CS 188
Spring 2020

Midterm Review Search

Q1. Search

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

(i) [*true* or *false*] Depth-first graph search is guaranteed to return an optimal solution.

(ii) [*true* or *false*] Breadth-first graph search is guaranteed to return an optimal solution.

(iii) [*true* or *false*] Uniform-cost graph search is guaranteed to return an optimal solution.

(iv) [*true* or *false*] Greedy graph search is guaranteed to return an optimal solution.

(v) [*true* or *false*] A* graph search is guaranteed to return an optimal solution.

(vi) [*true* or *false*] A* graph search is guaranteed to expand no more nodes than depth-first graph search.

(vii) [*true* or *false*] A* graph search is guaranteed to expand no more nodes than uniform-cost graph search.

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

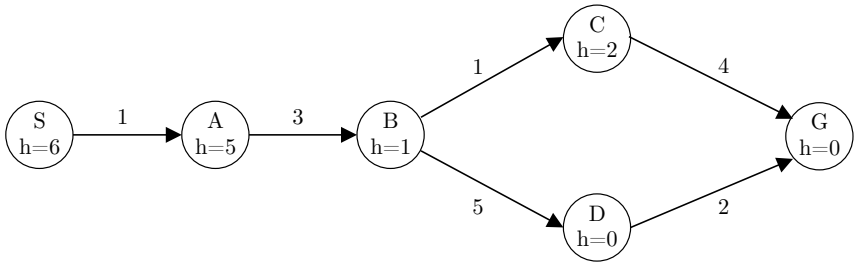
(i) [*true* or *false*] The solution found by A* tree search with h_2 is guaranteed to be an optimal solution.

(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.

(iii) [*true* or *false*] The solution found by A* graph search with h_2 is guaranteed to be an optimal solution.

(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?

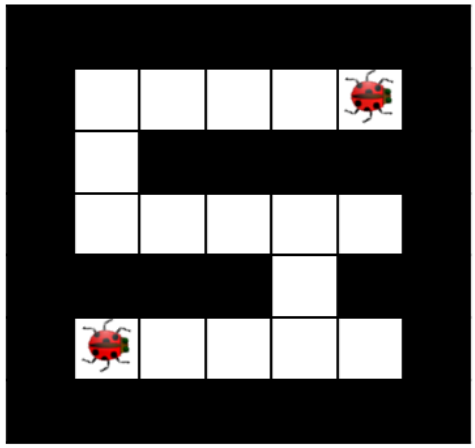
State: [ ] Range: [ ]



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.

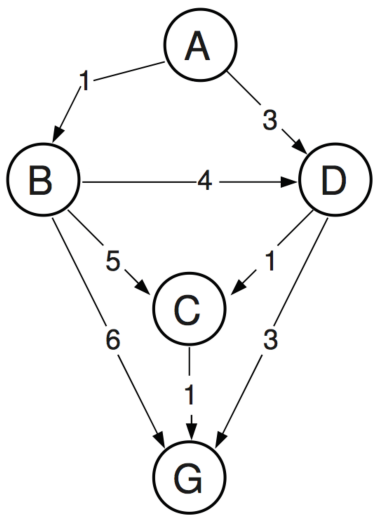
(a) Give a *minimal* state representation for the above search problem.

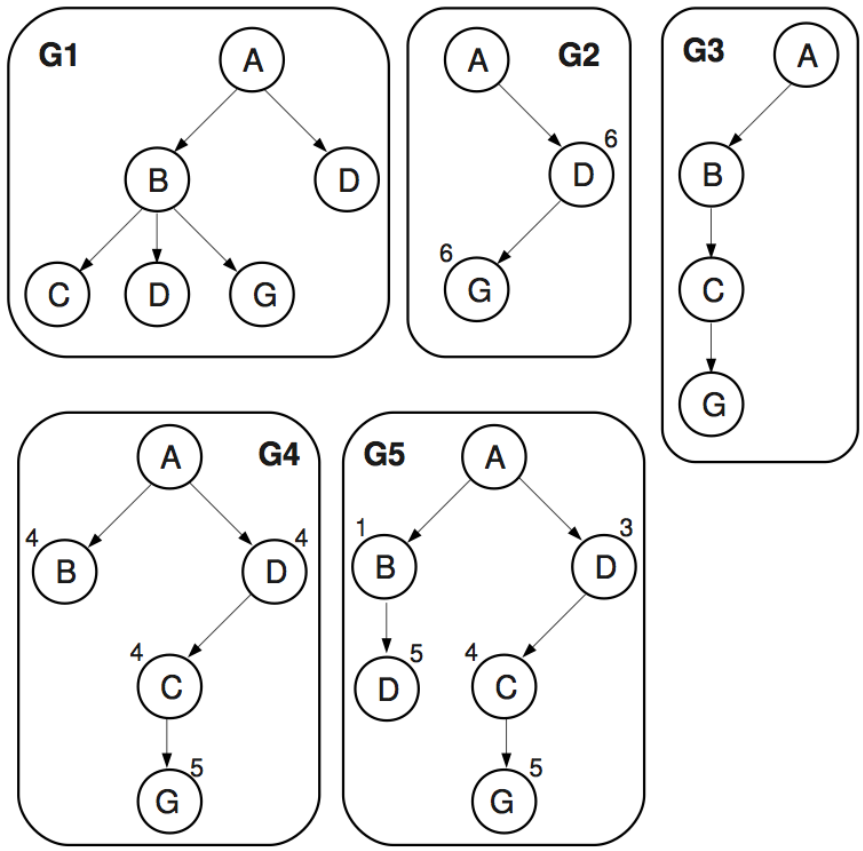
(b) Give the size of the state space for this search problem.

(c) Give a nontrivial admissible heuristic for this search problem.

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

   H1 $= \{h(A) = 3,\ h(B) = 6,\ h(C) = 4,\ h(D) = 3,\ h(G) = 0\}$

   H2 $= \{h(A) = 3,\ h(B) = 3,\ h(C) = 0,\ h(D) = 1,\ h(G) = 0\}$
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:
2. Heuristic (if any):
3. Did it find least-cost path? If not, why?

(b) **G2**:

1. Algorithm:
2. Heuristic (if any):
3. Did it find least-cost path? If not, why?

(c) **G3**:

1. Algorithm:
2. Heuristic (if any):
3. Did it find least-cost path? If not, why?

(d) **G4**:

1. Algorithm:
2. Heuristic (if any):
3. Did it find least-cost path? If not, why?

(e) **G5**:

1. Algorithm:
2. Heuristic (if any):
3. Did it find least-cost path? If not, why?