2. (25 pts total, -5 pts each error, but not negative) Search Properties. Fill in the values of the four evaluation criteria for each search strategy shown. Assume a tree search where $b$ is the finite branching factor; $d$ is the depth to the shallowest goal node; $m$ is the maximum depth of the search tree; $l$ is the depth limit; step costs are identical and equal to some positive $\epsilon$; in bidirectional search both directions use breadth-first search. Note: These assumptions are the same as in Figure 3.21 of your textbook.

Note: These assumptions are the same as in Figure 3.21 of your textbook.

<table>
<thead>
<tr>
<th>Search Strategy</th>
<th>Complete?</th>
<th>Time complexity</th>
<th>Space complexity</th>
<th>Optimal?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Depth-First</td>
<td>No</td>
<td>$O(b^m)$</td>
<td>$O(bm)$</td>
<td>No</td>
</tr>
<tr>
<td>Breadth-First</td>
<td>Yes</td>
<td>$O(b^d)$</td>
<td>$O(b^d)$</td>
<td>Yes</td>
</tr>
<tr>
<td>Uniform-Cost</td>
<td>Yes</td>
<td>$O(b^{(1+\text{floor}(C*/\epsilon))})$</td>
<td>$O(b^{(d+1)})$ also OK</td>
<td>Yes</td>
</tr>
<tr>
<td>Depth-Limited</td>
<td>No</td>
<td>$O(b^l)$</td>
<td>$O(bl)$</td>
<td>No</td>
</tr>
<tr>
<td>Iterative Deepening</td>
<td>Yes</td>
<td>$O(b^d)$</td>
<td>$O(bd)$</td>
<td>Yes</td>
</tr>
<tr>
<td>Bidirectional (if applicable)</td>
<td>Yes</td>
<td>$O(b^{(d/2)})$</td>
<td>$O(b^{(d/2)})$</td>
<td>Yes</td>
</tr>
</tbody>
</table>

3. (30 pts total, -5 pts each error, but not negative) Reasoning about Search.

Note: Assumptions are DIFFERENT from problem 2 above. REASON about them.

Assume that you are doing Tree Search, the state space is infinitely deep, the branching factor is finite, there are cycles and loops, multiple goal nodes exist with different costs, step costs may differ from each other and are always greater than some given positive constant, in bidirectional search both directions use breadth-first search, and the heuristic function is consistent.

These assumptions represent a typical ill-conditioned search space.

3a. Is depth-first search complete? ___N___ optimal? ___N___ ("Y" = yes, "N" = no)
3b. Is breadth-first search complete? ___Y___ optimal? ___N___ ("Y" = yes, "N" = no)
3c. Is uniform-cost search complete? ___Y___ optimal? ___N___ ("Y" = yes, "N" = no)
3d. Is depth-limited search complete? ___N___ optimal? ___N___ ("Y" = yes, "N" = no)
3e. Is iterated-deepening search complete? ___Y___ optimal? ___N___ ("Y" = yes, "N" = no)
3f. Is bidirectional search complete? ___Y___ optimal? ___N___ ("Y" = yes, "N" = no)
3g. Is greedy best-first search complete? ___N___ optimal? ___N___ ("Y" = yes, "N" = no)
3h. Is A* search complete? ___Y___ optimal? ___Y___ ("Y" = yes, "N" = no)
4. (45 pts total, 9 pts each) Execute Tree Search through this graph (i.e., do not remember visited nodes). Step costs are given next to each arc. Heuristic values are given next to each node (as h=x). The successors of each node are indicated by the arrows out of that node. Successors are returned in left-to-right order. Specifically, the children of S are (A, B) and the children of B are (C, B), in that order.

For each search strategy below, show the order in which nodes are expanded (i.e., to expand a node means that its children are generated), ending with the goal node that is found. Show the path from start to goal, or write "None". Give the cost of the path found, if any. The first one is done for you, as an example.

4.a. DEPTH FIRST SEARCH.
Order of node expansion: S A C G
Path found: S A C G  Cost of path found: 25

4.b. (9 pts) BREADTH FIRST SEARCH.
Order of node expansion: S A B C G
Path found: S A C G  Cost of path found: 25

4.c. (9 pts) UNIFORM COST SEARCH.
Order of node expansion: S B C A B G
Path found: S B C G  Cost of path found: 14

4.d. (9 pts) GREEDY (BEST-FIRST) SEARCH.
Order of node expansion: S B B B B B B B ...
Path found: none Cost of path found: none

4.e. (9 pts) ITERATED DEEPENING SEARCH.
Order of node expansion: S S A B S A C G
Path found: S A C G  Cost of path found: 25

4.f. (9 pts) A* SEARCH.
Order of node expansion: S B C G
Path found: S B C G  Cost of path found: 14

Is the heuristic admissible? (Yes or No)  Y

Minor errors will receive partial credit. The Reader will determine how much credit to allow in any particular case.

Please see the lecture slides for Uninformed Search, topic “When to do Goal-Test? When generated? When popped?” for clarification about exactly what to do in practical cases.

BFS does the Goal-test before the child is pushed onto the queue. The goal is found when C is expanded.

IDS does the Goal-test before the child is pushed onto the queue. The goal is found when D is expanded.

UCS does goaltest when node is popped off queue.

A always has lower h(=4) than any other node on queue.

IDS does the Goal-test before the child is pushed onto the queue. The goal is found when D is expanded.

A* does goaltest when node is popped off queue.

See section 3.4.3.

See section 3.4.1.

See section 3.4.2.

See section 3.4.5.

See section 3.5.1.

See section 3.5.2.