CS 188 Summer 2020 Introduction to Artificial Intelligence Midterm 1

- You have approximately 110 minutes.
- The exam is open book, open calculator, and open notes.
- In the interest of fairness, we want everyone to have access to the same information. To that end, we will not be answering questions about the content or making clarifications.
- For multiple choice questions,
  - ☐ means mark **all options** that apply
  - ○ means mark a **single choice**

First name	
Last name	
SID	

For staff use only:

Q1.	Plants vs. Zombies	/22
Q2.	Climbing	/21
Q3.	RL: Stochastic Policies	/20
Q4.	Pacfriends Unite	/18
Q5.	Searching for a Problem...	/19
	Total	/100

THIS PAGE IS INTENTIONALLY LEFT BLANK

Q1. [22 pts] Plants vs. Zombies

Dr. Zomboss wants your brain and is having zombies invade your backyard!

(a) [2 pts] You currently have the Peashooter **(P)**, but to defend yourself from the zombies, you need to purchase more plants from Crazy Dave! He has a plant lottery **(L)**, which gives either the Cherry Bomb **(C)**, the Wall-nut **(W)**, or the Snow Pea **(S)** with equal probability of each.

Assume rational preferences $P \succ W$, $L \succ P$, and $S \succ C$. Which of the following are guaranteed to be correct?

☐ $L \succ C$
☐ $C \sim L$
☐ $C \succ L$
☐ $P \succ C$
☐ $C \sim P$
☐ $C \succ P$
[✓ $S \succ L$]
○ None of the Above

Let $U(S) = 175$, $U(C) = 150, U(P) = 100$, and $U(W) = 50$, and we have $U(L) = 125$. Then, we do have $P \succ W$, $L \succ P$, and $S \succ C$ and in this case, $U(C) > U(L) > U(P)$.
Let $U(S) = 325$, $U(C) = 75, U(P) = 100$, and $U(W) = 50$, and we have $U(L) = 150$. Then, we do have $P \succ W$, $L \succ P$, and $S \succ C$ and in this case, $U(L) > U(P) > U(C)$.
Hence neither $L \succ C$, $C \sim L$, $C \succ L$, $P \succ C$, $C \sim P$, or $C \succ P$ is guaranteed.
$L \succ P \succ W$, which means $\frac{U(S)+U(C)+U(W)}{3} > U(P) > U(W)$.

$$\begin{aligned}
&\frac{U(S)+U(C)+U(W)}{3} > U(W)\\
\iff& U(S)+U(C)+U(W) > 3\cdot U(W)\\
\iff& U(S)+U(C) > 2\cdot U(W)\\
\iff& U(S)+U(C)+2\cdot U(S)+2\cdot U(C) > 2\cdot U(W)+2\cdot U(S)+2\cdot U(C)\\
\iff& 3\cdot(U(S)+U(C)) > 2\cdot(U(W)+U(S)+U(C))\\
\iff& 0.5\cdot(U(S)+U(C)) > \frac{1}{3}\cdot(U(W)+U(S)+U(C))\\
\iff& 0.5\cdot(U(S)+U(C)) > U(L)
\end{aligned}$$

And since $S \succ C$, $U(S) > U(C)$ so $U(S) > 0.5\cdot(U(S)+U(C)) > U(L)$.
Another way to think: if S is not preferred over L, then neither S nor C is preferred over L, and since L is a lottery of S, C and W, W cannot be less preferred over L, which contradicts with $L \succ P \succ W$.

Now it's time to battle the zombies!

(b) You have M grass lanes and N types of plants. In each turn, Dr. Zomboss chooses a grass lane and puts a zombie on the right end of the lane. Then, you choose a plant to put on the left-most slot of that lane. You are trying to maximize utility and Dr. Zomboss is trying to minimize your utility.

Consider the game tree for one turn, and run alpha-beta pruning.

(i) [2 pts] For $M = 5$ and $N = 3$:

the maximum possible number of leaf nodes pruned = $\boxed{8}$.

the minimum possible number of leaf nodes pruned = $\boxed{0}$.

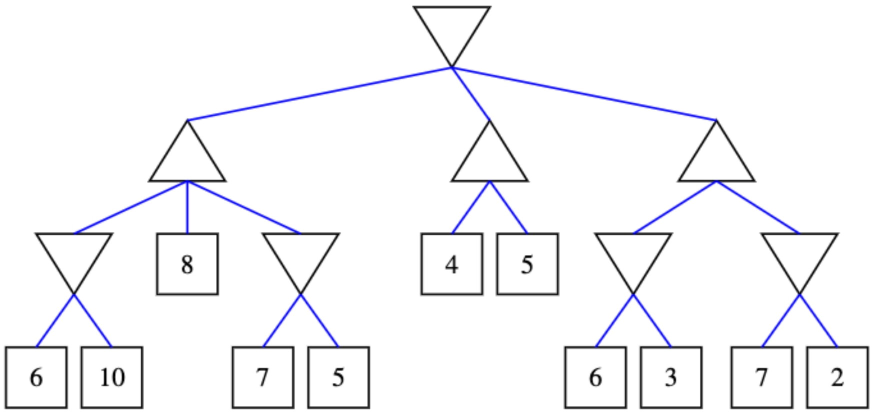
(ii) [1 pt] For $M = 6$ and $N = 10$:

the maximum possible number of leaf nodes pruned = $\boxed{45}$.

In minimax, it's always possible that none of the nodes can be pruned, so the minimum possible pruning is 0.

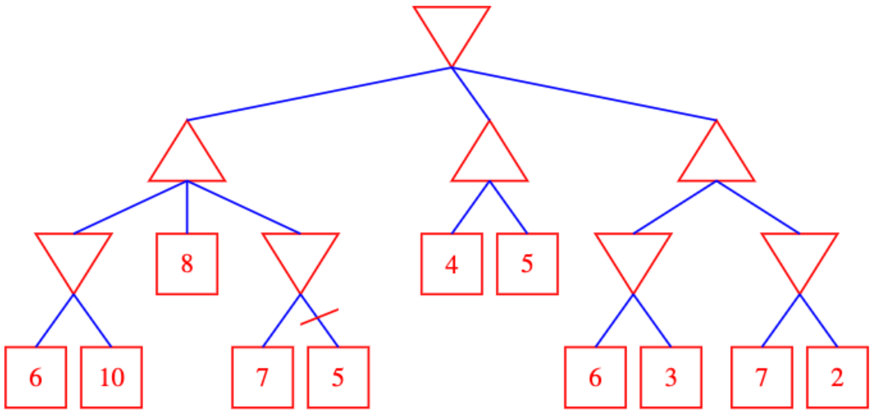
The maximum pruning for a depth 2 minimax is that except we need to see all the leaf nodes of the first branch of the root, for each other branch of the root, we prune all but the first leaf node. That is, we prune $(M-1)(N-1)$ nodes. Plug this in for your value of M and N.

(c) Consider the following game tree:



(i) [1 pt] Using alpha beta pruning, how many leaf nodes can we prune? Assume that branches are visited in left to right order.

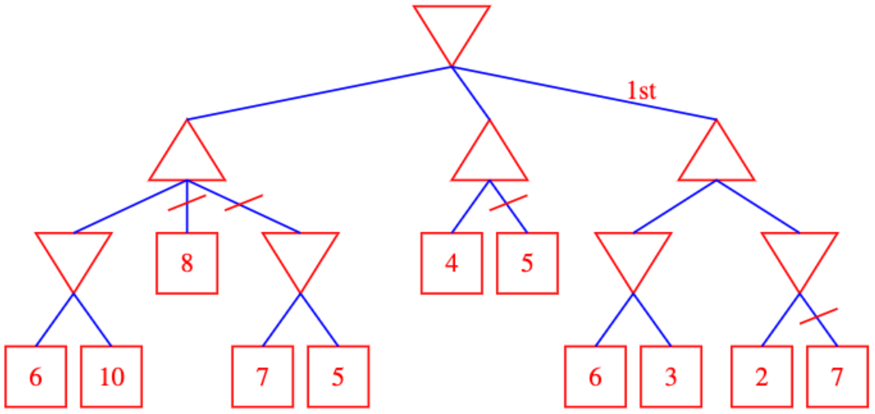
$\boxed{1}$.



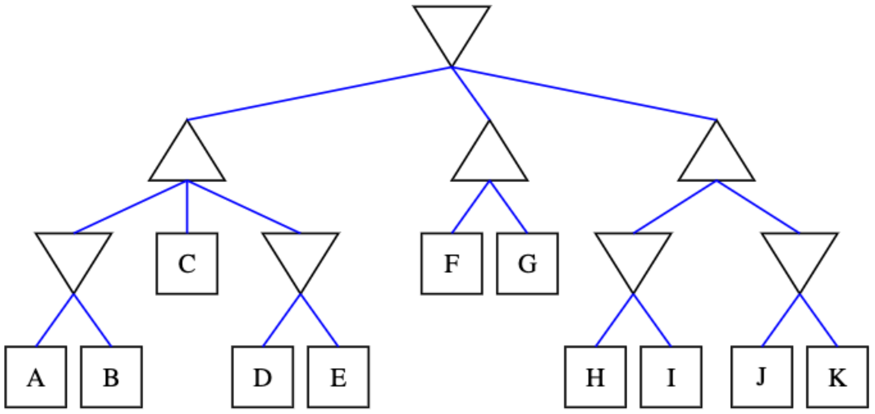
(ii) [2 pts] If the branches of every minimizer node are reordered such that we prune the maximum number of leaf nodes, how many leaf nodes can now be pruned? Assume that children of maximizer nodes are visited from left to right, and that we are not pruning on equality.

$\boxed{5}$.

The right branch of the root contains the value that got propagated to the root, so let's visit that branch first. Note that this doesn't mean that visiting other branches first cannot prune maximally. For simplicity, we are just giving one ordering.
In the right branch, swap "7" and "2" so that "7" gets pruned.
It doesn't really matter whether we visit the middle branch next or the left branch.
In the middle branch, we can prune "5" since upon seeing "4", we know that the maximizer mode is at least a 4, so the root would not choose this branch. We cannot prune more than this.
In the left branch, after getting the value of the first minimizer (6), we know that the maximizer mode is at least a 4, so the root would not choose this branch. Hence, we are not checking the 3 leaf nodes corresponding to "8", "7" or "5". If we can reorder branches of max nodes, we could be able to prune 4 leaf nodes, but that's not what we can do for this problem. No reordering of branches of min nodes can prune more than these 3 leaf nodes.



(d) **Assume that we are not pruning on equality.** In this part, we have a fixed traversal order from left to right. We start with the tree in the previous part, and **shuffle the values of all the leaf nodes** such that we **check as few nodes as possible** without changing the structure of the tree. The initial ordering (as in the tree presented in the previous part) is 6, 10, 8, 7, 5, 4, 5, 6, 3, 7, 2, and the new ordering is $A, B, C, D, E, F, G, H, I, J, K$.



(i) [2 pts] For all possible new orderings, which of the results (root values) are possible?

