CS 188 Spring 2019 Introduction to Artificial Intelligence Midterm

- You have 110 minutes. The time will be projected at the front of the room. You may not leave during the last 10 minutes of the exam.
- Do NOT open exams until told to. Write your SIDs in the top right corner of every page.
- If you need to go to the bathroom, bring us your exam, phone, and SID. We will record the time.
- 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. If a clarification is needed, it will be projected at the front of the room. **Make sure to periodically check the clarifications**.
- The exam is closed book, closed laptop, and closed notes except your one-page cheat sheet. Turn off and put away all electronics.
- Mark your answers ON THE EXAM ITSELF IN THE DESIGNATED ANSWER AREAS. We will not grade anything on scratch paper.
- The last two sheet in your exam packet is a sheet of scratch paper. Please detach it from your exam.
- For multiple choice questions:
  - ☐ means mark ALL options that apply
  - ◯ means mark ONE choice
  - When selecting an answer, please fill in the bubble or square COMPLETELY (● and ■)

First name	
Last name	
SID	
Student to the right (SID and Name)	
Student to the left (SID and Name)	

For staff use only:

Q1.	Search Party	/13
Q2.	Bike Bidding Battle	/15
Q3.	How do you Value It(eration)?	/17
Q4.	Q-uagmire	/12
Q5.	Dudoku	/18
Q6.	In What Worlds?	/13
Q7.	Proba-Potpourri	/12
	Total	/100

Q1. [13 pts] Search Party

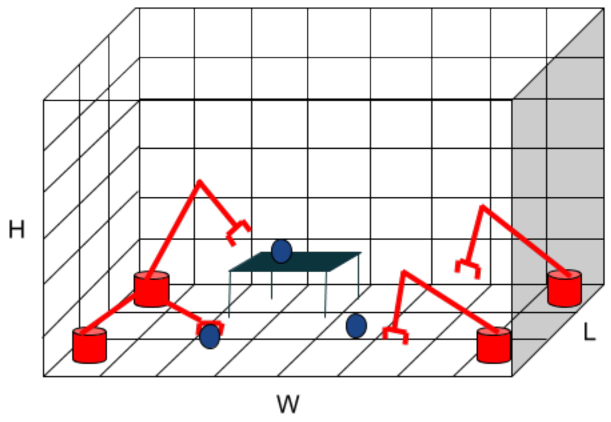
(a) [3 pts] First consider **general search problems**, which of the following are true?

- 🟥 Code that implements A* tree search can be used to run UCS.
- ☐ A* tree search is optimal with any heuristic function.
- ☐ A* tree search is complete with any heuristic function.
- ☐ A* graph search is guaranteed to expand no more nodes than DFS.
- 🟥 The max of two admissible heuristics is always admissible.
- ☐ A heuristic that always evaluates to $h(s) = 1$ for non-goal search nodes s is always admissible.

1. True. Because we can set heuristic function $h(s) = 0$.
2. False. Because we need admissible heuristic to guarantee the optimal solution.
3. False. Suppose h = -2g. Then f=-g and A* is depth-first search.
4. False. Because depth-first graph search could go a sub-optimal solution.
5. True. Because each of the two heuristics is guaranteed to underestimate the distance from the given node to the goal, and so therefore must their max.
6. False. Because the edge weights could be less than one as float number.

Now consider the following real-world scenario. Annie is throwing a party tonight, but she only has a couple hours to get ready. Luckily, she was recently gifted 4 one-armed robots! She will use them to rearrange her room for the guests while she is busy getting everything else ready. Here are the specifications:

- Her room is modeled as a W-by-L-by-H 3D grid in which there are N objects (which could be anywhere in the grid to start with) that need rearrangement.
- Each object occupies one grid cell, and no two objects can be in the same grid cell. Do not consider any part of the robot an "object."
- At each time-step, one robot may take an action $\in$ {move gripper to legal grid cell, close gripper, open gripper}. Moving the gripper does not change whether the gripper was closed/open.
- A robot can move an object by
  1. Moving an open gripper into the object's grid cell
  2. Closing the gripper to grasp the object
  3. Moving to desired location
  4. Opening the gripper to release the object in-hand.
- The robots do not have unlimited range. The arm can move to any point *within* the room that is strictly less than R grid cells from its base per direction along each axis. Explicitly, if $R = 2$ and a robot's base is at (0,0,0), the robot cannot reach (0,0,2) but can reach (1,1,1). Assume $R < W, L, H$.



(b) [4 pts] Annie stations one robot's stationary base at each of the 4 *corners* of the room (see figure for example of this with 3 objects). Thankfully, she knows where each of the N objects in the room should be and uses that to define the robots' goal. Complete the following expression such that it evaluates to the size of the minimal state space. Please approximate permutations as follows: X permute $Y \approx X^Y$. You may use scalars and the variables: W, L, H, R, and N in your answers.

$$2^{(a)} \cdot N^{(b)} \cdot R^{(c)} \cdot W^{(d)} \cdot L^{(e)} \cdot H^{(f)}$$

(a): $\boxed{}$ (b): $\boxed{}$ (c): $\boxed{}$

(d): $\boxed{}$ (e): $\boxed{}$ (f): $\boxed{}$

$(R^3)^4 \cdot (W \cdot L \cdot H)^N \cdot 2^4$.
We need to keep track of the positions of each of the robot's grippers one of which is R^3. This comes from the fact that they are stationed at the corners of the room. Then, they can only move R in each direction (since it's 3D there are 3 directions). Since there are 4 robots, we have $(R^3)^4$.
To keep track of the objects (which could be anywhere), we have $(W * L * H)$ possibilities for N objects.
We need to the open/close status (boolean) of the gripper because we need to know that for picking up/dropping objects to move them, this gives us 2^4.

