

- You have approximately 110 minutes.
- The exam is closed book, no calculator, and closed notes, other than one double-sided "crib sheets" that you may reference.
- For multiple choice questions,
 - means mark **all options** that apply
 - means mark a **single choice**

First name	
Last name	
SID	
Exam Room	
Name and SID of person to the right	
Name and SID of person to the left	
Discussion TAs (or None)	

Honor code: “As a member of the UC Berkeley community, I act with honesty, integrity, and respect for others.”

By signing below, I affirm that all work on this exam is my own work, and honestly reflects my own understanding of the course material. I have not referenced any outside materials (other than one double-sided crib sheet), nor collaborated with any other human being on this exam. I understand that if the exam proctor catches me cheating on the exam, that I may face the penalty of an automatic "F" grade in this class and a referral to the Center for Student Conduct.

Signature: _____

Point Distribution

Q1. Potpourri	21
Q2. CSP	8
Q3. Game	18
Q4. Bayes Nets	12
Q5. Slides and Ladders	8
Q6. State Spaces and Uninformed Search	13
Total	80

THIS PAGE IS INTENTIONALLY LEFT BLANK

Q1. [21 pts] Potpourri

(a) True or False

- (i) [1 pt] An environment is *deterministic* if the next state of the environment is completely determined by the current state and the action executed by the agents.
 True False
- (ii) [1 pt] A reflex agent selects actions based on only the current percept and ignores the rest of the percept history.
 True False
- (iii) [1 pt] Depth-first Tree Search is optimal for finite state spaces.
 True False
- (iv) [1 pt] There exists a problem with a finite solution in which a constant heuristic $h(n) = c$ for some strictly positive c is a consistent heuristic.
 True False
 False, the heuristic at the goal node has to be 0 for it to be admissible, and since consistency implies admissibility, not admissible implies not consistent.
- (v) [1 pt] In Uniform Cost Tree Search, when a goal state is added to the frontier, it must be the goal state that the optimal solution reaches.
 True False
- (vi) [1 pt] In a general CSP with n variables, each taking d possible values, the asymptotic runtime of naive backtracking search is guaranteed to be better than the asymptotic runtime of depth first graph search.
 True False
- (vii) [1 pt] The worst case time complexity of AC-3 (arc consistency) algorithm is $O(e^2d^3)$ where e is the number of arcs (directed edges in constraint graph) and d is the size of the largest domain.
 True False
- (viii) [1 pt] The LCV heuristic stands for Least Constraining Variable, i.e., the variable with the largest number of remaining values in its domain.
 True False
- (ix) [1 pt] Backtracking search is a method of informed search.
 True False
- (x) [1 pt] Using the same fixed variable and value ordering, backtracking search with forward checking vs. AC3 will always return the same solution to a CSP.
 True False
- (xi) [1 pt] When combining MRV and LCV heuristics together, we can always improve the asymptotic runtime of naive backtracking search with random ordering.
 True False
 2 is correct since using the same ordering guarantees that the leftmost solution in the search tree is always chosen. Filtering only limits the amount of subtree searching, it does not affect the order that solutions are visited in.
- (xii) [1 pt] If random variables A and B are independent, then we have $P(A, B) = P(A)P(B)$.
 True False

- (b) (i) [2 pts] Select all of the following search algorithms which are complete.
You may assume that all heuristics are finite and bounded, the branching factor is always finite, and all edge costs are finite values $\geq \epsilon > 0$ for some constant ϵ . But do not make any assumptions on the state space size (might be infinite).

Recall that an algorithm is complete if it is guaranteed to find a solution if one exists.

- DFS Graph Search
- BFS Graph Search
- Greedy Graph Search with a consistent heuristic
- A* Tree Search with a bad heuristic
- None of the above

- (ii) [2 pts] If a heuristic h is consistent for search problem A , and search problem B is a relaxed version of problem A , then h is guaranteed to be consistent for problem B .

Note: A relaxed problem is one that contains fewer restrictions on the action space. Formally, the available actions at every state for the relaxed version B is a superset of those in problem A ($\mathcal{A}_B(s) \supseteq \mathcal{A}_A(s), \forall s \in S$). All the other elements of the search problem, such as state space and action costs (for shared actions), are the same in A and B .

