Written Assignment 1

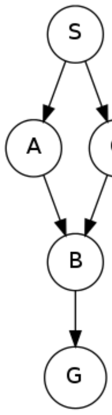
Deadline: October 16th, 2019

Instruction: You may discuss these problems with classmates, but please complete the write-ups individually. (This applies to BOTH undergraduates and graduate students.) Remember the collaboration guidelines set forth in class: you may meet to discuss problems with classmates, but you may not take any written notes (or electronic notes, or photos, etc.) away from the meeting. Your answers must be **typewritten**, except for figures or diagrams, which may be hand-drawn. Please submit your answers (pdf format only) on **Canvas**.

Q1. Uninformed Search (25 points)

Q1.1 Search Trees (5 points)

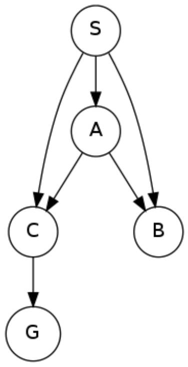
How many nodes are in the complete search tree for the following state space graph? The start state is **S**. You may find it helpful to draw out the search tree on a piece of paper.



Q1.2 Depth-First Graph Search (10 points)

Consider a depth-first graph search on the graph below, where **S** is the start and **G** is the goal state. Assume that ties are broken alphabetically (so a partial plan $S \to X \to A$ would be expanded before $S \to X \to B$ and $S \to A \to Z$ would be expanded before $S \to B \to A$).

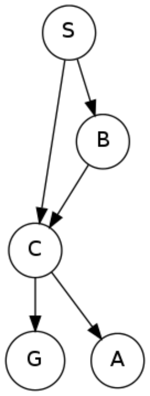
Please provide the final path returned by depth-first graph search. Your answer should be a string with **S** as your first character and **G** as your last character.



Q1.3 Breadth-First Graph Search (10 points)

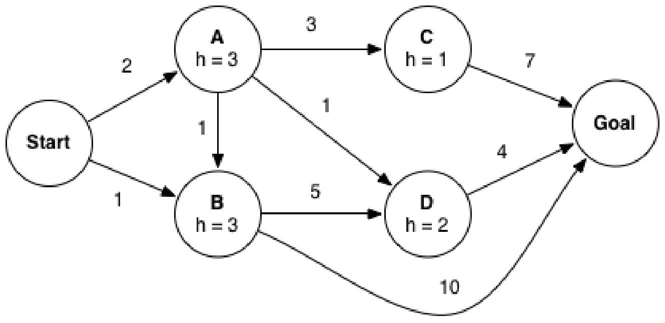
Consider a breadth-first graph search on the graph below, where **S** is the start and **G** is the goal state. Assume that ties are broken alphabetically.

Please provide the final path returned by breadth-first graph search.



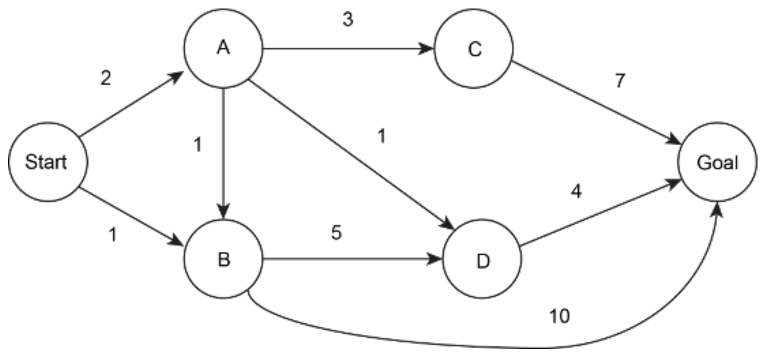
Q2. Informed Search (20 points)

Q2.1 A* Graph Search (10 points)



Consider A* graph search on the graph above. Arcs are labeled with action costs and states are labeled with heuristic values. Assume that ties are broken alphabetically. On what order are states expanded by A^* graph search? What path does $A*$ graph search return?

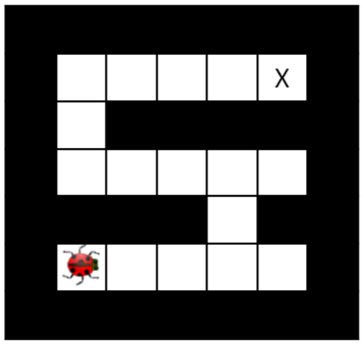
Q2.2 Uniform-Cost Graph Search (10 points)



Consider the graph above. Arcs are labeled with their weights. Assume that ties are broken alphabetically. In what order are states expanded by Uniform Cost Search? What path does uniform cost search return?

Q3. Hive Minds: Lonely Bug (20 points)

You control a single insect in a rectangular maze-like environment with dimensions $M \times N$ as shown in the figure below. The insect must reach a designated target location X, also known as the hive. There are no other insects moving around. At each time step, an insect can either (a) move into an adjacent square if that square is currently free, or (b) stay in its current location. Squares may be blocked by walls, but the map is known. Optimality is always in terms of time steps; all actions have cost 1 regardless of the number of insects moving or where they move.