☐ 2 ☐ 3 [✓ 4] [✓ 5] [✓ 6] ☐ 7 ☐ 8 ☐ 10

The maximal pruning can have E, G, J and K pruned.
The root is selecting the minimum in the three maximizers, so one of the maximizers must have the result, and the other two maximizers are larger. If that maximizer is the middle or right one, the result could be 3 (but not 2, since the maximizer is never choosing 2). But in this case, we cannot prune node E. The smallest result is therefore 4, where we have min(A, B), C, and min(D, E) each taking a value from 2, 3 and 4.

The largest result is 6. The right branch of the root must be smaller than the maximum leaf in that branch, since the maximum leaf cannot get selected by either 2 minimizers in that branch. The best we can have is the second largest in the branch. Let one of F or G is large and C is large. If the right branch of the root has result > 6, it has two of 7, 8, 10 in the branch, and one of the left/middle branch of the root would have max 6. Else, the right branch of the root has max 6. So for the root, 7, 8, and 10 are impossible.

(ii) [3 pts] Suppose that for the new ordering, the value of the root is 4. Which of the following leaf nodes are guaranteed to have value ≤ 5?

☐ A ☐ B [✓ C] [✓ D] ☐ E ☐ F ☐ G
☐ H ☐ I ☐ J ☐ K ○ None of the above

(iii) [3 pts] Suppose that for the new ordering, the value of the root is 5. Which of the following leaf nodes are guaranteed to have value > 5?

☐ A ☐ B ☐ C ☐ D ☐ E [✓ F] ☐ G
[✓ H] [✓ I] ☐ J ☐ K ○ None of the above

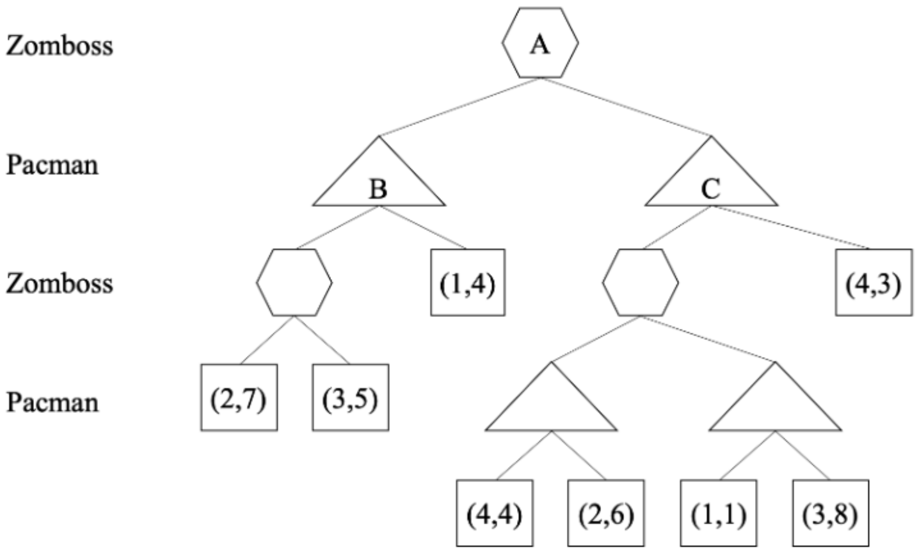
For root = 4, it's sufficient and necessary to have one of min(A, B) and C to be "4", and the other is "2" or "3" while D also takes "2" or "3". So C and D are guaranteed to be ≤ 5, but A and B can take any value as long as one of them is < 5. All other nodes may take values > 5.

For root = 5, it’s sufficient and necessary to have one of min(A, B) and C to be “5", and the other ≤ 5, while D takes “2", “3" or “4" since we do not prune on equality. F, H, and I are larger than 5, so that G, J, K are pruned.

(e) You are traveling and have asked Pacman to be in charge of your backyard, but Dr. Zomboss thinks you are the one at home, and is again coming for your brain!

Pacman knows Dr. Zomboss's utility, since you've been talking about it all the time, but **Dr. Zomboss doesn't know Pacman's utility**. However, Pacman does not care if zombies come into the house; he only wants more plants in the backyard. Both Pacman and Dr. Zomboss are playing optimally, trying to maximize their own utilities.

All nodes are in the format of **(Dr. Zomboss's utility, Pacman's utility)**. Complete the tree below. In the case of a tie, choose the leftmost branch:



(i) [2 pts] Write node B as a tuple $(B_{Zomboss}, B_{Pacman})$.

$B_{Zomboss} = \boxed{3}$, $B_{Pacman} = \boxed{5}$

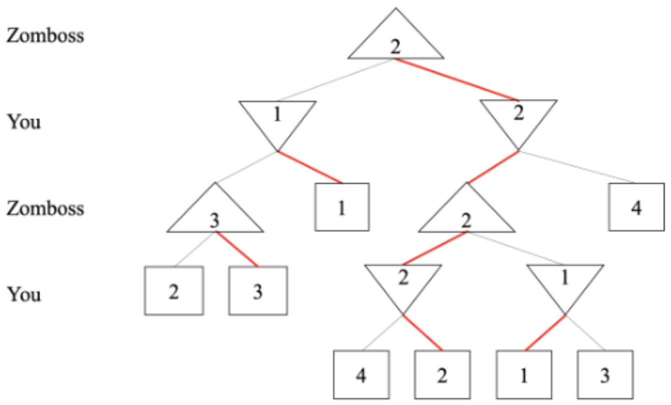
(ii) [2 pts] Write node C as a tuple $(C_{Zomboss}, C_{Pacman})$.

$C_{Zomboss} = \boxed{2}$, $C_{Pacman} = \boxed{6}$

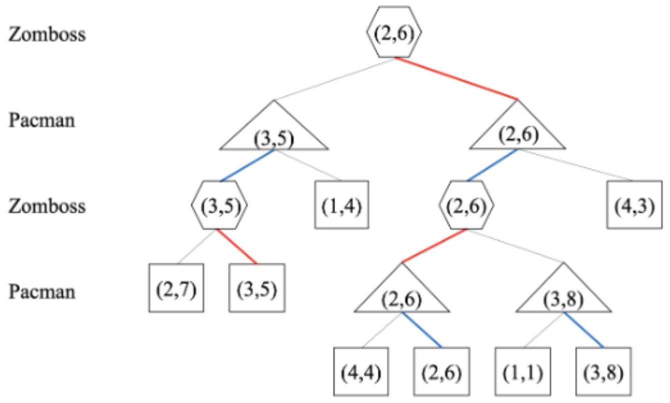
(iii) [2 pts] Write node A as a tuple $(A_{Zomboss}, A_{Pacman})$.

$A_{Zomboss} = \boxed{2}$, $A_{Pacman} = \boxed{6}$

In Zomboss' perspective, the tree is a minimax, as follows:



The red branches are the ones being selected by the agents.
Pacman knows this, so all he need to do is to maximize his utility with the information of the branches that are going to be selected by Zomboss.

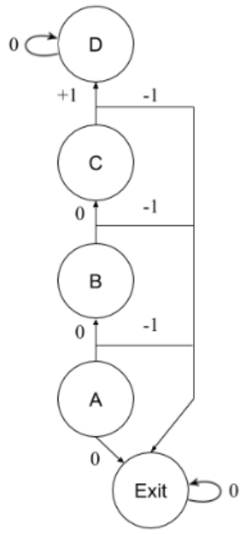


Essentially, to solve this problem, we first solved a minimax, then solved a general-sum game where one of the agent's moves are known.

You might observed some peculiar behaviors of this tree: despite both agents being rational, sometimes Zomboss ends up selecting a branch that does not maximize his own utility, but maximizes Pacman's utility; sometimes Zomboss selects a branch that minimizes both his own utility and Pacman's. That's a result of not having complete information.

SID: ______________

Q2. [21 pts] Climbing



Alice is deciding whether to spend a day at the Berkeley Ironworks to practice climbing. Since she recently took and mastered CS188, she decides to test her newfound knowledge to compute whether it will be optimal, from the point of view of her utility function, to go and practice. She decides to model her problem as the Markov Decision Process (MDP) shown on the right.

State A represents Alice's starting state, where Alice can either **stay** put on the ground and receive a reward of 0, or **climb** and advance to state B, also with a reward of 0. If Alice stays, she moves to a special "exit" state. If Alice begins climbing, she cannot stop, and must continue to advance to the next state until she reaches the top at state D, upon which she receives a reward of +1. However, while she is climbing (from A to B, from B to C, and from C to D), after attempting to move to the next state, there is a chance p_{fall} she may fall and receive a reward of −1, also causing Alice to enter the "exit" state.

Let's begin by assuming a falling probability of $p_{\text{fall}} = 0.1$ and a discount factor of $\gamma = 0.5$.

(a) [2 pts] (0.5 pt each) Assume we initialize the values of all states to 0. What is the value of each state after running a single iteration of value iteration?

- $V_1(A) =$ ____0____
- $V_1(B) =$ ____−0.1____
- $V_1(C) =$ ____0.8____
- $V_1(D) =$ ____0____

For a single iteration, the only values that will update are $V_1(C)$, since it is the only state where you can gain positive reward, and $V_1(B)$, which has a negative reward for falling. For C, we can calculate this by using $p_{\text{fall}} = 0.1$ as our transition probability to get $(0.9 * 1) + (0.1 * -1) = 0.8$. For B, we can calculate this as $(0.1 * -1) = -0.1$.

To compute for randomized values of p_{fall} and γ, plug their values into the general cases below:

$V_1(A) = max\{(1 - p_{\text{fall}}) \cdot [0 + \gamma \cdot 0] + p_{\text{fall}} \cdot [-1 + \gamma \cdot 0], 0\} = max\{-p_{\text{fall}}, 0\} = 0$ for all p_{fall}

$V_1(B) = (1 - p_{\text{fall}}) \cdot [0 + \gamma \cdot 0] + p_{\text{fall}} \cdot [-1 + \gamma \cdot 0]) = -p_{\text{fall}}$

$V_1(C) = (1 - p_{\text{fall}}) \cdot [1 + \gamma \cdot 0] + p_{\text{fall}} \cdot [-1 + \gamma \cdot 0]) = 1 - 2p_{\text{fall}}$

$V_1(D) = 0$

(b) [4 pts] (1 pt each) What is the optimal value function at each state?

- $V^*(A) =$ ____0.017____
- $V^*(B) =$ ____0.26____
- $V^*(C) =$ ____0.8____
- $V^*(D) =$ ____0____

$V^*(C)$ and $V^*(D)$ will be the same as in the previous part since their possible successor states will always have values of 0.

