

Midterm I, Fall 2016

This test has **8** questions worth a total of **100** points, to be completed in 110 minutes. The exam is closed book, except that you are allowed to use a single two-sided hand written cheat sheet. No calculators or other electronic devices are permitted. Give your answers and show your work in the space provided.

Write the statement out below in the blank provided and sign. You may do this before the exam begins. Any plagiarism, no matter how minor, will result in an F.

“I have neither given nor received any assistance in the taking of this exam.”

Signature: _____

Name: _____ **Your EdX Login:** _____
SID: _____ **Name of person to left:** _____
Exam Room: _____ **Name of person to right:** _____
Primary TA: _____

- ○ indicates that only one circle should be filled in.
- □ indicates that more than one box may be filled in.
- Be sure to fill in the ○ and □ boxes completely and erase fully if you change your answer.
- There may be partial credit for incomplete answers. Write as much of the solution as you can, but bear in mind that we may deduct points if your answers are much more complicated than necessary.
- There are a lot of problems on this exam. Work through the ones with which you are comfortable first. **Do not get overly captivated by interesting problems or complex corner cases you’re not sure about.** Fun can come after the exam: There will be time after the exam is try them again.
- Not all information provided in a problem may be useful.
- Write the last four digits of your SID on each page in case pages get shuffled during scanning.

Problem	1	2	3	4	5	6	7	8
Points	11.5	9.5	14.5	12	8.5	16	19	9

Optional: Mark along the line to show your feelings Before exam: [:(_____ ☺]
 on the spectrum between :(and ☺ . After exam: [:(_____ ☺]

1. Munching (11.5 pts)

i) (7.5 pts) Suppose Pac-Man is given a map of a maze, which shows the following: it is an M -by- N grid with a single entrance, and there are W walls in various locations in the maze that block movement between empty spaces. Pac-Man also knows that there are K pellets in the maze, but the pellets are not on the map, i.e. **Pac-Man doesn't know their locations in advance**. Pac-Man wants to model this problem as a search problem. Pac-Man's goal is to plan a path, before entering the maze, that lets him eat the hidden pellets as quickly as possible. **Pac-Man can only tell he has eaten a pellet when he moves on top of one.**

Which of the following features should be included in a minimal correct state space representation? For each feature, if it should be included in the search state, calculate the size of this feature (the number of values it can take on). If it should **not** be included, **very briefly** justify why not.

Feature	Include?	Size (if included) or Justification (if not included)
Current location	<input checked="" type="radio"/> Yes / <input type="radio"/> No	Size: MN Justification: We need to keep track of where Pac-Man is.
Dimensions of maze	<input type="radio"/> Yes / <input checked="" type="radio"/> No	Justification: This is an input to the problem, and does not need to be included.
Locations of walls	<input type="radio"/> Yes / <input checked="" type="radio"/> No	Justification: This is an input to the problem, and does not need to be included.
Places explored	<input checked="" type="radio"/> Yes / <input type="radio"/> No	Size: 2^{MN} Justification: We need to visit every square in the maze in order to ensure that we eat all of the pellets, so we must track where we've explored.
Number of pellets found	<input type="radio"/> Yes / <input checked="" type="radio"/> No	Justification: We don't know where the pellets are, so we <i>can't</i> know when we've found a pellet when we're planning a path.
Time elapsed	<input type="radio"/> Yes / <input checked="" type="radio"/> No	Justification: This does not need to be in the state, but is instead part of the cost of transitioning between states.

ii) (1 pt) Which approach is best suited for solving the K hidden pellets problem? Completely fill in the circle next to one of the following:

- State Space Search** **CSP** **Minimax** **Expectimax** **MDP** **RL**

iii) (2 pts) Now assume that there are K_R red pellets, K_G green pellets, and K_B blue pellets, that Pac-Man **knows the locations and colors of all the pellets**, and that Pac-Man wants **the shortest path that lets him eat at least one pellet of each color**, i.e. one red, one green, and one blue. What is the total size of the search state space, assuming a minimal correct state space representation?

$2^3 MN$

Solution: The state needs to keep track of where Pac-Man is, and, for each color, whether or not Pac-Man has eaten any pellets of that color.

iv) (1 pt) Which approach is best suited for solving the one-pellet-of-each-color problem? Completely fill in the circle next to one of the following:

- State Space Search** **CSP** **Minimax** **Expectimax** **MDP** **RL**

2. Uninformed Search (9.5 pts)

Part A: Depth Limited Search (DLS) is a variant of depth first search where nodes at a given depth L are treated as if they have no successors. We call L the **depth limit**. Assume all edges have uniform weight.

i) (1 pt) Briefly describe a major advantage of depth limited search over a simple depth first tree search that explores the entire search tree.

Solution: Because we define a depth limit in DLS, we won't get stuck exploring an infinite path.

ii) (1 pt) Briefly describe a major disadvantage of depth limited search compared to a simple depth first tree search that explores the entire search tree.

Solution: Because we define a depth limit in DLS, if the solution is beyond the depth limit, we will not reach it.

iii) (2 pts) Is DLS optimal? Yes / No Is DLS complete? Yes / No

Solution: As noted in part (ii), DLS might not find the solution.