(c) Each of the following describes a modification to the scenario previously described and depicted in the figure. **Consider each modification independently (that is, the modifications introduced in (i) are *not* present in (ii)).** For each scenario, give the size of the new minimal state space.
Please use the symbol X as a proxy for the correct answer to (b) in your answers.

(i) [3 pts] The robots are given wheels, so each base is able to slide along the floor (they still cant jump) from their original corners. That is, at each time-step, a robot has a new action that allows them to move its (once stationary) base arbitrarily far across the floor. When the robot slides its base, the relative arm position and status of the gripper remain the same.

$\boxed{}$

$(W \cdot L \cdot)^4 \cdot ((2R) \cdot (2R) \cdot R)^4 \cdot (W \cdot L \cdot H)^N \cdot 2^4$
We need to keep track of where the base is now and it's no longer restricted, this is the $(W \cdot L)^4$ factor. Then we need to keep track of the relative position of the gripper to its base since it has limited range. $2R$ is technically incorrect but it's a simplifying approximation. Correct answer also given credit this is $(2R - 1)$ if it can move R in each direction). This is where $((2R)(2R)R)^4$ comes from. Then we still need the $(WLH)^N$ and 2^R from earlier.

(ii) [3 pts] *One* robot is defective and can move for a maximum of T timesteps before it must rest for at least S timesteps. You may use S or T in your expression.

$\boxed{}$

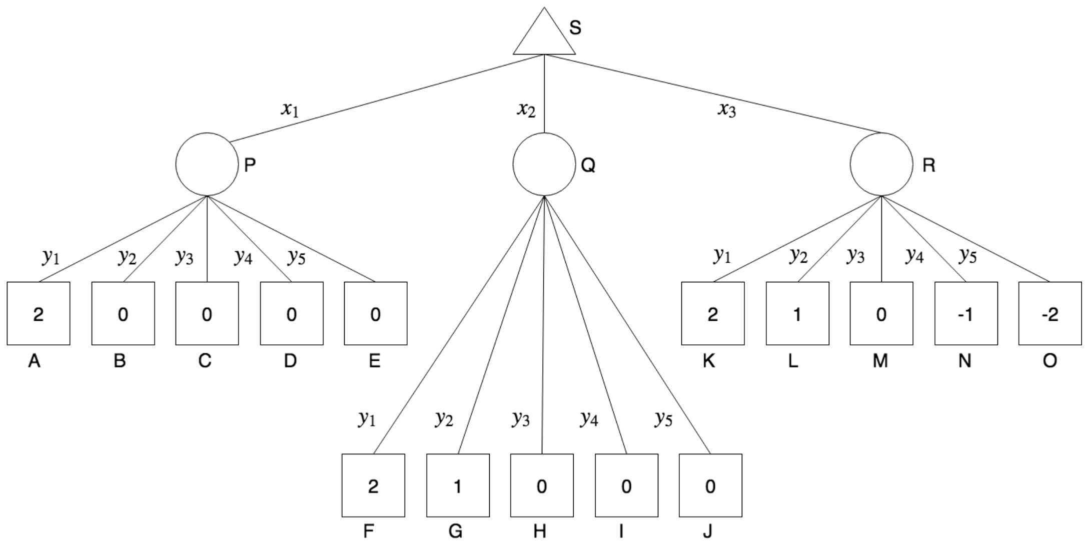
$(S+T) \cdot (R^3)^4 \cdot (W \cdot L \cdot H)^N \cdot 2^4$

We need to keep track of how long we've been moving/resting for. The minimal way to represent this is with a range 1 to S+T where 1 represents that we're ready to move again and S+T means we've moved since resting. Every step in between decrements. The successor function takes in a state and action, we need to know this piece in the state in order to use it in the successor function (it is changing unlike the range restriction from part (i))

Q2. [15 pts] Bike Bidding Battle

Alyssa P. Hacker and Ben Bitdiddle are bidding in an auction at Stanley University for a bike. Alyssa will either bid x_1, x_2, or x_3 for the bike. She knows that Ben will bid y_1, y_2, y_3, y_4, or y_5, but she does not know which. All bids are nonnegative.

(a) Alyssa wants to maximize her payoff given by the expectimax tree below. The leaf nodes show Alyssa's payoff. The nodes are labeled by letters, and the edges are labeled by the bid values x_i and y_i. The maximization node S represents Alyssa, and the branches below it represent each of her bids: x_1, x_2, x_3. The chance nodes P, Q, R represent Ben, and the branches below them represent each of his bids: y_1, y_2, y_3, y_4, y_5.



(i) [2 pts] Suppose that Alyssa believes that Ben would bid any bid with equal probability. What are the values of the chance (circle) and maximization (triangle) nodes?

1. Node P ___0.4___
2. Node Q ___0.6___
3. Node R ___0___
4. Node S ___0.6___

(ii) [1 pt] Based on the information from the above tree, how much should Alyssa bid for the bike?

○ x_1 [✓ x_2] ○ x_3

(b) [3 pts] Alyssa does expectimax search by visiting child nodes from left to right. Ordinarily expectimax trees cannot be pruned without some additional information about the tree. Suppose, however, that Alyssa knows that the leaf nodes are ordered such that payoffs are non-increasing from left to right (the leaf nodes of the above diagram is an example of this ordering). Recall that if node X is a child of a maximizer node, a child of node X may be pruned if we know that the value of node X will never be $>$ some threshold (in other words, it is $\leq$ that threshold). Given this information, if it is possible to prune any branches from the tree, mark them below. Otherwise, mark "None of the above."