$V^*(B) = p_{\text{climb}}[r_{\text{climb}} + \gamma V^*(C)] + p_{\text{fall}}[r_{\text{fall}} + \gamma V^*(\text{exit})] = 0.9[0 + 0.5(0.8)] + 0.1[-1 + 0.5(0)] = 0.26$

$V^*(A) = p_{\text{climb}}[r_{\text{climb}} + \gamma V^*(B)] + p_{\text{fall}}[r_{\text{fall}} + \gamma V^*(\text{exit})] = 0.9[0 + 0.5(0.26)] + 0.1[-1 + 0.5(0)] = 0.017$

To compute for randomized values of p_{fall} and γ, plug their values into the general cases below:

$V^*(A) = max\{(1 - p_{\text{fall}}) \cdot [0 + \gamma \cdot V^*(B)] + p_{\text{fall}} \cdot [-1 + \gamma \cdot 0], 0\} = max\{(1 - p_{\text{fall}}) \cdot \gamma V^*(B) - p_{\text{fall}}, 0\}$

$V^*(B) = (1 - p_{\text{fall}}) \cdot [0 + \gamma \cdot V^*(C)] + p_{\text{fall}} \cdot [-1 + \gamma \cdot 0] = (1 - p_{\text{fall}}) \cdot \gamma V^*(C) - p_{\text{fall}}$

$V^*(C) = (1 - p_{\text{fall}}) \cdot [1 + \gamma \cdot 0] + p_{\text{fall}} \cdot [-1 + \gamma \cdot 0]) = 1 - 2p_{\text{fall}}$

$V^*(D) = 0$

Alice wants to bring her friend Bob, but realizes that Bob might have a different skill level from her (p_{fall}) or might value time differently (γ). Therefore, Alice wants to estimate whether Bob will agree to **climb** or instead choose to **stay**.

There is only a single decision point at state A, which makes computing the value function much simpler. The value of going up from state A is γ^2 times the probability of making it to the top, minus the total probability of falling, which results in $\gamma^2(1-p_{\text{fall}})^3 - \gamma^2(1-p_{\text{fall}})^2 p_{\text{fall}} - \gamma(1-p_{\text{fall}})p_{\text{fall}} - p_{\text{fall}}$. In order to answer the following questions, we need to check when this quantity is less than 0.

(c) [3 pts] Assume $p_{\text{fall}} = 0.1$. For which of the following values of γ will Bob choose to **stay** from state A?

■ 0.2 ☐ 0.6
■ 0.4 ☐ 0.8

Plugging in $p_{\text{fall}} = 0.1$, we get $\gamma^2 0.729 - \gamma^2 0.081 - \gamma 0.9 - 0.1$. Numerically, the values for which this is less than zero are $\gamma < 0.468$, which corresponds to the cases when Bob would choose to stay.

For $p_{\text{fall}} = 0.1, \gamma = 0.468$, so the answers are $0.2, 0.4$.

For $p_{\text{fall}} = 0.155, \gamma = 0.709$, so the answers are $0.2, 0.4, 0.6$.

For $p_{\text{fall}} = 0.05, \gamma = 0.279$, so the answer is 0.2.

(d) [3 pts] Assume $\gamma = 0.5$. For which of the following values of p_{fall} will Bob choose to **stay** from state A?

☐ 0.05 ☐ 0.15 ■ 0.25 ■ 0.35
☐ 0.1 ■ 0.2 ■ 0.3 ■ 0.4

Plugging in $\gamma = 0.5$, we get $0.25(1-p)^3 - 0.25(1-p)^2 p - 0.5(1-p)p - p$. Numerically, the values for which this is less than zero are $p > 0.108$, which corresponds to the cases when Bob would choose to stay.

For $\gamma = 0.5, p_{\text{fall}} = 0.108$, so the answers are everything above 0.1.

For $\gamma = 0.4, p_{\text{fall}} = 0.082$, so the answers are everything above 0.05.

For $\gamma = 0.8, p_{\text{fall}} = 0.173$, so the answers are everything above 0.15.

Alice wants to try a different method for computing her total utility. Instead of using the discounted sum of rewards, Alice wishes to use the *average* reward she accumulates while climbing.

That is, rather than using the discounted sum of rewards to measure her total utility over an episode of k steps:

$$U^\gamma([r_0, \cdots, r_k]) = r_0 + \gamma r_1 + \gamma^2 r_2 + \gamma^3 r_3 + \ldots + \gamma^k r_k$$

She wishes to use the *average* utility:

$$\bar{U}([r_0, \cdots, r_k]) = \frac{1}{k+1}[r_0 + r_1 + r_2 + r_3 + \ldots + r_k]$$

(e) [3 pts] Assume $p_{\text{fall}} = 0.5$, and when we reach the exit state or the top state D, the episode terminates (rather than looping forever). What is Alice's expected *average* utility if she starts from state A, assuming Alice's policy is to always climb from A? *(Hint: the expectation is taken over randomness in falling (which can lead to some episodes being longer than others), and the average utility is taken over the length of an episode.)*

-0.625

There are 4 possible outcomes:

- Receive a reward of 1 for reaching state D, over 3 steps (average utility $1/3$). This has a probability of $(1-p_{\text{fall}})^3$.
- Fall at state A, receiving a reward of -1 over 1 step (average utility -1). This has a probability of p_{fall}.
- Fall at state B, receiving a reward of -1 over 2 steps (average utility $-1/2$). This has a probability of $(1-p_{\text{fall}})p_{\text{fall}}$.
- Fall at state C, receiving a reward of -1 over 3 steps (average utility $-1/3$). This has a probability of $(1-p_{\text{fall}})^2 p_{\text{fall}}$.

Thus, the average reward is in total,

$$\frac{1}{3}(1-p_{\text{fall}})^3 - p_{\text{fall}} - \frac{1}{2}(1-p_{\text{fall}})p_{\text{fall}} - \frac{1}{3}(1-p_{\text{fall}})^2 p_{\text{fall}}$$

Plugging in $p_{\text{fall}} = 0.5$, we get a value of -0.625.
Plugging in $p_{\text{fall}} = 0.4$, we get a value of -0.496.
Plugging in $p_{\text{fall}} = 0.2$, we get a value of -0.152.
Plugging in $p_{\text{fall}} = 0.1$, we get a value of 0.071.
Partial credit (2 points) was given for not averaging over the timestep.

$$(1-p_{\text{fall}})^3 - p_{\text{fall}} - (1-p_{\text{fall}})p_{\text{fall}} - (1-p_{\text{fall}})^2 p_{\text{fall}}$$

Plugging in $p_{\text{fall}} = 0.4$, we get a value of -0.568.
Plugging in $p_{\text{fall}} = 0.2$, we get a value of 0.024.
Plugging in $p_{\text{fall}} = 0.1$, we get a value of 0.458.

Recall that the total discounted utility can be written as a recursion.

$$U^{\gamma}([r_0, \cdots, r_k]) = \sum_{t=0}^{k} \gamma^t r_t = r_0 + \sum_{t=1}^{k} \gamma^t r_t = r_0 + \gamma U([r_1, \cdots, r_k])$$

This gives rise to the Bellman operator for policy evaluation, which computes this utility in expectation (the value function represents the total *expected* utility). Recall that in policy evaluation, we are given the value at iteration k, $V_k^{\pi}(s)$, and can use the following update to compute the value for iteration $k+1$, $V_{k+1}^{\pi}(s)$.

$$V_{k+1}^{\pi}(s) \leftarrow \sum_{s'} T(s, \pi(s), s')[R(s, \pi(s), s') + \gamma V_k^{\pi}(s')]$$

We are interested in deriving the corresponding update for the average utility case.

(f) [6 pts] (1 pt each; **(C)** and **(D)** worth 1.5 pts each)

Alice wishes to derive an algorithmic procedure for finding the average utility, $\bar{V}^{\pi}$. For each letter **(A), (B), (C), (D)** and **(E)**, fill in a single entry for the term corresponding to the correct equation to implement policy evaluation for the average utility case. Select one bubble per row to form the whole equation. *(Hint: try to think about how you can write down the average utility as a recursion, just like in the discounted utility case)*

$$\bar{V}_{k+1}^{\pi}(s) \leftarrow \textbf{(A)}\ \textbf{(B)}\ [\textbf{(C)} + \textbf{(D)}\ \bar{V}_k^{\pi}(s')] + \textbf{(E)}\ \bar{V}_k^{\pi}(s')$$

- **(A)** : ○ $\max_{\pi(s)}$ [✓ $\sum_{s'}$] ○ $\max_{s'}$ ○ 1
- **(B)** : [✓ $T(s, \pi(s), s')$] ○ $R(s, \pi(s), s')$ ○ 1
- **(C)** : ○ $R(s, \pi(s), s')$ [✓ $\frac{1}{k+1} R(s, \pi(s), s')$] ○ $\frac{k}{k+1} R(s, \pi(s), s')$ ○ 0 ○ 1
- **(D)** : ○ γ ○ $\frac{1}{k+1}$ [✓ $\frac{k}{k+1}$] ○ 0 ○ 1
- **(E)** : ○ γ ○ $\frac{1}{k+1}$ ○ $\frac{k}{k+1}$ [✓ 0] ○ 1

We can compute the average utility recursively, just like in the discounted case:

$$\bar{U}([r_0, \cdots, r_k]) = \frac{1}{k+1} \sum_{t=0}^{k} r_t = \frac{1}{k+1} r_0 + \frac{1}{k+1} \sum_{t=1}^{k} r_t = \frac{1}{k+1} r_0 + \frac{k}{k+1} \bar{U}([r_1, \cdots, r_k])$$

In expectation (formally, by via linearity of expectations), this corresponds to the update

$$\bar{V}^{\pi}_{k+1}(s) = \sum_{s'} T(s, \pi(s), s')[\frac{1}{k+1} R(s, a, s') + \frac{k}{k+1} \bar{V}^{\pi}_{k}(s')]$$

Q3. [20 pts] RL: Stochastic Policies

Max and her friend Tubs are discussing solving Markov Decision Processes (MDP) and Reinforcement Learning (RL) problems. Max is wondering if she can arrive at the optimal value faster by changing the policy. Normally, in RL we use a **policy** $\pi(s)$ to choose an action at a given state: $\pi : s \mapsto a$. When estimating the value of a state, she knows that exploration and exploitation is an important trade-off. Max, knowing that we can incorporate exploration into Q-learning, wants to bring it to policy iteration. She decides to mimic epsilon-greedy exploration by making a **stochastic policy**. A stochastic policy, rather than return a single action, returns a probability distribution over potential actions.

$$\widetilde{\pi} : s \mapsto \mathcal{P}(a) \tag{1}$$

When taking an action from state s following policy $\widetilde{\pi}$, an agent samples an action at random according to the probability distribution $\widetilde{\pi}(s)$.

The initial MDP is below, and Max is annoyed that the policy she arrived at is clearly not optimal. Her friend Tubs suggests a change to stochastic policies, but she is skeptical that it'll help.