- True, h is an underestimate for every cost in A , and the costs in B are at least as much as in A , so h must be consistent for B .
- True for another reason.
- False, but admissibility is guaranteed in problem B .
- False, and admissibility is also not guaranteed in problem B .

If heuristic h is consistent for B , and B is a relaxed version of A , then h is consistent for A , because every action in B costs at least as much as it does in A .

- (iii) [2 pts] Consider a search problem where the start state is x_1 and goal state is x_n . For a heuristic h , if we have $h(x_1) > h(x_2) > \dots > h(x_n)$ along every path $x_1 - x_2 - \dots - x_n$ from x_1 to x_n , and $h(x_n) = 0$, then h is consistent.

- True, since the heuristic is strictly decreasing along each path to the goal.
- True, for another reason.
- False, but if we are additionally given that h is admissible, then we can conclude that h is consistent.
- False, and even if we are additionally given that h is admissible, we still cannot conclude that h is consistent.

False, the heuristic must underestimate the cost of each action. Monotonicity is a consequence of every heuristic satisfying the triangle inequality.

- (c) [3 pts] Consider a two player game of rock paper scissors where each player wants to win against the other, and neither player has any information on what move the other player might choose. Select all the statements that are true.

- This problem can be modeled as a single player search problem.
- This problem can be modeled as an Expectimax search.
- This problem can be modeled as a Minimax search.
- None of the above

For this question, we will give students full score as long as they don't choose A and D.

Q2. [8 pts] CSP

Pacman wants to try out some food around Berkeley. There are three Restaurants $\{A, B, C\}$ and two Drinks $\{D, E\}$ that he wants to assign to 4 time slots $\{1, 2, 3, 4\}$. Each time slot might be assigned more than one restaurant or drink, if not otherwise specified in the constraints. Pacman has the following constraints that he wants to follow:

1. Any two restaurants *cannot* be taken at the same time slot.
2. Any two drinks *cannot* be taken at the same time slot.
3. Restaurant A doesn't open for time slot 1.
4. The time slot number that C is assigned to must be greater than the time slot numbers that A and B are assigned to.
5. Drink D and restaurant A should be assigned to the same time slot.

(a) (i) [1 pt] Which constraint(s) are *unary*?

1 2 3 4 5

(ii) [4 pts] Pacman wants to find a valid arrangement by using backtracking search. Suppose all variables are unassigned initially. Pacman first assigns 2 to A and enforces arc consistency. What values remain in the domain of each variable? (A is omitted since we already assigned a value to A)

(1) B : 1 2 3 4

(2) C : 1 2 3 4

(3) D : 1 2 3 4

(4) E : 1 2 3 4

(b) (i) [2 pts] Now Pacman wants to solve this problem using a different approach: local search. Suppose we start with the following assignment $\{A: 4, B: 1, C: 3, D: 2, E: 2\}$. Which of the variables are conflicted in this assignment?

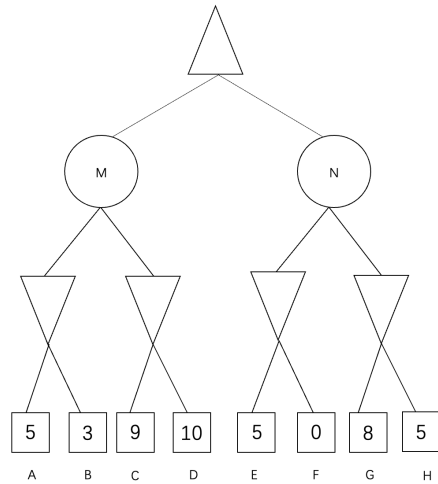
A B C D E

(ii) [1 pt] Assume that we decide to change the value of variable A . What new value should be assigned to A , based on the min-conflict heuristic?

1 2 3 4

Q3. [18 pts] Game

- (a) Consider the following game tree where all leaf nodes are between 0 and 10 (inclusive) and both expectation nodes choose the left branch with chance 0.8.



