CS-171, Intro to A.I., SS-1, 2018 — Quiz # 1 — 20 minutes

NAME: _____

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

1. (36 pts total, 12 pts each) SEARCH STRATEGIES AND THE FRONTIER. (Adapted from Poole, Mackworth, & Goebel, 1998.) This question asks you to think about search strategies and how they interact with the frontier (= fringe, open-list, or queue). Say that a search strategy is **"fair"** if any node on the frontier eventually will be expanded. Specifically, take a "snapshot" of the queue at any time *t* — then the strategy is **fair** if there is some later time *t'* such that every node in the snapshot taken at time *t* has been expanded by time *t'*. (Of course, if a goal node is found before time *t'* then the search will stop and return without expanding the remaining nodes on the queue, so we also will assume that no goal node is found before time *t'*.)

You are doing tree search, i.e., do not remember visited nodes. Recall that the branching factor *b* is always finite. Assume that all step costs are $\geq \varepsilon > 0$.

⇒ Mark X next to every fair search strategy in each condition below:

1.a. (12 pts total, 2 pts each) The search space is finite and has no loops.

__X__ Depth-first; __X__ Breadth-first; __X__ Uniform cost;

__X__ Iterated deepening; __X__ Greedy best first; __X__ A*

1.b. (12 pts total, 2 pts each) The search space is finite and does have loops.

_____ Depth-first; __X__ Breadth-first; __X__ Uniform cost;

__X__ Iterated deepening; _____ Greedy best first; __X__ A*

1.c. (12 pts total, 2 pts each) The search space is infinite and may or may not have loops.

_____ Depth-first; __X__ Breadth-first; __X__ Uniform cost;

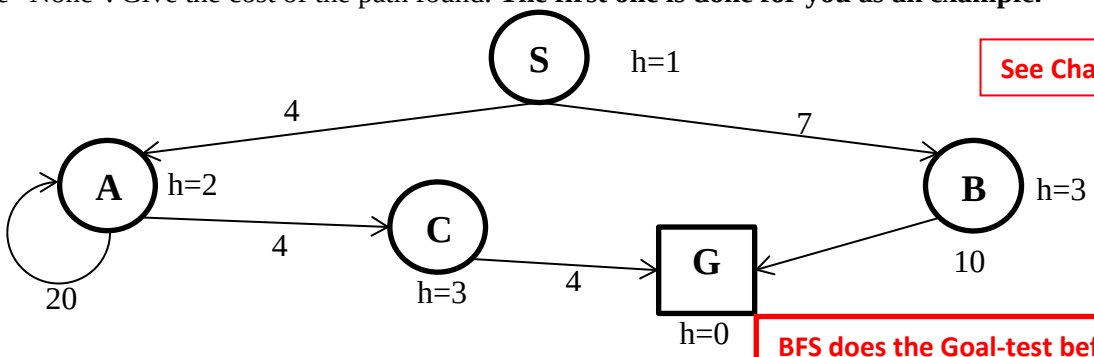
__X__ Iterated deepening; _____ Greedy best first; __X__ A*

Note that the "fair" search strategies in 1.b and 1.c are also the ones that are complete in such spaces.

****** TU[illegible]HER SIDE ******

2. (64 pts total, 16 pts each) STATE-SPACE SEARCH STRATEGIES. Execute Tree Search through this graph (i.e., do not remember visited nodes). Step costs are given next to each arc. Heuristic values are 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.

For each search strategy, 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. **The first one is done for you as an example.**



See Chapter 3.

2.a. (example) BREADTH FIRST SEARCH

See Section 3.4.1 and Fig. 3.11.

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

Order of node expansion: S A B (G)

Path found: S B G Cost of path found : 17

2.b. (16 pts) UNIFORM COST SEARCH:

See Section 3.4.2 and Fig. 3.14.

UCS does goal-test when node is popped off queue.

(10 pts) Order of node expansion: S A B C (G)

(4 pts) Path found: S A C G **(2 pts)** Cost of path found: 12

2.c. (16 pts) GREEDY (BEST-FIRST) SEARCH:

See Section 3.5.1 and Fig. 3.23.

A always has lower h [h(A)=2] than any node on the queue.

(10 pts) Order of node expansion: S A A A A A etc.

(4 pts) Path found: None **(2 pts)** Cost of path found: None

2.d. (16 pts) ITERATED DEEPENING SEARCH

See Sections 3.4.4-5 and Figs. 3.18-19.

IDS does the Goal-test iteratively on each child as generated, keeping the queue on the stack.

(10 pts) Order of node expansion: S S A B (G)

(4 pts) Path found: S B G **(2 pts)** Cost of path found: 17

2.e. (16 pts) A* SEARCH:

See Section 3.5.2 and Figs. 3.24-25.

A* does goal-test when node is popped off queue.

(10 pts) Order of node expansion: S A B C (G)

(4 pts) Path found: S A C G **(2 pts)** Cost of path found: 12

TECHNICAL NOTE: Technically, the goal node is not expanded, because no children of a goal node are generated. The goal node is listed in "Order of node expansion" for your convenience. Your answer is correct if you do not show the goal node in "Order of node expansion" — but it is a nicety to do so. Nevertheless, "Path found" *always* must show the goal node, because a path to a goal always must end in a goal.