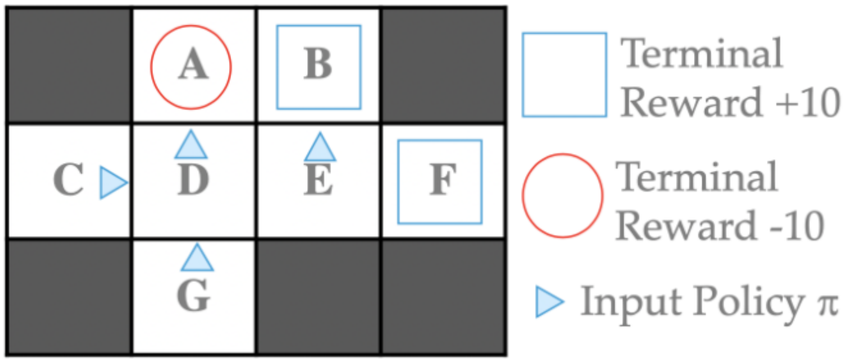


Figure 1: The MDP that Max is trying to solve.

state	policy
C	right
D	up
E	up
G	up

Table 1: Deterministic policy π.

state	$\mathcal{P}$(up)	$\mathcal{P}$(right)	$\mathcal{P}$(down)	$\mathcal{P}$(left)
C	0.1	0.8	0.1	0.0
D	0.7	0.1	0.1	0.1
E	0.4	0.4	0.1	0.1
G	0.8	0.1	0.0	0.1

Table 2: Stochastic policy, $\widetilde{\pi}$ for non-terminal states.

Above: first set of policies for Max.

Throughout this problem, we will operate with the following assumptions:
1) The discount factor, $\gamma = 1$.
2) The living reward is 0.
3) The transition functions, $\mathcal{T}(s, a, s')$, are 'simple,' where the resulting motion is always in the direction of the action. For example, $\mathcal{T}(C, right, D) = 1$, $\mathcal{T}(D, up, A) = 1$, $\mathcal{T}(E, down, E) = 1$.
*4) Taking an action toward an invalid location will result in no movement, i.e.: the successor state for moving **left** or **up** from state C in this MDP would also be state C.*
*5) **Arriving in a terminal state immediately grants the terminal reward and ends the episode.** For example, $R(E, up, B) = 10$. There is no "exit" action in this MDP.*

The Bellman equation for a fixed policy, shown below, may be useful during this problem.

$$V^{\pi}(s) = \sum_{s'} \mathcal{T}(s, \pi(s), s')\left[\mathcal{R}(s, \pi(s), s') + \gamma V^{\pi}(s')\right] \tag{2}$$

(a) [2 pts] Assuming $V_0^{\pi}(s) = 0$ for all states s, compute the value of the following states after two steps of **policy evaluation**, conditioned on the **deterministic policy**, π (shown in Table 1).

- $V_2^{\pi}(C) =$ $\underline{\quad -10 \quad}$
- $V_2^{\pi}(D) =$ $\underline{\quad -10 \quad}$
- $V_2^{\pi}(E) =$ $\underline{\quad 10 \quad}$
- $V_2^{\pi}(G) =$ $\underline{\quad -10 \quad}$

The values at step 0 are 0 for the states of interest.

$$V_0^{\pi}(C) = 0 \tag{3}$$
$$V_0^{\pi}(D) = 0 \tag{4}$$
$$V_0^{\pi}(E) = 0 \tag{5}$$
$$V_0^{\pi}(G) = 0 \tag{6}$$

Step 1: The states adjacent to the terminal states get the full reward because the discount factor is 1. *An important item to note is that when the reward is received entering a terminal state, there is no more reward to be received, so V(terminal state) = 0.*

$$V_1^{\pi}(C) = 0 \tag{7}$$
$$V_1^{\pi}(D) = 1\big[-10 + 1 \cdot 0\big] = -10 \tag{8}$$
$$V_1^{\pi}(E) = 10 \tag{9}$$
$$V_1^{\pi}(G) = 0 \tag{10}$$

Step 2: The reward propagates once more.

$$V_2^{\pi}(C) = -10 \tag{11}$$
$$V_2^{\pi}(D) = -10 \tag{12}$$
$$V_2^{\pi}(E) = 10 \tag{13}$$
$$V_2^{\pi}(G) = -10 \tag{14}$$

(b) [4 pts] Assuming $V_0^{\tilde{\pi}}(s) = 0$ for all states s, compute the value of the following states after two steps of **policy evaluation**, conditioned on the **stochastic policy**, $\tilde{\pi}$ (shown in Table 2). *(Hint: $V_k^{\tilde{\pi}}(s)$ is the expected utility when taking at most k actions according to the policy $\tilde{\pi}$.)*

- $V_2^{\tilde{\pi}}(C) =$ $\underline{\quad -5.6 \quad}$
- $V_2^{\tilde{\pi}}(D) =$ $\underline{\quad -6.2 \quad}$
- $V_2^{\tilde{\pi}}(E) =$ $\underline{\quad 7.3 \quad}$
- $V_2^{\tilde{\pi}}(G) =$ $\underline{\quad -5.6 \quad}$

The uncertainty in the policy is handled in the Bellman update as transition uncertainty is. The values at step 0 are 0 for the states of interest.

$$V_0^{\tilde{\pi}}(C) = 0 \tag{15}$$
$$V_0^{\tilde{\pi}}(D) = 0 \tag{16}$$
$$V_0^{\tilde{\pi}}(E) = 0 \tag{17}$$
$$V_0^{\tilde{\pi}}(G) = 0 \tag{18}$$

Step 1: The states adjacent to the terminal states get the reward proportional to the likelihood of stepping there (and the discount factor is 1, living reward is 0).

$$V_1^{\tilde{\pi}}(C) = 0.1[0 + 1 \cdot 0] + 0.8[0 + 1 \cdot 0] + 0.1[0 + 1 \cdot 0] + 0.0[0 + 1 \cdot 0] \quad = 0 \tag{19}$$
$$V_1^{\tilde{\pi}}(D) = 0.7[-10 + 1 \cdot 0] + 0.1[0 + 1 \cdot 0] + 0.1[0 + 1 \cdot 0] + 0.1[0 + 1 \cdot 0] \quad = -7 \tag{20}$$
$$V_1^{\tilde{\pi}}(E) = 0.4[10 + 1 \cdot 0] + 0.4[10 + 1 \cdot 0] + 0.1[0 + 1 \cdot 0] + 0.1[0 + 1 \cdot 0] \quad = 8 \tag{21}$$
$$V_1^{\tilde{\pi}}(G) = 0.8[0 + 1 \cdot 0] + 0.1[0 + 1 \cdot 0] + 0.0[0 + 1 \cdot 0] + 0.1[0 + 1 \cdot 0] \quad = 0 \tag{22}$$

Step 2: The reward propagates once more.

$$V_2^{\tilde{\pi}}(C) = 0.1[0+1\cdot 0] + 0.8[0+1\cdot -7] + 0.1[0+1\cdot 0] + 0.0[0+1\cdot 0] \qquad = -5.6 \tag{23}$$
$$V_2^{\tilde{\pi}}(D) = 0.7[-10+1\cdot 0] + 0.1[0+1\cdot 8] + 0.1[0+1\cdot 0] + 0.1[0+1\cdot 0] \qquad = -6.2 \tag{24}$$
$$V_2^{\tilde{\pi}}(E) = 0.4[10+1\cdot 0] + 0.4[10+1\cdot 0] + 0.1[0+1\cdot 0] + 0.1[0+1\cdot -7] \qquad = 7.3 \tag{25}$$
$$V_2^{\tilde{\pi}}(G) = 0.8[0+1\cdot -7] + 0.1[0+1\cdot 0] + 0.0[0+1\cdot 0] + 0.1[0+1\cdot 0] \qquad = -5.6 \tag{26}$$

This part has different solutions depending on the stochastic policy in the exam version, with each version below.

Policy 2:

$$V_1^{\tilde{\pi}}(C) = 0.1[0+1\cdot 0] + 0.7[0+1\cdot 0] + 0.1[0+1\cdot 0] + 0.1[0+1\cdot 0] \qquad = 0 \tag{27}$$
$$V_1^{\tilde{\pi}}(D) = 0.4[-10+1\cdot 0] + 0.2[0+1\cdot 0] + 0.2[0+1\cdot 0] + 0.2[0+1\cdot 0] \qquad = -4 \tag{28}$$
$$V_1^{\tilde{\pi}}(E) = 0.5[10+1\cdot 0] + 0.3[10+1\cdot 0] + 0.1[0+1\cdot 0] + 0.1[0+1\cdot 0] \qquad = 8 \tag{29}$$
$$V_1^{\tilde{\pi}}(G) = 0.7[0+1\cdot 0] + 0.1[0+1\cdot 0] + 0.1[0+1\cdot 0] + 0.1[0+1\cdot 0] \qquad = 0 \tag{30}$$

Policy 2 - Step 2: The reward propagates once more.

$$V_2^{\tilde{\pi}}(C) = 0.1[0+1\cdot 0] + 0.7[0+1\cdot -4] + 0.1[0+1\cdot 0] + 0.1[0+1\cdot 0] \qquad = -2.8 \tag{31}$$
$$V_2^{\tilde{\pi}}(D) = 0.4[-10+1\cdot 0] + 0.2[0+1\cdot 8] + 0.2[0+1\cdot 0] + 0.2[0+1\cdot 0] \qquad = -2.4 \tag{32}$$
$$V_2^{\tilde{\pi}}(E) = 0.5[10+1\cdot 0] + 0.3[10+1\cdot 0] + 0.1[0+1\cdot 0] + 0.1[0+1\cdot -4] \qquad = 7.6 \tag{33}$$
$$V_2^{\tilde{\pi}}(G) = 0.7[0+1\cdot -4] + 0.1[0+1\cdot 0] + 0.1[0+1\cdot 0] + 0.1[0+1\cdot 0] \qquad = -2.8 \tag{34}$$

Policy 3:

$$V_1^{\tilde{\pi}}(C) = 0.2[0+1\cdot 0] + 0.5[0+1\cdot 0] + 0.2[0+1\cdot 0] + 0.1[0+1\cdot 0] \qquad = 0 \tag{35}$$
$$V_1^{\tilde{\pi}}(D) = 0.3[-10+1\cdot 0] + 0.2[0+1\cdot 0] + 0.3[0+1\cdot 0] + 0.2[0+1\cdot 0] \qquad = -3 \tag{36}$$
$$V_1^{\tilde{\pi}}(E) = 0.3[10+1\cdot 0] + 0.5[10+1\cdot 0] + 0.1[0+1\cdot 0] + 0.1[0+1\cdot 0] \qquad = 8 \tag{37}$$
$$V_1^{\tilde{\pi}}(G) = 0.7[0+1\cdot 0] + 0.1[0+1\cdot 0] + 0.1[0+1\cdot 0] + 0.1[0+1\cdot 0] \qquad = 0 \tag{38}$$

Policy 3 -Step 2: The reward propagates once more.

