CSCE 420 Homework 1 (2023 Fall)

See submission instructions on Canvas.

Search

Total: 100 points

* Note: In the questions below, the node list should have the form `[a, b, c, d, ... ]` where ``a'' is the first element of this list that gets taken out for inspection, and the rest are ordered from left to right.
That is, `Get-First-Node([a, b, c, d ...]) = a`.

* Note: Depth is 0 at the root, and increases by 1 as you follow the edge downward. That is, depth equals the number of operators you executed to reach the current level.

* Visit: The act of taking out a node from the node list for goal check is counted as a visit. Nodes that are expanded and put in the nodelist are treated as not yet visited.

1 Uninformed Search

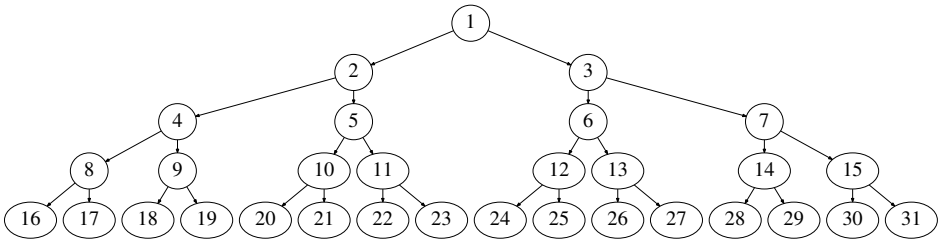


Figure 1: **Search Trees.**

Consider the search tree in Fig. 1. Assume that the exploration of the children of a particular node proceeds from the left to the right for all search methods in this section.

Problem 1 (Written; 5 pts): Consider depth first search. When the goal is (18) and the node is visited (taken out of node list), (1) what are the nodes that remain in the node list? (list them in the correct order). Also (2) which nodes have been visited until then, in what order?

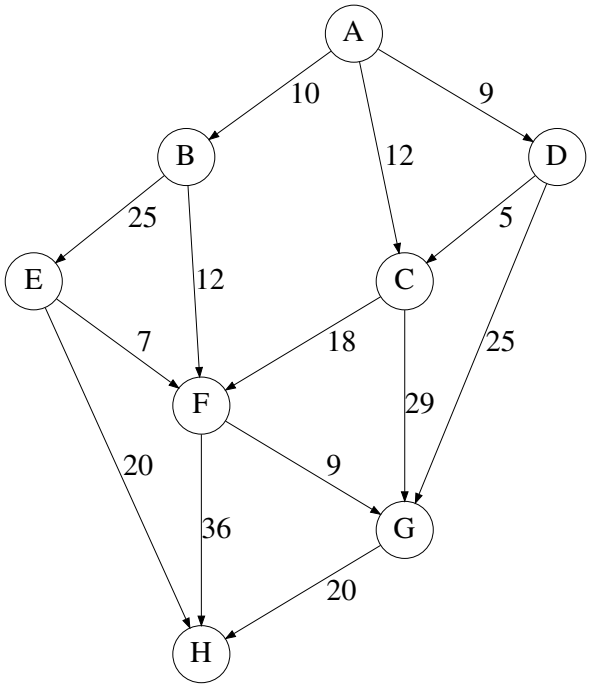
Problem 2 (Written; 5 pts): Consider breadth first search. When the goal is (6) and the node is visited (taken out of node list), (1) what are the nodes that remain in the node list? (list them in the correct order). Also (2) which nodes have been visited until then, in what order?

Problem 3 (Written; 5 pts): Why is the space complexity of BFS $O(b^{d+1})$, not $O(b^d)$, where b is the branching factor and d is the goal depth?

Problem 4 (Written; 5 pts): Can depth limited search become incomplete in the case of the finite search tree above? If so, give an example (use Figure 1). If not, explain why not (use Figure 1).

Problem 5 (Written; 5 pts): Consider iterative deepening search. When the goal is (12), how many nodes are visited before reaching that node? Hint: Include repeated visits in the count. Include the visit to the goal (9) in the count.

2 Informed Search



Node	h(n)
A	41
B	40
C	37
D	38
E	17
F	24
G	19
H	0

Figure 2: **Informed Search.**

Problem 6 (Written; 5 pts): For the problem shown in Fig. 2, show that the heuristic is admissible ($h(n) \leq h^*(n)$ for all n). Actual cost from node to node are shown as edge labels. The heuristic function value for each node is shown in a spearate table to the right. Note: You have to compute $h^*(n)$ for each n and compare to the $h(n)$ table.

Hint: It is best to work backwards from the goal, where $h^*(H) = 0$ (already at goal), $h^*(G) = 20$ (true minimum cost from (G) to (H)), $h^*(F) = 9 + 20 = 29$ (true minimum cost from (F) to (G) to (H): note that there is another path (F) to (H) which has a cost of 36), etc. Once you have the true cost for (F) and (G), you can easily compute the values for (C) and (E), etc.

Problem 7 (Written; 15 pts): Manually conduct greedy best-first search on the graph below (Fig. 2), with initial node (A) and goal node (H). Show:

1. Node list content at each iteration.
2. Node visit order
3. Solution path

4. Cost of the final solution.

Note: When sorting the node list by hand based on the $h(\cdot)$ values, some tied values may or may not appear. If they appear, put the oldest node ahead of new nodes.

Problem 8 (Written; 15 pts): (1) Repeat the problem right above with A* search. (2) In addition, show the $f(n)$ value for all nodes expanded (you need this to sort them in the node list). (3)Which one gives a lower cost solution: Greedy best-first or A*?

Note: Note that the same node can appear in the node list with a different $f(n)$ value, depending on the path taken. For example, $f(F)$ will be different if you followed a different path to reach the node: $A \to B \to F$ (where $f(F) = 10 + 12 + 24 = 36$) vs. $A \to C \to F$ (where $f(F) = 12 + 18 + 24 = 54$). Due to this, you may need to track which path you followed to reach node n and calculate the $f(n)$ value accordingly. It helps to write the node F as ${}_{AB}(\mathrm{F})$ to indicate the path in the subscript (path = $A \to B \to F$). Also, for sorting, it helps to indicate the f value as a subscript. For example, ${}_{AB}(\mathrm{F})_{f=36}$.

Note: When sorting the node list by hand based on the $f(\cdot)$ values, some tied values may or may not appear. If they occur, put the oldest node ahead of new nodes.

Problem 9 (Programming; 20 pts): Using the `dfs.ipynb` code, implement greedy-best-first search, and solve the problem in (Fig. 2). Print out the same output as required by Problem 7. See the Homework #1 Explanations page on Canvas (linked from hw1 description).

Problem 10 (Programming; 20 pts): Using the `dfs.ipynb` code, implement A* search, and solve the problem in (Fig. 2). Print out the same output as required by Problem 8.

IMPORTANT: For Problem 9 and 10, to get full credit, you must submit the screenshot of your pinned revision (colab) or history (Jypyter notebook). See Canvas instructions for details.