CSC384 Sample Questions

Summer 2019

1 Search

1. It would seem that iterative deepening search should have a higher asymptotic time complexity than breadth-first search because every time the depth-bound is increased it must start its search from scratch. However this is not true given the algorithms presented in class. Why?

2. If $h()$ is admissible and s is the start node, how is $h(s)$ related to the cost of the solution found by A* search?

3. What happens if we use a heuristic $h()$ in A* search that does not have the guarantee that $h(n) \leq h^*(n)$ for all states n?

4. How do depth-first, breadth-first and depth-first with iterative deepening search compare in terms of their asymptotic time complexity? In terms of their asymptotic space complexity? Please indicate any assumptions you are using (i.e. justifications from the textbook, from other sources, etc).

5. Prove that if $h(n) = h^*(n)$ for all n, then whenever A^* expands a node n, n must lie on an optimal path to a goal.

6. Let h be an admissible function and let $f(n) = w * g(n) + (1 - w) * h(n)$ for $0 <= w <= 1$. Will A* find an optimal solution when $w = 1$?, $w = 1/2$?, $w = 3/4$?

7. Consider the problem of finding a path in the grid shown below from the position s to the position g. The robot can move on the grid horizontally and vertically, one square at a time (each step has a cost of one). No step may be made into a forbidden shaded area. The search space and the shaded areas are illustrated in Figure 1.

   - On the grid, number the nodes in the order in which they are removed from the frontier in a depth-first search from s to g, given that the order of the operators you will test is: up, left, right, then down. Assume there is a cycle check.
   - Number the nodes in order in which they are taken off the frontier for an A* search for the same graph. Manhattan distance should be used as the heuristic function. That is, $h(n)$ for any node n is the Manhattan distance from n to g. The Manhattan distance between two points is the distance in the x-direction plus the distance in the y-direction. It corresponds to the distance traveled along city streets arranged in a grid. For example, the Manhattan distance between g and s is 4. What is the path that is found by the A* search?

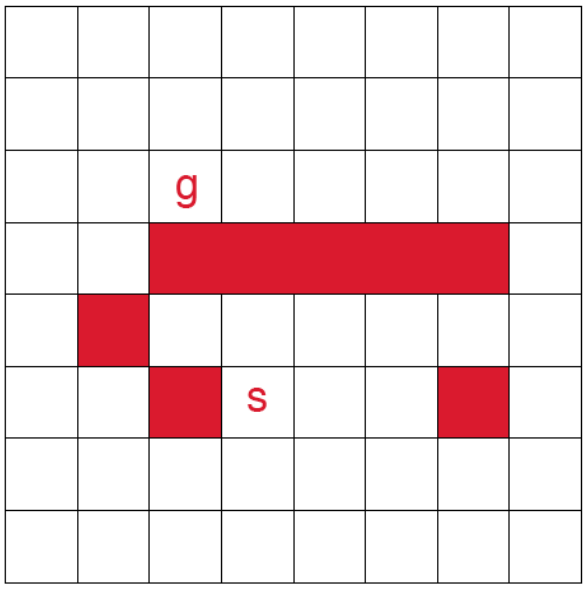
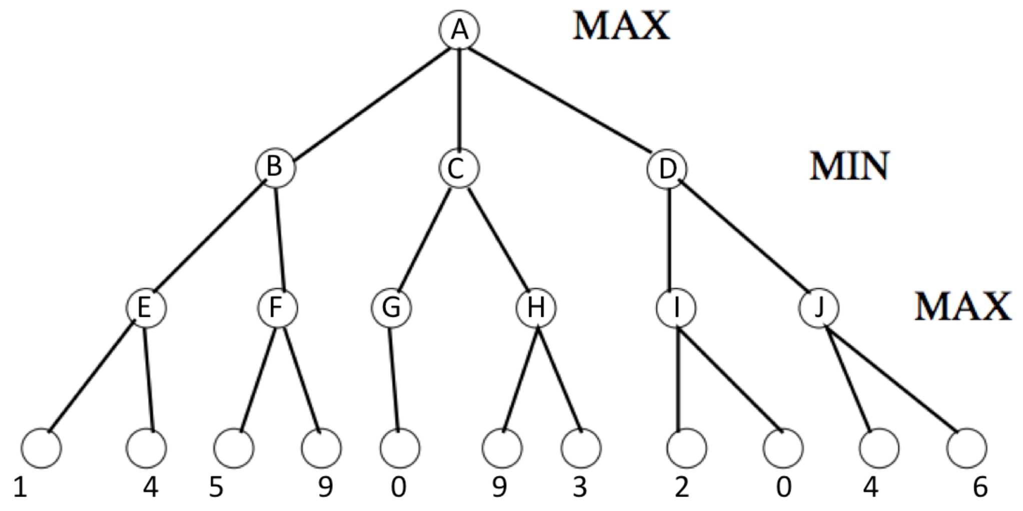


Figure 1: Search Space

- Suppose that the graph extended infinitely in all directions. That is, there is no boundary, but s, g, and the forbidden area are in the same relative positions to each other. Which search methods would no longer find a path? Would A*, or would depth-first search? Which would be the better method, and why?