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

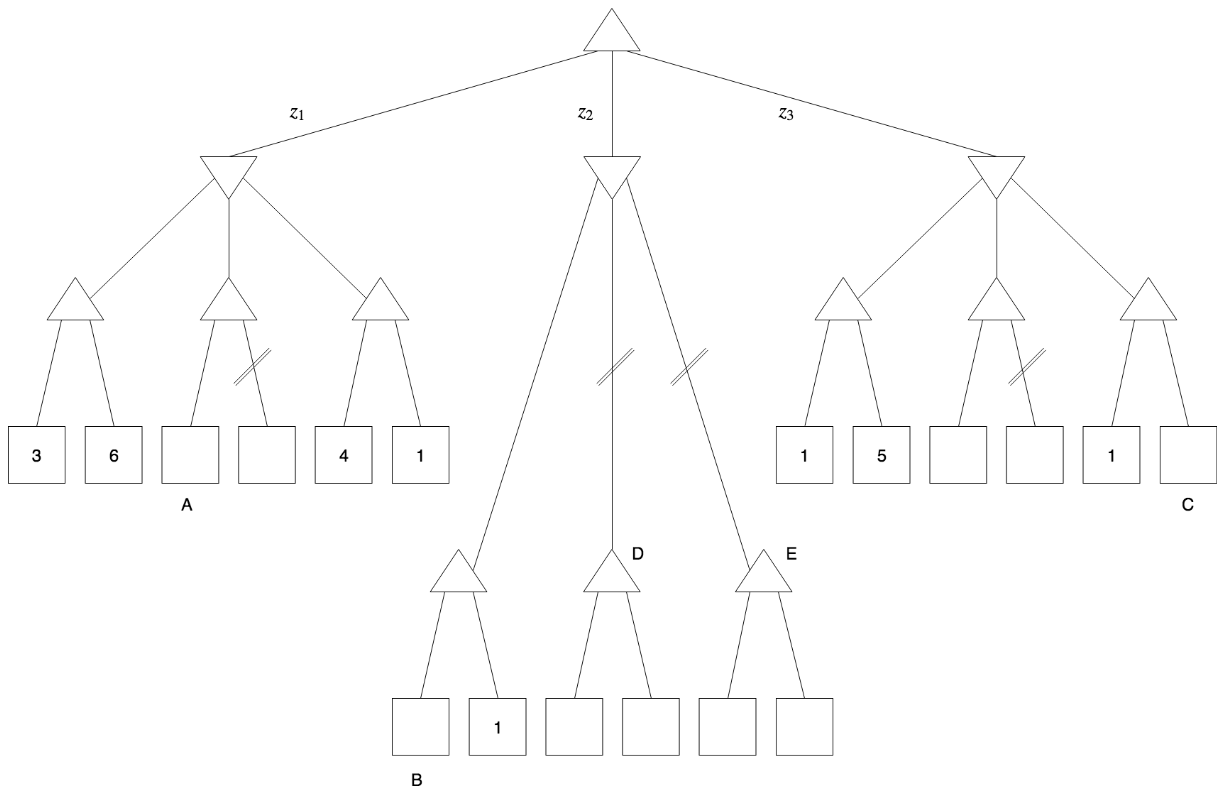
To prune the children of a chance node in an expectimax tree, Alyssa would need to keep track of a threshold on the value of the chance node: if at some point while searching left to right, she realizes that the value of the chance node will never be higher than its threshold, she can prune the remaining children of the chance node.

Alyssa needs to search the entire left subtree because she does not have a threshold against which to compare the value of P.

When Alyssa searches the center subtree, she knows that the maximizing node will only consider taking action x_2 if the value of Q is higher than the value of P, which is 0.4. If at some point Alyssa realizes that the value of Q will never be higher than 0.4, she can prune the remaining children. After exploring node G, Alyssa knows that nodes H, I, and J are ≤ 1, which means that the value of node Q is at most 1.2. After exploring node H, Alyssa knows that nodes I and J are ≤ 0, which means that the value of node Q is at most 0.6. After exploring node I, Alyssa knows that node J is ≤ 0, which means that the value of node Q is at most 0.6. This is not enough information to prune any of the nodes in the center subtree because at no point does Alyssa know for sure that the value of Q is ≤ 0.4.

When Alyssa searches the right subtree, if at some point Alyssa ralizes that the value of R will never be higher than 0.6, then she can prune the remaining of children of R. After exploring node L, Alyssa knows that the nodes M, N, and O are ≤ 1, which means that the value of node R is at most 1.2. After exploring node M, Alyssa knows that nodes N and O are ≤ 0, which means that the value of node Q is at most 0.6. At this point, Alyssa can prune nodes N and O because they can only make the value of R lower than the value of Q.

(c) [4 pts] Unrelated to parts (a) and (b), consider the minimax tree below. whose leaves represent payoffs for the maximizer. The crossed out edges show the edges that are pruned when doing naive alpha-beta pruning visiting children nodes from left to right. Assume that we prune on equalities (as in, we prune the rest of the children if the current child is $\leq \alpha$ (if the parent is a minimizer) or $\geq \beta$ (if the parent is a maximizer)).



Fill in the inequality expressions for the values of the labeled nodes A and B. Write ∞ and $-\infty$ if there is no upper or lower bound, respectively.

