path from vertex A to either F or G (assume both F and G are goal states).



Here are four possible heuristic functions h1(n) through h4(n):

| | A | B | C | D | F | G |
|---|---|---|---|---|---|---|
| h1 | 0 | 0 | 0 | 0 | 0 | 0 |
| h2 | 11 | 7 | 7 | 3 | 0 | 0 |
| h3 | 13 | 9 | 7 | 1 | 0 | 0 |
| h4 | 15 | 10 | 11 | 5 | 0 | 0 |

(a) Which, if any, of the heuristics are admissible?

(b) Which, if any, of the heuristics are consistent?

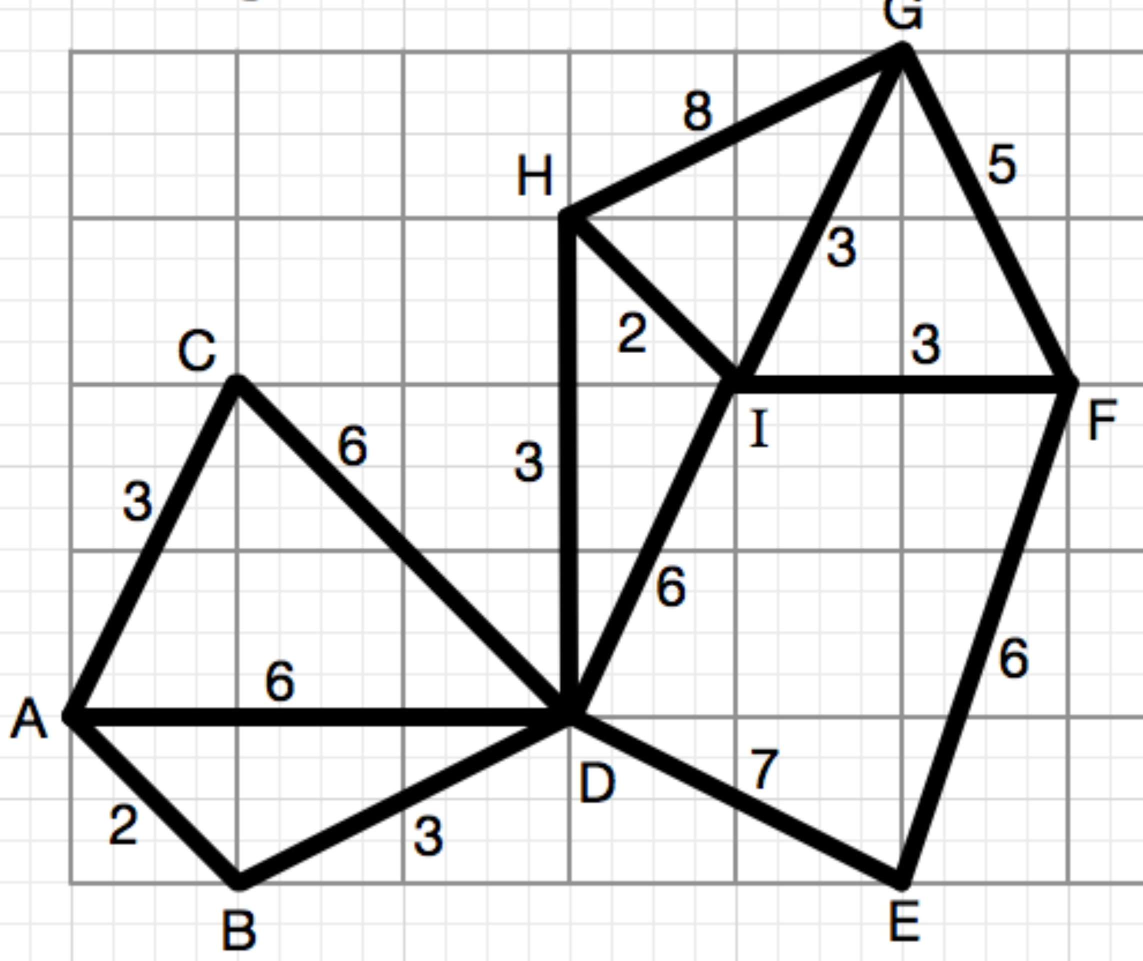
(c) Run A* (by hand) using h3, showing the search tree generated, and the frontier and the reached map. Make sure I can tell from the frontier in which order the states were added to it.

What is the best path returned by A*, and what it its f(n) cost?

(d) Recall that the term best-first search refers to any search algorithm that uses a heuristic function to prioritize the frontier by which nodes would be best to examine first. A* is an example of an algorithm that falls into this category. Consider a different algorithm that also falls into this category, called *greedy best-first search*. This algorithm sorts the frontier by f(n) = h(n), rather than f(n) = g(n) + h(n). Run greedy best first search by hand using h3, following the same procedure as part (c).

What is the best path returned by greedy best-first search and what is its f(n) cost?

# A* Algorithm



h(n) estimates for distance from n to A

| n | h(n) |
|---|---|
| A | 0 |
| B | 1.4 |
| C | 2.2 |
| D | 3 |
| E | 5.1 |
| F | 6.3 |
| G | 6.4 |
| H | 4.2 |
| I | 4.5 |