CS 411: Artificial Intelligence I
Written Homework 1 Solutions
Dues: Sunday of Week 2 11:59 PM

You may discuss the assignment with other students, but if you do you must note on your submission who you discussed it with. The actual submission must be entirely your own work. It must be submitted via gradescope. Please make sure to tag which page(s) each problem is answered on at the appropriate step in the submission process.

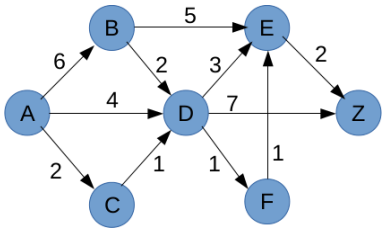
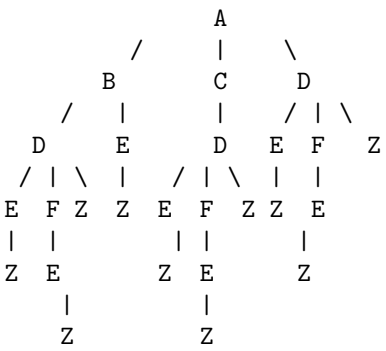


Figure 1: Search cost graph.

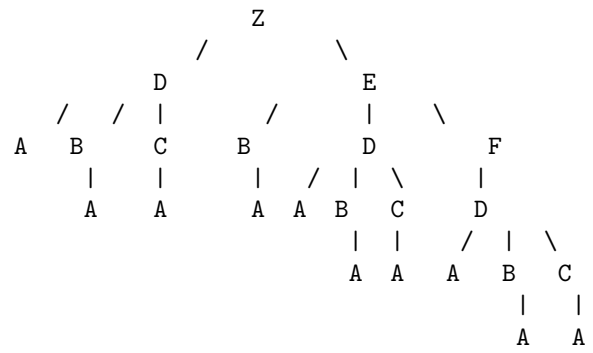
1. **Search tree** Draw the complete search tree for this graph starting from A. Please list children in the tree in alphabetical order from left to right.



2. **Breadth first tree search.** Indicate which nodes of the search tree will be expanded using breadth first tree search from A to Z.

   A, AB, AC, AD, ABD, ABE, ACD, ADE, ADF, ADZ

3. **Reverse search tree** Sometimes searching backward can be more efficient. Reverse the direction of the edges of the search cost graph and show the resulting search tree from Z to A.



4. **Reverse uniform cost tree search** Indicate which nodes of the reverse search tree will be expanded using uniform cost tree search from Z to A.

   Z, ZE, ZEF, ZEFD, ZED, ZEFDC, ZEFDB, ZEDC, ZD, ZEB, ZEDB, ZEFDCA

5. **Bidirectional uniform cost tree search.** Another useful trick is to search forward (A to Z) and backward (Z to A) simultaneously until a node is expanded by one search while some other path to that node is in the fringe of the other search. This idea can be implemented by using a single UCS priority queue and starting with the root node for each search tree enqueued. Show the portions of the forward and backward search trees that are explored to find this common node.

   In addition to the tie-breaking about which order to visit nodes with the same priority during a search, we now need to decide which search should expand a node first when both searches have the same priority for the next node. In this example the difference comes when we need to start expanding nodes with cost 3. If we have the forward search go first we get

   Forward: A, AC, ACD, ACDF*

   Reverse: Z, ZE, ZEF

   while if we have the reverse search go first we get

   Reverse: Z, ZE, ZEF, ZEFD*

   Forward: A, AC, ACD

   Thus this decision can actually change the node at which the searches meet.

   Another tricky point is when to stop the search. We are doing tree search, so the only information we have available is our fringe. So the check somehow needs to look for an intersection between the fringes. This could be done either at the time a node is selected for expansion or when nodes are added to the fringe. The above assumes the former, and so the starred nodes are not actually expanded but rather the node found in the fringe of the other search.

   Because there are a number of choices that could be made in this problem, a number of variants of the above solution are also acceptable.

6. **Iterative Deepening Search** Prove that if there is a solution at depth d and the branching factor of the search tree is b then iterative deepening search expands $O(b^d)$ nodes

   Iterative deepening search will run depth first search d times, once for each non-zero depth until the solution is reached. It visits the nodes at depth 0 and 1 all d times, those at depth 2, $d-1$ times, and the nodes at depth i $d-i+1$ times. Thus the total number of expansions is at most $d+\sum_{i=1}^{d}(d-i+1)b^i$. As this is a polynomial in b, only the highest order term matters, which is the $i = d$ term, $(d-d+1)b^d = b^d$. Thus, the algorithm expands $O(b^d)$ nodes.

Grading standards:

- Full Credit (1.0) - All questions attempted, most with minor or no errors
- Partial Credit (0.5) - A number of questions may have significant errors
- Some Credit (0.25) - Many questions attempted with answers demonstrating some understanding of relevant concepts
- Insufficient evidence (0) - Many questions either not attempted or with answers not demonstrating understanding of relevant concepts