1. $\boxed{6} \leq \text{A} \leq \boxed{\infty}$

2. $\boxed{-\infty} \leq \text{B} \leq \boxed{4}$

(d) [1 pt] Suppose node B took on the largest value it could possibly take on and still be consistent with the pruning scheme above. After running the pruning algorithm, we find that the values of the left and center subtrees have the same minimax value, both 1 greater than the minimax value of the right subtree. Based on this information, what is the numerical value of node C?

○ 1 ○ 2 [✓ 3] ○ 4 ○ 5 ○ 6 ○ 7 ○ 8 ○ 9 ○ 10

(e) [4 pts] For which values of nodes D and E would choosing to take action z_2 be *guaranteed* to yield the same payoff as action z_1? Write ∞ and $-\infty$ if there is no upper or lower bound, respectively (this would correspond to the case where nodes D and E can be any value).

1. $\boxed{4} \leq \text{D} \leq \boxed{\infty}$

2. $\boxed{4} \leq \text{E} \leq \boxed{\infty}$

When doing naive alpha-beta pruning, the values propagated up to the parent nodes are not necessarily exact, but rather bounds. If $D < 4$ or $E < 4$, then the true minimax value of the center subtree is less than the true minimax value of the left subtree.

Q3. [17 pts] How do you Value It(eration)?

(a) Fill out the following True/False questions.

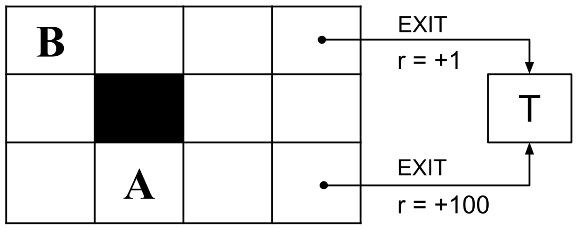
(i) [1 pt] [✓ ● True] ○ False: Let A be the set of all actions and S the set of states for some MDP. Assuming that $|A| \ll |S|$, one iteration of value iteration is generally faster than one iteration of policy iteration that solves a linear system during policy evaluation. One iteration of value iteration is $O(|S|^2|A|)$, whereas one iteration of policy iteration is $O(|S|^3)$, so value iteration is generally faster when $|A| \ll |S|$

(ii) [1 pt] ○ True [✓ ● False]: For any MDP, changing the discount factor does not affect the optimal policy for the MDP. Consider an infinite horizon setting where we have 2 states A, B, where we can alternate between A and B forever, gaining a reward of 1 each transition, or exit from B with a reward of 100. In the case that $\gamma = 1$, the optimal policy is to forever oscillate between A and B. If $\gamma = \frac{1}{2}$, then it is optimal to exit.

The following problem will take place in various instances of a grid world MDP. Shaded cells represent walls. In all states, the agent has available actions $\uparrow, \downarrow, \leftarrow, \rightarrow$. Performing an action that would transition to an invalid state (outside the grid or into a wall) results in the agent remaining in its original state. In states with an arrow coming out, the agent has an *additional* action $EXIT$. In the event that the $EXIT$ action is taken, the agent receives the labeled reward and ends the game in the terminal state T. Unless otherwise stated, all other transitions receive no reward, and all transitions are deterministic.

For all parts of the problem, assume that value iteration begins with all states initialized to zero, i.e., $V_0(s) = 0 \ \forall s$. **Let the discount factor be** $\gamma = \frac{1}{2}$ **for all following parts.**

(b) Suppose that we are performing value iteration on the grid world MDP below.



(i) [2 pts] Fill in the optimal values for A and B in the given boxes.

$V^*(A)$: $\boxed{25}$ $V^*(B)$: $\boxed{\frac{25}{8}}$

(ii) [2 pts] After how many iterations k will we have $V_k(s) = V^*(s)$ for all states s? If it never occurs, write "never". Write your answer in the given box.

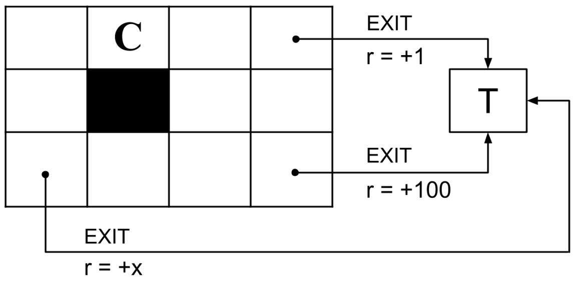
$\boxed{6}$

(iii) [3 pts] Suppose that we wanted to re-design the reward function. For which of the following new reward functions would the optimal policy **remain unchanged**? Let $R(s, a, s')$ be the original reward function.

- [✓] $R_1(s, a, s') = 10R(s, a, s')$
- [✓] $R_2(s, a, s') = 1 + R(s, a, s')$
- [✓] $R_3(s, a, s') = R(s, a, s')^2$
- ☐ $R_4(s, a, s') = -1$
- ☐ None

R_1: Scaling the reward function does not affect the optimal policy, as it scales all Q-values by 10, which retains ordering
R_2: Since reward is discounted, the agent would get more reward exiting then infinitely cycling between states
R_3: The only positive reward remains to be from exiting state +100 and +1, so the optimal policy doesn't change
R_4: With negative reward at every step, the agent would want to exit as soon as possible, which means the agent would not always exit at the bottom-right square.

(c) For the following problem, we add a new state in which we can take the $EXIT$ action with a reward of $+x$.



(i) [3 pts] For what values of x is it *guaranteed* that our optimal policy π^* has $\pi^*(C) = \leftarrow$? Write ∞ and $-\infty$ if there is no upper or lower bound, respectively. Write the upper and lower bounds in each respective box.

