CSC384 - Introduction to Artificial Intelligence
Sample Problems
(Winter 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. Why?

   If the children of breadth first search are expanded as BFS executes, the order of BFS will be $O(b^{d+1})$. Iterative deepening may execute many repeated searches at each iteration, but none will be more than $O(b^d)$. A fixed number of searches that are each bound by $O(b^d)$ takes $O(b^d)$ time to run, collectively.

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?

   $h(s)$ must be less than $h^*(s)$ where $h^*(s)$ is the true cost of the path from s to the goal.

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?

   If it is not the case that $h(n) \leq h^*(n)$ for all nodes, our search will favor some nodes for expansion that are on suboptimal paths to the goal over nodes that are on optimal paths to the goal. This means A-star search will no longer be guaranteed to yield an optimal solution.

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).

   (a) *BFS: $O(d^{b+1})$ in both space and time (with some caveats. Assumes we expand nodes in layer d prior to discovering the goal. May however be $O(b^d)$ as per RN ed. 3. Also, this analysis assumes state space may not be able to be explicitly represented).*

   (b) *DFS: $O(b^{dMax})$ time but only $O(b * dMax)$ space.*

   (c) *UCS: $O(b^{C^*/e+1})$ space and time (with some caveats. Maintenance of ordered frontier adds to space and time complexity ... typically employs a priority queue (which takes logarithmic time to update). Additional caveat is that, if the state space can be explicitly*

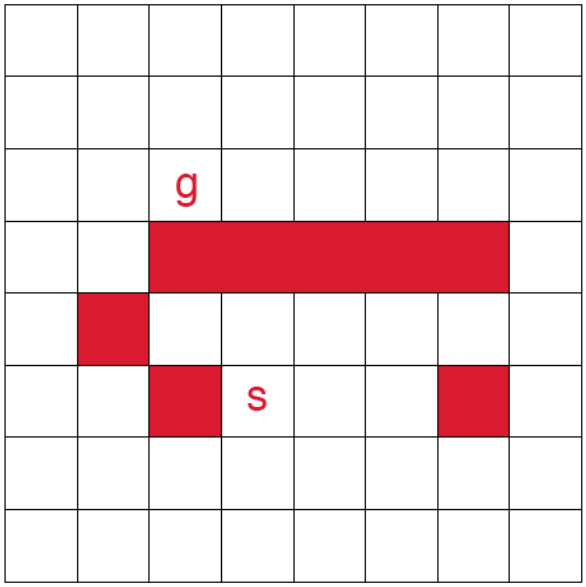


Figure 1: Search Space

represented, time and and space bounds can be reduced. Complexity can also be reduced if we institute goal tests prior to placing successors in the queue.

(d) Iterative Deepening: $O(d^b)$ time and $O(b * d)$ space.

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.

   Suppose $h(n) = h*(n)$ and let N be a node we just expanded on a path to goal that is sub-optimal, i.e. $f(N) = g(N) + h(N) > f*$. Now consider any node K on a path to an optimal goal that remains to be expanded. Because the node is still on the queue, we have $f(K) = g(K) + h(K) > f(N) = g(N) + h(N)$. But $f(K) = g(K) + h(K) = f < f(N)$. This leads to a contradiction. Hence, no node that is not on an optimal path will be expanded if $h = h*$.*

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? *Yes, Yes, Yes*

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.
     Refer to Figure 2.