Part B: IDDFS. The iterative deepening depth first search (IDDFS) algorithm repeatedly applies depth limited search with increasing depth limits L : **First it performs DLS with $L=1$, then $L=2$, and so forth, until the goal is found.** IDDFS is neither BFS nor DFS, but rather an entirely new search algorithm. Suppose we are running IDDFS on a tree with branching factor B , and **the tree has a goal node located at depth D .** Assume all edges have uniform weight.

i) (2 pts) Is IDDFS optimal? Yes / No Is IDDFS complete? Yes / No

Solution: If the solution is at depth D , IDDFS is guaranteed to find it on the D th iteration of the algorithm (when it runs DLFS with depth D).

ii) (1 pt) In the worst case asymptotically, which algorithm will **use more memory** looking for the goal?

IDDFS / BFS

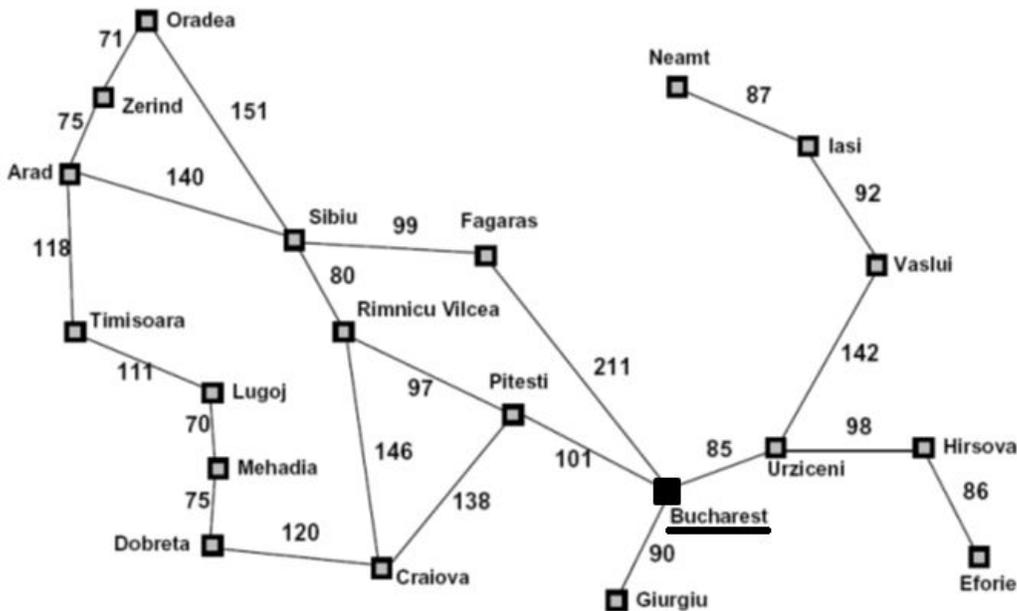
Solution: DLS and IDDFS both use asymptotically the same amount of memory as DFS (which only requires keeping track of the path to the current state), which is less than BFS (which requires keeping track of the entire fringe.)

iii) (2.5 pts) In project 2, you used DLS to implement the minimax algorithm to play Pac-Man. Suppose we are instead writing a minimax AI to play in a speed chess tournament. Unlike the Pac-Man project, there are many pieces that you or your opponent might move, and they may move in more complicated ways than Pac-Man. Furthermore, you are given only 60 seconds to select a move, otherwise you are given a random move. **Explain why** we'd prefer to implement minimax using IDDFS instead of DLS for this problem, and **explain how** would we need to modify IDDFS (as described above) so that it works for this problem.

Solution: With a pre-set depth L , DLS could either run out of time or not utilize all the time (L could be set too deep or too shallow). IDDFS will not have this problem. It would have to be modified such that the best found answer would be returned once time is up. (Possible answers for this include: keeping track of the time, keeping track of the best found answer at each layer)

3. Aliens in Romania (14.5 pts)

Here's our old friend, a road map of Romania. The **edges** represent **travel time along a road**. Our goal in this problem is to find the shortest path from some arbitrary city to Bucharest (underlined in black). You may assume that the heuristic at Bucharest is always 0. In this question, $L(X, Y)$ represents the travel time if we went in a straight line from city X to city Y, ignoring roads, teleporters, wind, birds, etc. You should make no other assumptions about the values of $L(X, Y)$.



Part A: Warm up.

i) (6 pts) For each of the heuristics below, completely fill in the circle next to “Yes” or “No” to indicate whether the heuristic is admissible and/or consistent for A* search. You should fill in 6 answers (one has been provided for you). **Reminder, $h(\text{Bucharest})$ is always zero.**

Heuristic $h(C) =$	[except $C = \text{Bucharest}$]	Admissible	Consistent
0		<input checked="" type="radio"/> Yes / <input type="radio"/> No	<input checked="" type="radio"/> Yes / <input type="radio"/> No
90		<input type="radio"/> Yes / <input checked="" type="radio"/> No	<input type="radio"/> Yes / <input checked="" type="radio"/> No
$L(C, \text{Bucharest})$		<input checked="" type="radio"/> Yes / <input type="radio"/> No	<input checked="" type="radio"/> Yes / <input type="radio"/> No
$\min(L(C, \text{Bucharest}), 146)$		<input checked="" type="radio"/> Yes / <input type="radio"/> No	<input type="radio"/> Yes / <input type="radio"/> No

ii) (2.5 pts) For what values of x is $h(C) = \max(L(C, \text{Bucharest}), x)$ guaranteed to be admissible? Your answer should be in the form of an expression on x (e.g. “ $x = 0$ OR $x > 102$ ”).