$$V_2^{\tilde{\pi}}(C) = 0.2[0+1\cdot 0] + 0.5[0+1\cdot -3] + 0.2[0+1\cdot 0] + 0.1[0+1\cdot 0] \qquad = -1.5 \tag{39}$$
$$V_2^{\tilde{\pi}}(D) = 0.3[-10+1\cdot 0] + 0.2[0+1\cdot 8] + 0.3[0+1\cdot 0] + 0.2[0+1\cdot 0] \qquad = -1.4 \tag{40}$$
$$V_2^{\tilde{\pi}}(E) = 0.3[10+1\cdot 0] + 0.5[10+1\cdot 0] + 0.1[0+1\cdot 0] + 0.1[0+1\cdot -3] \qquad = 7.7 \tag{41}$$
$$V_2^{\tilde{\pi}}(G) = 0.7[0+1\cdot -3] + 0.1[0+1\cdot 0] + 0.1[0+1\cdot 0] + 0.1[0+1\cdot 0] \qquad = -2.1 \tag{42}$$

Policy 4:

$$V_1^{\tilde{\pi}}(C) = 0.2[0+1\cdot 0] + 0.6[0+1\cdot 0] + 0.1[0+1\cdot 0] + 0.1[0+1\cdot 0] \qquad = 0 \tag{43}$$
$$V_1^{\tilde{\pi}}(D) = 0.2[-10+1\cdot 0] + 0.3[0+1\cdot 0] + 0.3[0+1\cdot 0] + 0.2[0+1\cdot 0] \qquad = -2 \tag{44}$$
$$V_1^{\tilde{\pi}}(E) = 0.2[10+1\cdot 0] + 0.5[10+1\cdot 0] + 0.2[0+1\cdot 0] + 0.1[0+1\cdot 0] \qquad = 7 \tag{45}$$
$$V_1^{\tilde{\pi}}(G) = 0.5[0+1\cdot 0] + 0.2[0+1\cdot 0] + 0.1[0+1\cdot 0] + 0.2[0+1\cdot 0] \qquad = 0 \tag{46}$$

Policy 4 - Step 2: The reward propagates once more.

$$V_2^{\widetilde{\pi}}(C) = 0.2[0+1\cdot 0] + 0.6[0+1\cdot -2] + 0.1[0+1\cdot 0] + 0.1[0+1\cdot 0] \quad = -1.2 \tag{47}$$
$$V_2^{\widetilde{\pi}}(D) = 0.2[-10+1\cdot 0] + 0.3[0+1\cdot 7] + 0.3[0+1\cdot 0] + 0.2[0+1\cdot 0] \quad = 0.1 \tag{48}$$
$$V_2^{\widetilde{\pi}}(E) = 0.2[10+1\cdot 0] + 0.5[10+1\cdot 0] + 0.2[0+1\cdot 0] + 0.1[0+1\cdot -2] \quad = 6.8 \tag{49}$$
$$V_2^{\widetilde{\pi}}(G) = 0.5[0+1\cdot -2] + 0.2[0+1\cdot 0] + 0.1[0+1\cdot 0] + 0.2[0+1\cdot 0] \quad = -1 \tag{50}$$

(c) [2 pts] Given a deterministic, optimal policy, π^*, will changing to any probabilistic policy, $\widetilde{\pi}$, ever increase the **expected** reward an agent achieves?

Yes, always. Sometimes. [✓ No, never.]

A change to stochastic policies could improve on a non-optimal policy (as shown in this problem), but by definition of an optimal policy there is no way to change it and obtain higher expected total reward.

Now Max goes to a new MDP. She wants to learn the optimal policy and thinks that maybe stochastic policies will help in this case. For deterministic policies, we update the policy from a policy-conditioned value with *policy improvement*:

$$\pi_{i+1}(s) = \arg\max_a \sum_{s'} \mathcal{T}(s,a,s')\big[\mathcal{R}(s,a,s') + \gamma \mathbb{E}_{\pi(s)}[V^{\pi}(s')]\big] \tag{51}$$

We need to do something different for stochastic policies, so instead of returning the maximum action, we increase its probability. In the context of this problem, when we run policy improvement on the stochastic policy $\widetilde{\pi}$, we increase the probability of the best action by 0.3, and decrease all others by 0.1. (*In the case that an individual action is trying to decrease in probability below 0.0, its probability stays the same, and the change in probability for the best action is adjusted such that the distribution still sums to 1.0. For example: if exactly one action does not decrease, then the best action only increases by 0.2.*)

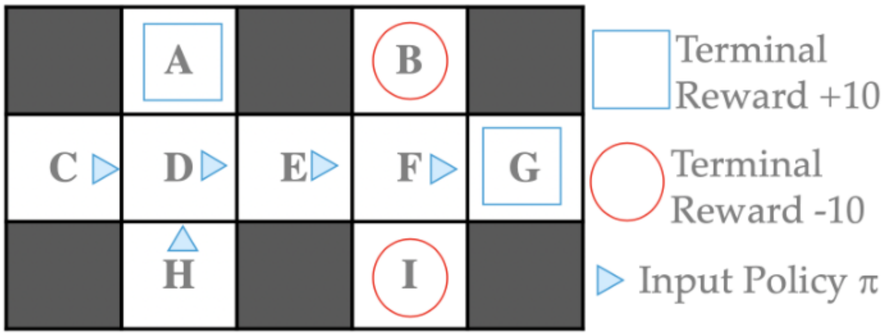


Figure 2: The MDP that Max is trying to solve.

In this part, the living reward is -1.

The other assumptions that we made previously still hold however:
1) The discount factor, $\gamma = 1$.
2) The transition functions, $\mathcal{T}(s,a,s')$, *are 'simple,' where the resulting motion is always in the direction of the action. For example,* $\mathcal{T}(C, right, D) = 1$, $\mathcal{T}(D, up, A) = 1$, $\mathcal{T}(E, down, E) = 1$.
3) Taking an action toward an invalid location will result in no movement, i.e.: the successor state for moving ***left*** *or* ***up*** *from state C in the current MDP would also be state C.*
4) ***Arriving in a terminal state immediately grants the terminal reward and ends the episode.*** *For example,* $R(F, right, G) = 10$. *There is no "exit" action in this MDP.*

state	policy
C	right
D	right
E	right
F	right
H	up

Table 3: Deterministic policy π.

state	$\mathcal{P}$(up)	$\mathcal{P}$(right)	$\mathcal{P}$(down)	$\mathcal{P}$(left)
C	0.1	0.7	0.1	0.1
D	0.1	0.7	0.1	0.1
E	0.1	0.3	0.1	0.5
F	0.1	0.5	0.1	0.3
H	0.7	0.1	0.1	0.1

Table 4: Stochastic policy, $\widetilde{\pi}$ for non-terminal states.

	sequence
1	C → D → E → F → G
2	H → D → E → F → G

Table 5: Deterministic policy roll-outs.

	sequence
1	C → D → A
2	H → D → E → F → E → F→ G
3	C → D → E → F → I

Table 6: Stochastic policy roll-outs.

(d) Compute the values of the listed states after TD-learning for the deterministic policy π in Table 3 and the stochastic policy $\widetilde{\pi}$ in Table 4. Use the provided episodes in Table 5 for π and the provided episodes in Table 6 for $\widetilde{\pi}$. ***In this part, the living reward is*** -1**.**

Hint: recall that TD-learning is performed after each transition **only**.

$V_0(s) = 0$ for all states (including terminal states).

$V^{\pi}(s) \leftarrow (1-\alpha)V^{\pi}(s) + \alpha \cdot sample$, where $\alpha = .5$.

Note: in these parts it may be easiest to leave your answer as a fraction.

(i) [2 pts] Deterministic Policy

- State D ____-1____
- State E ____$\frac{7}{4}$____

To compute the state value with TD learning, one state gets updated at each step. Below will go episode-by-episode to show how the values for each state are computed. The final value for each state is shown with $(*)$. Each sample is given by $(R(s,a,s') + \gamma V(s'))$

$$\text{episode 1}: \begin{cases} V(C) &= \frac{1}{2}(0) + \frac{1}{2}\big(-1+0\big) &= -\frac{1}{2} & * \\ V(D) &= \frac{1}{2}(0) + \frac{1}{2}\big(-1+0\big) &= -\frac{1}{2} \\ V(E) &= \frac{1}{2}(0) + \frac{1}{2}\big(-1+0\big) &= -\frac{1}{2} \\ V(F) &= \frac{1}{2}(0) + \frac{1}{2}\big(10+0\big) &= \frac{10}{2} \end{cases}$$

$$\text{episode 2}: \begin{cases} V(H) &= \frac{1}{2}(0) + \frac{1}{2}\big(-1+-\frac{1}{2}\big) &= \frac{-3}{4} & * \\ V(D) &= \frac{1}{2}(-\frac{1}{2}) + \frac{1}{2}\big(-1+-\frac{1}{2}\big) &= -1 & * \\ V(E) &= \frac{1}{2}(-\frac{1}{2}) + \frac{1}{2}\big(-1+\frac{10}{2}\big) &= \frac{7}{4} & * \\ V(F) &= \frac{1}{2}(\frac{10}{2}) + \frac{1}{2}\big(10+0\big) &= 7.5 & * \end{cases}$$

For $V(D)$ and $V(E)$, this can be expressed with the following equations:

$$V(D) = -2\alpha$$

$$V(E) = -2\alpha + 11\alpha^2$$

Plugging in the randomized values of α, we get the following:

For $\alpha = 0.2$: $V(D) = -0.4; V(E) = 0.04$.
For $\alpha = 0.3$: $V(D) = -0.6; V(E) = 0.39$.
For $\alpha = 0.5$: $V(D) = -1; V(E) = 1.75$.
For $\alpha = 0.7$: $V(D) = -1.4; V(E) = 3.99$.
For $\alpha = 0.9$: $V(D) = -1.8; V(E) = 7.11$.

NOTE: There was some discrepancy about whether the living reward should be added to the terminal state reward when moving to the terminal state. For this, the correct equation for $V(E)$ would be:

$$V(E) = -2\alpha + 10\alpha^2$$

.

Thus, the following answers also received full credit:

For $\alpha = 0.2$: $V(D) = -0.4; V(E) = 0$.
For $\alpha = 0.3$: $V(D) = -0.6; V(E) = 0.3$.
For $\alpha = 0.5$: $V(D) = -1; V(E) = 1.5$.
For $\alpha = 0.7$: $V(D) = -1.4; V(E) = 3.5$.
For $\alpha = 0.9$: $V(D) = -1.8; V(E) = 6.3$.