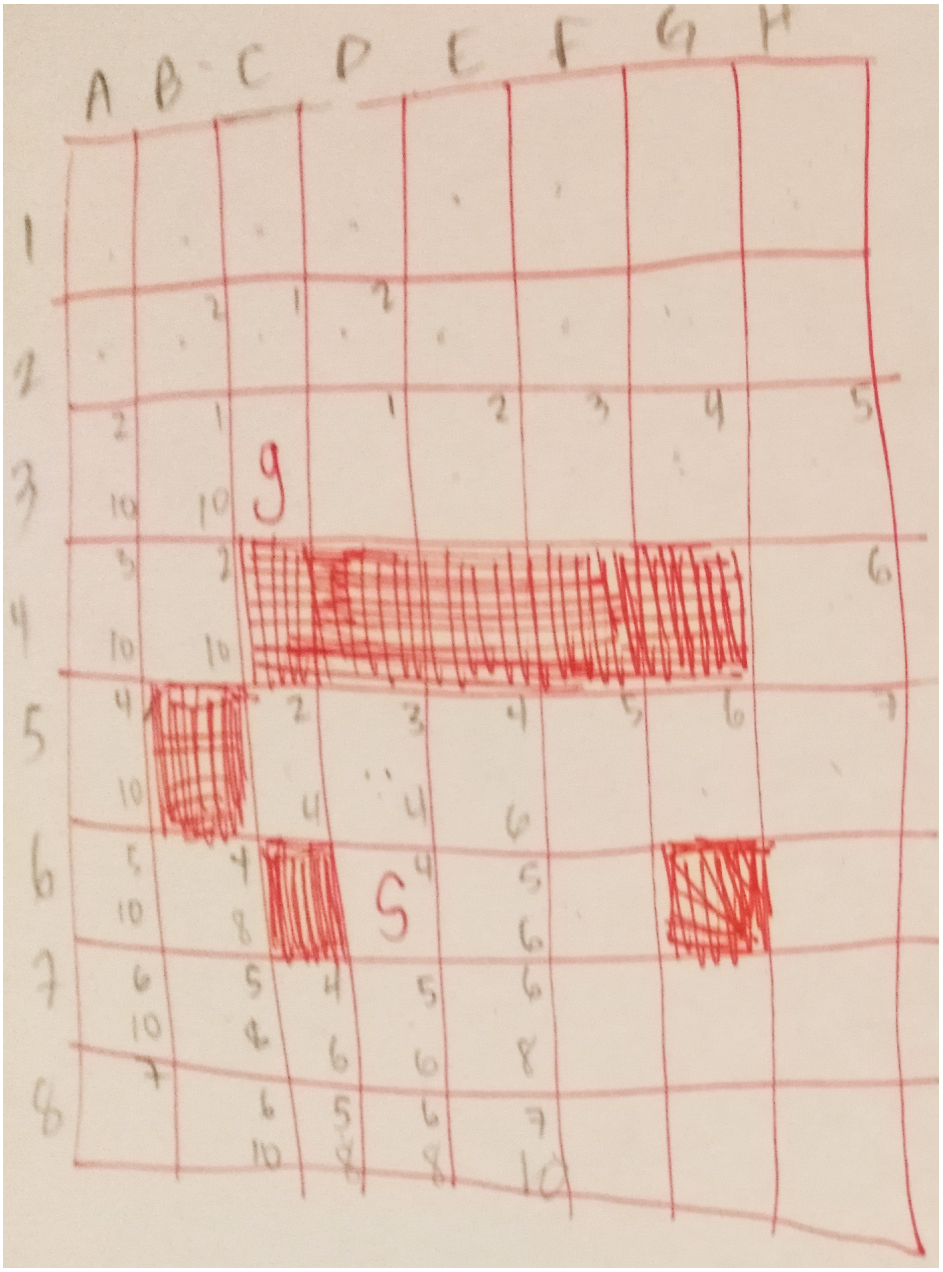


Figure 2: This illustrates the numbering scheme used for the solutions. Note that in the top right of each corner is the Manhattan Distance heuristic value to the goal. The bottom right numbers are "f-values" (i.e. the Manhattan Distance heuristic value + cost from the start.

D6,D5,C5,E5,D5,G5,H5,H4,H3,H2,H1,G1,
F1,E1,D1,C1,B1,A1,A2,B2,C2,D2,E2,F2,
G2,G3,F3,E3,D3,C3.

- 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?

  Refer to Figure 2.

  Order of nodes pulled off frontier depends and will vary depending on how people choose to break ties between equal f-values.

  first f-contour (value is 4) (nodes are first to be expanded): D6,D5

  second f-contour (value is 6) (nodes are second to be expanded): C5,E5,E6,D7,C7

  third f-contour (value is 8) (nodes are next to be expanded): E7,D8,C8,B7,B6

  final f-contour (value is 10) (nodes are next to be expanded): E8,B8,A7,A6,A5,A4,B4,B3,A3,C3

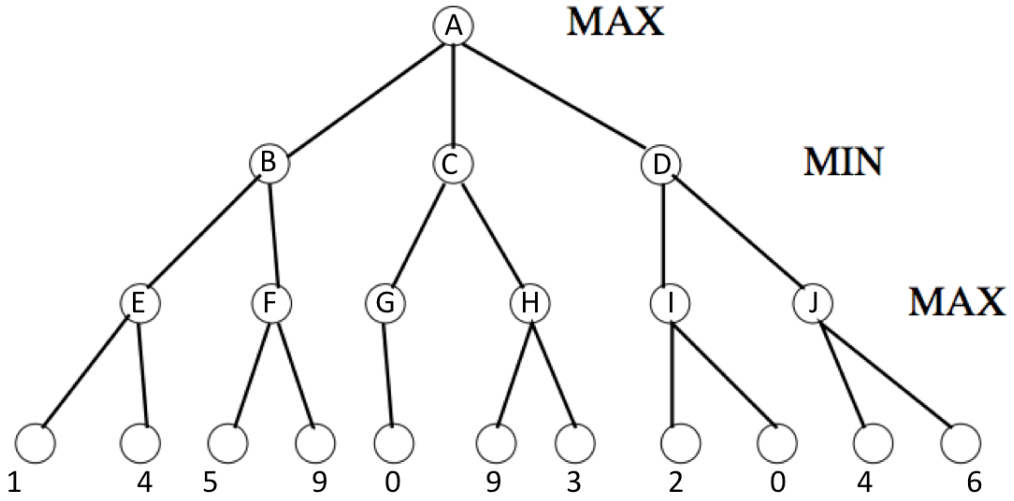
  path discovered: D6,D7,C7,B7,B6,A6,A5,A4,B4,B3,C3

- 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?

  DFS will travel infinitely off the board at the upper right corner. It will not find a path and is therefore not the best method to use. A, by contrast, will be drawn to states that are relatively close to the goal (i.e. that have a low f-value). It will find the optimal path, even if the board is infinite.*

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	4	B	4	C	0
D	2	E	4	F	9
G	0	H	9	I	2
J	6				

(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. *You can prune 9 (the child of F). You can also prone H and J entirely.*

(c) What value does player MAX expect to get out of the game if MIN plays perfectly? *4*

(d) What value does player MAX expect to get out of the game if MIN always moves to state E from state B? *4*

(e) What value does player MAX expect to get out of the game if MIN always moves to state F from state B? *9*

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'? *A heuristic function for A* estimates the cost that will be accrued to get to a node from a given state. A heuristic for a game estimates payoff to be derived at an underlying terminal. Good A* heuristics may be admissible; good games heuristics are accurate and orders terminals appropriately*

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]

   Not Same, Not 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? *No, only linear transforms that are applied to all terminals will preserve order.*

5. Why is minimax search generally not used to play real games? *Resuires traversing entire tree to terminals, which may be unfeasible if game is large (big branching factor and/or depth)*

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. *True*

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

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? *2d*

   (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? *same depth as minimax, in worst case ordering*

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. *False*