Solution: If $x > 85$, then $h(\text{Urziceni})$ will be greater than 85, which would be an overestimate of the true cost to reach Bucharest from Urziceni (which is at most 85). Thus, we want $x \leq 85$. (Notice that this is a strict inequality; $x = 85$ is fine.)

Part B: Teleporters

Aliens build a teleporter from Oradea to Vaslui. This means that we can now get from Oradea (top left) to Vaslui (near the top right) with zero cost.

i) (1 pt) Update the drawing on the **previous page** so that your graph now reflects the existence of the teleporter.

Solution: An arc between Oradea and Vaslui with weight 0.

ii) (5 pts) Completely fill in the circle next to “Yes” or “No” to indicate which of the following are guaranteed admissible for A* search, given the existence of the alien teleporter. **Reminder, $h(\text{Bucharest})$ is always zero, and $L(A, B)$ is the travel time between A and B assuming a straight line between A and B (“as a bird flies”).**

Heuristic $h(C) =$ [except $C = \text{Bucharest}$]	Admissible
77	<input checked="" type="radio"/> Yes / <input type="radio"/> No
$L(C, \text{Bucharest})$	<input type="radio"/> Yes / <input checked="" type="radio"/> No
$\max(L(C, \text{Bucharest}), 80)$	<input type="radio"/> Yes / <input checked="" type="radio"/> No
$\max(L(C, \text{Bucharest}), L(C, \text{Oradea}))$	<input type="radio"/> Yes / <input checked="" type="radio"/> No
$\min(L(C, \text{Bucharest}), L(C, \text{Oradea}))$	<input checked="" type="radio"/> Yes / <input type="radio"/> No

Solution: A key insight is that from C, we need to either go directly to Bucharest without using the teleporter, or go to Oradea before using the teleporter. However, neither, by itself is an admissible heuristic, so a *max* expression involving $L(C, \text{Bucharest})$ is necessarily inadmissible.

Survey Completion

Enter the secret code for survey completion: _____

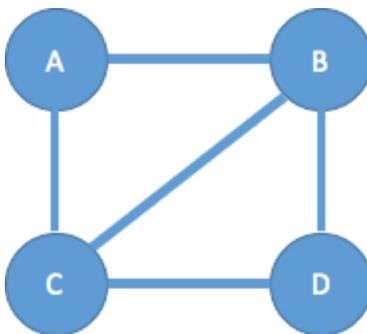
If you took the survey but don’t have the secret code, explain:

4. CSPs (12 pts)

In a combined 3rd-5th grade class, students can be 8, 9, 10, or 11 years old. We are trying to solve for the ages of Ann, Bob, Claire, and Doug. Consider the following constraints:

- No student is older in years than Claire (but may be the same age).
- Bob is two years older than Ann.
- Bob is younger in years than Doug.

The figure below shows these four students schematically (“A” for Ann, “B” for Bob, etc).



i) (2 pts) In the figure, draw the arcs that represent the binary constraints described in the problem.

Note: Two directed arcs in place of each undirected arc is also acceptable.

ii) (1 pt) Suppose we’re using the AC-3 algorithm for arc consistency. How many total arcs will be enqueued when the algorithm begins execution? Completely fill in the circle next to one of the following:

- 0 / 5 / 6 / 8 / 10 / 12

Solution comment: Remember that a relationship between two variables will result in two arcs (since arcs are directed.)

iii) Assuming all ages {8, 9, 10, or 11} are possible for each student before running arc consistency, manually run arc consistency on **only** the arc from A to B.

A. (2 pts) What values on A remain viable after this operation? Fill in all that apply.

- 8 / 9 / 10 / 11

B. (2 pts) What values on B remain viable after this operation? Fill in all that apply.

- 8 / 9 / 10 / 11

Solution comment: Remember that enforcing the arc $A \rightarrow B$ should never change the viable domain of B.

C. (1 pt) Assuming there were no arcs left in the list of arcs to be processed, which arc(s) would be added to the queue for processing after this operation?

Solution: $B \rightarrow A, C \rightarrow A$ (in either order). They should be directed arcs, since that’s critical to the algorithm.

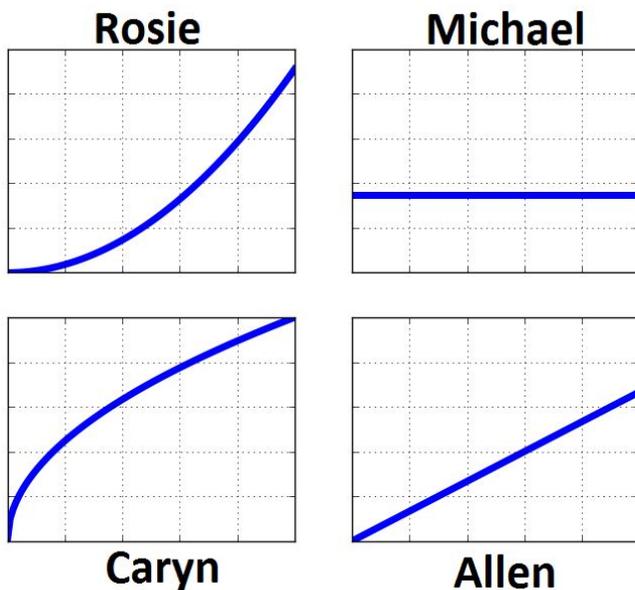
iv) (4 pts) Suppose we enforce arc consistency on all arcs. What ages remain in each person’s domain?

