CS 188 Spring 2020 Introduction to Artificial Intelligence — Written HW 1

Due: Wednesday 02/19/2020 at 11:59pm (submit via Gradescope).

Policy: Can be solved in groups (acknowledge collaborators) but must be written up individually

Submission: Your submission should be a PDF that matches this template. Each page of the PDF should align with the corresponding page of the template (page 1 has name/collaborators, question 1 begins on page 2, etc.). **Do not reorder, split, combine, or add extra pages.** The intention is that you print out the template, write on the page in pen/pencil, and then scan or take pictures of the pages to make your submission. You may also fill out this template digitally (e.g. using a tablet.)

First name	Shiv
Last name	Shankar
SID	3032662765
Collaborators	

For staff use only:

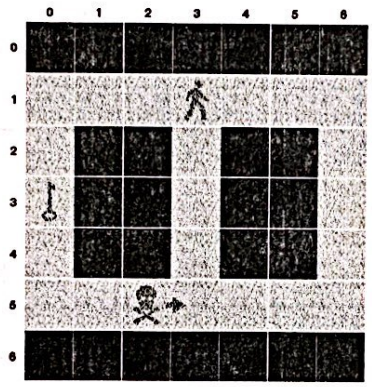
Q1.	Uninformed Search and Heuristics	/50
Q2.	Expectimax Yahtzee	/50
	Total	/100

Q1. [50 pts] Uninformed Search and Heuristics

Consider the following simplified version of the classic Atari video game, *Montezuma's Revenge*: It is played on the board illustrated below. An agent (represented by the person icon in cell (1,3)) wishes to grab the key (in cell (3,0)). A skull starts in cell (5,2) and moves to the right by one cell after each action is executed until it ends up in the rightmost cell, at which point it starts moving to the left, and repeats this pattern back and forth.

The agent can be facing either left or right. There are 10 possible actions for the agent: 2 turning actions ($turn\_left, turn\_right$) and 8 moving actions ($left, right, up, down, left\_up, left\_down, right\_up, right\_down$). The agent can move up or down while facing either direction, but can move sideways or diagonally only if facing in that direction. For example, if the agent is facing right but tries to move $left\_up$, the agent will not move and nothing will happen. Furthermore, if the agent is already facing $left$ and a $turn\_left$ action is taken, nothing happens.

Lastly, the agent cannot move into a cell **currently occupied** by the skull, or a wall.



(a) Answer the following questions for the Montezuma's revenge board above:

(i) [4 pts] Let N be the number of possible cell locations that the agent can be in, and let M be the number of possible cell locations that the skull can be in. Recall that for "pacman pathing", the representation of the state was (x, y) where x was the row and y was the column of pacman's position.
Describe a representation of a state in the state space for this game and give an expression for the size of the state space.

Representation of the state space: a tuple with the position of the agent, a tuple with the position of the skull, and the agents direction

Size of the state space: $2 \cdot N \cdot M$

Explanation of each term in the size of the state space:

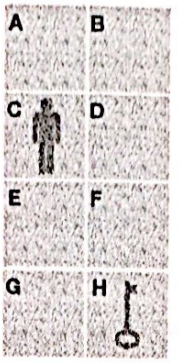
2 = possible facing directions of the agent
N = possible cell locations of the agent
M = possible cell locations of the skull

(ii) [4 pts] Please fill in the following pseudocode for the **getSuccessor** function for this game:

```
procedure GETSUCCESSOR(state)
    successors ← empty list
    for action ∈ { turn_left, turn_right, left, right, up, down,
                   left_up, left_down, right_up, right_down }  do
        if action ∈ {left, left_up, left_down} and state.facing_direction == right  then
            pass
        if action ∈ {right, right_up, right_down} and state.facing_direction == left  then
            pass
        if action ∈ {turn_right} and state.facing_direction == right  then
            pass
        if action ∈ {turn_left} and state.facing_direction == left  then
            pass
        next_state ← state.apply_action(action)
        if next_state .agent_position != state.skull  and next_state .agent_position is not wall  then
            successors.append(next_state)
    return successors
```

(iii) [2 pts] What is the goal test?

state.agent_position == key_position



(b) Now, consider a simplified version of the board above, which has **no skull** and **no facing-direction for the agent** (i.e., the agent can move in any of the 8 directions as long as it remains in the board). For the three following graph search algorithms, perform the search procedure yourself (**please show your work**) and provide answers to the questions below regarding the nodes expanded during the search as well as the final path found by the algorithm.

