Q1. Search

For this problem, assume that all of our search algorithms use tree search, unless specified otherwise.

(a) For each algorithm below, indicate whether the path returned after the modification to the search tree is guaranteed to be identical to the unmodified algorithm. Assume all edge weights are non-negative before modifications.

(i) Adding additional cost $c > 0$ to every edge weight.

	Yes	No
BFS	○	○
DFS	○	○
UCS	○	○

(ii) Multiplying a constant $w > 0$ to every edge weight.

	Yes	No
BFS	○	○
DFS	○	○
UCS	○	○

(b) **For part (b)**, two search algorithms are defined to be **equivalent** if and only if they expand the same states in the same order and return the same path. **Assume all graphs are directed and acyclic.**

(i) Assume we have access to costs c_{ij} that make running UCS algorithm with these costs c_{ij} equivalent to running BFS. How can we construct new costs c'_{ij} such that running UCS with these costs is equivalent to running DFS?

○ $c'_{ij} = 0$ ○ $c'_{ij} = 1$ ○ $c'_{ij} = c_{ij}$

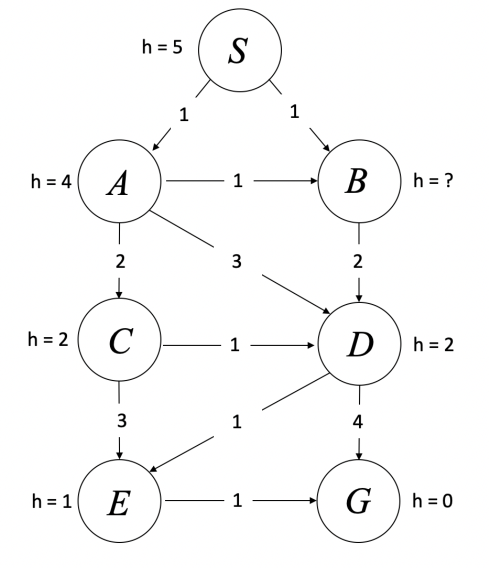
○ $c'_{ij} = -c_{ij}$ ○ $c'_{ij} = c_{ij} + \alpha$ ○ Not possible

(ii) Given edge weight $c_{ij} = h(j) - h(i)$, where $h(n)$ is the value of the heuristic function at node n, running UCS on this graph is equivalent to running which of the following algorithm on the same graph?

○ DFS ○ BFS ○ Iterative Deepening

○ Greedy ○ A* ○ None of the above.

(c) Consider the following graph. $h(n)$ denotes the heuristic function evaluated at node n.



(i) Given that G is the goal node, and heuristic values are fixed for all nodes other than B, for which values of $h(B)$ will A* tree search be guaranteed to return the optimal path? Fill in the lower and upper bounds or select "impossible."

________ $\leq h(B) \leq$ ________ ○ Impossible

(ii) With the heuristic values fixed for all nodes other than B, for which values of $h(B)$ will A* graph search be guaranteed to return the optimal path? Either fill in the lower and upper bound or select "impossible."

________ $\leq h(B) \leq$ ________ ○ Impossible

Q2. SpongeBob and Pacman (Search Formulation)

Recall that in Midterm 1, Pacman bought a car, was speeding in Pac-City, and the SpongeBob wasn't able to catch him. Now Pacman has run out of gas, his car has stopped, and he is currently hiding out at an undisclosed location.

In this problem, you are on the SpongeBob side, tryin' to catch Pacman!

There are still p SpongeBob cars in the Pac-city of dimension m by n. In this problem, **all SpongeBob cars can move, with two distinct integer controls: throttle and steering, but Pacman has to stay stationary**. Once one SpongeBob car takes an action which lands him in the same grid as Pacman, Pacman will be arrested and the game ends.

Throttle: $t_i \in \{1, 0, -1\}$, corresponding to {Gas, Coast, Brake}. This controls the **speed** of the car by determining its acceleration. The integer chosen here will be added to his velocity for the next state. For example, if a SpongeBob car is currently driving at 5 grid/s and chooses Gas (1) he will be traveling at 6 grid/s in the next turn.
Steering: $s_i \in \{1, 0, -1\}$, corresponding to {Turn Left, Go Straight, Turn Right}. This controls the **direction** of the car. For example, if a SpongeBob car is facing North and chooses Turn Left, it will be facing West in the next turn.

(a) Suppose you can **only control 1 SpongeBob car**, and have absolutely no information about the remainder of $p - 1$ SpongeBob cars, or where Pacman stopped to hide. Also, the SpongeBob cars can travel up to 6 grid/s so $0 \leq v \leq 6$ at all times.

(i) What is the **tightest upper bound** on the size of state space, if your goal is to use search to plan a sequence of actions that guarantees Pacman is caught, no matter where Pacman is hiding, or what actions other SpongeBob cars take. Please note that your state space representation must be able to represent **all** states in the search space.

(ii) What is the maximum branching factor? Your answer may contain integers, m, n.

(iii) Which algorithm(s) is/are guaranteed to return a path passing through all grid locations on the grid, if one exists?

- ☐ Depth First Tree Search
- ☐ Breadth First Tree Search
- ☐ Depth First Graph Search
- ☐ Breadth First Graph Search

(iv) Is Breadth First Graph Search guaranteed to return the path with the shortest number of **time steps**, if one exists?

○ Yes ○ No

(b) Now let's suppose you can control **all** p SpongeBob cars at the same time (and know all their locations), but you still have no information about where Pacman stopped to hide

(i) Now, you still want to search a sequence of actions such that the paths of p SpongeBob car combined **pass through all** $m * n$ **grid locations**. Suppose the size of the state space in part (a) was N_1, and the size of the state space in this part is N_p. Please select the correct relationship between N_p and N_1

○ $N_p = p * N_1$ ○ $N_p = p^{N_1}$ ○ $N_p = (N_1)^p$ ○ None of the above

(ii) Suppose the maximum branching factor in part (a) was b_1, and the maximum branching factor in this part is b_p. Please select the correct relationship between b_p and b_1

○ $b_p = p * b_1$ ○ $b_p = p^{b_1}$ ○ $b_p = (b_1)^p$ ○ None of the above