- (i) [3 pts] Determine the values of M , N , and the root node.

$$M = \boxed{4.2} \quad N = \boxed{1} \quad \text{root} = \boxed{4.2}$$

$$M = 3 * 0.8 + 9 * 0.2 = 4.2$$

$$N = 0 * 0.8 + 5 * 0.2 = 1$$

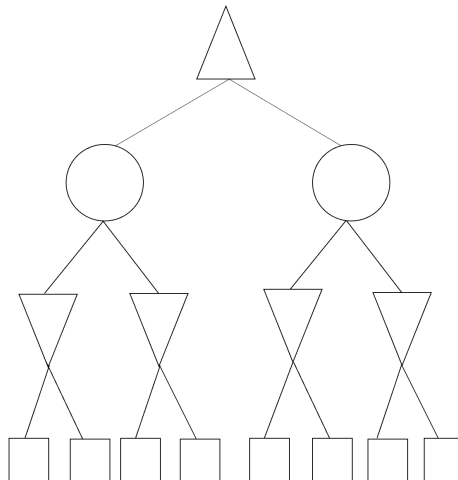
$$\text{root} = \max(4.2, 1) = 4.2$$

- (ii) [2 pts] Given the above expectimax problem, which node(s) can be pruned in alpha-beta pruning?

A B C D E F G H None

We know for sure that the left expectimax is 4.2. Given $E = 5$, $F = 0$, we know that the value of the left minimax is 0, so no matter what G and H is, the maximum value for N is 2, which is smaller than M .

- (iii) [2 pts] Now consider the general case where leaf values can take on any real value between 0 and 10 (inclusive). Do not prune on equality.



What is the maximum number of nodes that can be pruned?

1 2 3 4 5 6 7 8 None

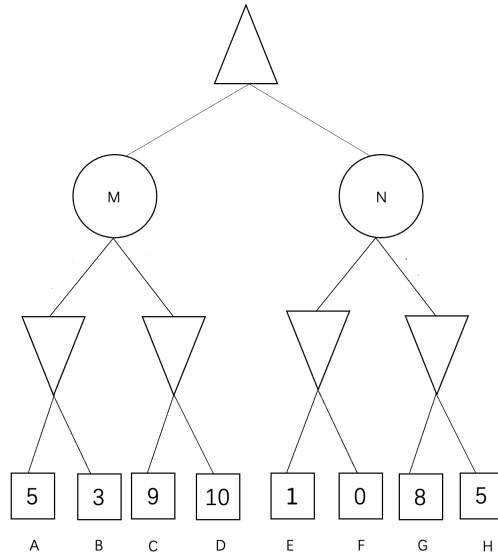
In this question, we shouldn't prune on equality. Thus, all the four children on the left should not be pruned in order to determine the value of the expectimax on the left. Then, if the fifth leaf is way too small, we don't need to look

at the remaining nodes. One example would be 10, 10, 10, 10, 0, x, x, x. From this example, we know that the left expectimax is 10, and the maximum of the right expectimax would be 2.

Also accepted the answer 4 since we did not clarify leaf nodes (since the minimizer should be pruned).

- (iv) [2 pts] Now, let p be the probability that each expectation node chooses the left branch, and assume p cannot be 0 or 1. For the following game tree shown below (note this is different from the tree in part (i)), determine the bounds on p such that the maximum number of leaf nodes can be pruned.

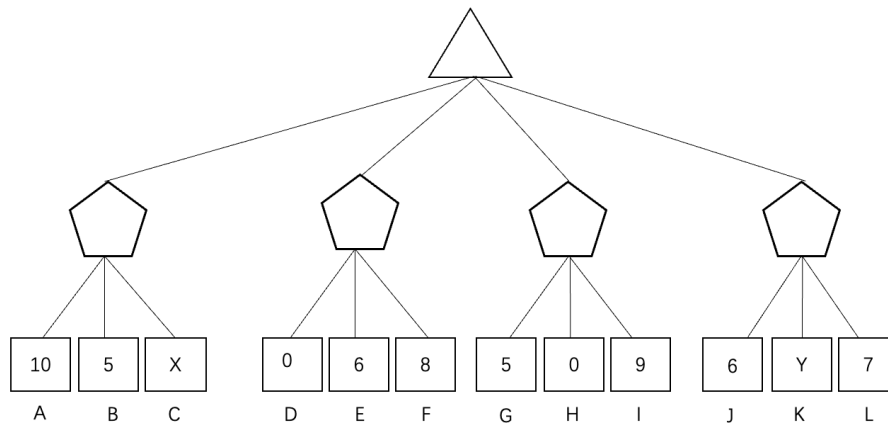
Recall that probabilities must be between 0 and 1.



