CS-171, Intro to A.I., Winter Quarter, 2014 — Quiz # 1 — 20 minutes

1. NAME: _____

YOUR ID: _____ ID TO RIGHT: _____ ROW: _____ NO. FROM RIGHT: _____

2. (25 pts total, -5 pts each error, but not negative) Search Properties. Fill in the values of the four evaluation criteria for each searc[illegible] search where b is the finite branching factor; d is t[illegible] is the maximum depth of the search tree; l is the d[illegible] equal to some positive ε; in bidirectional search bo[illegible]

Your answer will be considered correct if it differs from that shown below by no more than ±1, e.g., O(b^d) vs. O(b^(d+1)).

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

	Complete?	Time complexity	Space complexity	Optimal?
Depth-First	No	O(b^m)	O(bm)	No
Breadth-First	Yes	O(b^d)	O(b^d)	Yes
Uniform-Cost	Yes	O(b^(1+floor(C*/ε))) O(b^(d+1)) also OK	O(b^(1+floor(C*/ε))) O(b^(d+1)) also OK	Yes
Depth-Limited	No	O(b^l)	O(bl)	No
Iterative Deepening	Yes	O(b^d)	O(bd)	Yes
Bidirectional (if applicable)	Yes	O(b^(d/2))	O(b^(d/2))	Yes

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? __Y__ ("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)

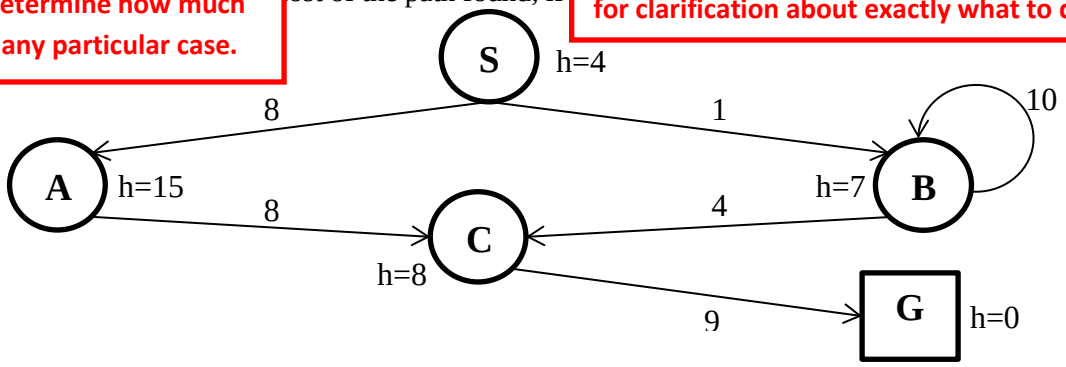
****** TURN PAGE OVER AND CONTINUE ON THE OTHER SIDE ******

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 ated), ending with the go ost of the path found, if

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.



4.a. DEPTH FIRST SEARCH.

See section 3.4.3.

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.

See section 3.4.1.

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

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.

UCS does goaltest when node is popped off queue.

Order of node expansion: S B C A B G

See section 3.4.2.

Path found: S B C G Cost of path found: 14

4.d. (9 pts) GREEDY (BEST-FIRST) SEARCH.

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

Order of node expansion: S B B B B B B B ...

See section 3.5.1.

Path found: none Cost of path found: none

4.e. (9 pts) ITERATED DEEPENING SEARCH.

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

Order of node expansion: S S A B S A C G

See section 3.4.5

Path found: S A C G Cost of path found: 25

4.f. (9 pts) A* SEARCH.

See section 3.5.2.

A* does goaltest when node is popped off queue.

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