(ii) [2 pts] Stochastic Policy

- State D $-\frac{1}{16}$
- State E $\frac{5}{4}$

TODO: Nikita thinks the answer with new formulation $R(\cdot, \cdot, terminal state) = +10 \text{or} - 10$ is: State D 0, State E 5/4.

$$\text{episode 1 :} \begin{cases} V(C) & = \frac{1}{2}(0) + \frac{1}{2}\big(-1+0\big) & = -\frac{1}{2} \\ V(D) & = \frac{1}{2}(0) + \frac{1}{2}\big(10+0\big) & = \frac{10}{2} \end{cases}$$

$$\text{episode 2 :} \begin{cases} V(H) & = \frac{1}{2}(0) + \frac{1}{2}\big(-1+\frac{10}{2}\big) & = \frac{8}{4} & * \\ V(D) & = \frac{1}{2}(\frac{10}{2}) + \frac{1}{2}\big(-1+0\big) & = \frac{8}{4} \\ V(E) & = \frac{1}{2}(0) + \frac{1}{2}\big(-1+0\big) & = -\frac{1}{2} \\ V(F) & = \frac{1}{2}(0) + \frac{1}{2}\big(-1-\frac{1}{2}\big) & = -\frac{3}{4} \\ V(E) & = \frac{1}{2}(-\frac{1}{2}) + \frac{1}{2}\big(-1-\frac{3}{4}\big) & = -\frac{9}{8} \\ V(F) & = \frac{1}{2}(-\frac{3}{4}) + \frac{1}{2}\big(10+0\big) & = \frac{37}{8} \end{cases}$$

$$\text{episode 3 :} \begin{cases} V(C) & = \frac{1}{2}(-\frac{1}{2}) + \frac{1}{2}\big(-1+\frac{8}{4}\big) & = \frac{1}{4} & * \\ V(D) & = \frac{1}{2}(\frac{8}{4}) + \frac{1}{2}\big(-1-\frac{9}{8}\big) & = -\frac{1}{16} & * \\ V(E) & = \frac{1}{2}(-\frac{9}{8}) + \frac{1}{2}\big(-1+\frac{37}{8}\big) & = \frac{5}{4} & * \\ V(F) & = \frac{1}{2}(\frac{37}{8}) + \frac{1}{2}\big(-10+0\big) & = -\frac{43}{16} & * \end{cases}$$

For $V(D)$ and $V(E)$, this can be expressed with the following equations:

$$V(D) = -\alpha^4 + 10\alpha^3 - 21\alpha^2 + 8\alpha$$

$$V(E) = 2\alpha^4 - \alpha^3 + 11\alpha^2 - 3\alpha$$

Plugging in the randomized values of α, we get the following:

For $\alpha = 0.2$: $V(D) = 0.8384; V(E) = -0.1648$.
For $\alpha = 0.3$: $V(D) = 0.7719; V(E) = 0.0792$.
For $\alpha = 0.7$: $V(D) = -1.5001; V(E) = 3.4272$.
For $\alpha = 0.9$: $V(D) = -3.1761; V(E) = 6.7932$.

NOTE: There was some discrepancy about whether the living reward should be added to the terminal state reward when moving to the terminal state. For this, the correct equation for $V(D)$ and $V(E)$ would be:

$$V(D) = -\alpha^4 + 9\alpha^3 - 19\alpha^2 + 7\alpha$$

$$V(E) = 2\alpha^4 - \alpha^3 + 10\alpha^2 - 3\alpha$$

Thus, the following answers also received full credit:

For $\alpha = 0.2$: $V(D) = 0.7104$; $V(E) = -0.2048$.
For $\alpha = 0.3$: $V(D) = 0.6249$; $V(E) = -0.0108$.
For $\alpha = 0.5$: $V(D) = -0.1875$; $V(E) = 1$.
For $\alpha = 0.7$: $V(D) = -1.5631$; $V(E) = 2.9372$.
For $\alpha = 0.9$: $V(D) = -3.1851$; $V(E) = 5.9832$.

(e) Compute the updated stochastic policy, using the value estimates V^π calculated from 3 episodes of TD learning (like in the previous subpart). ***In this part, the living reward is*** -1. The updated actions are very similar to that of a deterministic policy. Observe the policy improvement equation, and then think of a 'soft' argmax term. The soft argmax increases the returned argument, but cannot set its probability to 1.0 in all cases.

(i) [4 pts] State D.

- $\mathcal{P}(\text{up}) =$ 0.4
- $\mathcal{P}(\text{down}) =$ 0.0
- $\mathcal{P}(\text{right}) =$ 0.6
- $\mathcal{P}(\text{left}) =$ 0.0

For state D, the optimal action is up, so the stochastic policy increases that probability by 0.3, and lowers all others by 0.1.
This part has different solutions depending on the stochastic policy in the exam version, with each version below.
Policy 1:

- $\mathcal{P}(\text{up}) =$ 0.4
- $\mathcal{P}(\text{down}) =$ 0.0
- $\mathcal{P}(\text{right}) =$ 0.6
- $\mathcal{P}(\text{left}) =$ 0.0

Policy 2:

- $\mathcal{P}(\text{up}) =$ 0.4
- $\mathcal{P}(\text{down}) =$ 0.0
- $\mathcal{P}(\text{right}) =$ 0.6
- $\mathcal{P}(\text{left}) =$ 0.0

Policy 3:

- $\mathcal{P}(\text{up}) =$ 0.4
- $\mathcal{P}(\text{down}) =$ 0.1
- $\mathcal{P}(\text{right}) =$ 0.5
- $\mathcal{P}(\text{left}) =$ 0.0

Policy 4:

- $\mathcal{P}(\text{up}) =$ 0.5
- $\mathcal{P}(\text{down}) =$ 0.1
- $\mathcal{P}(\text{right}) =$ 0.3
- $\mathcal{P}(\text{left}) =$ 0.1

(ii) [4 pts] State E.

- $\mathcal{P}(\text{up}) =$ 0.0
- $\mathcal{P}(\text{down}) =$ 0.0
- $\mathcal{P}(\text{right}) =$ 0.2
- $\mathcal{P}(\text{left}) =$ 0.8

For state E, the optimal action is left, so the stochastic policy increases that probability by 0.3, and lowers all others by 0.1.
This part has different solutions depending on the stochastic policy in the exam version, with each version below.
Policy 1:

- $\mathcal{P}$(up) = 0.0
- $\mathcal{P}$(right) = 0.2
- $\mathcal{P}$(down) = 0.0
- $\mathcal{P}$(left) = 0.8

Policy 2:

- $\mathcal{P}$(up) = 0.0
- $\mathcal{P}$(right) = 0.3
- $\mathcal{P}$(down) = 0.0
- $\mathcal{P}$(left) = 0.7

Policy 3:

- $\mathcal{P}$(up) = 0.0
- $\mathcal{P}$(right) = 0.1
- $\mathcal{P}$(down) = 0.0
- $\mathcal{P}$(left) = 0.9

Policy 4:

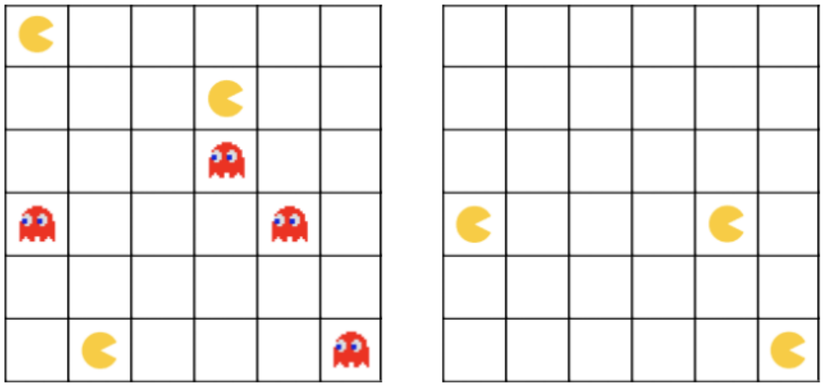
- $\mathcal{P}$(up) = 0.0
- $\mathcal{P}$(right) = 0.1
- $\mathcal{P}$(down) = 0.0
- $\mathcal{P}$(left) = 0.9

Q4. [18 pts] Pacfriends Unite

Pacman and his Pacfriends have decided to combine forces and go on the offensive, and are now chasing ghosts instead! In a grid of size M by N, Pacman and $P-1$ of his Pacfriends are moving around to collectively eliminate **all** of the ghosts in the grid by stepping on the same square as each of them. Moving onto the same square as a ghost will eliminate it from the grid, and move the Pacman into that square.

Every turn, Pacman and his Pacfriends may choose one of the following four actions: *left, right, up, down*, but may not collide with each other. In other words, any action that would result in two or more Pacmen occupying the same square will result in no movement for either Pacman or the Pacfriends. Additionally, Pacman and his Pacfriends are **indistinguishable** from each other. There are a total of G ghosts, which are indistinguishable from each other, and cannot move.

Treating this as a search problem, we consider each configuration of the grid to be a state, and the goal state to be the configuration where **all** of the ghosts have been eliminated from the board. Below is an example starting state, as well as an example goal state:


(a) Possible Start State (b) Possible Goal State

Assume each of the following subparts are **independent** from each other. **Also assume that regardless of how many Pacmen move in one turn, the total cost of moving is still 1.**

(a) [8 pts] Suppose that Pacman has no Pacfriends, so $P = 1$.

(i) [2 pts] What is the size of the minimal state space representation given this condition? Recall that $P = 1$.

○ MN	○ $(MN)^G$	○ 2^{MN}	○ $G(2)^{MN}$
○ MNG	○ $(MN)^{G+1}$	○ 2^{MN+G}	[✓ $MN(2)^G$]

Since $P = 1$, we only need to keep track of the position of Pacman, as well as whether each of the G ghosts has been eliminated. We can keep track of this with MN and 2^G respectively; therefore, the size of the minimal representation is the product of the two, which is $MN(2)^G$. Note that we do not need to keep track of the ghosts' positions since they are stationary.

For each of the following heuristics, select whether the heuristic is only admissible, only consistent, neither, or both. Recall that $P = 1$.

Note that consistency implies admissibility, so just 'consistent' will never be the correct answer.

(ii) [2 pts] $h(n)$ = the sum of the Manhattan distances from Pacman to every ghost.

○ only admissible ○ only consistent [✓ neither] ○ both

In this case, we can consider a dummy example, such that all the ghosts are adjacent and lined up horizontally, and Pacman is one square to their left. If there are three ghosts, then the sum of their Manhattan distances from Pacman is $1 + 2 + 3 = 6$, which is an overestimate of the actual cost 3, since the Pacman only needs to move three spaces to actually eliminate all the ghosts.

(iii) [2 pts] $h(n)$ = the number of ghosts times the maximum Manhattan distance between Pacman and any of the ghosts.

○ only admissible ○ only consistent [✓ neither] ○ both

Considering the same example as in the previous part, this heuristic also overestimates the cost, since the number of ghosts × Manhattan distance between Pacman and the furthest ghost is $3 \times 3 = 9 > 3$.
This question was randomized to say 'minimum Manhattan distance' as well, but the answer is still neither. Consider the case where Pacman is very far from the closest ghost, but all the ghosts are next to each other. The heuristic will give us an overestimate to collect all the ghosts, because going the minimum distance value to each ghost is an overestimate once you've reached the closest ghost.

(iv) [2 pts] $h(n)$ = the number of remaining ghosts.

○ only admissible ○ only consistent ○ neither [✓ both]

Without his Pacfriends, Pacman can only eliminate at most one ghost each turn, and therefore the number of ghosts will be a lower bound on the number of moves necessary to eliminate all the ghosts, making this heuristic admissible. Additionally, since between moves, the number of ghosts can only be reduced by at most one, the heuristic is also consistent.

