CS 188 Spring 2020 Introduction to Artificial Intelligence

Written HW 1 Sol.

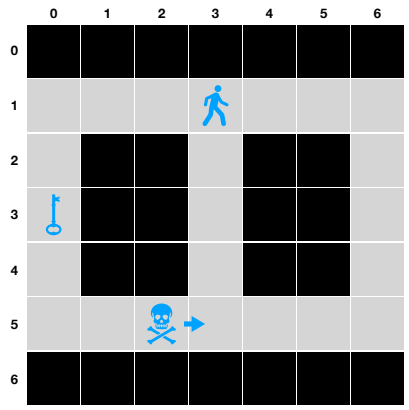
Solutions for HW 1 (Written)

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: $(x, y, facing\_direction, x\_skull, y\_skull)$

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

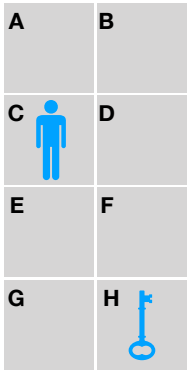
Explanation of each term in the size of the state space: There are N possible positions for the agent, 2 directions that the agent can face, and M possible positions for the skull.

(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, left_up, left_down, right, right_up, right_down, up, down] **do**
    if *action* ∈ {*left*, *left_up*, *left_down*} and *state.facing_direction* == [right] **then**
      continue
    if *action* ∈ {*right*, *right_up*, *right_down*} and *state.facing_direction* == [left] **then**
      continue
    if *action* ∈ {*turn_right*} and *state.facing_direction* == [right] **then**
      continue
    if *action* ∈ {*turn_left*} and *state.facing_direction* == [left] **then**
      continue
    next_state ← *state.apply_action*(*action*)
    if *next_state* [not in wall] and *next_state* [not collide with skull] **then**
      successors.append(*next_state*)
  return *successors*

(iii) [2 pts] What is the goal test?
Is our current location $(x_H, y_H) = (3, 0)$?



(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.

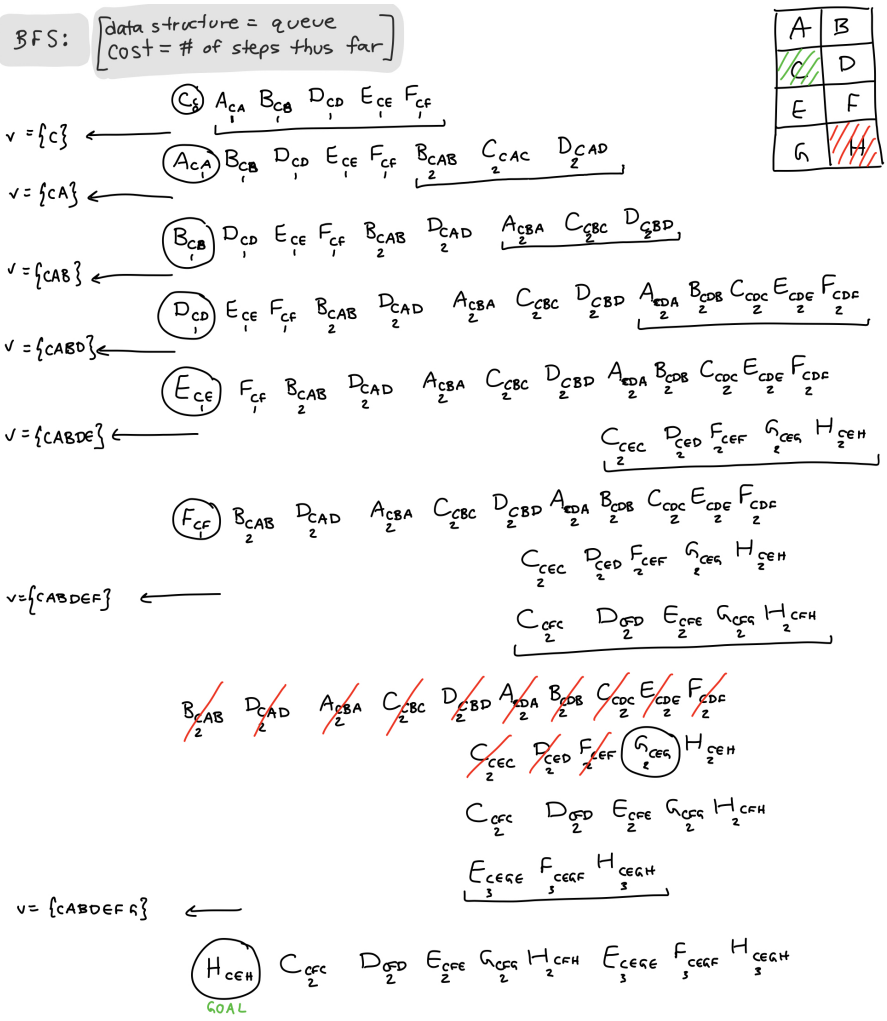
Finally, when listing the order/number of nodes expanded, do not include nodes which are taken off the fringe but discarded immediately due to already having been visited.

