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

Q2. SpongeBob and Pacman (Search Formulation)

Pacman bought a car, was speeding in Pac-City, and 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

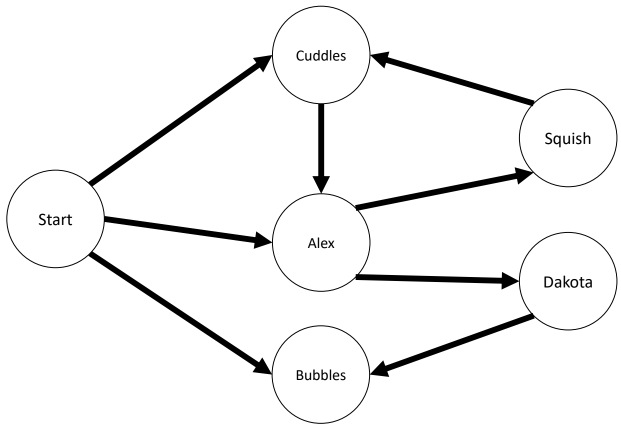
○ $N_p = p * N_1$ ○ $N_p = p^{N_1}$ ○ $N_p = (N_1)^p$ ○ None of the above

(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

Q3. Search: Snail search for love

Scorpblorg the snail is looking for a mate. It can visit different potential mates based on a trail of ooze to nearby snails, and then test them for chemistry, as represented in the below graph, where each node represents a snail. In all cases, nodes with equal priority should be visited in alphabetical order.



(a) Simple search

In this part, assume that the only match for Scorpblorg is Squish (i.e. Squish is the goal state). Which of the following are true **when searching the above graph**?

(i) BFS Tree Search expands more nodes than DFS Tree Search ◯ True ◯ False

(ii) DFS Tree Search finds a path to the goal for this graph ◯ True ◯ False

(iii) DFS Graph Search finds the shortest path to the goal for this graph ◯ True ◯ False

(iv) If we remove the connection from Cuddles → Alex, can DFS Graph Search find a path to the goal for the altered graph? ◯ Yes ◯ No

(b) Third Time's A Charm

Now we assume that Scorpblorg's mate preferences have changed. The new criteria she is looking for in a mate is that she has **visited the mate twice before** (i.e. when she visits any state for the third time, she has found a path to the goal).

(i) What should the most simple yet sufficient new state space representation include?

- ☐ The current location of Scorpblorg
- ☐ The total number of edges travelled so far
- ☐ An array of booleans indicating whether each snail has been visited so far
- ☐ An array of numbers indicating how many times each snail has been visited so far
- ☐ The number of distinct snails visited so far

(ii) DFS Tree Search finds a path to the goal for this graph ◯ True ◯ False

(iii) BFS Graph Search finds a path to the goal for this graph ◯ True ◯ False

(iv) If we remove the connection from Cuddles → Alex, can DFS Graph Search finds a path to the goal for the altered graph? ◯ Yes ◯ No

We continue as in part (b) where the goal is still to find a mate who is visited for the third time.

(c) Costs for visiting snails

Assume we are using Uniform cost search and we can now add costs to the actions in the graph.

(i) Can one assign (non-negative) costs to the actions in the graph such that the goal state returned by UCS (Tree-search) changes? ◯ Yes ◯ No

(ii) Can one assign (potentially negative) costs to the actions in the graph such that UCS (Tree-search) will never find a goal state? ◯ Yes ◯ No

Q4. Search Party

Annie is throwing a party tonight, but she only has a couple hours to get ready. Luckily, she was recently gifted 4 one-armed robots! She will use them to rearrange her room for the guests . Here are the specifications:

- Her room is modeled as a W-by-L-by-H 3D grid in which there are N objects (which could be anywhere in the grid to start with) that need rearrangement.
- Each object occupies one grid cell, and no two objects can be in the same grid cell. Do not consider any part of the robot an "object."
- At each time-step, one robot may take an action $\in$ {move gripper to legal grid cell, close gripper, open gripper}. Moving the gripper does not change whether the gripper was closed/open.
- A robot can move an object by
  1. Moving an open gripper into the object's grid cell
  2. Closing the gripper to grasp the object
  3. Moving to desired location
  4. Opening the gripper to release the object in-hand.
- The robots do not have unlimited range. The arm can move to any point *within* the room that is strictly less than R grid cells from its base per direction along each axis. Explicitly, if $R = 2$ and a robot's base is at (0,0,0), the robot cannot reach (0,0,2) but can reach (1,1,1). Assume $R < W, L, H$.

(a) Annie stations one robot's stationary base at each of the 4 *corners* of the room. Thankfully, she knows where each of the N objects in the room should be and uses that to define the robots' goal. Complete the following expression such that it evaluates to the size of the minimal state space. Please approximate permutations as follows: X permute $Y \approx X^Y$. You may use scalars and the variables: W, L, H, R, and N in your answers.

$$2^{(a)} \cdot N^{(b)} \cdot R^{(c)} \cdot W^{(d)} \cdot L^{(e)} \cdot H^{(f)}$$

(a): ______ (b): ______ (c): ______

(d): ______ (e): ______ (f): ______

(b) Each of the following describes a modification to the scenario previously described and depicted in the figure. **Consider each modification independently (that is, the modifications introduced in (i) are *not* present in (ii)).** For each scenario, give the size of the new minimal state space.
Please use the symbol X as a proxy for the correct answer to (b) in your answers.

(i) The robots are given wheels, so each base is able to slide along the floor (they still can't jump) from their original corners. That is, at each time-step, a robot has a new action that allows them to move its (once stationary) base arbitrarily far across the floor. When the robot slides its base, the relative arm position and status of the gripper remain the same.

(ii) *One* robot is defective and can move for a maximum of T timesteps before it must rest for at least S timesteps. You may use S or T in your expression.