$\boxed{50} < x < \boxed{\infty}$

We go left if $Q(C, \leftarrow) > Q(C, \rightarrow)$. $Q(C, \leftarrow) = \frac{1}{8}x$, and $Q(C, \rightarrow) = \frac{100}{16}$. Solving for x, we get $x > 50$.

(ii) [3 pts] For what values of x does value iteration take the **minimum** number of iterations k to converge to V^* for all states? Write ∞ and $-\infty$ if there is no upper or lower bound, respectively. Write the upper and lower bounds in each respective box.

$\boxed{50} \leq x \leq \boxed{200}$

The two states that will take the longest for value iteration to become non-zero from either $+x$ or $+100$, are states C, and D, where D is defined as the state to the right of C. C will become nonzero at iteration 4 from $+x$, and D will become nonzero at iteration 4 from +100. We must bound x so that the optimal policy at D does not choose to go to $+x$, or else value iteration will take 5 iterations. Similar reasoning for D and $+x$. Then our inequalities are $\frac{1}{8}x \geq \frac{100}{16}$ and $\frac{1}{16}x \leq \frac{100}{8}$. Simplifying, we get the following bound on x: $50 \leq x \leq 200$

(iii) [2 pts] Fill the box with value k, the **minimum** number of iterations until V_k has converged to V^* for all states.

$\boxed{4}$

See the explanation for the part above

Q4. [12 pts] Q-uagmire

Consider an unknown MDP with three states (A, B and C) and two actions ($\leftarrow$ and $\rightarrow$). Suppose the agent chooses actions according to some policy π in the unknown MDP, collecting a dataset consisting of samples (s, a, s', r) representing taking action a in state s resulting in a transition to state s' and a reward of r.

s	a	s'	r
A	$\rightarrow$	B	2
C	$\leftarrow$	B	2
B	$\rightarrow$	C	-2
A	$\rightarrow$	B	4

You may assume a discount factor of $\gamma = 1$.

(a) Recall the update function of Q-learning is:

$$Q(s_t, a_t) \leftarrow (1-\alpha)Q(s_t, a_t) + \alpha\left(r_t + \gamma \max_{a'} Q(s_{t+1}, a')\right)$$

Assume that all Q-values are initialized to 0, and use a learning rate of $\alpha = \frac{1}{2}$.

(i) [3 pts] Run Q-learning on the above experience table and fill in the following Q-values:

$Q(A, \rightarrow) =$ $5/2$ $\qquad Q(B, \rightarrow) =$ $-1/2$

$$Q_1(A,\rightarrow) = \frac{1}{2}\cdot Q_0(A,\rightarrow) + \frac{1}{2}\left(2 + \gamma \max_{a'} Q(B,a')\right) = 1$$
$$Q_1(C,\leftarrow) = 1$$
$$Q_1(B,\rightarrow) = \frac{1}{2}(-2+1) = -\frac{1}{2}$$
$$Q_2(A,\rightarrow) = \frac{1}{2}\cdot 1 + \frac{1}{2}\left(4 + \max_{a'} Q_1(B,a')\right)$$
$$= \frac{1}{2} + \frac{1}{2}(4+0) = \frac{5}{2}.$$

(ii) [2 pts] After running Q-learning and producing the above Q-values, you construct a policy π_Q that maximizes the Q-value in a given state:

$$\pi_Q(s) = \arg\max_a Q(s,a).$$

What are the actions chosen by the policy in states A and B?

$\pi_Q(A)$ is equal to:

- ○ $\pi_Q(A) = \leftarrow$.
- ● $\pi_Q(A) = \rightarrow$.
- ○ $\pi_Q(A) =$ Undefined.

$\pi_Q(B)$ is equal to:

- ● $\pi_Q(B) = \leftarrow$.
- ○ $\pi_Q(B) = \rightarrow$.
- ○ $\pi_Q(B) =$ Undefined.

Note that $Q(B,\leftarrow) = 0 > -\frac{1}{2} = Q(B,\rightarrow)$.

(b) [3 pts] Use the empirical frequency count model-based reinforcement learning method described in lectures to estimate the transition function $\hat{T}(s,a,s')$ and reward function $\hat{R}(s,a,s')$. (Do not use pseudocounts; if a transition is not observed, it has a count of 0.)

Write down the following quantities. You may write N/A for undefined quantities.

$\hat{T}(A, \rightarrow, B) =$ 1 $\hat{R}(A, \rightarrow, B) =$ 3

$\hat{T}(B, \rightarrow, A) =$ 0 $\hat{R}(B, \rightarrow, A) =$ N/A

$\hat{T}(B, \leftarrow, A) =$ N/A $\hat{R}(B, \leftarrow, A) =$ N/A

(c) This question considers properties of reinforcement learning algorithms for *arbitrary* discrete MDPs; you do not need to refer to the MDP considered in the previous parts.

(i) [2 pts] Which of the following methods, at convergence, provide enough information to obtain an optimal policy? (Assume adequate exploration.)

■ Model-based learning of $T(s, a, s')$ and $R(s, a, s')$.

☐ Direct Evaluation to estimate $V(s)$.

☐ Temporal Difference learning to estimate $V(s)$.

■ Q-Learning to estimate $Q(s, a)$. Given enough data, model-based learning will get arbitrarily close to the true model of the environment, at which point planning (e.g. value iteration) can be used to find an optimal policy. Q-learning is similarly guaranteed to converge to the optimal Q-values of the optimal policy, at which point the optimal policy can be recovered by $\pi^*(s) = \arg\max_a Q(s, a)$. Direct evaluation and temporal difference learning both only recover a value function $V(s)$, which is insufficient to choose between actions without knowledge of the transition probabilities.

