Q1. Search

Each True/False question is worth 1 points. Leaving a question blank is worth 0 points. **Answering incorrectly is worth** \(-1\) **points.**

(a) Consider a graph search problem where for every action, the cost is at least \(\epsilon\), with \(\epsilon > 0\). Assume the used heuristic is consistent.

(i) [true or false] Depth-first graph search is guaranteed to return an optimal solution.
False. Depth first search has no guarantees of optimality. Further, it measures paths in length and not cost.

(ii) [true or false] Breadth-first graph search is guaranteed to return an optimal solution.
False. Breadth first search has no guarantees of optimality unless the actions all have the same cost, which is not the case here.

(iii) [true or false] Uniform-cost graph search is guaranteed to return an optimal solution.
True. UCS expands paths in order of least total cost so that the optimal solution is found.

(iv) [true or false] Greedy graph search is guaranteed to return an optimal solution.
False. Greedy search makes no guarantees of optimality. It relies solely on the heuristic and not the true cost.

(v) [true or false] A* graph search is guaranteed to return an optimal solution.
True, since the heuristic is consistent in this case.

(vi) [true or false] A* graph search is guaranteed to expand no more nodes than depth-first graph search.
False. Depth-first graph search could, for example, go directly to a sub-optimal solution.

(b) Let \(h_1(s)\) be an admissible A* heuristic. Let \(h_2(s) = 2h_1(s)\). Then:

(i) [true or false] The solution found by A* tree search with \(h_2\) is guaranteed to be an optimal solution.
False. \(h_2\) is not guaranteed to be admissible since only one side of the admissibility inequality is doubled.

(ii) [true or false] The solution found by A* tree search with \(h_2\) is guaranteed to have a cost at most twice as much as the optimal path.
True. The heuristic could help to guide the search and reduce the number of nodes expanded. In the extreme case where the heuristic function returns zero for every state, A* and UCS will expand the same number of nodes. In any case, A* with a consistent heuristic will never expand more nodes than UCS.

Let \(h_1(s)\) be an admissible A* heuristic. Let \(h_2(s) = 2h_1(s)\). Then:

(i) [true or false] The solution found by A* tree search with \(h_2\) is guaranteed to be an optimal solution.
False. \(h_2\) is not guaranteed to be admissible since only one side of the admissibility inequality is doubled.

(ii) [true or false] The solution found by A* tree search with \(h_2\) is guaranteed to have a cost at most twice as much as the optimal path.
True. In A* tree search we always have that as long as the optimal path to the goal has not been found, a prefix of this optimal path has to be on the fringe. Hence, if a non-optimal solution is found, then at time of popping the non-optimal path from the fringe, a path that is a prefix of the optimal path to the goal is sitting on the fringe. The cost \(\bar{g}\) of a non-optimal solution when popped is its \(f\)-cost. The prefix of the optimal path to the goal has an \(f\)-cost of \(g + h_0 = g + 2h_1 \leq 2(g + h_1) \leq 2C^*\), with \(C^*\) the optimal cost to the goal. Hence we have that \(\bar{g} \leq 2C^*\) and the found path is at most twice as long as the optimal path.
(iii) [true or false] The solution found by A* graph search with \( h_2 \) is guaranteed to be an optimal solution.

False. \( h_2 \) is not guaranteed to be admissible and graph search further requires consistency for optimality.

(c) The heuristic values for the graph below are not correct. For which single state (S, A, B, C, D, or G) could you change the heuristic value to make everything admissible and consistent? What range of values are possible to make this correction?

State: B
Range: [2,3]
Q2. Hive Minds: Redux

Let’s revisit our bug friends. To recap, you control one or more insects in a rectangular maze-like environment with dimensions $M \times N$, as shown in the figures below. At each time step, an insect can move North, East, South, or West (but not diagonally) into an adjacent square if that square is currently free, or the insect may stay in its current location. Squares may be blocked by walls (as denoted by the black squares), but the map is known.

For the following questions, you should answer for a general instance of the problem, not simply for the example maps shown.

You now control a pair of long lost bug friends. You know the maze, but you do not have any information about which square each bug starts in. You want to help the bugs reunite. You must pose a search problem whose solution is an all-purpose sequence of actions such that, after executing those actions, both bugs will be on the same square, regardless of their initial positions. Any square will do, as the bugs have no goal in mind other than to see each other once again. Both bugs execute the actions mindlessly and do not know whether their moves succeed; if they use an action which would move them in a blocked direction, they will stay where they are. Unlike the flea in the previous question, bugs cannot jump onto walls. Both bugs can move in each time step. Every time step that passes has a cost of one.

(a) Give a minimal state representation for the above search problem.

A list of boolean variables, one for each position in the maze, indicating whether the position could contain a bug. You don’t keep track of each bug separately because you don’t know where each one starts; therefore, you need the same set of actions for each bug to ensure that they meet.

(b) Give the size of the state space for this search problem.

$2^{MN}$

(c) Give a nontrivial admissible heuristic for this search problem.
$h_{\text{friends}} = \text{the maximum Manhattan distance of all possible pairs of points the bugs can be in.}$
Q3. Search Traces

Each of the trees (G1 through G5) was generated by searching the graph (below, left) with a TREE search algorithm. Assume children of a node are visited in alphabetical order. Each tree shows only the nodes that have been expanded. Numbers next to nodes indicate the relevant “score” used by the algorithm’s priority queue. The start state is A, and the goal state is G.

For each tree, indicate:

1. Whether it was generated with depth first search, breadth first search, uniform cost search, or \( A^* \) search. Algorithms may appear more than once.

2. If the algorithm uses a heuristic function, say whether we used

   \[
   H_1 = \{ h(A) = 3, h(B) = 6, h(C) = 4, h(D) = 3, h(G) = 0 \} 
   \]

   \[
   H_2 = \{ h(A) = 3, h(B) = 3, h(C) = 0, h(D) = 1, h(G) = 0 \} 
   \]

3. For all algorithms, say whether the result was an optimal path (assuming we want to minimize sum of link costs). If the result was not optimal, state why the algorithm found a suboptimal path.

Please fill in your answers on the next page.
(a) G1:
1. Algorithm: Breadth-First search
2. Heuristic (if any): None
3. Did it find least-cost path? If not, why? No. The least-cost solution is A-D-C-G. Breadth-first search will only find a path with the minimum number of edges. It does not consider edge cost at all.

(b) G2:
1. Algorithm: A* search
2. Heuristic (if any): H1
3. Did it find least-cost path? If not, why? No. A* tree search is only guaranteed to find an optimal solution if the heuristic is admissible. H1 is not admissible.

(c) G3:
1. Algorithm: Depth-First Search
2. Heuristic (if any): None
3. Did it find least-cost path? If not, why? No. Depth first search simply finds any solution - there are no guarantees of optimality.

(d) G4:
1. Algorithm: A* search
2. Heuristic (if any): H2
3. Did it find least-cost path? If not, why? Yes. H2 is an admissible heuristic; therefore, A* tree search finds the optimal solution.

(e) G5:
1. Algorithm: Uniform Cost Search
2. Heuristic (if any): None
3. Did it find least-cost path? If not, why? Yes. Uniform cost (graph/tree) search is guaranteed to find a shortest-cost path.