(b) [10 pts] Suppose that Pacman has exactly one less Pacfriend than there are number of ghosts; therefore $P = G$. Recall that Pacman and his Pacfriends are indistinguishable from each other.

(i) [2 pts] What is the size of the minimal state space representation given this condition? Recall that $P = G$.

○ MNP	○ $(MN)^G P$	○ $\binom{MN}{P}(MN)^G$
○ $MNGP$	○ $(MN)^{G+1}$	○ $\binom{MN}{P}\binom{MN}{G}$
○ $(MN)^G$	○ $(MN)^{(G+1)P}$	○ 2^{MN}
○ $(MN)^{(G+P)}$	○ $\binom{MN}{P}$	○ 2^{MN+G+P}
○ $(MN)^P 2^G$	[✓ $\binom{MN}{P}2^G$]	○ $GP(2)^{MN}$

Since P can be any number of Pacman and Pacfriends, arbitrarily equal to G, we need to keep track of the position of all the Pacmen, as well as whether each of the G ghosts has been eliminated; however, since the Pacmen are **indistinguishable** from each other, we can consider states where different Pacmen have swapped positions to be the same. Therefore, we can keep track of unique Pacmen configurations using $\binom{MN}{P}$, which is the number of ways to select P Pacman positions from a grid of M by N. Therefore, the size of the minimal representation is the product of this and 2^G, which is $\binom{MN}{P}2^G$.

For each of the following heuristics, select whether the heuristic is only admissible, only consistent, neither, or both. Recall that $P = G$.
Note that consistency implies admissibility, so just 'consistent' will never be the correct answer.

(ii) [2 pts] $h(n)$ = the largest of the Manhattan distances between each Pacman and its closest ghost.

○ only admissible ○ only consistent [✓ neither] ○ both

Consider the case where we have two Pacmen and two ghosts, and one Pacman is very close to both ghosts, while the other is very far away. This heuristic would overestimate the cost since it would take the distance from the furthest Pacman to the ghosts as its metric, when the closer Pacman could just eliminate both ghosts quicker.

(iii) [2 pts] $h(n)$ = the smallest of the Manhattan distances between each Pacman and its closest ghost.

○ only admissible ○ only consistent ○ neither [✓ both]

At minimum, it will take the smallest of the distances between each Pacman and its closest ghost for any of the ghosts to be eliminated. Therefore this heuristic will always underestimate the true cost and is therefore admissible. To see that this is consistent, we can say that the difference between $h(a) - h(b)$ for going from any state to the next state is an underestimate of the cost between states. In fact, consider the scenario with two Pacmen and two ghosts, where one ghost is next to one Pacman and far from the other, while the other ghost is far from both. The heuristic for this state will be 1, since one of the ghosts is adjacent to one of the Pacmen, but as soon as that Pacman eliminates that ghost, the subsequent state will have a much larger heuristic value, since the minimum Manhattan distance between either Pacman and the remaining ghost will be very high; even in this edge case, $h(A) - h(B)$ will be negative, meaning that it is still an underestimate of the cost between states. Therefore, the heuristic is consistent.

(iv) [2 pts] $h(n) =$ the number of remaining ghosts.

○ admissible ○ only consistent [✓ neither] ○ both

Since multiple ghosts could be eliminated in one move because there is more than one Pacman eliminating them, this heuristic can overestimate the cost to the goal state, and is therefore neither consistent nor admissible.

(v) [2 pts] $h(n) = \frac{\text{number of remaining ghosts}}{P}$.

○ only admissible ○ only consistent ○ neither [✓ both]

Since we are given that $P = G$, the maximum value of this heuristic is 1, since $h_6(n)$ is at most G. Therefore this heuristic is admissible since it has a value always less than or equal to 1, the cost of taking a turn, and when there are no remaining ghosts left, the heuristic evaluates to 0, so it will underestimate the cost to the goal state. It is also consistent since the heuristic value decreases fractionally in increments less than 1, and so it always underestimates the cost between two states.

Q5. [19 pts] Searching for a Problem...

(a) [2 pts] Mark the statement(s) below that are correct. Assume all search problems are on finite graphs for this problem.

- 🟥 DFS graph search is guaranteed to find a solution (if one exists).
- 🟥 A* graph search is guaranteed to find a solution (if one exists) with any admissible heuristic.
- 🟥 UCS does not always find the optimal solution.
- 🟥 There are cases where BFS tree search finds a solution but DFS tree search doesn't.
- ☐ There are cases where DFS tree search finds a solution but BFS tree search doesn't.

DFS graph search will not get stuck in loops, since we keep track of explored states.

A* graph search is complete. For finite graphs, since there will be a finite amount of nodes whose estimated path cost will be less than the actual goal cost.

UCS is optimal if we assume that edge weight are non-negative. On graphs with negative edge weights, it may not find the optimal solution.

BFS is complete, so BFS will always find a solution if one exists, regardless of if DFS can find one or not.

(b) [7 pts] Turtle Tee is crawling within an $M \times N$ coral habitat, located in the middle of an $X \times Y$ area. Tee knows that his dinner has been hidden by a naughty fish within the coral habitat, and Tee needs to find it to stay alive. The food will be consumed once (and only once) when Turtle Tee reaches its position. Even though Tee cannot leave the coral habitat, he is determined to survive and would like your help!

(i) [3 pts] Please complete the following expression that evaluates to the size of the minimum state space by filling in the blanks. For example, writing 0 in all the blanks will result in taking all of these variables to the 0th power:

$$2^{(a)} \cdot M^{(b)} \cdot N^{(c)} \cdot X^{(d)} \cdot Y^{(e)}$$

(a) $\boxed{0}$ **(b)** $\boxed{1}$ **(c)** $\boxed{1}$ **(d)** $\boxed{0}$ **(e)** $\boxed{0}$

MN would be the correct minimal state space, because the only thing we need to keep track of is the location of Tee, MN. We don't need to keep track of whether or not the food has been consumed using a boolean 2 because we would be able to tell if Tee's location matches the food's location, and the search problem would end once we Tee reaches this goal location. The X and Y terms are extraneous and unnecessary, since Tee would never move anywhere within that full area to get the food.

(ii) [4 pts] We now know that the naughty fish also set up $K > 10$ traps inside the coral region (many could be overlapping, and could be right on top of the food Tee is looking for). **Traps are visible, stationary, and each trap will trigger every time it is stepped on.** If Tee steps on 10 traps before he finds his dinner, then he would be so severely injured that he couldn't move anymore, and would then starve! We still want to make sure that the food can only be consumed once (when and only when Turtle Tee steps on that square).

Please complete the following expression that evaluates to the size of the minimum state space where **(g)** represents a constant and A **represents the size of the correct minimal state space from the previous part:**

$$A \cdot 2^{(\mathbf{a})} \cdot M^{(\mathbf{b})} \cdot N^{(\mathbf{c})} \cdot X^{(\mathbf{d})} \cdot Y^{(\mathbf{e})} \cdot K^{(\mathbf{f})} \cdot (\mathbf{g})$$

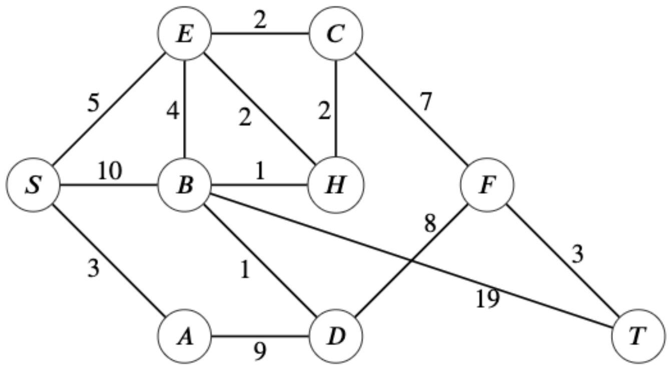
(a) $\boxed{0}$ **(b)** $\boxed{0}$ **(c)** $\boxed{0}$ **(d)** $\boxed{0}$ **(e)** $\boxed{0}$ **(f)** $\boxed{0}$ **(g)** $\boxed{11}$

Since each trap is stationary and triggers multiple times, we don't need to actually keep track of their locations or individually track booleans, and instead just count the number of traps that we have currently stepped on. This can be encompassed by a constant of size 11, representing number of traps stepped on from 0 to 10.

Essentially, this means that the only thing that will differ between this and the previous part is the constant 11, so the remaining values should just be 0. Since A represents the answer to the previous part, we don't need to repeat the values from part b.i.

The number of traps was randomized between versions, but the correct answer for (g) will be the number of traps plus one.

(c) [10 pts] Suppose the numbers in the following graph represent the cost of traversing its corresponding edge. In this problem, we use a priority queue representation for our fringe in **all** of the following search algorithms, and break ties alphabetically for items at the same priority. We start from node S (leftmost), and the goal is to find node T (rightmost). Also provided are the heuristic values for each node.



Node	S	A	B	C	D	E	F	H	T
h(Node)	19	15	18	9	10	12	3	2	1

For the following parts, list the order of nodes expanded using the specified search algorithm separated by commas (no spaces), e.g., an example ordering is S,A,D,F,T. Note that we're using a priority queue representation for each search algorithm, and breaking ties at the same priority alphabetically. This affects tiebreaking orders of when nodes are expanded.

(i) [2 pts] What is the order of node expansion using BFS graph search?

$\boxed{S,A,B,E,C,D,H,T}$

For brevity, nodes that were previous expanded or added to the fringe will not be readded to the fringe, since BFS wouldn't expand them in that order anyway. For BFS, node priority will be their level.
Starting at node S:
Fringe: $(S:0)$; Expanded: None
Fringe: $(A:1),(B:1),(E:1)$; Expanded: S
Fringe: $(B:1),(E:1),(D:2)$; Expanded: S,A
Fringe: $(E:1),(D:2),(H:2),(T:2)$; Expanded: S,A,B
Fringe: $(C:2),(D:2),(H:2),(T:2)$; Expanded: S,A,B,E
Fringe: $(D:2),(H:2),(T:2),(F:3)$; Expanded: S,A,B,E,C
Fringe: $(H:2),(T:2),(F:3)$; Expanded: S,A,B,E,C,D
Fringe: $(T:2),(F:3)$; Expanded: S,A,B,E,C,D,H
Fringe: $(F:3)$; Expanded: $S,A,B,E,C,D,H,T \rightarrow$ expanded goal state!

(ii) [2 pts] What is the order of node expansion using DFS graph search?

$\boxed{S,A,D,B,E,C,F,T}$

To use a priority queue for DFS, we can set the priority of a node in DFS to be one less than that of the parent. As such, we prioritize expanding the most recently added nodes. Starting at Node S:
Fringe: $(S:0)$; Expanded: None
Fringe: $(A:-1),(B:-1),(E:-1)$; Expanded: S
Fringe: $(D:-2),(B:-1),(E:-1)$; Expanded: S,A
Fringe: $(B:-3),(F:-3),(E:-1)$; Expanded: S,A,D
Fringe: $(E:-4),(H:-4),(F:-3)$; Expanded: S,A,D,B
Fringe: $(C:-5),(H:-5)$; Expanded: S,A,D,B,E
Fringe: $(F:-6),(H:-6)$; Expanded: S,A,D,B,E,C
Fringe: $(T:-7),(H:-6)$; Expanded: S,A,D,B,E,C,F
Fringe: $(H:-6)$; Expanded: $S,A,D,B,E,C,F,T \rightarrow$ expanded goal state!