On this board, assume that a diagonal move has a cost of 3, whereas moving left, right, up, or down has a cost of 1. Do notice the difference in costs, and recall which algorithms use this cost information and which algorithms do not.

Remember that the search procedure should begin at the agent's starting position (C). To break ties when adding nodes of equal cost to the fringe, follow alphabetical order.

Finally, when listing the order/number of nodes expanded, do not include nodes which are taken off the fringe but discarded immediately due to already having been visited.

(i) [3 pts] **Breadth first graph search**
Fringe Data structure: queue
Recall that BFS computes the smallest number of steps, $b(v)$, taken to get to a node v from the start node.

Order of nodes expanded:

C, A, D, E, G, B, F, H

Number of nodes expanded:

8

Path returned:

C E H

Length of path:

2

Cost of path:

4

What is $b(A), b(B), \ldots b(H)$?

State s	A	B	C	D	E	F	G	H
$b(s)$	1	3	0	1	1	3	2	4

(ii) [3 pts] **Uniform cost graph search**
Fringe data structure: priority queue (make sure you update/reorder the whole PQ after each addition)
Recall that UCS keeps track of the lowest cost, $c(v)$, to get from the start node to the node v.

Order of nodes expanded:

C, A, D, E, B, F, G, H

Number of nodes expanded:

8

Path returned:

C E G H

Length of path:

3

Cost of path:

3

What is $c(A), c(B), \ldots, c(H)$?

State s	A	B	C	D	E	F	G	H
$c(s)$	1	2	0	1	1	2	2	3

(iii) [4 pts] **A* graph search (with Manhattan distance to the goal as the heuristic)**
Fringe data structure: priority queue (make sure you update/reorder the whole PQ after each addition)
Recall that A* computes $f(v)$ for the nodes v that it expands, with $f(v) = c(v) + h(v)$ where $c(v)$ is the lowest cost to reach v from the start node and $h(v)$ is an estimate of the distance from v to the goal.

Order of nodes expanded during the search:

C, D, E, F, H

Number of nodes expanded during the search:

5

Path returned by the search:

C, D, F, H

Length of path returned by the search:

3

Cost of path returned by the search:

3

What is $f(A), f(B), \ldots, f(H)$? Note that here, we are asking for the true $f(v)$ values as dictated by the definition, which is the value populated by the search algorithm only if it were to expand every node. This particular search problem doesn't end up expanding all nodes, so the $f(v)$ estimate maintained by the algorithm on the queue is not the true $f(v)$ value that we're asking for. Hint: you can fill out these values directly by looking at the board.

State s	A	B	C	D	E	F	G	H
$f(s)$	5	5	3	3	3	3	3	3

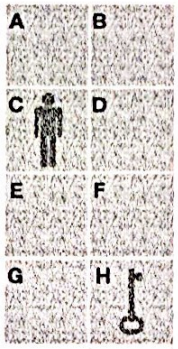
(c) [10 pts] Given your answers above, what are the qualitative differences between the results achieved by BFS, UCS, and A*? Which one finds the shortest path (in number of steps)? Which one finds the optimal path (in cost)?

Both UCS and BFS expand 8 nodes, but UCS has a lower path cost than BFS.

A* expands fewer nodes than UCS and it has the same length shortest path and path cost.

