Q1. [16 pts] Search: Project Groups

You're taking a class with n students, and you want to find a 4-person project group (you and 3 other students). You decide to formulate your problem as a search problem, as follows:

- **State space**: An unordered set of students in the group so far. For example, {you, Inky, Blinky} and {you, Blinky, Inky} are equivalent states.
- **Successor function**: You pick someone from the class (who is not already in your group), and add them to your group. If your group already has 4 students, there are no further successor states.
- **Start state**: The project group is just you, and nobody else.
- **Goal test**: Your group has exactly 4 students, and everyone in the group gets along. (You can assume that the goal test is somehow able to determine whether a group gets along.)

Consider drawing the search **graph** for this problem.

(a) [2 pts] How many nodes are in the search graph?

Hint: Recall that $\binom{n}{k} = \frac{n!}{(n-k)!k!}$ is the number of ways to choose k elements from a set of n elements. Also, $0! = 1$.

- ◯ $(n-1)$
- ◯ $(n-1)^4$
- ◯ $\binom{n-1}{3}$
- ◯ $\binom{n-1}{0} + \binom{n-1}{1} + \binom{n-1}{2} + \binom{n-1}{3}$
- ◯ Infinitely many

(b) [2 pts] Does the search graph contain any cycles?

- ◯ Yes, the cycles allow us to backtrack from groups that don't pass the goal test.
- ◯ Yes, the successor function allows us to move back-and-forth between two adjacent states.
- ◯ No, because search graphs by definition never contain cycles.
- ◯ No, because it is only possible to move to states with more students in the group.

Now, consider drawing the search **tree** for this problem.

(c) [2 pts] What is the maximum branching factor for this search tree?

- ◯ $(n-1)$
- ◯ $(n-1)^4$
- ◯ $\binom{n-1}{3}$
- ◯ $\binom{n-1}{0} + \binom{n-1}{1} + \binom{n-1}{2} + \binom{n-1}{3}$
- ◯ Could be infinite

(d) [2 pts] What is the maximum depth of this search tree, assuming the root node is at depth 0?

- ◯ 1
- ◯ 2
- ◯ 3
- ◯ $n-1$
- ◯ $(n-1)^4$
- ◯ Could be infinite

Now, suppose that you want k students in your project group ($k < n$).

Also, suppose that any group of k students passes the goal test.

(e) [2 pts] As n and k grow very large, which search algorithm will expand fewer nodes to find a solution?

- ◯ Depth-first tree search
- ◯ Breadth-first tree search
- ◯ Not enough information

(f) [2 pts] Approximately how many nodes will be expanded by the search algorithm you selected before it finds a solution?

- ◯ $O(k)$
- ◯ $O(n)$
- ◯ $O(nk)$
- ◯ $O(k^n)$
- ◯ $O(n^k)$

Now, suppose that some pairs of students in the class cannot work together because they dislike each other. We'd like to modify our search problem definition to account for this.

(g) [4 pts] Which of the following changes would, if implemented individually, ensure that any complete search algorithm still finds a correct solution, if one exists?

- ☐ Modify the goal test to check that no such pairs exist in the state being tested.
- ☐ Modify the successor function so that it only adds students who can work with anybody.
- ☐ Modify the successor function so that it only adds students who can work with anybody already in the group.
- ☐ Require that states be ordered lists rather than sets.
- ◯ None of the above. A search algorithm cannot solve this problem. You would need a CSP algorithm.

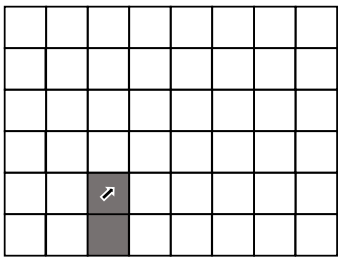
Q2. [18 pts] Search: Color a Dinosaur

Consider a computer screen with a grid of squares.

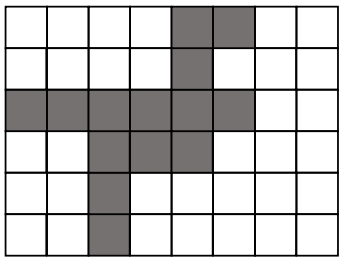
We start with a blank grid and the mouse cursor in the bottom-left square. At each time step, we have several actions to choose from, each costing 1:

- Move the mouse cursor to an adjacent square (north, south, east, west).
- Click the mouse to fill in the currently selected square.