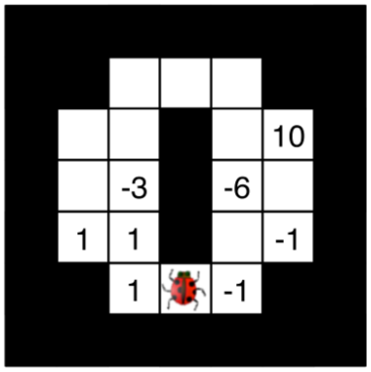


1. Provide a minimal correct state space representation? What is the size of the state space? (10 points)

2. Provide two heuristics which are admissible. (10 points)

Q4. Hive Minds: Time Limit (20 points)

In this problem, the ladybug has a limited number of timesteps remaining. For each time step, there is no moving penalty, but the remaining time will decrease by one. The ladybug will gain or lose points when it lands on positive or negative values, respectively. The ladybug must move to a new square for each timestamp, and the ladybug cannot move to a square that it has already visited. Your goal is to find the optimal score (higher is better) for a given timestep limit.



1. What is the optimal score for a timestep of 2? (4 points)
2. What is the optimal score for a timestep of 5? (6 points)
3. What is the optimal score for a timestep of 8? (10 points)

Q5. Lookahead Graph Search (15 points)

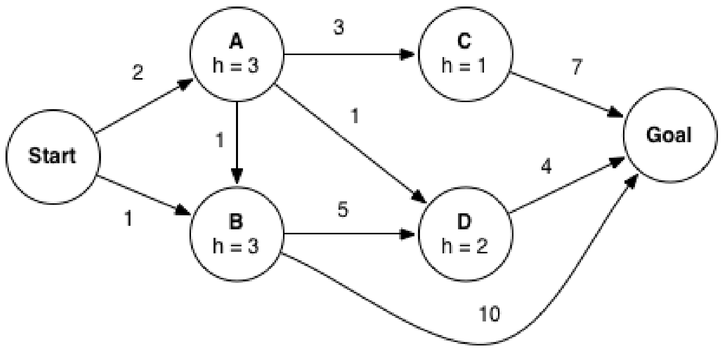
Recall from lecture the general algorithm for Graph Search reproduced below.

```
function GRAPH-SEARCH(problem, fringe, strategy) return a solution, or failure
    closed ← an empty set
    fringe ← INSERT(MAKE-NODE(INITIAL-STATE[problem]), fringe)
    loop do
        if fringe is empty then return failure
        node ← REMOVE-FRONT(fringe, strategy)
        if GOAL-TEST(problem, STATE[node]) then return node
        if STATE[node] is not in closed then
            add STATE[node] to closed
            for child-node in EXPAND(STATE[node], problem) do
                fringe ← INSERT(child-node, fringe)
            end
    end
```

Using GRAPH-SEARCH, when a node is expanded it is added to the closed set. This means that even if a node is added to the fringe multiple times it will not be expanded more than once. Consider an alternative version of GRAPH-SEARCH, LOOKAHEAD-GRAPH-SEARCH, which saves memory by using a "fringe-closed-set" keeping track of which states have been on the fringe and only adding a child node to the fringe if the state of that child node has not been added to it at some point. Concretely, we replace the highlighted block above with the highlighted block below.

```
function LOOKAHEAD-GRAPH-SEARCH(problem, fringe, strategy) return a solution, or failure
    fringe-closed ← an empty set
    fringe ← INSERT(MAKE-NODE(INITIAL-STATE[problem]), fringe)
    add INITIAL-STATE[problem] to fringe-closed
    loop do
        if fringe is empty then return failure
        node ← REMOVE-FRONT(fringe, strategy)
        if GOAL-TEST(problem, STATE[node]) then return node
        for child-node in EXPAND(node, problem) do
            if STATE[child-node] is not in fringe-closed then
                add STATE[child-node] to fringe-closed
                fringe ← INSERT(child-node, fringe)
        end
    end
```

Now, we've produced a more memory efficient graph search algorithm. However, in doing so, we might have affected some properties of the algorithm. To explore the possible differences, consider the example graph below.



1. If using LOOKAHEAD-GRAPH-SEARCH with an A* node expansion strategy, which path will this algorithm return? (Suggestion: step through the execution of the algorithm on a scratch sheet of paper and keep track of the fringe and the search tree as nodes get added to the fringe.) (15 points for undergrads / 6 points for grads)

2. **(Grads only)** Assume you run LOOKAHEAD-GRAPH-SEARCH with the A* node expansion strategy and a consistent heuristic, select all statements that are true. (9 points)

    (a) The EXPAND function can be called at most once for each state.

    (b) The algorithm is complete.

    (c) The algorithm will return an optimal solution.