$$\boxed{1/3} < p < \boxed{1}$$

We need $3p + 9(1 - p) \geq 1p + 10(1 - p)$.

- (b) Now consider a new problem with the game tree shown below, where pentagon nodes choose the median value of all its children. Once again, assume that all leaf values are between 0 and 10 (inclusive).



$$\boxed{6}$$

- (i) [1 pt] For this part only, let $X = 1$ and $Y = 2$. What is the value of the root node?

The first medianer is 5, the second medianer is 6, the third medianer is 5, the fourth medianer is 6. Thus, the root value is 6.

For all the remaining subparts, assume that X and Y can be any integer between 0 and 10 (inclusive). In all pruning subparts, we do not prune on equality.

- (ii) [2 pts] Select all the possible value(s) that the root node could be.

- 1
 2
 3
 4
 5
 6
 7
 8
 9
 10

We know that the second median value is 6 and the third median value is 5. The values for the left median are [5, 10](5 when $X \leq 5$, X when X is [6, 10]). The value for the right most median are 6 and 7(6 when $Y \leq 6$, 7 when $Y \geq 7$).

(iii) [2 pts] Select all of the leaf node(s) that are guaranteed to be pruned for any values of X and Y .

- A B C D E F G H I J K L
 None

In the medianmax, it is only possible to prune the third child of a medianer because we need the first two children to specify the range. From DEF we know that the minimum value that the root can take on is 6. G and H makes sure that the value for the third medianer is $[0, 5]$, so it is guaranteed that the third medianer will not be chosen.

(iv) [4 pts] Determine all possible values for X and Y such that the maximum number of leaf nodes can be pruned.

(In other words, the value $X = x$ should be selected if there exists a value $y \in [0, 10]$ such that $X = x$ and $Y = y$ causes the maximum set of nodes to be pruned in the above tree. And vice versa for Y .)

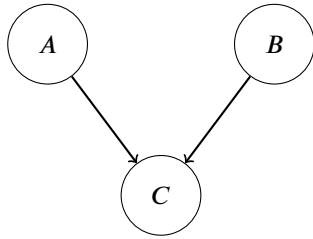
(1) X : 1 2 3 4 5 6 7 8 9 10

(2) Y : 1 2 3 4 5 6 7 8 9 10

The maximum number of leaf nodes that can be pruned is 3, which are F , I , and L . To prune these three nodes, the value for the first medianer should be larger than 6 (note again: we don't prune on equality, which means that if the first medianer and the second medianer are both 6, we don't prune any node). To prune L , the value for Y can be less than or equal to 6.

Q4. [12 pts] Bayes Nets

Consider the following Bayes net of three binary random variables A , B , and C . The conditional probabilities tables associated with this Bayes Net, $P(A)$, $P(B)$, and $P(C|A, B)$, are provided below.



$P(A)$		$P(B)$	
$A = a$	0.1	$B = b$	0.1
$A = \neg a$	0.9	$B = \neg b$	0.9

A	B	C	$P(C A, B)$
a	b	c	0.8
a	b	$\neg c$	0.2
a	$\neg b$	c	0.3
a	$\neg b$	$\neg c$	0.7
$\neg a$	b	c	0.3
$\neg a$	b	$\neg c$	0.7
$\neg a$	$\neg b$	c	0
$\neg a$	$\neg b$	$\neg c$	1

(a) For each of the following sub-parts, calculate the given probability. Please box your final answer. You may leave your answer as an expression.