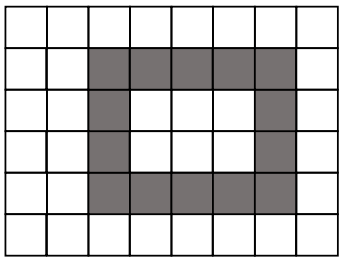
For example, starting from the bottom-left, the action sequence East, East, Click, North, Click produces the grid in Figure 1(a).



(a) Configuration after East, East, Click, North, Click. The arrow denotes the mouse cursor.



(b) A possible goal configuration. The cursor location is ignored in the goal test.



(c) Another possible goal configuration.

Figure 1: Examples for the coloring problem.

We are provided with a list of squares, and we'd like to color all of those squares to form a picture, such as the dinosaur in Figure 1(b) or the rectangle in Figure 1(c), using the fewest number of actions. (Don't laugh! Pixels are expensive!)

A square is *unfinished* if it still needs to be colored.

For each piece of information listed below, select the part of the search problem definition where that information should be stored.

(a) [1 pt] The mouse cursor starts in the bottom-left square.

- ◯ State space
- ◯ Successor function
- ◯ Goal test
- ◯ Start state

(b) [1 pt] The current location of the mouse cursor.

- ◯ State space
- ◯ Successor function
- ◯ Goal test
- ◯ Start state

(c) [1 pt] The list of squares colored so far.

- ◯ State space
- ◯ Successor function
- ◯ Goal test
- ◯ Start state

(d) [1 pt] The original list of squares that need to be colored.

- ◯ State space
- ◯ Successor function
- ◯ Goal test
- ◯ Start state

For each heuristic provided, select whether it is admissible, and whether it is consistent.

(e) [3 pts] h_1 = Number of unfinished squares

- ◯ Both admissible and consistent
- ◯ Consistent, but not admissible
- ◯ Admissible, but not consistent
- ◯ Neither admissible nor consistent

(f) [3 pts] h_2 = 2 times the number of unfinished squares

- ◯ Both admissible and consistent
- ◯ Consistent, but not admissible
- ◯ Admissible, but not consistent
- ◯ Neither admissible nor consistent

(g) [3 pts] h_3 = maximum Manhattan distance between any two unfinished squares

- ◯ Both admissible and consistent
- ◯ Consistent, but not admissible
- ◯ Admissible, but not consistent
- ◯ Neither admissible nor consistent

(h) [2 pts] Consider h_1 and h_3. Select all true statements:

- ☐ h_1 dominates h_3.
- ☐ h_3 dominates h_1.
- ☐ Neither h_1 nor h_3 dominates the other.
- ☐ If h_1 and h_3 are admissible, h^* dominates $\max(h_1, h_3)$.
- ◯ None of the above.

(i) [3 pts] Consider another heuristic:

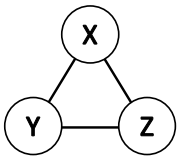
h_4 = Manhattan distance to nearest unfinished square + 2 times the number of unfinished squares

Select all true statements:

- ☐ Greedy **tree** search with h_4 will always find a solution (possibly not optimal) if one exists.
- ☐ Greedy **graph** search with h_4 will always find a solution (possibly not optimal) if one exists.
- ☐ If the goal is the rectangle in Figure 1(c), greedy tree search with h_4 will return the optimal solution.
- ◯ None of the above.

Q3. [18 pts] Logic: Map Coloring

We'd like to express the map coloring problem in propositional logic. Consider the following map:



We'd like to assign colors to nodes X, Y, and Z such that no two adjacent nodes have the same color.

We have three colors available to use: R (Red), G (Green), and B (Blue).

Let the proposition symbol N_C represent node N being assigned color C. For example, $X_R = \textit{true}$ means that node X is colored Red.

(a) [2 pts] How many symbols are needed to express this problem in propositional logic?

Your answer should be an integer.

(b) [1 pt] For a general map coloring problem with n nodes and c colors, how many symbols are needed to express the problem in propositional logic?

Your answer should be an expression, possibly in terms of n and/or c.

Now we'll formulate several logical sentences that represent this map coloring problem.

(c) [2 pts] Below is a sentence that represents part of this map coloring problem. Fill in the blanks with $\vee$ or $\wedge$.

$$(X_R \_\_\_\_\_ X_G \_\_\_\_\_ X_B) \_\_\_\_\_ (Y_R \_\_\_\_\_ Y_G \_\_\_\_\_ Y_B) \_\_\_\_\_ (Z_R \_\_\_\_\_ Z_G \_\_\_\_\_ Z_B)$$

(d) [2 pts] What does the sentence in the previous subpart represent? You can answer in 10 words or fewer.