(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, B, D, E, F, G, H (can omit H)
Number of nodes expanded: 8 (or 7 if omit H)
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	1	0	1	1	1	2	2

SOLUTION:

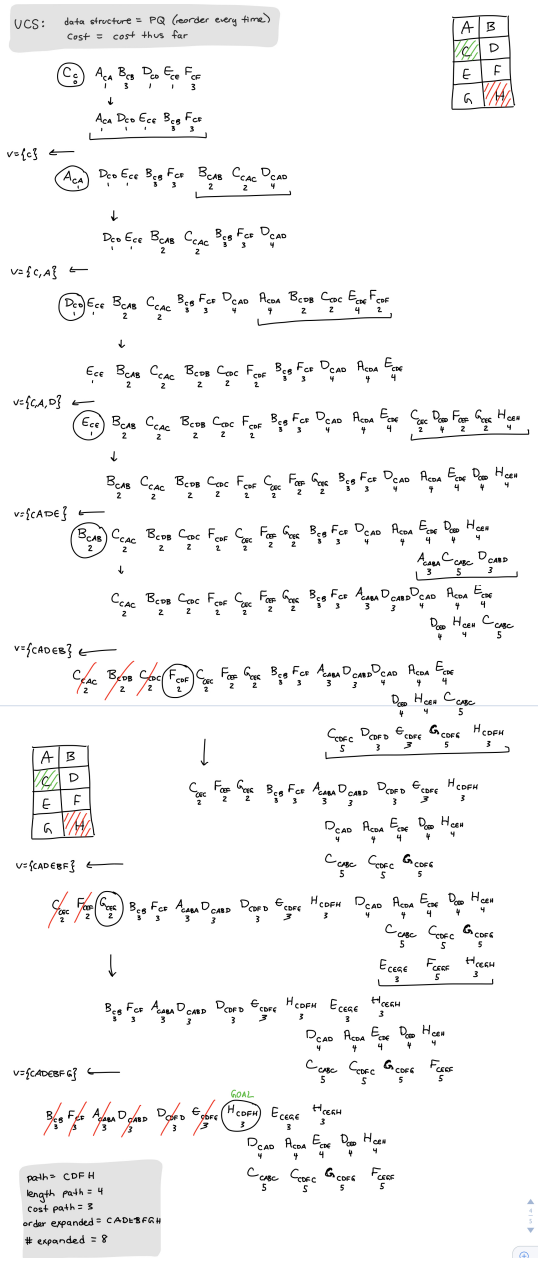


(ii) [3 pts] **Uniform cost graph search**
Fringe data structure: priority queue (make sure you update/reorder the whole PQ after each addition)
Recall that UCS keeps track of the lowest cost, $c(v)$, to get from the start node to the node v.

Order of nodes expanded: C, A, D, E, B, F, G, H (can omit H)
Number of nodes expanded: 8 (or 7 if omit H)
Path returned: C → D → F → 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

SOLUTION:

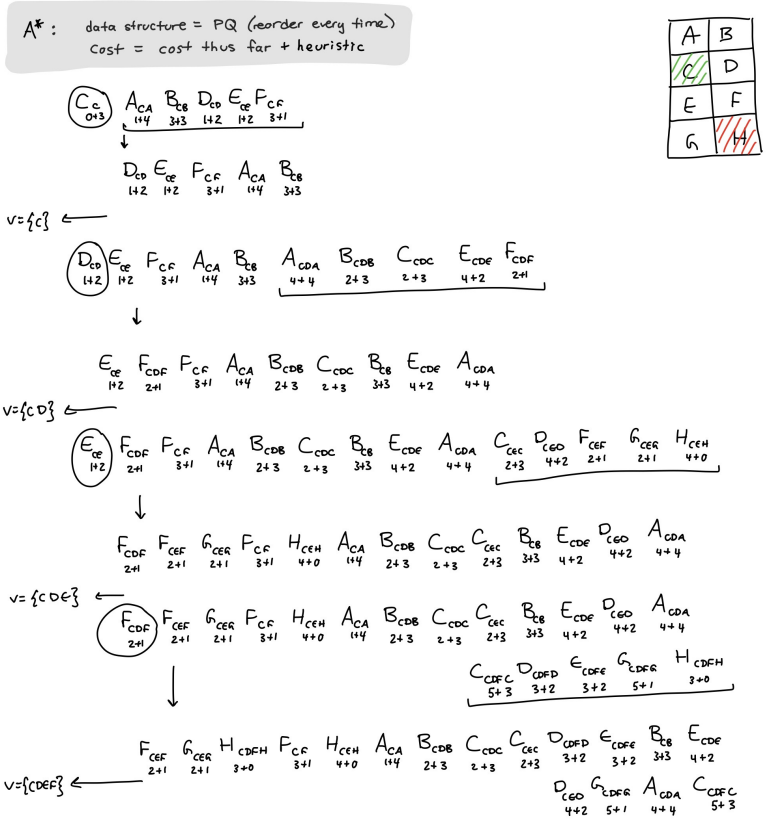


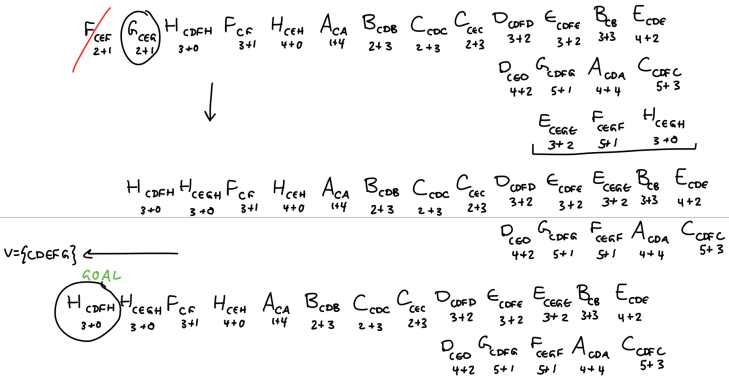
(iii) [4 pts] **A* graph search (with Manhattan distance to the goal as the heuristic)**
Fringe data structure: priority queue (make sure you update/reorder the whole PQ after each addition)
Recall that A* computes $f(v)$ for the nodes v that it expands, with $f(v) = c(v) + h(v)$ where $c(v)$ is the 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, G, H (can omit H)
Number of nodes expanded during the search: 6 (or 5 if omit H)
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
$f(s)$	1+4	2+3	0+3	1+2	1+2	2+1	2+1	3+0

SOLUTION:

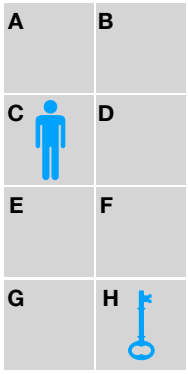




path = CDFH
length path = 4
cost path = 3
Order expanded = CDEFGH
expanded = 6

(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)?

BFS is able to find the most direct path (in number of steps).
UCS takes into account the cost of transitions and finds the path of optimal/least cost to the key. However, it has to expand nodes in the opposite direction of the key as it does not take into account any problem-specific information aside from path cost.
A* search finds the same optimal path as UCS but because of the heuristic, is able to do so while expanding fewer nodes.



(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: Manhattan Distance is admissible and consistent. Answers might vary

(ii) [4 pts] Admissible but inconsistent

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

Explanation: Manhattan Distance, except reduced $h(A)$ and $h(B)$. This makes them remain an underestimate, but this causes $A \to B$ and $A \to C$ to be overestimated. True cost is 1 for these, but difference in heuristic cost says 2, so we are overestimating the transition cost of the edge. Answers might vary but need to overestimate at least one edge

(iii) [3 pts] Inadmissible and inconsistent

State s	A	B	C	D	E	F	G	H
Heuristic $h(s)$	5	2	2	2	2	1	1	0

Explanation: Manhattan Distance, except overestimated cost-to-goal for A, overestimated $A \to B$. Answers might vary but need to 1. overestimate cost-to-goal from one location, 2. overestimate at least one edge

(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?