2 Game Tree Search

1. Consider the following game tree, where the utility of each terminal node is specified.



   (a) For each min and max node, list its minimax value in the space below (one mark off for each incorrect answer):

A	B	C
D	E	F
G	H	I
J		

   (b) On the diagram, put a check besides the nodes that will be visited during a minmax depth-first search that uses *alpha-beta* pruning. You should also mark any terminal node that is visited. Draw a line across each edge that leads to a sub-tree pruned by an alpha or beta bound.

   (c) What value does player MAX expect to get out of the game if MIN plays perfectly?

   (d) What value does player MAX expect to get out of the game if MIN always moves to state E from state B?

   (e) What value does player MAX expect to get out of the game if MIN always moves to state F from state B?

2. What is the difference in the definition of a heuristic value for a game state, and for a state in A* search? What properties make a heuristic in either situation 'good'?

3. Consider an evaluation function that returns the square of a state's true minimax value. Consider three cases:

   (i) In planning a move (or strategy) for a certain state S, the agent does not hit any terminal state.

   (ii) In planning a move (or strategy) for a certain state S, the agent only hits a terminal state.

   (iii) In planning a move (or strategy) for a certain state S, the agent sometimes hits the terminal state.

   Answer the following questions:

   (a) Will the search result in the same strategy in cases (i) and (ii)? [Same/Not Same]
   (b) Will the search result in the same strategy in cases (i) and (iii)? [Same/Not Same]
   (c) Will the search result in the same strategy in cases (ii) and (iii)? [Same/Not Same]

4. Assume you have come up with a minimax policy after traversing to all the terminals of a game tree. Someone then comes along and doubles the value of every terminal over some threshold. Will you come up with the same policy if you run minimax again? What if someone doubles only values of terminals that are even?

5. Why is minimax search generally not used to play real games?

6. True or False: When executing the alpha-beta algorithm on a game tree which is traversed from left to right, the leftmost branch will never be pruned.

7. True or False: The alpha-beta algorithm will always result in at least one branch of the game tree being pruned.

8. Consider a game tree constructed for our Pacman game, where b is the branching factor and where depth is greater than d. Say a minimax agent (without alpha-beta pruning) has time to explore all game states up to and including those at level d. At level d, this agent will return estimated minimax values from the evaluation function.

   (a) In the best case scenario, to what depth would alpha-beta be able to search in the same amount of time?

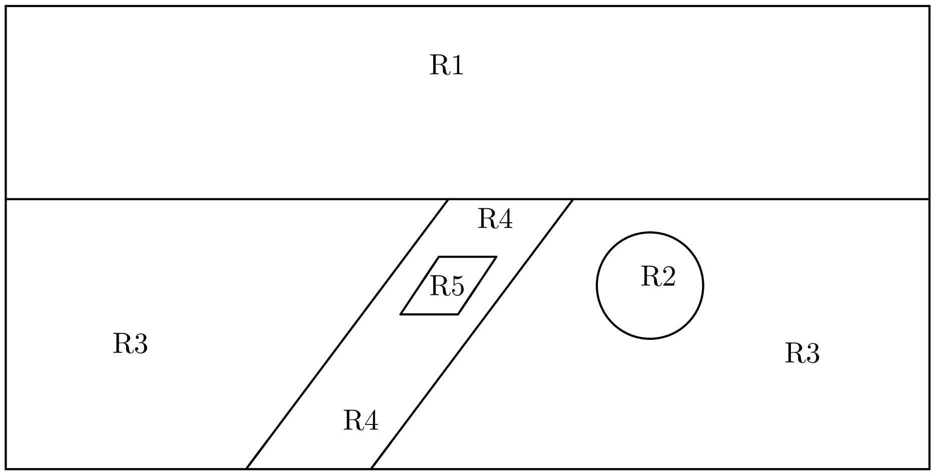
   (b) In the worst case scenario, to what depth would alpha-beta be able to search in the same amount of time? How might this compare with the minimax agent without alpha-beta pruning?

9. **True or False:** Consider a game tree where the root node is a max agent, and we perform a minimax search to terminals. Applying alpha-beta pruning to the same game tree may alter the minimax value of the root node.

3 Constraint Satisfaction Problems

1. Consider the map shown below. There are five regions, and the following information is available about what they can represent.

   (a) R1 can only be the sky or grass,
   (b) R2 can only be trees or a road,
   (c) R3 can only be grass or trees,
   (d) R4 can only be road or grass,
   (e) R5 can only be a car.



   Furthermore, we have the following knowledge about the real world, and the map:

   (a) A car cannot be next to grass.
   (b) No two neighboring regions can be the same.
   (c) Only one region is a road.

   The problem is to find what each region in the figure represents.

   (a) Represent the above problem as a CSP. You must specify the meaning of each variable assignment. (That is, in our representation, how can one read off a solution to the problem from an assignment of values to the variables).
   (b) Apply forward checking to solve the problem. Use the heuristic of always assigning next the variable with fewest remaining values (you can break ties as you choose). You *do not* need to show the updated current domains, only show the nodes of the search tree visited by forward checking and the assignments made at those nodes.
   (c) How many solutions are there, and what are they?

2. Consider a CSP with the following variables and constraints:

   - Variables A, B, C, D, E with all variables having the domain $1, 2, 3, 4$

- Constraints:
  - $E - A$ is even.
  - $C \neq D$
  - $C > E$
  - $C \neq A$
  - $B > D$
  - $D > E$
  - $B > C$

Draw these variables and constraints as a constraint graph. Then, perform GAC-Enforce on the graph. Write the domains for each variable that exist after you've completed GAC-Enforce.