(i) [1 pt] $P(A = \neg a, B = b, C = c)$.

$$0.9 * 0.1 * 0.3 = 0.027$$

(ii) [2 pts] $P(C = c|B = b)$.

$$P(A = a) * P(C|A = a, B = b) + P(A = \neg a) * P(C|A = \neg a, B = b) = 0.1 * 0.8 + 0.9 * 0.3 = 0.35$$

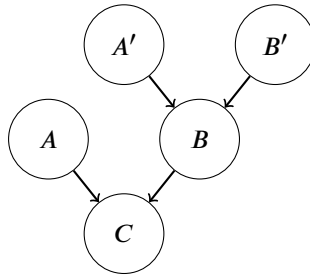
(iii) [1 pt] $P(C = c|B = \neg b)$.

$$P(A = a) * P(C|A = a, B = \neg b) + P(A = \neg a) * P(C|A = \neg a, B = \neg b) = 0.1 * 0.3 + 0.9 * 0 = 0.03$$

(iv) [2 pts] $P(B = b|C = c)$.

$$0.35 * 0.1 / (0.8 * 0.01 + 0.3 * 0.18) = 35/62$$

(b) [2 pts] Consider the following Bayes net of binary random variables A' , B' , A , B , and C . $P(A') = P(B') = P(A)$, and $P(B|A', B') = P(C|A, B)$.



Assume $P(C = c|B = b) = x$ and $P(C = c|B = \neg b) = y$.

Represent the value of $P(B = b|C = c, B' = b)$ in terms of x , and y . Box the final answer expression.

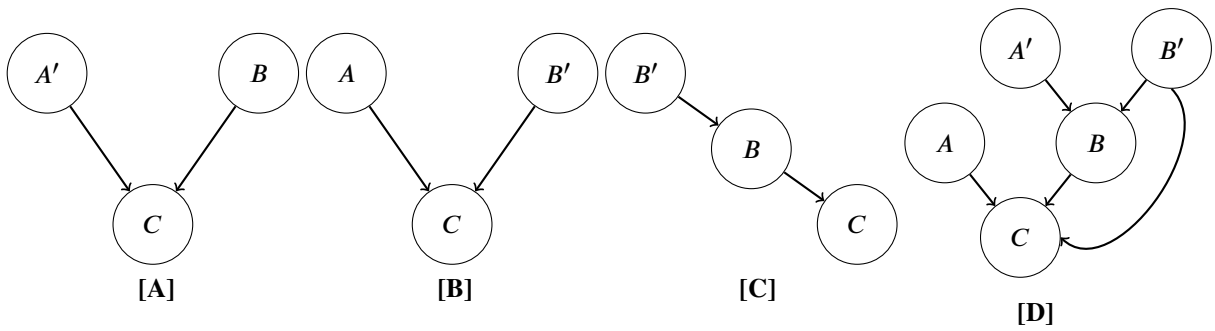
$$x*x / (x*x + (1-x)*y)$$

(c) (i) [2 pts] Consider the Bayes net of the previous part (the one containing A' and B'). Which of the following conditional independence relations are guaranteed by the Bayes net graph?

- $A' \perp\!\!\!\perp C$
- $A' \perp\!\!\!\perp B' | C$
- $A' \perp\!\!\!\perp A$
- $B' \perp\!\!\!\perp A | B$
- None of the above

(ii) [2 pts] Which of the following new Bayes nets violate at least one (conditional) independence relation implied by the original Bayes Net? Note that the answer choices in the previous part are not a comprehensive list of all possible conditional independence relations.

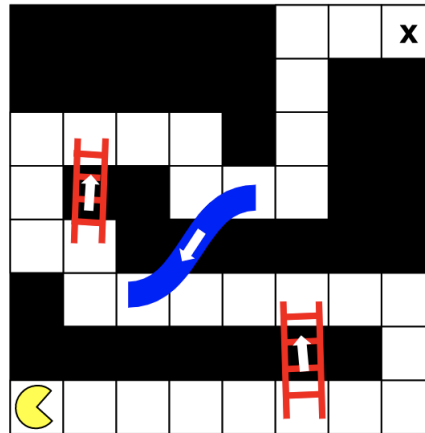
- A
- B
- C
- D
- None



Q5. [8 pts] Slides and Ladders

Pacman is in a two-dimensional grid world with ladders and slides. There is a single path to his goal, and the goal is at the end of the path so there are no squares after the goal. Ladders carry Pacman to a square on the path that is closer to his goal, and slides carry him to a square that is farther. Ladders and slides may cross, but each square can only be the beginning or the end of one ladder or slide. (Note: this figure is only for demonstration purposes. For all the questions below, the size of the grid, as well as the locations of ladders and slides, may vary.)