Ann: 8 Bob: 10 Claire: 11 Doug: 11

Solution comment: If we enforce the arcs $B \rightarrow A, B \rightarrow D, D \rightarrow B, A \rightarrow B, C \rightarrow D$, then we can quickly find the only satisfying solution.

5. Utilities (8.5 pts)

Part A: Interpreting Utility. (3.5 pts) Suppose Rosie, Michael, Caryn, and Allen are four people with the utility functions for marshmallows as described below (e.g. Michael doesn't care how many marshmallows he gets). The x-axis is the number of marshmallows, and the y-axis is the utility. Rank our four TAs from **least risk-seeking (at the bottom)** to most risk-seeking (at the top) using the four lines provided. **If two people have the same degree of risk-seeking, put them on the same line.** You may not need all four blanks (e.g. if two people are on the same line).



Rank in terms of risk-seeking behavior:

Most: _____ Rosie _____

_____ Michael, Allen _____

_____ Caryn _____

Least: _____

Solution explanation: This question asked the student to identify if a function was risk-seeking, risk-neutral, or risk-averse, based on the function's graph. Visually, we can see that Rosie's utility function is concave up (i.e. convex) and thus must be risk-seeking, Caryn's utility function is concave down and thus risk-averse, while both Michael's and Allen's functions are linear, and thus both have risk-neutral utility functions.

For visual intuition as to why concave up functions are risk-seeking: imagine connecting two points on the function's curve with a straight line segment. Notice that this line segment is above the curve. Now, for any point on this line segment, we can imagine a lottery between the two endpoints, such that the coordinates of this point are (the expectation of the lottery, the expected utility of the lottery), while the point below it on the curve is the utility of the expectation of the lottery.

Conversely, a concave down function is risk-averse, because the line segment (the expected utility of the lottery) is now below the curve (the utility of the expected value of the lottery), while a linear function is risk-neutral, because the line segment coincides with the curve.

Part B: Of Two Minds. (5 pts) Some utility functions are sometimes risk-seeking, and sometimes risk-averse. Show that $U(x) = x^3$ is one such function, by providing two lotteries: one for which it is risk-seeking, and one for which it is risk-averse. Provide justification by filling out the table below. Averse is just the opposite of “seeking”. Specifically, it means “having a strong dislike for something,” in this case, risk.

Example solution:

	Risk-seeking:	Risk-averse:
Lottery	$[1/2, 0; 1/2, 2]$	$[1/2, 0; 1/2, -2]$
Utility of lottery	$\frac{1}{2} * 0^3 + \frac{1}{2} * 2^3 = 4$	$\frac{1}{2} * 0^3 + \frac{1}{2} * (-2)^3 = -4$
Utility of expected value of lottery	$(\frac{1}{2} * 0 + \frac{1}{2} * 2)^3 = 1$	$(\frac{1}{2} * 0 + \frac{1}{2} * -2)^3 = -1$

Solution explanation: In general, for the given utility function, any lottery that can only take nonnegative values is risk-seeking, and any lottery that can only take on nonpositive values is risk-averse. (Lotteries that can take on both positive and negative values require a bit more effort to assess.)

The intention was for students to identify that, because the utility function is an odd function, simply negating the risk-seeking lottery would acquire a risk-averse function.

Common Mistakes:

- Not knowing how to identify a risk-seeking vs risk-averse function.
- More specifically: providing a fixed value as a lottery. (Note that we don’t consider fixed values as lotteries, but even if we did, such a lottery would never, under any utility function, be anything other than risk-neutral.)
- Not knowing what “utility of lottery” vs “utility of expected value of lottery” means; more students had trouble with “utility of expected value of lottery”.
- When calculating the expected value of the lottery: some students gave lotteries with a 50-50 chance between two rewards, correctly divided by two when calculating $E[L]$, but then divided by two again after calculating $U(E[L])$.

Advice:

- As noted above, having one lottery be the negative version of the other halves the needed amount of computation for this problem.
- In general, choosing lotteries with easy numbers makes computation easier as work, For example, 0^3 is easy to compute; 5^3 is slightly less so. Multiple students had 100 as an outcome, and then lost a 0 or had some other careless error when calculating 100^3 .
- Similarly, choosing simple lotteries (e.g. only two possible outcomes that are equally likely) simplifies things as well.

6. Games / Mini-Max: The Potato Game (16 pts)

Aldo and Becca are playing the game below, where the left value of each node is the number of potatoes that Aldo gets, and the right value of each node is the number of potatoes that Becca gets (i.e. Aldo: left, Becca: right). Unlike prior scenarios where having more potatoes results in more utility, Becca and Aldo will have a more complex view of what makes a “good” distribution of potatoes.