(ii) [2 pts] In the limit of infinite timesteps, under which of the following exploration policies is Q-learning guaranteed to converge to the optimal Q-values for all state? (You may assume the learning rate α is chosen appropriately, and that the MDP is ergodic: i.e., every state is reachable from every other state with non-zero probability.)

■ A fixed policy taking actions uniformly at random.

☐ A greedy policy.

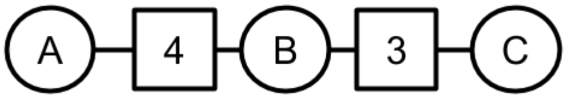
■ An ϵ-greedy policy

☐ A fixed optimal policy. For Q-learning to converge, every state-action pair (s, a) must occur infinitely often. A uniform random policy will achieve this in an ergodic MDP. A fixed optimal policy will not take any suboptimal actions and so will not explore enough. Similarly a greedy policy will stop taking actions the current Q-values suggest are suboptimal, and so will never update the Q-values for supposedly suboptimal actions. (This is problematic if, for example, an action most of the time yields no reward but occasionally yields very high reward. After observing no reward a few times, Q-learning with a greedy policy would stop taking that action, never obtaining the high reward needed to update it to its true value.)

Q5. [18 pts] Dudoku

Here we introduce Dudokus, a type of CSP problem. A Dudoku consists of variables and summation constraints. The circles indicate variables that can take integer values in a specified range and the boxes indicate summation constraints, which specify that the variables connected to the constraint need to add up to the number given in the box.

(a) Let's begin with linear Dudokus, where the variables can be arranged in a linear chain with constraints between adjacent pairs. For example, in the linear Dudoku below, variables A, B, and C need values assigned to them in the set $\{1, 2, 3\}$ such that $A + B = 4$ and $B + C = 3$.



(i) [2 pts] How many solutions does this Dudoku have?

○ 0 ○ 1 [✓ 2] ○ 3 ○ more than 3

We can enumerate possible values of A and propagate to find the values of the other variables (or a contradiction). A=1, B=3 leaves no value for C. A=2, B=2, C=1. A=3, B=1, C=2. So two solutions.

(ii) [1 pt] Consider the general case of a linear Dudoku with n variables $X_1, \dots, X_n$, each taking a value in $\{1, \dots, d\}$. What is the complexity for solving such a Dudoku using the generic tree-CSP algorithm?

○ $\mathcal{O}(nd^3)$ ○ $\mathcal{O}(n^2d^2)$ [✓ $\mathcal{O}(nd^2)$] ○ $\mathcal{O}(d^n)$

This is the standard complexity for tree-CSP, from a backward pass running $O(n)$ arc consistency checks costing $O(d^2)$ each.

(iii) [2 pts] One proposal for solving linear Dudokus is as follows: for each possible value i of the first variable X_1 in the chain, instantiate X_1 with that value and then run generic arc consistency beginning with the pair (X_2, X_1) until termination; keep going until a solution is found or there are no more values to try for X_1. Which of the following are true?

[✓] This will correctly detect any unsolvable Dudoku.

[✓] This will always solve any solvable Dudoku.

☐ This will sometimes terminate without finding a solution when one exists.

[✓] The runtime is $O(nd^3)$.

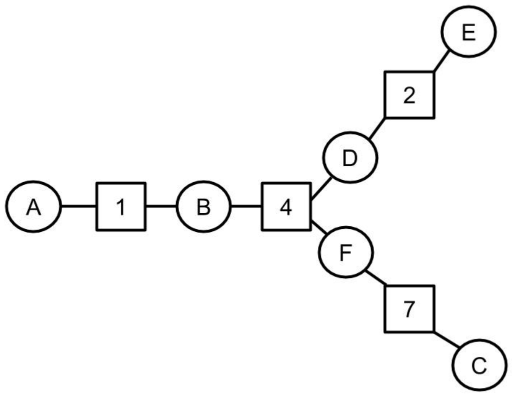
d iterations, each $O(nd^2)$ for n consistency checks.

(iv) [2 pts] Binary Dudoku constraints are *one-to-one*, meaning that if one variable in a binary constraint has a known value, there is only one possible value for the other variable. Suppose we modify arc consistency to take advantage of one-to-one constraints instead of checking all possible pairs of values when checking a constraint. Now the runtime of the algorithm in the previous part becomes:

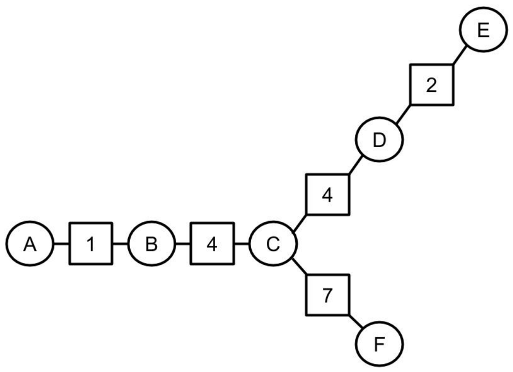
[✓ $\mathcal{O}(nd)$] ○ $\mathcal{O}(nd^3)$ ○ $\mathcal{O}(n^2d^2)$ ○ $\mathcal{O}(nd^2)$

The cost of a consistency check becomes $O(1)$, so d iterations, each $O(n)$ for n consistency checks.

(b) [4 pts] Branching Dudokus



Example A



Example B

A branching Dudoku is one in which multiple linear chains are joined together. Chains can be joined at a summation node, as in example A above, or at a variable, as in example B above. Recall that a cutset is a set of nodes that can be removed from a graph to ensure some desired property. Which of the following are true?