At each timestep, Pacman normally moves forward one step towards the goal along the path. However, if Pacman is at a square where a slide or ladder begins, he can instead choose to get transported to the end of the slide or ladder at the next time step. Pacman wants to minimize the number of time steps it takes to reach the goal.



For each of the following heuristics $h_i(s)$, where s is the input state, indicate whether A* graph search, A* tree search, both algorithms, or neither algorithm will be optimal under the heuristic h_i . The different scenarios below should be viewed independently. Let $(x(s), y(s))$ be Pacman's location in state s , (x_{goal}, y_{goal}) be the goal location, and $d(s)$ be the Manhattan distance between (x_{goal}, y_{goal}) and $(x(s), y(s))$.

- (a) Let $B(s)$ and $E(s)$ be lists of squares which contains all the beginnings and ends of the ladders along the path from state s to the goal, respectively. $B(s)[i]$ and $E(s)[i]$ refer to the i th element of $B(s)$ and $E(s)$, respectively. Note that $|X|$ represents the number of elements in the list X .

- (i) [2 pts] $h_1(s) = |B(s)|$

- A* graph search Both
 A* tree search Neither

This heuristic is inadmissible. Consider the case where Pacman is at a ladder that takes him directly to the goal, and there are any number of ladders beginning between.

- (ii) [2 pts]

$$h_2(s) = \begin{cases} d(s) & \text{if } |B(s)| = 0 \\ \min_{0 \leq i < |B(s)|} \text{Manhattan}((x(s), y(s)), B(s)[i]) + \min_{0 \leq i < |B(s)|} \text{Manhattan}((x_{goal}, y_{goal}), E(s)[i]) & \text{otherwise} \end{cases}$$

- A* graph search Both
 A* tree search Neither

The heuristic is consistent. While there are ladders ahead, Pacman must move forward by at least the minimum Manhattan distance to a ladder beginning and at least the minimum Manhattan distance between ladder endings and the goal. If there are no more ladders, he must travel at least the Manhattan distance to the goal. The edge forward is not overestimated and there is no edge backward.

- (b) [2 pts] Assume that all ladders go straight up (only their y coordinates change) and the length of the longest ladder is l_{max} . If no ladders exist, let $l_{max} = 1$.

$$h_3(s) = \frac{|y(s) - y_{goal}|}{l_{max}} + |x(s) - x_{goal}|$$

- A* graph search Both
 A* tree search Neither

This heuristic is consistent. The largest step we can take upward is the length l_{max} , and we must take at least the current distance sideways.

- (c) [2 pts] Assume now that Pacman can choose to ignore any slide entrances that he encounters up to Z times and after that, he must transport down the slide if he enters a square containing a slide beginning. Let $C(s)$ and $F(s)$ be lists of squares which contain all the beginnings and ends of the slides along the path from state s to the goal, respectively. Let $l_{max} =$ the max ladder length or 1 if none exist, and z be the number of times that Pacman has already ignored a slide entrance. As a reminder, $d(s)$ is the Manhattan distance between (x_{goal}, y_{goal}) and $(x(s), y(s))$.

$$h_4(s) = \begin{cases} d(s)/l_{max} & \text{if } |C(s)| = 0 \text{ or } z \leq Z \\ d(s)/l_{max} + \min_{0 \leq i < |C(s)|} (\text{Manhattan}(C(s)[i], F(s)[i])) & \text{otherwise} \end{cases}$$