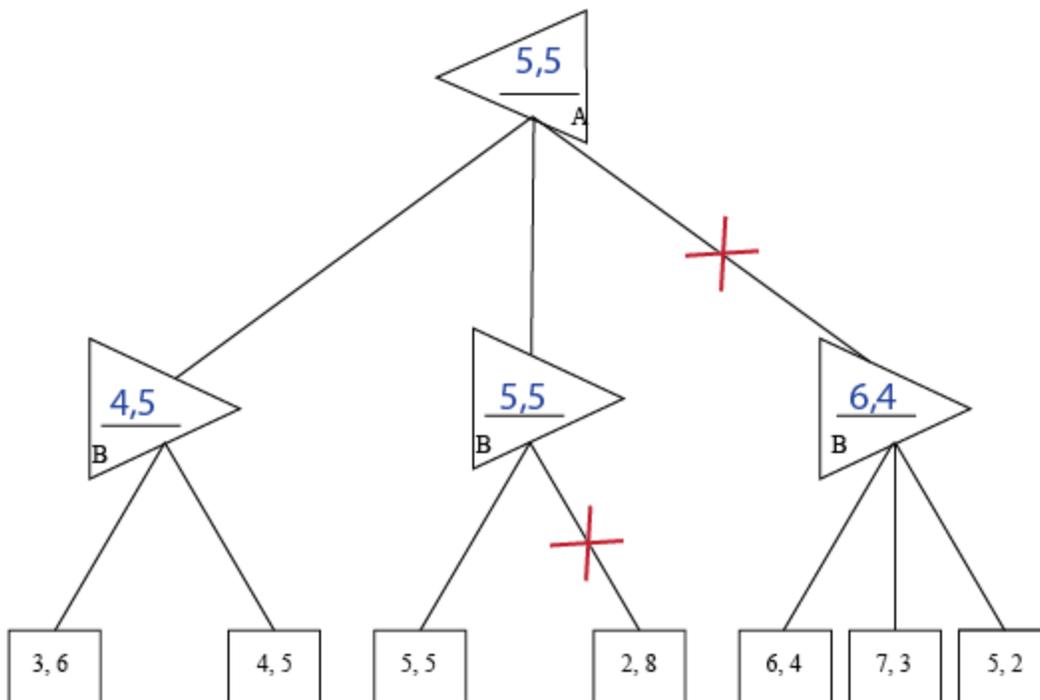
Becca will use the following thought process to decide which move to take:

- Among all choices where she gets at least as many potatoes as Aldo, she’ll pick the one that maximizes Aldo’s number of potatoes (very nice!).
- If there are no choices where she gets at least as many potatoes as Aldo, she’ll simply maximize her own potato count (ignoring Aldo’s value).

Aldo will do the same thing, but substituting “he” for “she”, “her” for “his”, and “Becca” for “Aldo”. The rules above effectively tell us how Becca (and Aldo) rank the utility of any set of choices.

i) (4 pts) Fill in the blanks below with the choice that each player would make at each stage. Assume that both Aldo and Becca are trying to maximize their own utility as described above.

ii) (3 pts) Assume that Aldo and Becca know that **the sum at any leaf node is no more than 10**. Cross out the edges to any nodes that can be pruned (if none can be pruned then write “no pruning possible” below the tree).