☐ Dudoku A is a binary CSP. No, it has a ternary constraint

🟥 Dudoku B is a binary CSP. Yes, all constraints are binary.

☐ Dudoku A is a tree-structured CSP. No, this concept applies only to binary CSPs

🟥 Dudoku B is a tree-structured CSP. Yes, no cycles in the constraint graph

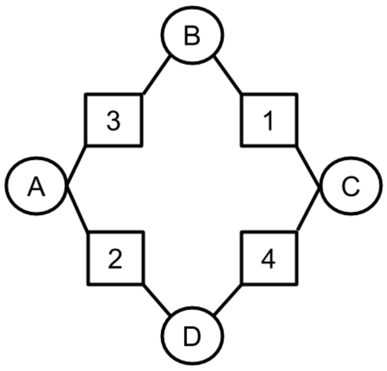
🟥 If variables B, D, and F are merged into a single megavariable, Dudoku A will be a tree-structured CSP. Yes, all remaining constraints will be binary

☐ If variables A and E are merged into a single megavariable, Dudoku B will be a tree-structured CSP. No, this creates a cycle

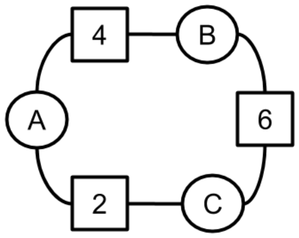
☐ The minimum cutset that turns Dudoku A into a set of disjoint linear Dudokus contains 3 variables. No, the smallest cutset has size 1 (any of B, D, F)

🟥 The minimum cutset that turns Dudoku B into a set of disjoint linear Dudokus contains 1 variable. Yes, any one of B, C, D, or F.

(c) Circular Dudokus



Example 1



Example 2

(i) [2 pts] The figure above shows two examples of circular Dudokus. If we apply cutset conditioning with a minimal cutset and a tree-CSP solver, what is the runtime for a circular Dudoku with n variables and domain size d?

○ $\mathcal{O}(d^{n-1})$ ○ $\mathcal{O}(n^2d^2)$ ○ $\mathcal{O}(nd)$ [✓ ● $\mathcal{O}(nd^3)$]

The cutset size is 1, any variable will do. Instantiate it d times and solve $O(nd^2)$ tree-CSP, so $O(nd^3)$.

(ii) [2 pts] Suppose that the variables in the circular Dudokus in the figure are assigned values such that all the constraints are satisfied. Assume also that the variable domains are integers in $[-\infty, \infty]$. Now consider what happens if we add 1 to the value of variable A in each Dudoku and then re-solve with A fixed at its new value. Which of the following are true?

☐ Dudoku 1 now has no solution. False—we can subtract 1 from B and D and add 1 to C.

[✓ ■ Dudoku 2 now has no solution.] True—we can subtract 1 from B and C but then the 6 constraint is violated.

(iii) [4 pts] What can you conclude about the number of solutions to a circular Dudoku with n variables?

For even n, if there are any solutions, there will be infinitely many solutions, since we can add any number to A and make alternating changes around the circle to re-satisfy the constraints. For odd n, there can be at most one solution, because adding any number x to A and then propagating around the circle causes x to be subtracted again from A.

Q6. [13 pts] In What Worlds?

(a) We wish to come up with hypotheses that entail the following sentences:

- S_1: $X_1 \wedge X_2 \implies Y$
- S_2: $\neg X_1 \vee X_2 \implies Y$

In this problem, we want to come up with a hypothesis H such that $H \models S_1 \wedge H \models S_2$.

(i) [3 pts] Assume we have the hypothesis H: $Y \iff X_1 \vee X_2$.

Does H entail S_1? [✓ Yes] No

Does H entail S_2? Yes [✓ No]

By looking at the truth table, you see that the for all worlds H is true, S_1 is true. However, H does not entail S_2. One example is $X_1 = false$, $X_2 = false$,$Y = true$.

(ii) [3 pts] Pretend that we have obtained a magical solver, $SAT(s)$ which takes in a sentence s and returns $true$ if s is satisfiable and $false$ otherwise. We wishes to use this solver to determine whether a hypothesis H' entails the two sentences S_1 and S_2. Mark all of the following expressions that correctly return $true$ if and only if $H' \models S_1 \wedge H' \models S_2$. If none of the expressions are correct, select "None of the above".

☐ $SAT(H' \wedge \neg(S_1 \wedge S_2))$ ☐ $SAT(\neg H' \vee (S_1 \wedge S_2))$

[✓ $\neg SAT(H' \wedge \neg(S_1 \wedge S_2))$] ☐ $\neg SAT(\neg H' \vee (S_1 \wedge S_2))$

☐ None of the above

Recall for H' to entail both S_1 and S_2, it must hold that $H' \wedge \neg(S_1 \wedge S_2)$ is **not** satisfiable.

Four people, Alex, Betty, Cathy, and Dan are going to a famliy gathering. They can bring dishes or games. They have the following predicates in their vocabulary:

- $Brought(p, i)$: Person p brought a dish or game i.
- $Cooked(p, d)$: Person p cooked dish d.
- $Played(p, g)$: Person p played game g.

(b) Select which first-order logic sentences are syntactically correct translations for the following English sentences. You must use the syntax shown in class (eg. ∀, ∃, ∧, ⇒, ⇔). **Please select all that apply**.

(i) [2 pts] At least one dish cooked by Alex was brought by Betty.

