James Yang

CSE 573

Artificial Intelligence

Written Homework #1

1. **Let's define the procedure of hill-climbing. You start at a random location on a hill, your goal is to get to the highest point on the hill. At each time step, you will take a step toward the location next to you that is higher than your current location.
Is hill-climbing complete? Why? If not, is there any way to improve the performance in the discrete problem space? (3 points)**

   If the hill were monotonically increasing, the problem would not be complete. An example is if the person starts at the top of the hill. Then the remaining nodes would never be explored. If the hill isn't monotonically increasing and had multimodal bumps, it still wouldn't be complete because it wouldn't be able to reach any sections lower than what it is already at. The only time it would be complete is if the starting location was already at the bottom and there were no multimodal distribution bumps. We can improve the problem space by making it a circuit hill that loops around back to the start; this idea is assuming that the next highest point is simply the next step in the hill. We can also randomly restart the problem.

**Pac-Man wants to get to the goal location from some initial position in a 2D grid.
a. If Pac-Man wants to get to the goal location with the shortest path, what is the simplest state representation? Please use mathematical notations. (2 point)
b. If Pac-Man instead wants to first get to all four corners, then go to the goal location, should the state representation above change? If so, how? (2 point)**

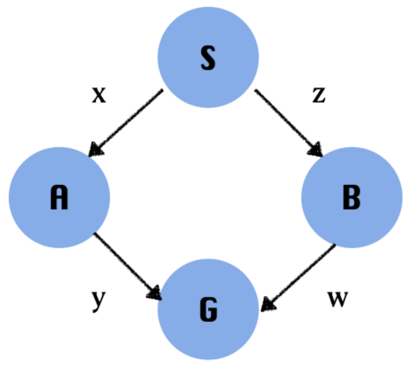
a. Like we have done in PS1, we would have a state where the first value correlates to its position in an (x,y) state. Ex: $State = (x_1, y_1)$. Then to get the path, we would use Manhattan distance or Euclidean distance.
b. Like we have done in PS1 for the cornersHeuristic, we would have an array of state where the first value correlates to its position in an (x,y) state and then a list of coordinates that we are trying to approach for the goal.
   Ex: $State =$ [(x_1, y_1), list((x_1, y_1), (x_2, y_2), … (x_n, y_n))]

In what circumstances is Greedy Search preferred over Uniform Cost search? Write down two circumstances. (2 points)

1. Greedy search picks the lowest heuristic value. When developing heuristic values for nodes, if someone is super educated on the game they were developing (where their heuristic values are almost perfect), then greedy search would be preferred over uniform cost search and the grid has paths where *cost of multiple cheap nodes > heuristic path.*
2. If we are continuing off the previous statement, lets say we don't need to explore the entirety of the map (assuming finite cost and minimum cost are

positive), then greedy search would be preferred because it doesn’t necessarily explore completely which could save computation time.

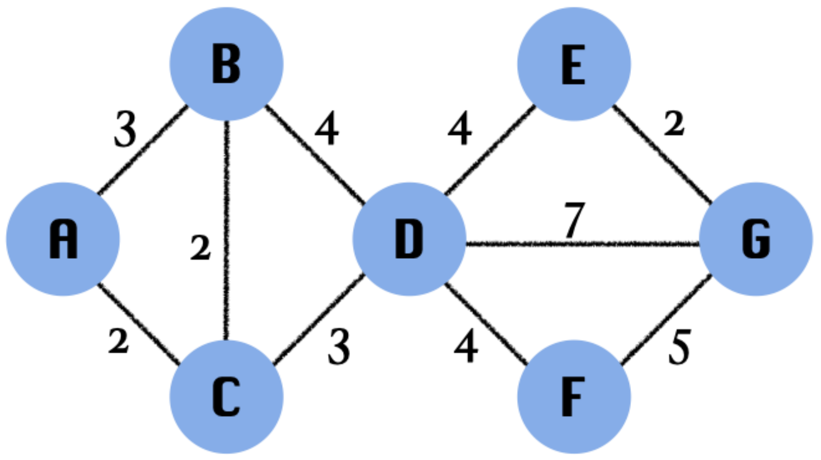
You are given the graph shown below, and the heuristics functions h. Your start from State S, and your goal is to go to State G.
Find non-negative edge weights, x, y, z, and w, such that it satisfies each of the scenarios:



Heuristic
h(S) = 20
h(A) = 2
h(B) = 15

a. Scenario 1: (3 points)
Both greedy search and A* find the optimal solution.
x: 1
y: 1
z: 2
w: 2
b. Scenario 2: (3 points)
A* search finds the optimal solution, but greedy search doesn’t.
x: 100
y: 100
z: 1
w: 1

2.