In terms of number of steps BFS has fewer steps than both UCS and A*

But UCS and A* find the optimal cost path



(d) For the same board and setting as part (b), give an example for each of the following types of heuristics. Please briefly explain why the example you chose satisfies the requested properties.

(i) [3 pts] Admissible and consistent.
Note: You can use a heuristic that we have frequently used in this class, or you can just assign any numbers that qualify as an admissible and consistent heuristic.

State s	A	B	C	D	E	F	G	H
Heuristic $h(s)$	4	3	3	2	2	1	1	0

Explanation: We will use Manhattan Distance to the goal as our heuristic
Let X & Y be 2 nodes, WLOG assume X is further from the goal than Y.
Case 1: X & Y are adjacent. This means the difference in their Manhattan distance to the goal is 1, which is the cost of edge E(X,Y) for adjacent nodes which is 1
Case 2: X & Y are diagonal. This means the difference their Manhattan distance to goal is 2, which is less than the edge cost of edge E(X,Y) which is 3
So $h(x) - h(y) \leq \text{cost}(x,y)$

(ii) [4 pts] Admissible but inconsistent

State s	A	B	C	D	E	F	G	H
Heuristic $h(s)$	4	2	3	2	2	1	1	0

Explanation: for every state s, $h(s) \leq h^*(s)$ guarantees admissibility
However, for states A and B $h(A) - h(B) = 2 > \text{cost}(A,B) = 1$
so the heuristic is inconsistent

(iii) [3 pts] Inadmissible and inconsistent

State s	A	B	C	D	E	F	G	H
Heuristic $h(s)$	10	3	3	2	2	1	1	0

Explanation:
the heuristic is inadmissible (and therefore inconsistent)
because $h(A) = 10 > h^*(A) = 4$

(e) [10 pts]

In the previous questions, perhaps you used "relaxed" heuristics; that is, you estimated the true cost of something by evaluating the cost for a simpler version of the problem (which is easier to compute). For example, using euclidean distance would be a "relaxed" heuristic to estimate the length of the shortest path from Arad to Bucharest in Romania.

Formally, we will define a **relaxed heuristic** as a function on a state that returns a value that is always admissible and consistent. In this problem, we will consider two changes ("skull" and "teleporation") to the board/game above, and we will reason about the effect of these changes on the consistency of heuristics.

(i) [5 pts] **"Skull":**

For this new version of the board, your friend Ethan suggests adding the skull back to the old board setting from part (b), and having the skull move back and forth between the cells E and F.

1. How does the presence of this skull change the state space of the game from part (b), which was (x, y)? Does anything need to be added or removed?
2. Ethan argues that in this new board, at least one previously consistent heuristic can become inconsistent. Is Ethan right?

○ Yes, and I will give an example below.
◉ No, and I will provide a proof below.

Note: we define heuristics for this problem as being a function of **only** the cell location: They cannot incorporate the location of the skull, since that did not exist in the old version of the board that we are comparing to.

If you believe Ethan is right, give an example of a heuristic that used to be consistent on the old board but is no longer consistent on this new board. Make sure to explain why it is no longer consistent (perhaps with a drawing of a board state and an explanation.

State	A	B	C	D	E	F	G	H
$h(s)$								

If you believe Ethan is wrong, provide an argument for why a heuristic that was consistent in the old board must also remain consistent in this new board. Be specific about your reasoning and use mathematical quantities such as heuristic costs of states $h(a)$ and true costs of transitions $c(ab)$.

The addition of the skull means that the edge cost for each edge is strictly non decreasing. This means for pair of nodes that have an edge between each other that were previously consistent must still be consistent.

Let $cost_b(X,Y)$ be the edge cost of $E(X,Y)$ without the skull
$cost_c(X,Y)$ with

Then $h(X) - h(Y) \leq cost_b(X,Y) \leq cost_c(X,Y)$ which implies consistency

(ii) [5 pts] **"Teleportation":**

For this new version of the game, your friend Nancy suggests taking the old game setting from part (b) and now adding the ability for the agent to perform a maximum of 1 "teleportation" action during the game. That is, on one of the agent's moves, it can choose to jump from its current state to any other non-goal state on the board.