The location of the skull needs to be added to the state.

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 $c(ab)$.

For a consistent heuristic in the old board, $h(a) - h(b) \leq c(ab)$. In the new board, the cost of getting somewhere is strictly greater than or equal to the cost of getting there in the old board. Thus, $c_{new}(ab) \geq c(ab)$, which means $h(a) - h(b) \leq c_{new}(ab)$ and thus the heuristic is still consistent in this new board.

(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?

Whether or not the teleporation step has been taken should be added to the state.

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

The old heuristic will no longer be consistent because the true cost of any transition can now be 1, due to teleportation. In this case, $h(a) - h(b)$ using the old heuristic values could definitely overestimate this true cost.

Self-grade note: considering the cost of teleportation to be 0 or 1 are both acceptable.

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 $c(ab)$.

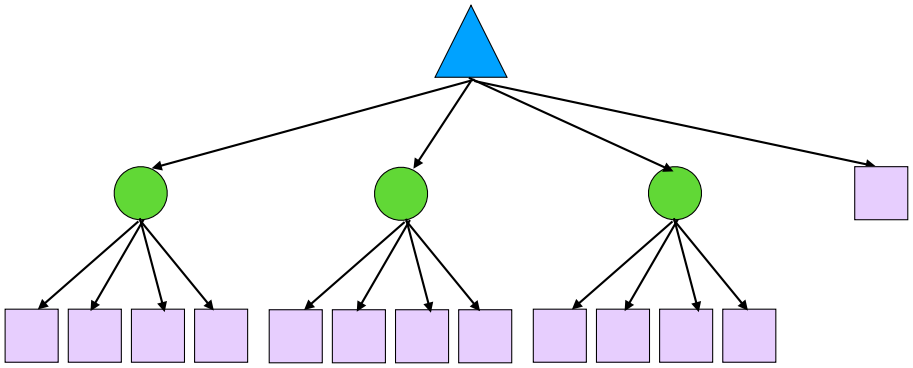
Q2. [50 pts] Expectimax Yahtzee

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.

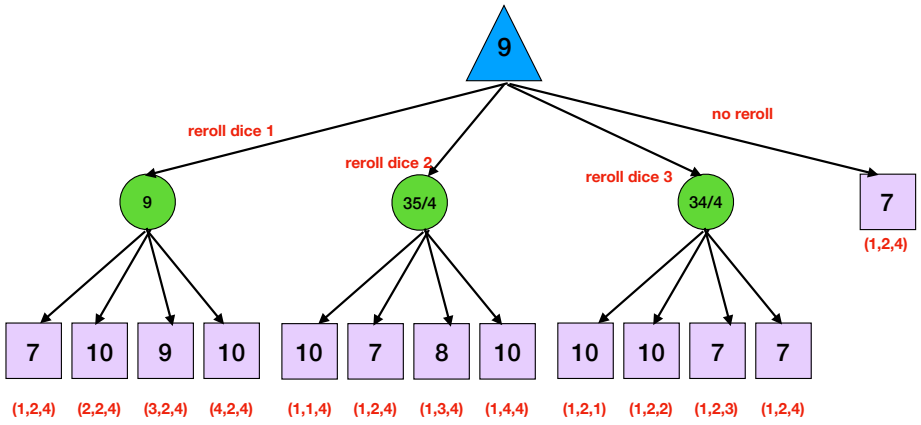
(i) [3 pts] The resulting tree for the problem is drawn below. Given a specific initial roll, what is the branching factor (of the player's decision) from this root node? What is the branching factor at the chance nodes, and what do those chance nodes represent?

The branching factor from the root node is 4. This corresponds to the different actions that may be taken after the initial roll (reroll die 1, reroll die 2, reroll die 3, don't reroll any). The branching factor from the chance nodes is also 4, corresponding to the 4 sides of the die.



(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.

The agent should reroll die 1.



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.

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

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

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

(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.

$R_H = \frac{5+5+5+10}{4} = 6.25$

$R_{RH} = \frac{(5+5+5+10)}{4} + \frac{3*10-(5+5+5)}{4}p = 6.25 + \frac{15}{4}p$

(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?

$R_{RH} \geq R_H$ as long as $p \geq 0$, therefore, the robot is being 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)?

The robot should intervene and not allow the human to reroll any die. $p=1$ will yield the largest R_{RH}

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.

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

$R_{RH} = \frac{(10+10+5+7)}{4} + \frac{3*7-(10+10+5)}{4}p = 8 - p$

(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?

