Name: **Student ID:**

CSE 573 Winter 2023 HW1

Due Thursday 2/2 by 11:59pm

100 points

Instructions:

1) The homework should be done individually. Don't forget to write your name.
2) We highly recommend typing your homework, but writing and scanning, or annotating a PDF are also acceptable.
3) Keep your answers brief but provide enough explanations and details to let us know that you have understood the topic.
4) The assignment is due on February 2.
5) You should upload your assignments through gradescope.

Topics:	Points
Short Problems	15
Search Algorithms	10
Heuristics for Informed Search	15
Alpha-Beta pruning	15
Evaluation Function	20
Expectimax	25

Problem 1. Short Problems [15 Points]

A) 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? (2 points)

B) 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? (1 point)

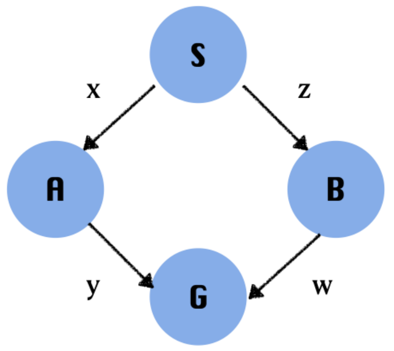
C) Search algorithm comparisons:

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

D) 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)

E) You are given the graph shown below, and the heuristics functions *h*. You 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 the graph satisfies each of the scenarios:

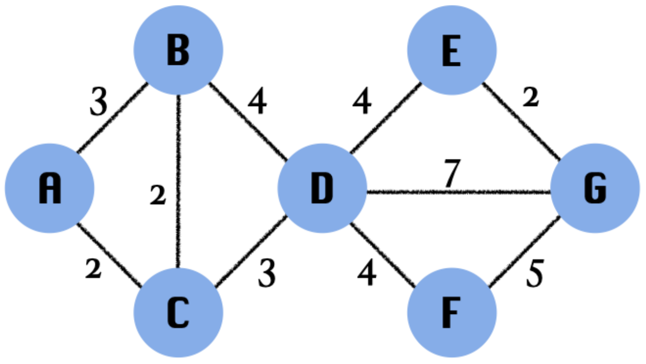


Heuristic
h(S) = 20
h(A) = 2
h(B) = 15

a. Scenario 1: (3 points)
Both greedy search and A* find the optimal solution.
x:
y:
z:
w:

b. Scenario 2: (3 points)
A* search finds the optimal solution, but greedy search doesn't.
x:
y:
z:
w:

Problem 2. Search Algorithms [10 Points]



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

Consider the state space graph shown above. A is the start state and G is the goal state. The costs for each edge are shown on the graph. Each edge can be traversed in both directions. Please refer to the search algorithms exactly **as presented on the lecture slides** as the ordering of the actions matters.

A) 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. (10 points)

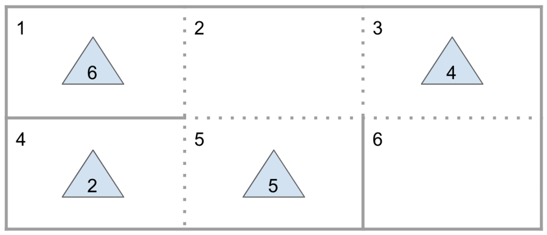
Algorithm	**A-B-D-G**	**A-C-D-G**	**A-C-D-E-G**
BFS			
DFS			
UCS			
Greedy with heuristic h_1			
Greedy with heuristic h_2			
A* with heuristic h_1			
A* with heuristic h_2			

Problem 3. Heuristics for Informed Search [15 Points]

A delivery robot is moving in an n*m maze. A simple version of the maze is shown in the figure. The robot is programmed to deliver multiple parcels to their destinations. Each parcel starts at some node in the maze and has its own delivery destination. The initial position of the parcels is shown in the figure, and the number on each parcel is its target destination. At every step the robot can take one of the following actions:

- Move: Move in one of these directions: {Up, Right, Down, Left}
- Pick: Pick-up a parcel in a location
- Drop: Put down a parcel at a location.

The cost of each move action is 1, and the costs of Pick and Drop are zero. The robot starts at square number 1 and can move through dotted lines - but the solid lines represent walls. The robot wants to deliver all parcels to their destinations with minimal cost.
Note that multiple parcels can be placed at the same square. You might come up with different ways of addressing these questions; For each part, there might be several possible heuristic functions. You will get credit if your heuristic follows requested requirements -- clearly write down the assumptions and conditions in which your solution is plausible.