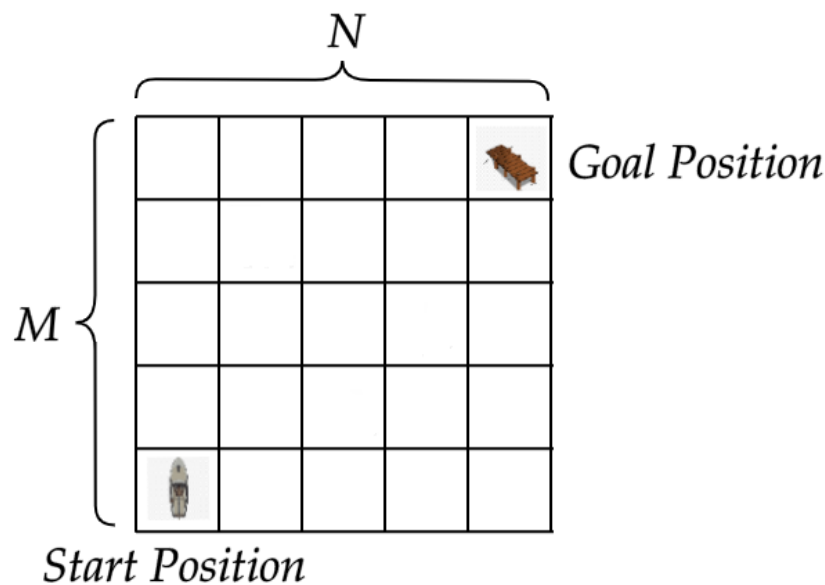
- A* graph search Both
 A* tree search Neither

This heuristic is inadmissible. While Pacman can avoid falling, he must travel at least his current Manhattan distance to the goal divided by his largest possible single step. Once he starts to fall down slides inevitably, the length of his journey may increase by the Manhattan distance of the shortest slides, but he might also encounter ladders, so optimality is not guaranteed for either algorithm.

Q6. [13 pts] State Spaces and Uninformed Search

A boat is sitting in an M by N grid lake. There are exactly K unique food pellets initially scattered in fixed positions in the lake, and the goal is to pick up all the food pellets and transport them to the goal position. Here are the problem specifications:

- The boat starts out in the bottom left with initially no food pellets and the goal position is in the top right.
- The boat can face 4 different directions. At each timestep, it can take one of the following actions: turn north, turn east, turn south, turn west, go forward, or stay still. **Each action has cost 1.** Note that the action "go forward" is not available if it will cause the boat to fall off the grid.
- The boat can carry at most C food pellets at a time. When the boat enters a square with a food pellet, it will automatically pick it up if possible; if not (i.e., the boat is already carrying C food pellets), the food pellet will remain in the cell.
- When the boat enters the goal position, all food pellets it currently carries will automatically be dropped off onto the goal position.



$$MN \cdot 3^K \cdot 4$$

- (a) [3 pts] Calculate the size of the minimum state space for this task.
- (b) [3 pts] Mark all of the following that are guaranteed to be true for the search problem described above. Assume that we break ties between nodes in the same manner for all searches.
- BFS tree search will always find an optimal solution to the goal.
 - BFS tree search will always return the same solution as iterative deepening search.
 - When conducting UCS, the first path to the goal which is added to the frontier will always be the optimal path.
 - None of the above

1. True bc the total action cost of a path is equivalent to the depth of the path and BFS guarantees minimum depth.
2. True bc IDS expands each node for the first time in the same order as BFS
3. True bc each edge has the same action cost of 1 so the first time the goal state is added to the frontier, it must be an optimal solution. (Also the same reason for why BFS is optimal.)

(c) Now, expanding on the same setup as before, E pirates have showed up to the lake. Each pirate follows a fixed action sequence: they always take the action "move forward" if possible; otherwise, they take the action "turn clockwise" if moving forward puts them off the grid. Pirate actions are known and deterministic; they happen simultaneously with boat actions. If the boat and a pirate occupy the same cell, the boat will lose all of the food pellets it is currently carrying and each food pellet will be reset to its initial location.