$R_{RH} \geq R_H$ if and only if $p=0$, therefore, the robot is being helpful for all $p=0$

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

The robot should never intervene and always allow human to reroll any die.

(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 $a, b, c, d, \geq$.

$3d - (a+b+c) \geq 0$

(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?

"no reroll" gives a higher reward than the average of rerolling one of the dice. (re-arrange the inequality to $d \geq \frac{a+b+c}{3}$

(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:

This question is a hard one (kind of unintentionally), and there are many ways to think about this. We list a few below.

1. Let "needing intervention" as an indicator variable X, where $P(X = 1) = 0.7$ and $P(X = 0) = 0.3$, and "robot actually intervening" as an indicator variable Y, where $P(Y = 1) = p$ and $P(Y = 0) = 1 - p$. It's pretty clear that our robot is helpful if and only if $X = Y$, and that happens when $X = 0$ and $Y = 0$, or $X = 1$ and $Y = 1$. Since X, Y are independent, $P(X, Y) = P(X)P(Y)$. Then the expectation of helpfulness with our parameter p can be written as $0.7p + 0.3(1 - p) = 0.3 + 0.4p$, which is linear in p, maximized at $p = 1$. So $p = 1$ is the correct answer if you went in this direction.

2. If one considers the actual expected value of the game reward, this is a lot more trickier. The big idea here is that we not only need to consider if $d \geq \frac{a+b+c}{3}$, but also how much of an advantage we get when the robot is actually helpful (intervened at the right time), and how much of an disadvantage we get when the robot is unhelpful (intervenes at the wrong time). If you use this interpretation, please refer to the thought process of part (2.e.ii).

3. (The meta level:) If you coded up Yahtzee in the problem setting and run the computer simulation by rolling out every individual cases and computing the percentage of time $d \geq \frac{a+b+c}{3}$ you will realize that number is not 70%. This means the dice we are rolling is BIASED!!! If you took this route, you will conclude that we need more information about the dice before we can determine the best value for p.

(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.

The expected reward after Ethanbot intervenes and performs its chosen action is $R_{EH} = \frac{a+b+c+d}{4} + \frac{3d-(a+b+c)}{6}p$.

You soon notice that the condition under which Ethanbot is helpful is actually the same as the condition under which our robot is helpful. Therefore, for any set of a, b, c, d such that $3d - (a + b + c) \geq 0$, our robot is more helpful than Ethanbot.

Self-grade note: You don't need to come up with the whole argument above. Just listing a set of a, b, c, d, calculate the correct R_{RH} and R_{EH} for those values, and show that $R_{RH} > R_{EH}$ is sufficient for the full 5 points on e(i). But if you got the whole proof, great!

Food for thoughts: What does this mean? Does this mean that Ethanbot is always less helpful than our robot?

(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**).

This question is fairly open-ended. Contrary to part 2.d, we are specifically dealing with increasing the end reward in expectation here.

From part (d), we know that Ethanbot increases our expected end reward and is considered helpful when $3d - (a + b + c) \geq 0$. Since the expected reward is linear on p, the way to maximize the expected reward is to set $p = 1$ when $\mathbb{E}[3d - (a + b + c)] \geq 0$ and $p = 0$ otherwise.

$\mathbb{E}[3d - (a + b + c)] = 3 * \mathbb{E}[d] - \mathbb{E}[a] - \mathbb{E}[b] - \mathbb{E}[c]$ from the linearity of expectation. So to find the optimal value for p, we only need to know the expected values for a, b, c, d and from there can extract whether p is 0 or 1.

However a, b, c, d are dependent on the initial roll which we don't know ahead of time. To deal with this uncertainty and calculate the expectation of a, b, c, d, we can do the following.

1. Explicitly roll out all possible initial rolls, and calculate the a, b, c, d of each initial roll, and compute R_H and R_{RH} for those quantities in each case. This yields an expression parameterized by p_e, and we simply need to calculate the p_e which maximizes this expression.

2. If procedure 1 requires too much computing power and you are unable to do a full roll out to compute the entire tree, you may instead use a sampling approach. Randomly roll out N initial rolls (install of all rolls), and perform the same procedure get the p_{eN}, the best estimate of p_e when we randomly roll out N initial rolls. Note p_{eN} will serve as a good estimate of the true p_e if the number of samples, N, is sufficiently large.