1. How does this new teleportation ability change the state space of the game from part (b), which was (x, y)? Does anything need to be added or removed?
2. Nancy argues that in this new game, at least one previously consistent heuristic can become inconsistent. Is Nancy right?

- (●) Yes, and I will give an example below.
- ( ) No, and I will provide a proof below.

Note: as in the "skull" question above, we define heuristics for this problem as being a function of **only** the cell location: They cannot incorporate anything that did not exist in the old version of the game that we are comparing to.

If you believe Nancy is right, give an example of a heuristic that used to be consistent in the old game but is no longer consistent in this new game. Make sure to explain why it is no longer consistent (perhaps with a drawing of a board state and an explanation).

State	A	B	C	D	E	F	G	H
$h(s)$	4	3	3	2	2	1	1	0

Suppose teleporting costs 1. (Stated on Piazza)

Consider cells diagonal to one another like A and D. Using a consistent heuristic like Manhattan Distance to goal. $h(A) - h(D) = 2$, but teleporting from A to D only costs 1 so $h(A) - h(D) = 2 \not\leq \text{cost}(A, D) = 1$ which means this is inconsistent.

If you believe Nancy is wrong, provide an argument for why a heuristic that was consistent in the old game must also remain consistent in this new game. Be specific about your reasoning and use mathematical quantities such as heuristic costs of states $h(a)$ and true costs of transitions $c(ab)$.

Q2. [50 pts] Expectimax Yahtzee

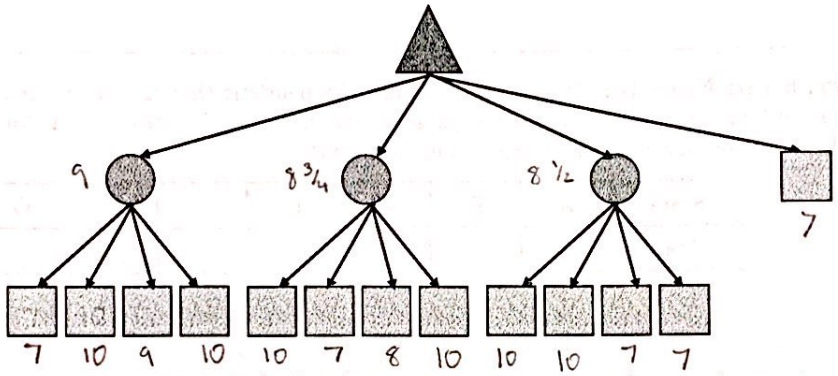
Consider a simplified version of the game *Yahtzee*. In this game, we have 3 dice with 4 sides each (numbered 1-4) and the game begins by rolling all 3 dice. At this point, a player can make a decision: pick one of the 3 dice to reroll, or don't reroll anything. Then, points are assigned as follows. A reward of 10 points is given for two-of-a-kind and a reward of 7 points is given for rolling a series (1-2-3 or 2-3-4). Otherwise, the score is equal to the sum of all 3 dice. If a special roll is achieved (series or two-of-a-kind) but the sum of dice is higher, the max is always taken.

(a) Formulate this problem as an expectimax tree.

(i) [3 pts] The resulting tree for the problem is drawn below. Given a specific initial roll, what is the branching factor (of the player's decision) from this root node? What is the branching factor at the chance nodes, and what do those chance nodes represent?