Heuristic

Node	h1	h2
A	12.5	11
B	12	10
C	11	9
D	5	6
E	1	2
F	4	4.5
G	0	0

For each of the following graph search strategies, mark with an X which (if any) of the listed paths it could return. Note that for some search strategies the specific path

returned might depend on tie-breaking behavior. In any such cases, make sure to mark all paths that could be returned under some tie-breaking scheme. (13 points)

Algorithm	**A-B-D-G**	**A-C-D-G**	**A-C-D-E-G**
BFS	**X**	**X**	
DFS	**X**	**X**	**X**
UCS			**X**
Greedy with Heuristic h1		**X**	
Greedy with Heuristic h2		**X**	
A* with Heuristic h1			**X**
A* with Heuristic h2			**X**

What is the cost of the optimal path for uniform cost search from A to G? (1 point)
A-C-D-E-G = 11 (ADD LATER)

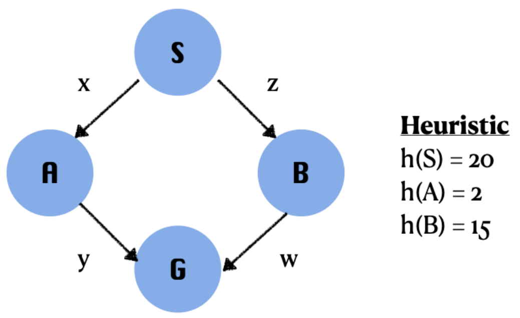
Is admissible? Is it consistent? Write an answer as well as the *h* reason for each. (2 points)
H1 is not admissible. Example: H(A) Heuristic cost is less than the actual cost. It is not consistent though. For example (b->d) 12-5 = 7 which is greater than 4.

Is admissible? Is it consistent? Write an answer as well as the *h* reason for each. (2 points)

H2 is admissible. Example: H(A) Heuristic cost is less than or equal to the actual cost (11 <= 11). It is also consistent. The reason is because all differences in nodes give you a value that is either equal or less than the true length.

For any given graph, is the path returned by greedy search always more expensive than the path returned by A* search? If you answer yes, explain; If you answer no, provide a simple counterexample. (Assume the heuristic used for A* is consistent and admissible.) (2 points)

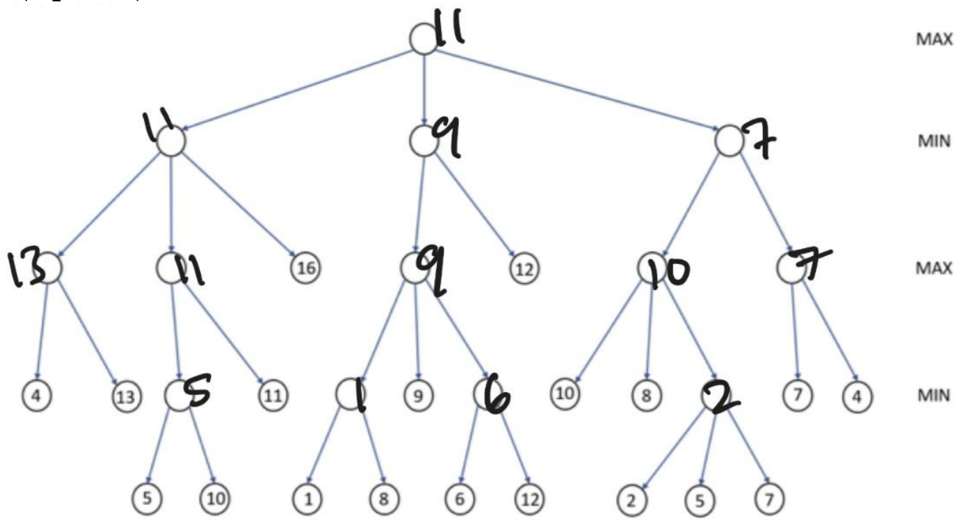
No the path given by greedy isn't always more expensive than the path returned by A* search. For example, the graph done in question 1 shown below:



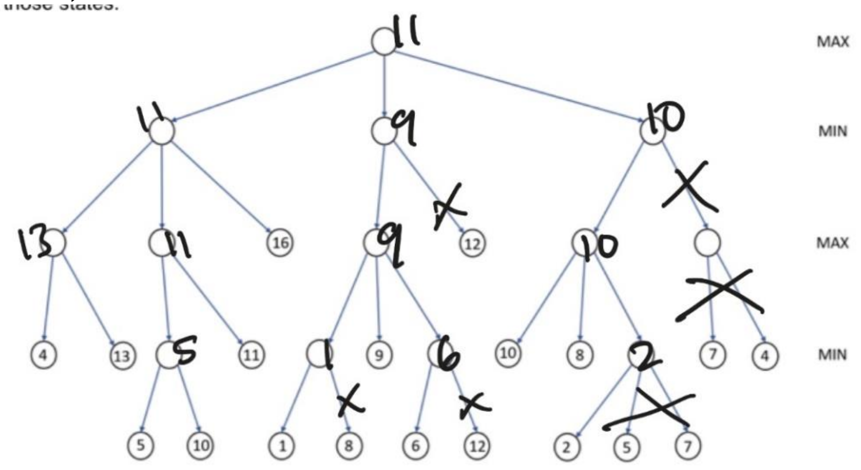
x: 1 **y:** 1 **z:** 2 **w:** 2, they both give the same path and they aren't more expensive than each other because they went the exact same way to the goal.

3.

Show the values of every intermediate node after performing the minimax algorithm. (5 points)



Use the Alpha-Beta pruning algorithm to determine the branches that need to be cut. (8 points)



For a general game tree (i.e., not limited to the above tree), are there any cases that the AlphaBeta algorithm gives a different value at the root node than the Minimax algorithm? If yes, show an example; if no, just say no. (2 points)

No I don't believe so.

4.

a. **Players can play optimally using a minimax algorithm. Why is expanding the whole game tree not practical? What are the things that we need to consider when we design the evaluation function so we can evaluate different stages of this game? (7 points)**

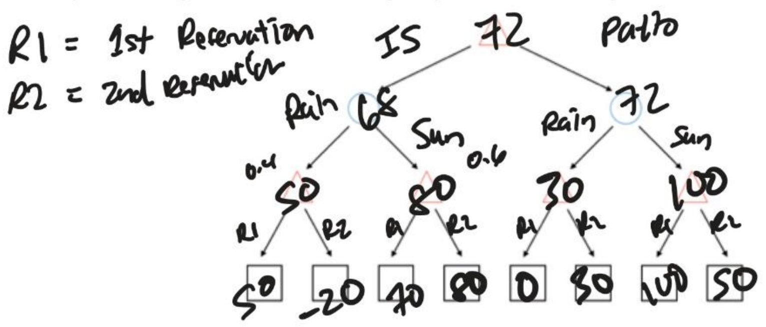
   There is no use in expanding the whole game tree because the game is revolved around nodes used and which nodes are considered of more "weighted value" based on their probabilistic chances of linking opposite sides of the board in an unbroken chain. We no longer need to consider nodes that are far away from the central chain being built, meaning any calculation of the outer nodes would be somewhat useless and expensive. Some things we need to consider when creating an evaluation function is the number of pieces needed for the opponent to connect to their two sides of the board. The less amount of pieces required, the stronger the evaluation. We can give it a score based on the difference between red and blue.

b. **Define an evaluation function that approximates the value of each state. (7 points)**
   We can give a value of Eval in terms of the approximation score of blue and the approximation score of red.
   ApproxR, ApproxB - gives the estimated amount of nodes needed to link itself in an unbroken chain.
   Eval score would be the **ApproxR – ApproxB** because if red needs to have 5 nodes still and blue say only needs 1, it would give you -4 which would give you how strong this state would be that you are currently in.

c. **If the size of the game is $n * n$ and each time the agent considers the next three moves (agents' move, minimizers' response, agents' subsequent move). What is the Big-θ time cost of the initial action? (6 points)**
   The initial cost would be O(n^2) because we would be iterating through the number of moves, the minimizers response might be less than O(n^2) but it will be overpowered by O(n^2). So the **Big(O) time would be approximately O(n^6).**