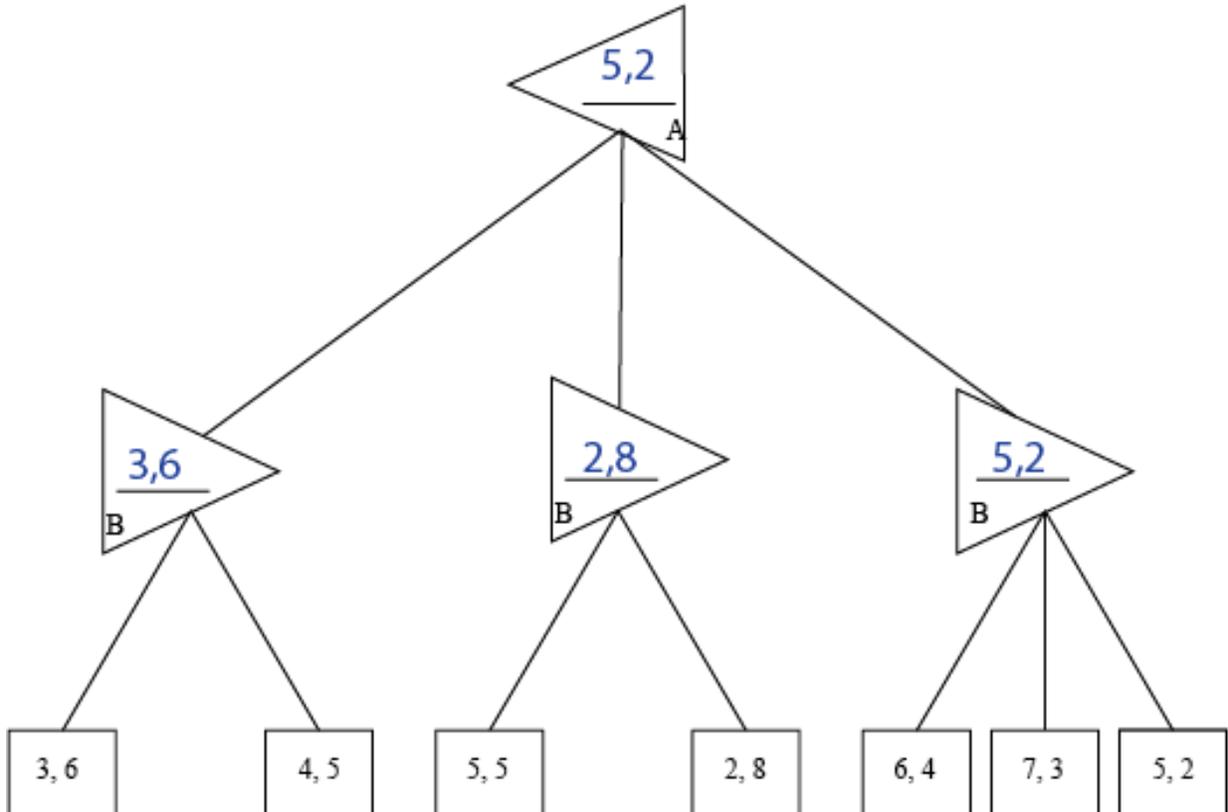


iii) (3 pts) Describe a general rule for pruning edges, assuming that the sum at any leaf node is no more than 10.

Solution: Prune all remaining edges after seeing a node with (5, 5).

Solution explanation: Since the sum at any leaf node is no more than 10, we know (5, 5) is the best possible (utility-maximizing) distribution of potatoes for both players, so we do not need to consider any other nodes. Note that based on the utility calculation rules, (5, 5) is better than other equal-potato distributions, like (4, 4) or (3, 3).

iv) (4 pts) Suppose that **Becca attempts to minimize Aldo's utility instead of maximizing her own utility, and that Aldo knows this, and tries to maximize his own utility -- taking into account Becca's strategy**. Repeat part ii. There is no need to cross out edges that are pruned.



v) (2 pts) Suppose that **Becca chooses uniformly randomly and Aldo knows this and tries to maximize his own utility -- taking into account Becca's strategy**. Which move will Aldo make to maximize his expected utility, assuming he treats Becca as a chance node: left, middle, or right? If there is not enough information, pick “Not Enough Information”.

- Left
 Middle
 Right
 Not Enough Information

Solution: We do not know what Aldo's specific utility function is, so we lack information to calculate his average utility between two outcomes, which we need to compute the assigned value for a chance node.

7. The Spice Must Flow (19 pts)

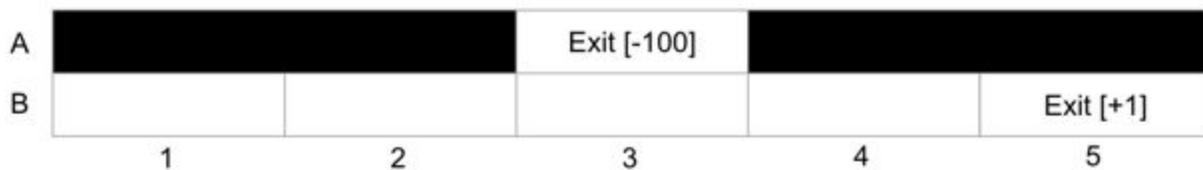
i) (3 pts) Suppose we have a robot that has 5 possible actions: $\{\uparrow, \downarrow, \leftarrow, \rightarrow, \text{Exit}\}$. If the robot picks a direction, it has an **80%** chance of going in that direction, a **10%** chance of going 90 degrees counterclockwise of the direction it picked (e.g. picked \rightarrow , but actually goes \uparrow), and a **10%** chance of going 90 degrees clockwise of the direction it picks.

If the robot hits a wall, it does not move this timestep, but time still elapses. For example, if the robot picks \downarrow in state B1 and is ‘successful’ (80% chance), it will hit the wall and not move this timestep. As another example, if it picks \rightarrow in state B1, but gets the 10% chance of going down, it will hit the wall and not move this timestep

In states B1, B2, B3, and B4, the set of possible actions is $\{\uparrow, \downarrow, \leftarrow, \rightarrow\}$. In states A3 and B5, the only possible action is $\{\text{Exit}\}$. Exit is always successful. The blackened areas (A1, A2, A4, and A5) represent walls.

All transition rewards are 0, except $R(A3, \text{Exit}) = -100$, $R(B5, \text{Exit}) = 1$.

In the boxes below, draw an optimal policy for states B1, B2, B3, and B4, if there is no discounting, i.e. $\gamma = 1$. Use the symbols $\{\uparrow, \downarrow, \leftarrow, \rightarrow\}$.



Solution: Any policy that went down at B3 and that never went left would work (i.e. guarantee never landing in the A3 exit and eventually reach the B5 exit).

ii) (2 pts) Give the expected utility for states B1, B2, B3, and B4 under the policy you gave above.

$V^*(B1)$: 1 $V^*(B2)$: 1 $V^*(B3)$: 1 $V^*(B4)$: 1

Solution: There is no discounting, so we don’t have to worry about how long it takes to get to the exit; as long as we can guarantee that we will *eventually* reach the good exit (as we can with the above optimal policy), the optimal value of every state is going to be 1.

iii) (2.5 pts) What is $V^*(B4)$, the expected utility of state B4, if we have a discount $0 < \gamma < 1$ and use the optimal policy? Give your answer in terms of γ .