Branching factor of root node = 4

Branching factor of chance node = 4



(ii) [7 pts] Given a starting roll (1,2,4) (each index in the tuple corresponding to die number 1, 2, and 3), what move should you take? Fill in the values of the expectimax tree above to justify your answer.

roll combo	score
1,2,4	7
2,2,4	10
3,2,4	9
4,2,4	10
1,1,4	10
1,3,4	8
1,4,4	10
1,2,1	10
1,2,2	10
1,2,3	7

Chance node reroll 1 = $\frac{1}{4}(7+10+9+10) = 9$

reroll 2 = $\frac{1}{4}(10+7+8+10) = 8\frac{3}{4}$

reroll 3 = $\frac{1}{4}(10+10+7+7) = 8\frac{1}{2}$

We should reroll the first die according to the expectimax tree

Now suppose the human player does not understand how to play the game, and as a result, they choose any action with uniform probability, regardless of the initial roll. We employ the use of a 'helpful robot' that has the power to perform the action selected by the human. Given a configuration of dice and the desired action from the human, this robot either actually implements the human's action (with probability $1-p$) or overrides it with a 'no reroll' action (with probability p).

(b) In the setup of this 'helpful robot', answer the following questions:

(i) [8 pts] Let a, b, c and d be the reward from the actions 'reroll die 1', 'reroll die 2', 'reroll die 3', and 'no reroll', respectively.

In terms of a, b, c and d, what is R_H, the expected reward after the human takes an action? What is R_{RH}, the expected reward after the robot intervenes and performs its chosen action? Please **show all steps of your work** and **write your expression into the form of** $R+Qp$, where R and Q are expressions that contain a, b, c and d but not p.

Hint: it might be helpful to draw out the tree.

$R_H =$ $\frac{1}{4}(a+b+c+d)$

$R_{RH} =$ $(1-p)R_H + p(d)$

$= (1-p)(\frac{1}{4}(a+b+c+d)) + pd$

$= \frac{1}{4}(a+b+c+d - p(a+b+c+d)) + pd$

$= \frac{1}{4}(a+b+c+d) + (-\frac{1}{4}(a+b+c) + \frac{3}{4}d)p$

So $R = \frac{1}{4}(a+b+c+d)$

$Q = -\frac{1}{4}(a+b+c) + \frac{3}{4}d$

(ii) [2 pts] What is the condition for our "helpful robot" being helpful (i.e., increasing expected reward)? Your answer should only contain R_H, R_{RH}, $\geq$.

$R_{RH} \geq R_H$

(c) For (i) to (iii) below, let's investigate the situation where $a = 5$, $b = 5$, $c = 5$ and $d = 10$. (numbers are chosen to simplify calculation).

(i) [2 pts] What is R_H and R_{RH}? Your answer may contain p.

$R_H =$ $\frac{1}{4}(5+5+5+10) = \frac{25}{4}$

$R_{RH} =$ $\frac{1}{4}(5+5+5+10) + \left(-\frac{1}{4}(5+5+5) + \frac{3}{4}(10)\right)p$

$= \frac{25}{4} + \frac{15}{4}p$

(ii) [1 pt] What is the range of values for p where our "helpful robot" is actually being helpful **in this situation?**
Hint: can p be greater than 1 or smaller than 0?

The robot is helpful for all $p \in [0,1]$

(iii) [2 pts] What's your interpretation of this range? At what value of p is R_{RH} the largest, and what does that mean (in words)?

Regardless of the p value, the robot will always be helpful.
R_H is largest when $p=1$ so we should always have the robot override the human with a no reroll action

For (iv) to (vi), let's investigate the situation where $a = 10$, $b = 10$, $c = 5$ and $d = 7$. (numbers are chosen to simplify calculation).

(iv) [2 pts] What is R_H and R_{RH}? Your answer may contain p.

$R_H =$ $\frac{1}{4}(10+10+5+7) = 8$

$R_{RH} =$ $\frac{1}{4}(10+10+5+7) + \left(-\frac{1}{4}(10+10+5) + \frac{3}{4}(7)\right)p$

$= 8 - p$

(v) [1 pt] What is the range of values for p where our 'helpful robot' is actually being helpful **in this situation?**
Hint: can p be greater than 1 or smaller than 0?

The robot can only be helpful if $p = 0$

(vi) [2 pts] What's your interpretation of this range?