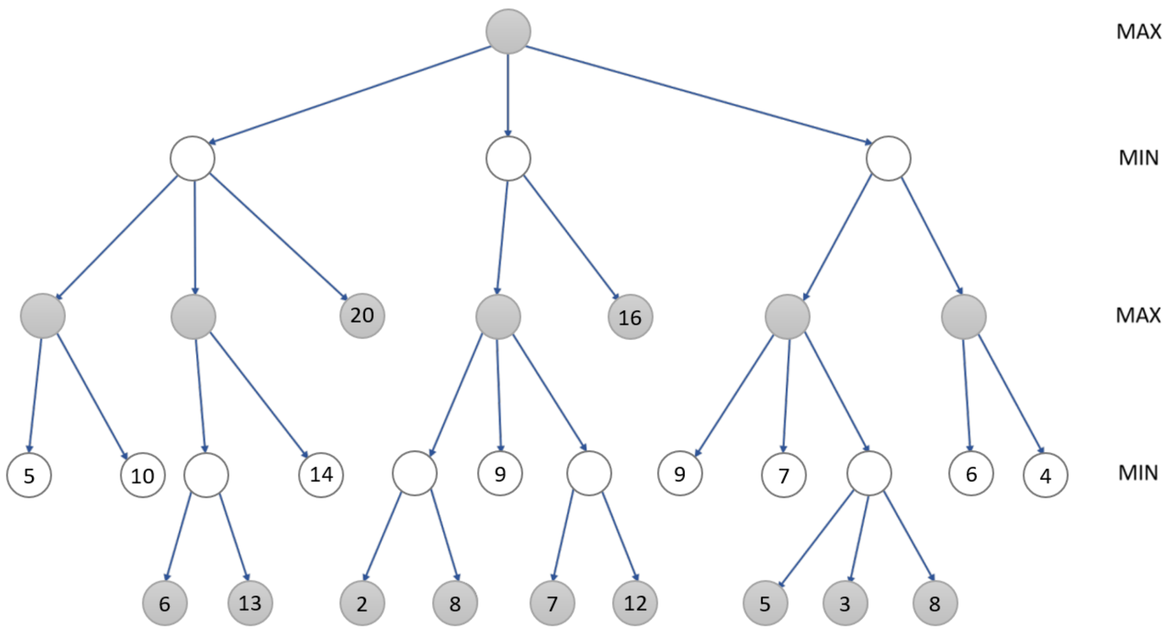
A) If the robot can carry only one parcel at a time, define an admissible heuristic function for searching the space. Explain in plain english why the heuristic is admissible. Is your heuristic consistent? Why? Make sure your heuristic is not h(x) = 0. (5 points)

B) If the robot can carry multiple parcels at a time, is the function h(x) = "count of packages that are not delivered" admissible and consistent? Why or why not? (4 points)

C) If the robot can carry multiple parcels at a time, define two admissible heuristic functions. Explain in plain english why each heuristic is admissible. Are your heuristics consistent? Why? Make sure your heuristic is not h(x) = 0, and it is not a function of actual cost because it is not practical to compute the actual cost in a general case. *Hint: the heuristic function can be a function of carried parcels and un-carried parcels.* (6 points)

Problem 4. Alpha-Beta pruning [15 points]

Below is the tree showing the states in a 2-player game played by two rational agents. This tree shows the 2-level expansion of decisions, and the values at the leaves are the utility values at those states.



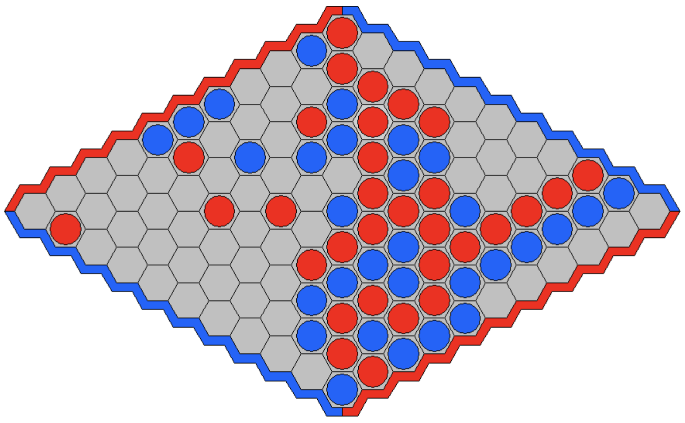
A) Show the values of every intermediate node after performing the minimax algorithm. (5 points)

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

C) 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)

Problem 5. Evaluation Function [20 Points]

The Game of Hex was invented by John Nash in the 1940s. Hex consists of a rhombus game map divided into $n * n$ hexagons. Each player in a 2-player game has a marker (Blue and red). At each round a player can place a marker on an unmarked hexagon, and players alternate turns. The goal for the players is to link their opposite sides of the board in an unbroken chain. Whoever connects their sides first wins and receives +1 point and the opponent receives -1. It has been proven that a draw is impossible in Hex, hence there's always a winner.