[✓ $\exists d\ Cooked(A, d) \wedge Brought(B, d)$]
☐ $[\exists d\ Cooked(A, d)] \wedge [\forall d' \wedge (d' = d)\ Brought(B, d')]$
☐ $\neg[\forall d\ Cooked(A, d) \vee Brought(B, d)]$
[✓ $\exists d_1, d_2\ Cooked(A, d_1) \wedge (d_2 = d_1)\ \wedge Brought(B, d_2)$]

(ii) [2 pts] At least one game played by Cathy is only played by people who brought dishes.

☐ $\neg[\forall g\ Played(C, g) \vee [\exists p\ Played(p, g) \implies \forall d\ Brought(p, d)]]$
☐ $\forall p\ \exists g\ Played(C, g) \wedge Played(p, g) \implies \exists d\ Brought(p, d)$
☐ $\exists g\ Played(C, g) \implies \forall p \exists d\ Played(p, g) \wedge Brought(p, d)$
[✓ $\exists g\ Played(C, g) \wedge [\forall p\ Played(p, g) \Rightarrow \exists d,\ Brought(p, d)]$]

(c) Assume we have the following sentence with variables A, B, C, and D, where each variable takes Boolean values:

$$S3 : (A \vee B \vee \neg C) \wedge (A \vee \neg B \vee D) \wedge (\neg B \vee \neg D)$$

(i) [2 pts] For the above sentence $S3$, state how many worlds make the sentence true. [Hint: you can do this and the next part without constructing a truth table!]

$\boxed{8}$ (1) Clauses disjoint (2) clauses with k literals remove 2^{n-k} models.

(ii) [1 pt] Does $S3 \models (A \wedge B \wedge D)$? ○ Yes [✓ No]

Q7. [12 pts] Proba-Potpourri

(a) [3 pts]

$$\begin{tikzcd} A \arrow[r] \arrow[d] & B \arrow[d] \\ C \arrow[r] & D \end{tikzcd}$$

Consider the Bayes' net shown above. In the box given below, write down an expression for $P(A|B,C,D)$ in terms of conditional probability distributions $P(B|A), P(C|A), P(A)$. If it is not possible to write down such an expression, mark the circle next to "Not possible".

$$\boxed{\frac{P(A)P(B|A)P(C|A)}{\sum_a (P(B|a)P(C|a)P(a))}}$$

○ Not possible.

$$P(A|B,C,D) = \frac{P(A,B,C,D)}{P(B,C,D)} = \frac{P(D|B,C)P(B|A)P(C|A)P(A)}{P(D|B,C)P(B,C)} = \frac{P(A)P(B|A)P(C|A)}{\sum_a (P(B|a)P(C|a)P(a))}$$

(b) [3 pts] Given the same Bayes' net as in part (a), write down an expression for $P(D|A)$ in terms of conditional probability distributions $P(D|B,C)$, $P(B|A)$ and $P(C|A)$. If it is not possible to write down such an expression, mark the circle next to "Not possible".

$$\boxed{\sum_b \sum_c (P(D|b,c)P(b|A)P(c|A))}$$

○ Not possible.

$$P(D|A) = \frac{P(A,D)}{P(A)} = \frac{\sum_b \sum_c P(A,B,C,D)}{P(A)} = \frac{\sum_b \sum_c (P(D|b,c)P(b|A)P(c|A)P(A))}{P(A)} = \sum_b \sum_c (P(D|b,c)P(b|A)P(c|A))$$

(c) [3 pts] Consider the Bayes' net shown below

$$\begin{tikzcd} A \arrow[r] \arrow[d] & B \\ C & D \arrow[u] \arrow[l] \end{tikzcd}$$

Write down an expression for $P(B|C)$ in terms of conditional probability distributions $P(B|A,D), P(C|A,D)$. If it is not possible to write such an expression, mark the circle next to "Not possible".

$$\boxed{}$$

● Not possible.

We also need access to $P(A)$ and $P(D)$.

$$P(B|C) = \frac{P(B,C)}{P(C)} = \frac{\sum_a \sum_d P(a,B,C,d)}{\sum_a \sum_d P(a,C,d)} = \frac{\sum_a \sum_d P(B|a,d)P(C|a,d)P(a)P(d)}{\sum_a \sum_d P(C|a,d)P(a)P(d)}$$

(d) [3 pts] Consider the Bayes' net shown below

$$\begin{tikzcd} C \arrow[r, "2"] & D \arrow[r, "5"] & E \\ B \arrow[u, "0"] \arrow[ur, "1"] \arrow[r, "3"] & A \arrow[ur, "4"] & \end{tikzcd}$$

You have access to all the conditional probability tables associated with the Bayes net above except $P(C|B)$, and you would like to determine $P(E|B)$ using this information. You are allowed to remove one edge from the given Bayes net. Select **all** possible single edges that you can remove in order to successfully compute $P(E|B)$ from $P(E|A, D), P(A|B), P(D|B, C)$ and $P(B)$. For examples, if you can remove the edge from B to C, or the edge from A to E, select 0 and 5. If there is no such single edge that you can remove, mark the circle next to "Not Possible".

☐ 0 ☐ 1 [✓ ■ 2] ☐ 3 ☐ 4 [✓ ■ 5] ○ Not Possible

The only missing conditional probability table is $P(B|C)$. In order to lose reliance on this conditional probability distribution, you can either directly remove the edge from C to D, or you can remove the edge from D to E, either of which which will make E conditionally independent of C given B.

SCRATCH PAPER – INTENTIONALLY BLANK – PLEASE DETACH ME

SCRATCH PAPER – INTENTIONALLY BLANK – PLEASE DETACH ME

SCRATCH PAPER – INTENTIONALLY BLANK – PLEASE DETACH ME