(i) [2 pts] Let S be the state space size from part (a). What is the minimum state space size for the new problem, expressed in terms of S (and any other variables)?

$$S \cdot (MN)^E \cdot 4^E \text{ or } S \cdot \binom{4MN}{E}$$

Either treating the pirates as distinguishable or indistinguishable.

(ii) [3 pts] Consider three heuristics for the original problem formulation (with no pirates): h_1 is admissible, h_2 is consistent, and h_3 is neither but results in an optimal solution when used in A* graph search. Select all of the following statements which are true about these heuristics for the new problem formulation with pirates.

- h_1 is still admissible for the new problem
- h_2 is still consistent for the new problem
- h_3 is still guaranteed to return an optimal solution when used in A* graph search for the new problem
- None of the above

The new problem can be viewed as the same state space as the original but with additional edges that point backward from a state closer to the goal to a state farther (when a boat loses all its pellets and is reset).

h_1 is still admissible and h_2 is still consistent since there are no edges in the new problem formulation that shorten any path to the goal.

h_3 may not guarantee an optimal solution. Consider a heuristic h_3 which only underestimates the heuristics for states along the optimal path to the goal in the original problem while overestimating all other states' heuristics. With the addition of ghosts, the optimal path may cross through different states that have overestimated heuristics even under the new problem formulation, which can prevent them from being searched.

(iii) [2 pts] You want to run UCS to find an optimal solution but your program has limited memory that allows only at most X states to be stored on the frontier. Let b be the branching factor for the problem and c be the cost of the optimal solution. Determine the correct upper bound on c such that UCS is guaranteed to find the optimal solution while staying within the space constraint.

$$c \leq \log_b X$$

Since all actions cost 1, UCS acts like BFS. We know from lecture that the space complexity of BFS is $O(b^d)$, where b is the branching factor and d is the max depth of the solution.

The problem states that given the memory constraint, UCS can only find solutions with cost less than c , and we want to find this upper bound. First, we note that $b^d \leq X$, since we have limited memory.

Next, we can take the \log_b of both sides, getting $d \leq \log_b X$. This means the maximum depth a solution can be is $\log_b X$ in terms of X .

Q1

- (a) (i) (ii) (iii) (iv)
 (v) (vi) (vii) (viii)
 (ix) (x) (xi) (xii)
- (b) (i) (ii) (iii)
- (c)





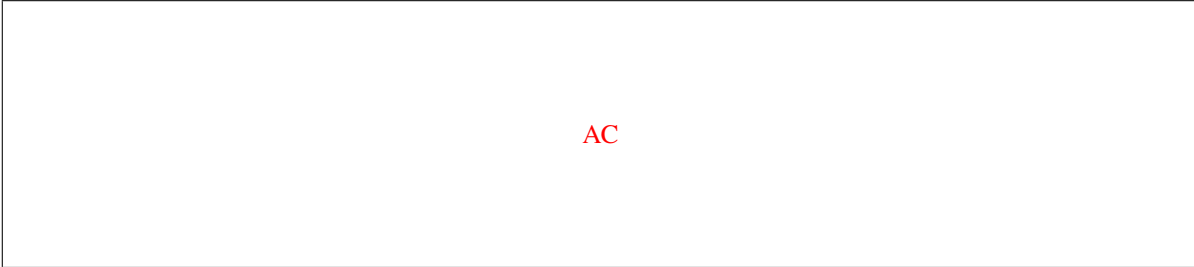


Q2

- (a) (i)
 (ii) (1)
 (2)
 (3)
 (4)
- (b) (i) (ii)





Q3

- (a) (i) (1) (2) (3)
- (ii)
- (iii)
- (iv) (1) (2)
- (b) (i)
 (ii)
 (iii)
 (iv) (1) (2)

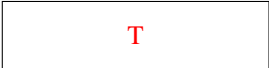
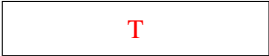


Q4

- (a) (i) 
- (ii) 
- (iii) 
- (iv) 
- (b) 
- (c) (i)  (ii) 

Q5

- (a) (i)  (ii) 
- (b) 
- (c) 

Q6

- (a) 
- (b) 
- (c) (i) 
- (ii) 
- (iii) 