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)

B) Define an evaluation function that approximates the value of each state. (7 points)

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)

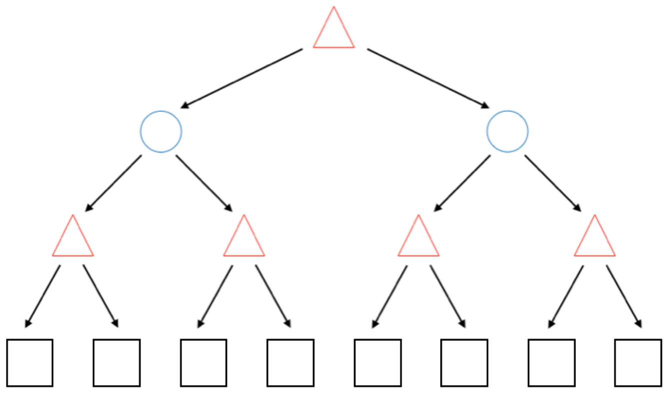
Problem 6 - Expectimax [25 points]

You have a friend visiting Seattle tomorrow. You want to make a dinner reservation for either patio or indoor seating. The weather forecast says it will rain with a 60% probability. Your friend's satisfaction will be as follows.

Weather	**Place**	**Your friend's satisfaction**
Rainy	Patio	0
Rainy	Indoor seating	50
Sunny	Patio	100
Sunny	Indoor seating	70

You can switch the reservation to the other seating option tomorrow, depending on the weather, but then your friend's satisfaction will be deducted by 20 due to a waiting time.

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. (3 points)



B) Fill in the values for interior nodes based on Expectimax search, and decide on whether you will reserve the patio or indoor seating. (3 points)

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. (3 Points)

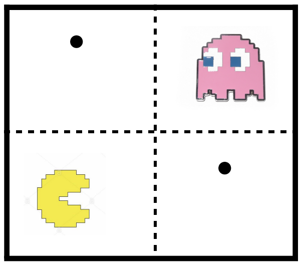
D) Suppose it will rain with probability 60, but the deduction to the satisfaction if you change the reservation from indoor seating to patio is d (instead of 20). Find the range of d that will change the optimal decision for the root max node. (4 Points)

Small PacMan Maze:

In the map below, the pacman and the ghost can move to any adjacent squares (the action space is $\{U, D, R, L\}$). The pacman and the ghost can move in any direction if there is no wall on the way and the solid lines represent the wall. The movement of the ghost is random. Pacman starts first and alternates turns with the ghost. The game starts in the shown figure, and it ends in either one of the two cases. Note that pacman and the ghost cannot move in the direction of a wall (for example, in the start state shown, pacman cannot move 'down' or 'left'; the only valid actions are 'up' or 'right').

- Winning: Pacman (maximizer) eats all the dots.
- Losing: The ghost catches the pacman.

Pacman will receive the score of +1 for eating a dot and a penalty score of -2 if it is caught by the ghost. The final score of the pacman is calculated as the number of dots eaten by the pacman and the penalty if the ghost caught the pacman.



A) Draw the expectimax search tree for the first four total turns (pacman moves first, then the ghost, then pacman again, and finally the ghost). The ghost moves horizontally with probability of *q* and it moves vertically with probability of *1-q*. Use the letter 'P' for pacman and 'G' for ghost when drawing game states. On the tree please distinguish the max-nodes, terminal nodes and the expectation nodes. Note that each of the pacman or the ghost's movements counts as the turn. You don't need to expand the whole tree; you can calculate the values of non-terminal states at depth 4. (5 points)

B) Calculate the expected score of Pacman if it plays optimally for q=⅓. Hint: the tree can be infinitely large, but you don't need to expand the whole tree in order to compute this expected score -- Pacman will get a score of +2 for eating both dots and winning the game. (5 points)

C) Explain in plain English what Pacman's strategy should be (2 points).