$V^*(B4)$: $\frac{0.8\gamma}{1 - 0.2\gamma}$

Solution: Any optimal policy requires going right from B4. We can calculate this in one of two ways:

Direct Approach: The expected reward from B4 is equal to γ^N where N is the number of time steps until we get to B5; following the optimal policy, we have N following a geometric distribution with $p = 0.8$. Thus, we can calculate:

$$V^*(B4) = E[\gamma^{Geom(0.8)}] = \sum_{i=1}^{\infty} \gamma^i * P(Geom(0.8) = i) = 0.8\gamma \sum_{i=0}^{\infty} (0.2\gamma)^i = \frac{0.8\gamma}{1-0.2\gamma}$$

Recurrence Approach: If we try to go right and move successfully, we will get a reward of $\gamma * 1$; otherwise, we are at the same location (and have expected reward $\gamma * V^*(B4)$). Thus, we can use the recurrence to get $V^*(B4) = 0.8 * \gamma + 0.2 * \gamma * V^*(B4)$, which we can solve to get $V^*(B4) = \frac{0.8\gamma}{1-0.2\gamma}$.

Part B: Spice. Now suppose we add a sixth special action **SPICE**. This action has a special property that it always takes the robot **back to its current state**, with a reward of 0. This action might seem useless, except for one thing - the robot will always get its top preference on its next action (instead of being subject to noise).

For example, suppose that the robot is in state **s1** and picks action **a** which has a chance **p2** of moving to **s2**, and a chance **p3** of moving to **s3**. If the robot takes the **SPICE** action at timestep t , and **a** at timestep $t + 1$, the robot will not end up randomly in **s2** or **s3**, but will instead go to its **preference** of **s2** or **s3**. It will still collect $R(s1, a, s')$, which is unchanged. Preferring **s2** or **s3** does not take an additional time step, e.g. if the robot is in B1 at timestep 0, chooses SPICE, then chooses \downarrow but prefers the outcome B2, the robot arrives in B2 at timestep 2.

The powers granted by **SPICE** last one timestep. **SPICE** may be used any number of times.

i) (2 pts) Give an optimal policy for B1, B2, B3, and B4, assuming no discounting, i.e. $\gamma = 1$. **On the left of each slash**, write the optimal action if the previous action was not **SPICE**, and on the **right side of each slash**, write the optimal action if the previous action was **SPICE**, **assuming the robot always “prefers” the outcome that points in the same direction as its action (e.g. if it picks \rightarrow , it prefers going right)**. In total, you should write 8 actions. Use the symbols $\{\uparrow, \downarrow, \leftarrow, \rightarrow, S\}$, where “S” represents indicate the spice action.

A			Exit [-100]		
B	/	/	/	/	Exit [+1]
	1	2	3	4	5

Solution: Since there was no discounting, there were a number of viable solutions. There were three particularly common solutions.

- 44% of students said to go right for every state except B3 without SPICE, where they said to use SPICE instead.
- 21% of students said to go right for every state except B3 without SPICE, where they said to go down instead.
- 19% of students simply said to always use SPICE to deterministically move right, regardless of location.

There are a few actions that would specifically *not* work:

- Going left, up, or right from B3 without SPICE all has a chance of accidentally leading to the bad exit.
- Whether or not it works depends on the other part of the policy, but going “left” was often incorrect because it would likely cause the robot to be caught in a loop and never be able to get to the exit.
- If SPICE is active, and the policy dictates for the robot to use SPICE again, then the robot would definitely be stuck in this state.

In addition, some students assumed that the robot started at B1 (or somewhere else), and then only wrote policies corresponding to a specific path. However, a policy should include an action for *all* states, even ones that might not be reachable from a particular starting state (not that there is a starting state in this problem at all).

ii) (2.5 pts) Assuming we have a discount $0 < \gamma < 1$, what is $V^*(B4)$ if the previous action was not **SPICE**? Give your answer in terms of γ . It is OK to leave your answer in terms of a max operation.

$$V^*(B4): \text{_____} \max\left(\frac{0.8\gamma}{1-0.2\gamma}, \gamma^2\right) \text{_____}$$

Solution: There are two potential policies: the robot can try right repeatedly until it reaches the exit, or it can take SPICE and move right. We thus want to compare the expected reward for each of these policies, and take the max over the two.

The value of the former is simply the same as the answer to part A.i: $\frac{0.8\gamma}{1-0.2\gamma}$.

The value of the latter is equal to γ^2 (since we will deterministically claim a reward of 1 in three time steps).

OFFICIAL RELAXATION SPACE. Draw or write anything here:

Part C: Bellman Equations. (You can do this problem even if you didn't do B, but it'll be harder)

In class, we derived the Bellman Equations for an MDP given below.

$$Q^*(s, a) = \sum_{s' \in S_{(s,a)}} T(s, a, s') [R(s, a, s') + \gamma V^*(s')]$$

$$V^*(s) = \max_{a \in A_s} Q^*(s, a)$$

You'll notice these look slightly different than the version on your cheat sheet, but these equations represent exactly the same idea. The only difference is that to improve the clarity of these equations, we've specified the sets over which we are summing and maximizing. Specifically, $S_{(s,a)}$ is the set of states that might result from the action (s, a) , and A_s is the set of actions that one can take in state s . **For this problem, assume that A_s does not include the special SPICE action.**

i) (3.5 pts) Derive $Q^*(s, \text{SPICE})$. Assume that the previous action was not **SPICE**. Your answer should have two max operators in it. Hint: Consider drawing the diagram we used to derive the Bellman Equation.