(iii) [2 pts] What is the order of node expansion using UCS graph search?

$\boxed{S,A,E,C,H,B,D,F,T}$

Starting at Node S:
Fringe: $(S:0)$; Expanded: None

Fringe: $(A : 3), (E : 5), (B : 10)$; Expanded: S
Fringe: $(E : 5), (B : 10), (D : 12)$; Expanded: S, A
Fringe: $(C : 7), (H : 7), (B : 9), (D : 12)$; Expanded: S, A, E
Fringe: $(H : 7), (B : 9), (D : 12), (F : 14)$; Expanded: S, A, E, C
Fringe: $(B : 8), (D : 12), (F : 14)$; Expanded: S, A, E, C, H
Fringe: $(D : 9), (F : 14), (T : 26)$; Expanded: S, A, E, C, H, B
Fringe: $(F : 14), (T : 26)$; Expanded: S, A, E, C, H, B, D
Fringe: $(T : 17)$; Expanded: S, A, E, C, H, B, D, F
Fringe: None; Expanded: $S, A, E, C, H, B, D, F, T \rightarrow$ expanded goal state!

(iv) [2 pts] What is the order of node expansion using A* graph search? $\boxed{S, E, H, C, F, A, T}$

Starting at Node S:
Fringe: $(S : 19)$; Expanded: None
Fringe: $(E : 17), (A : 18), (B : 28)$; Expanded: S
Fringe: $(H : 9), (C : 16), (A : 18), (B : 27)$; Expanded: S, E
Fringe: $(C : 16), (A : 18), (B : 26)$; Expanded: S, E, H
Fringe: $(F = 17), (A : 18), (B : 26)$; Expanded: S, E, H, C
Fringe: $(A : 18), (T : 18), (B : 26), (D : 32)$; Expanded: S, E, H, C, F
Fringe: $(T : 18), (D : 22), (B : 26)$; Expanded: S, E, H, C, F, A
Fringe: $(D : 22), (B : 26)$; Expanded: $S, E, H, C, F, A, T \rightarrow$ expanded goal state!

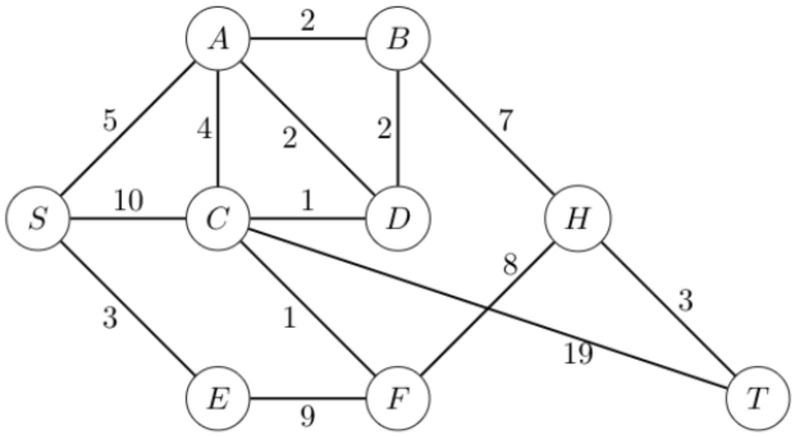
(v) [2 pts] Using the heuristics table associated with the graph, select the node(s) that have inadmissible heuristic value(s):

🟥 S	☐ C	☐ F
☐ A	☐ D	☐ H
🟥 B	☐ E	🟥 T

Any heuristic that overestimates the cost from that node to the goal will be inadmissible, so all nodes where $h(\text{Node}) >$ True Cost.

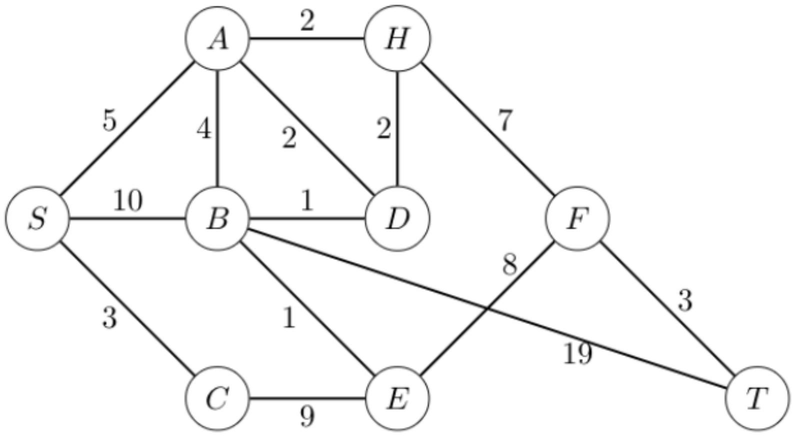
Node	h(Node)	True Cost
S	19	17
A	15	20
B	18	12
C	9	10
D	10	11
E	12	12
F	3	3
H	2	12
T	1	0

Solutions for alternate graphs:



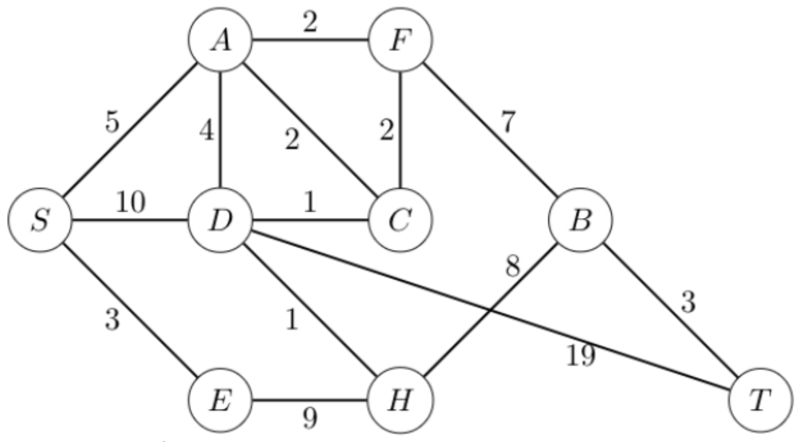
Node	S	A	B	C	D	E	F	H	T
$h(Node)$	19	12	9	18	2	15	10	3	1

Alternate 1:
BFS: S, A, C, E, B, D, F, T
DFS: S, A, B, D, C, F, E, H, T
UCS: S, E, A, B, D, C, F, H, T
A*: S, A, D, B, H, E, T
Inadmissible Heuristics: S, C, T



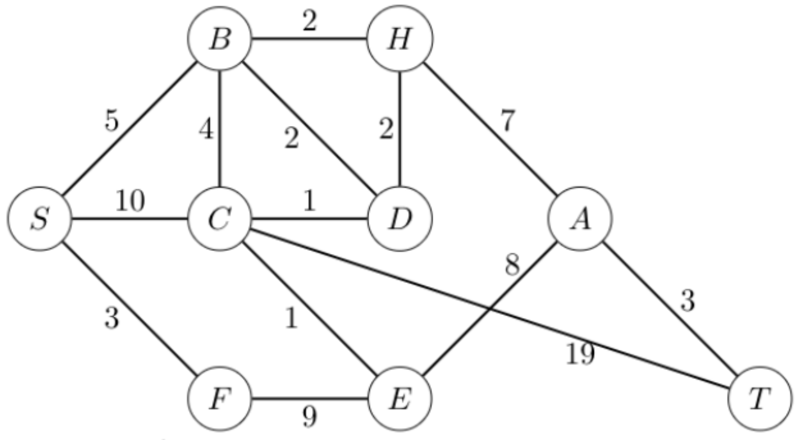
Node	S	A	B	C	D	E	F	H	T
$h(Node)$	19	12	18	15	2	10	3	9	1

Alternate 2:
BFS: S, A, B, C, D, E, H, T
BFS (partial): S, A, B, C, D, H, E, T (incorrect tiebreaking, but rest correct)
DFS: S, A, B, D, H, F, E, C, T
UCS: S, C, A, D, H, B, E, F, T
A*: S, A, D, H, F, C, T
Inadmissible Heuristics: S, B, T



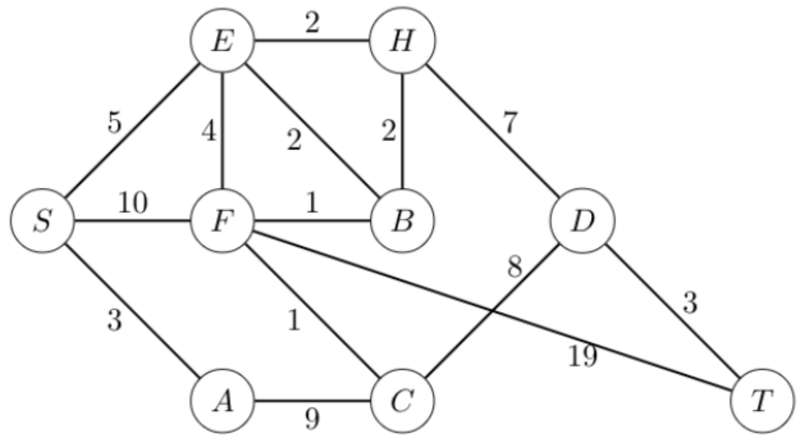
Node	S	A	B	C	D	E	F	H	T
$h(Node)$	19	12	3	2	18	15	9	10	1

Alternate 3:
BFS: S, A, D, E, C, F, H, T
DFS: S, A, C, D, H, B, F, T
UCS: S, E, A, C, F, D, H, B, T
A*: S, A, C, F, B, E, T
Inadmissible Heuristics: S, D, T



Node	S	A	B	C	D	E	F	H	T
$h(Node)$	19	3	12	18	2	10	15	9	1

Alternate 4:
BFS: S, B, C, F, D, E, H, T
BFS (partial): S, B, C, F, D, H, E, T (incorrect tiebreaking, but rest correct)
DFS: S, B, C, D, H, A, E, F, T
UCS: S, F, B, D, H, C, E, A, T
A*: S, B, D, H, A, F, T
Inadmissible Heuristics: S, C, T



Node	S	A	B	C	D	E	F	H	T
$h(Node)$	19	15	2	10	3	12	18	9	1

Alternate 5:
BFS: S, A, E, F, B, C, H, T
BFS (partial): S, A, E, F, C, B, H, T (incorrect tiebreaking, but rest correct)
DFS: S, A, C, D, H, B, E, F, T
UCS: S, A, E, B, H, F, C, D, T
A*: S, E, B, H, D, A, T
Inadmissible Heuristics: S, F, T