The robot is only helpful if H always chooses the humans action. and never overrides with no reroll

(d) Now let's step back and see what we have done with the helpful agents.

(i) [2 pts] Based on the math from previous parts, could you come up with a condition where $p \neq 0$ for our 'helpful robot'? Your condition should only contain $a, b, c, d, \geq$.

$R_{RH} \geq R_H \Rightarrow \frac{1}{4}(a+b+c+d) + (-\frac{1}{4}(a+b+c) + \frac{3}{4}d)p \geq \frac{1}{4}(a+b+c+d)$

$(-\frac{1}{4}(a+b+c) + \frac{3}{4}d)p \geq 0$

$\frac{3}{4}d \geq \frac{1}{4}(a+b+c)$

$d \geq \frac{1}{3}(a+b+c)$

(ii) [2 pts] In one sentence, please describe the situation when the condition above is true.
Hint: what does the reward of "no reroll" look like?

The reward of no reroll must be greater than or equal to the average of the rewards of rerolling the dice

(iii) [2 pts] Suppose that the condition you outline in (d)(i) and (d)(ii) occurs with 70% of all starting rolls. We would like to set a p such that our "helpful robot" will be the most helpful **in expectation**.
If you believe we can set such p, please specify its value and explain why it works. If you don't believe we can set such p, please explain why no such p exists.

The value for p, if exists = [ ]

● No such p exist (cannot distinguish the helpfulness between different p)

(iv) [4 pts] Explanation for the value of p or why no such p exist:

Since we do not the exact values a, b, c, and d will take on, we are unable to find find the expectation. In other words, even though 70% of the time d will be greater than $\frac{1}{3}(a+b+c)$, we do not know by how much. Nor do we know how much less d will be than $\frac{1}{3}(a+b+c)$ the other 30% of the time.

(e) Your friend Ethan (again) argues that our "helpful robot" should not only override the human player's "reroll" choice with probability p (and replace it with a "no reroll"), but also override the human player's "no reroll" choice with probability p (and replace it with the outcome of selecting one of the 3 dice at random and rerolling that dice).

Ethan claims that "Ethanbot", equipped wth this new feature, will help a random human player achieve better end reward **in expectation** than our "helpful robot."

(i) [5 pts] Give an example where "Ethanbot" is better (or worse) than our "helpful robot" in helping a random human player, if they override the human player's choice with the same p.

Intuitively, the difference with this new robot is that with probability p, it will override the humans decision to not reroll with some additional probability. This effectively reduces the weight on no reroll and increases the weight to rerolling. Thus, consider an example where the value of d is much greater than the average of $a+b+c$. Specifically if $d=4$ and $\frac{1}{3}(a+b+c)=3$

$R_{EH} = (a+b+c)(\frac{1}{4}-\frac{1}{4}p)+dp$

$= \frac{9}{4} - \frac{9}{4}p + 4p = \frac{9}{4} + \frac{7}{4}p$ so $R_{EH} > R_{new}$ so Ethanbot is worse than our helpful robot in this case

Ethan bot →

$R_E = (a+b+c)(\frac{1}{4}-\frac{1}{4}p) + \frac{1}{2}dp$

$= \frac{9}{4} - \frac{9}{4}p + 2dp = \frac{9}{4} + \frac{1}{2}p$

(ii) [5 pts] Describe a procedure to find the p_e where the "Ethanbot" will be the most helpful (increasing the end reward the most **in expectation**).

We can try an iterative hill climbing procedure

Initialize $p = 0.5$ and compute an expectimax game tree root value using Ethan bot with p equal to this probability.

Then increase p (perhaps by 0.05) and compute the expectimax value. If it is higher, repeat the procedure

If it is lower try the other direction and reduce p. If this is higher continue this procedure.

As we continue, reduce the amount that we change p by, each iteration.

Stop after n iterations.

Using this idea we will eventually converge on the most helpful value of p_e to maximize the reward in expectation.