Solution: We need to choose the next action to take, and SPICE allows us to also choose the next state after that action; this leads to the two maxes that the question suggests. The rest of the recurrence is like the usual Bellman equation, except that we are getting $R(s, a, s')$ at a one time-step delay, and s' is reached two time steps away:

$$\max_{a \in A} \max_{s' \in S_{(s,a)}} \gamma [R(s, a, s') + \gamma V(s')]$$

Common Mistakes:

- Some students included transition probabilities, or took the sum over multiple successor states; this goes against the point of consuming SPICE.
- Some students messed up on applying the discount factors correctly, most commonly forgetting the outermost γ .

ii) (3.5 pts) Derive $V^*(s)$, the value of a state in this MDP. You should account for the the fact that the **SPICE** action is available to the robot. Assume that the previous action was not **SPICE**.

Solution: The optimal value of s is acquired either by taking a non-SPICE action (where the recurrence is the same as the regular Bellman equation), or by taking SPICE. We can thus calculate this by taking the max over these two possibilities:

$$\max\{Q^*(s, \text{SPICE}), \max_{a \in A_s} Q^*(s, a)\}$$

We can also merge these two cases into a single max expression:

$$\max_{a \in A_s \cup \{\text{SPICE}\}} \{Q^*(s, a)\}$$

8. The Reinforcement of Brotherly Love (9 pts)

Consider two brother ghosts playing a game in an M-by-N grid world, Ghost Y is the Younger brother of Ghost O. Each ghost takes 100 steps, trying to pet cats which randomly appear in the maze. The reward for the game is equal to the number of cats petted. Each ghost has a policy for how to run through the maze. Ghost O being the older brother has a policy that has a **higher expected cumulative reward** (i.e. $V_Y(s) \leq V_O(s) \quad \forall s \in S$). However, Ghost O is **not necessarily optimal** (i.e. $V_O(s) \leq V^*(s) \quad \forall s \in S$). Ghost Y wants to learn from Ghost O instead of starting from scratch.

Part A: Q-Learning

We will now explore how Ghost Y can achieve this, while performing **Q-Learning**. Ghost Y starts out with a Q function **initialized randomly** and wants to catch up to Ghost O's performance. Denote the Q functions for each Ghost as Q_Y, Q_O . In order for Ghost Y to converge to the optimal policy, he must properly balance **exploring** and **exploiting**. Consider the following exploration strategy: At state s with probability $1-\epsilon$ choose $\max_a Q_Y(s, a)$ otherwise choose $\max_a Q_O(s, a)$.

(2 pts) Is this guaranteed to converge to the optimal value function V^* ? Briefly justify.

Yes / No _____

Solution: Ghost Y is not guaranteed to visit all states and actions infinite times each, which prevents it from observing a potentially better action to apply. In particular, Ghost Y will only ever consider the initial $\max_a Q_Y(s, a)$ action or the fixed $\max_a Q_O(s, a)$, which may not include the true optimal action.

Part B: Model-Based RL

Ghost Y decides model free learning is scary, and decides to try to learn a model instead. Ghost Y will watch Ghost O try to catch Pac-Man and estimate the **transition** probabilities and **rewards** from Ghost O's demonstrations (i.e. learn a model for $T(s,a,s')$ and $R(s,a,s')$).

i) (2 pts) For Ghost Y to learn the correct and exact values of $R(s,a,s')$ for all state/action/next_state transitions, what must be true about the episodes he watches Ghost O perform?

Solution: Ghost O must visit each state/action/next_state transition at least once. (Alternatively: ghost O must try each state/action pair enough times to see all possible successor states.)

Common Mistakes:

- Seeing the transition (s,a,s') once is enough to know exactly what $R(s,a,s')$ is. Some students incorrectly put that Ghost O needs infinite transitions, but that is only the case for learning $T(s,a,s')$ (as below.)
- Some students only tried every action from every state once. This didn't necessarily see every possible successor state.

ii) (2 pts) For Ghost Y to learn the correct and exact values of $T(s,a,s')$ for all state/action/next_state transitions, what must be true about the episodes he watches Ghost O perform?

Solution: Ghost O must visit each state/action/next_state transition infinite times each. (Alternatively: Ghost O must try each action from each state infinite times.)

Common Mistakes:

- Some answers were not specific enough, and only said that ghost O visits each transition "many" or "enough" times.
- Some answers mentioned that there needed to be an infinite number of episodes, and some also noted that the episodes must visit every transition at least once, but did not clearly specify that each transition is visited infinite times.
- Some answers said that each state needed to be visited infinite times, but did not talk about trying each *action*.
- Conversely, some answers said to try each action infinite times. This is different from trying each action *from each state* infinite times.
- A few answers said that Ghost Y needs episodes that include every state/action pair, and needed to watch these episodes infinite times. This does not guarantee that the finite number of episodes are representative of the true probability distributions.

iii) (2 pts) Assuming $T(s,a,s')$ and $R(s,a,s')$ are learned **exactly**, what algorithm/s below could be helpful for Ghost Y to decide which actions to take from each state, based **only** on information in his learned model?

Value Iteration Policy Iteration Q-Learning TD-Learning

Solution: Value Iteration and Policy Iteration. These both can generate a better policy and do not require further information to be gathered.

iv) (1 pt) Let π be any of the policies learned in part iii. Does following π result in maximized expected utility for every state, i.e. is $V^\pi(s) = V^*(s)$ for all states? [Consider all policies]

- Yes No Not enough information