5.

a. **Imagine this is a game between you and the weather. Complete the following game tree by describing what each node/edge represents and filling the terminal values. (2 points)**



The tree goes down with choosing either a patio or the indoor seating. From there, we decide whether this is the first reservation or the second reservation and if it is sunny or rainy.

b. **Fill in the values for interior nodes based on Expectimax search, and decide on whether you will reserve the patio or indoor seating. (5 points)**
With the diagram above, we see that we would eventually choose the patio seating as it has a better expectimax value.

c. **Suppose it will rain with probability p (instead of 60). Find the range of p that will change the optimal decision for the root max node. (5 Points)**

$$50(1-p) + 80p \geq 30(1-p) + 100p$$

$$20(1-p) \geq 20p$$

$$20 - 20p \geq 20p$$

$$20 \geq 40p$$

$$\boxed{0 \leq p \leq \frac{1}{2}}$$

d. **Suppose it will rain with probability 60, but the deduction to the satisfaction if you change the reservation from indoor seating to patio is d (instead of 20). Find d the range of d that will change the optimal decision for the root max node. (5 Points)**

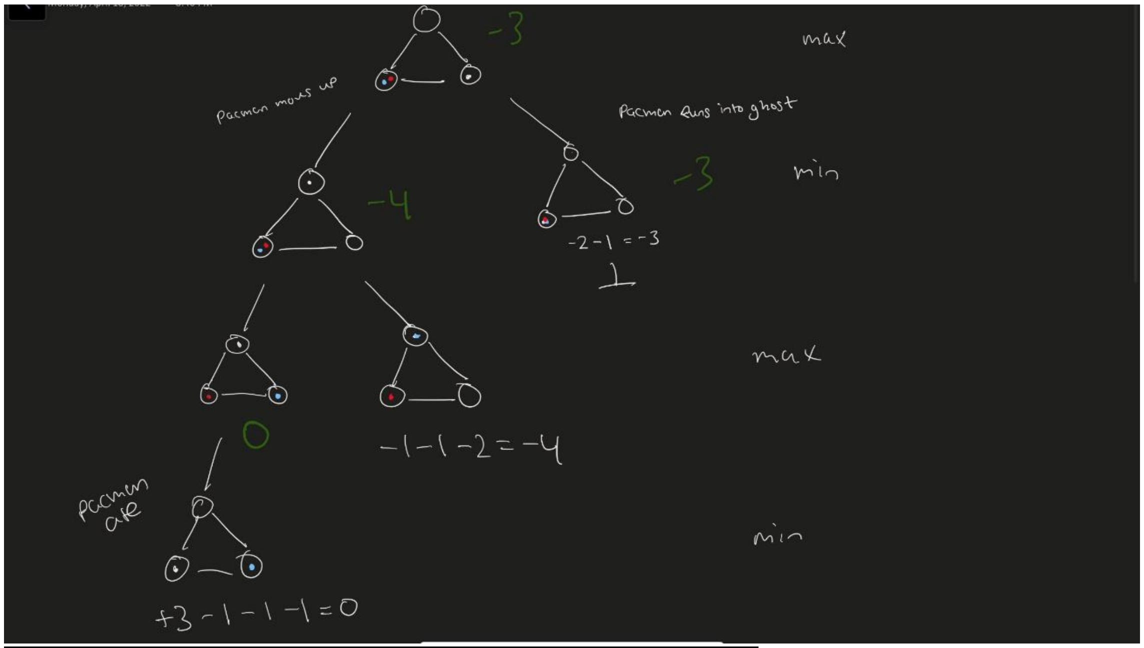
$$50(0.4) + (100-d)(0.6) \geq 72$$

$$20 + 60 - 0.6d \geq 72$$

$$-0.6d \geq -8$$

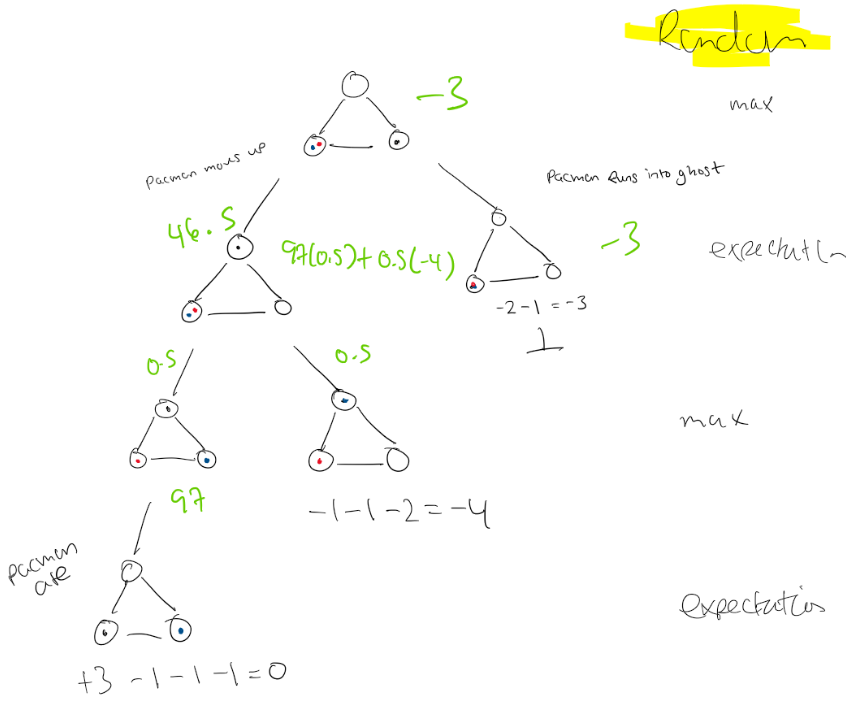
$$\boxed{0 \leq d \leq 15.\overline{3}}$$

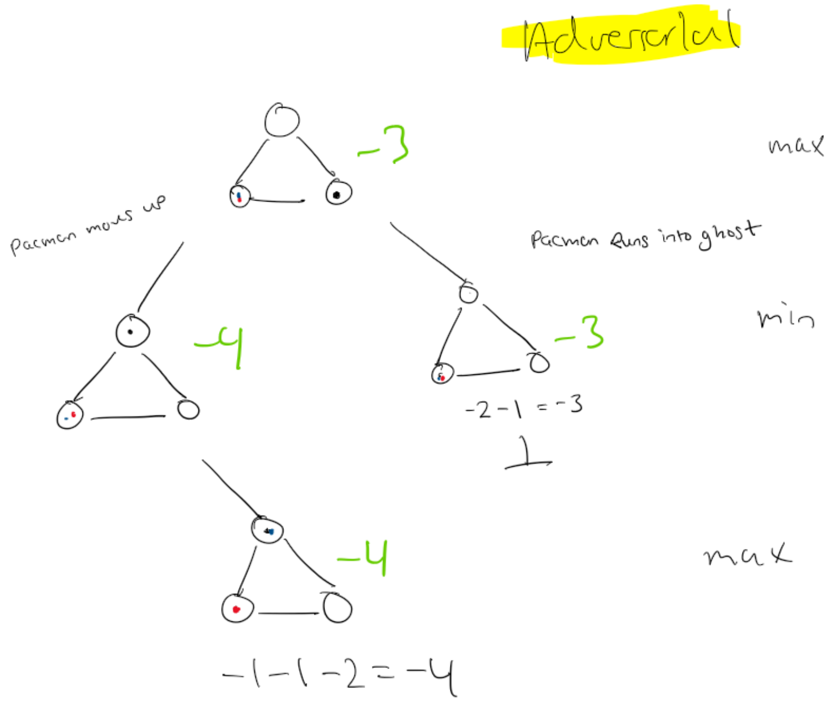
e. **Calculate the expected score of Pacman if it plays optimally either with Minimax or Expectimax, each in case of A=3 and A=100. Fill in the tables below. (8 points)**



	Adversarial Ghost	Random Ghost
Minimax Pacman	-3	-3
Expectimax Pacman	-3	-2

When A = 3





	Adversarial Ghost	Random Ghost
Minimax Pacman	-3	-3
Expectimax Pacman	-3	46.5

When A = 100

f. **Suppose the ghost will be random with a probability of and α adversarial with a probability of 1 − α. Explain what pacman's movement should be at its first turn. You should use a function of A and α. (5 points)**
Something I noticed is regardless of whether the ghost is adversarial or not, Pacman needs to move towards the top node. There is no other node for him to go without dying. **A(α) + A(α-1)** where A is the movement of pacman.