(e) [2 pts] Write a logical sentence that encodes the constraint "X and Y cannot have the same color". Your sentence does not need to be in CNF.

(f) [2 pts] Suppose we've additionally added sentences that encode "X and Z cannot have the same color," and "Y and Z cannot have the same color."

This problem is still missing some sentence(s). Describe the missing sentence(s) (you can answer in 10 words or fewer):

(g) [1 pt] How many symbols are assigned to *true* in an assignment that solves the problem?

Your answer should be an integer.

Pacman would like to prove the following statement: "If X is colored Red in this problem, then Y will not be colored Red," using only a SAT solver.

(h) [3 pts] In addition to the sentences defining the problem from subparts (c), (e), and (f), which other clauses should be part of the input to the SAT solver to ensure that Pacman gets the right answer? Select all that apply.

- ☐ (X_R)
- ☐ $(\neg X_R)$
- ☐ (Y_R)
- ☐ $(\neg Y_R)$
- ☐ $(\neg X_R \vee \neg Y_R)$
- ☐ $(X_R \vee Y_R)$
- ◯ None of the above.

(i) [3 pts] Select all true statements:

- ☐ Every discrete, finite CSP can be represented as a SAT problem.
- ☐ Every SAT problem can be represented as a CSP.
- ☐ A correct SAT representation of a discrete, finite CSP has exactly the same number of satisfying assignments as the CSP has distinct solutions.
- ◯ None of the above.

Q4. [15 pts] CSPs: The Zookeeper

You are a newly appointed zookeeper, and your first task is to find rooms for all of the animals.

The zoo has three animals: the Iguana (I), Jaguar (J), and Koala (K).

Each animal needs to be assigned to one of four rooms: the North room (N), East room (E), South room (S), or West room (W), subject to the following constraints:

1. The jaguar cannot share a room with any other animal.
2. The iguana and koala must be in different rooms.
3. The koala can only be in the East room or the South room.

(a) [2 pts] Consider the first constraint: "The jaguar cannot share a room with any other animal."

Can this constraint be expressed using only binary constraints on the three variables I, J, K?

- ◯ Yes, it can be expressed as 1 binary constraint.
- ◯ Yes, it can be expressed as 2 different binary constraints.
- ◯ Yes, it can be expressed as 4 different binary constraints.
- ◯ No, this is necessarily a unary constraint.
- ◯ No, this is necessarily a higher-order constraint.

(b) [3 pts] Suppose we enforce unary constraints, and then assign the jaguar to the South room. The remaining values in each domain would be:

Iguana:	North, East, South, West
Jaguar:	South
Koala:	East, South

In the table below, mark each value that would be **removed** by running forward-checking after this assignment.

	North	East	South	West
Iguana	☐	☐	☐	☐
Koala		☐	☐	

(c) [3 pts] Regardless of your answer to the previous subpart, suppose we've done some filtering and the following values remain in each variable's domain:

Iguana:	East, West
Jaguar:	South
Koala:	East

Are all the arcs in this CSP consistent?

- ◯ Yes, all arcs are consistent.
- ◯ No, only $K \to I$ is not consistent.
- ◯ No, only $I \to K$ is not consistent.
- ◯ No, $K \to I$ and $I \to K$ are both not consistent.
- ◯ No, only $J \to I$ is not consistent.
- ◯ No, only $I \to J$ is not consistent.
- ◯ No, $J \to I$ and $I \to J$ are both not consistent.

The constraints, repeated here for your convenience:

1. The jaguar cannot share a room with any other animal.
2. The iguana and koala must be in different rooms.
3. The koala can only be in the East room or the South room.

(d) [2 pts] Regardless of your answer to the previous subpart, suppose we start over and just enforce the third constraint. Then the remaining values in each domain are:

Iguana:	North, East, South, West
Jaguar:	North, East, South, West
Koala:	East, South

What does the minimum remaining values (MRV) heuristic suggest doing next?

- ◯ Assign North or West to a variable next.
- ◯ Assign East or South to a variable next.
- ◯ Assign a value to Koala next.
- ◯ Assign a value to Iguana or Jaguar next.

(e) [2 pts] Again, consider the CSP after just enforcing the third constraint:

Iguana:	North, East, South, West
Jaguar:	North, East, South, West
Koala:	East, South

Which assignment would the least constraining value (LCV) heuristic prefer?

- ◯ Assign North to Jaguar.
- ◯ Assign East to Jaguar.
- ◯ LCV is indifferent between these two assignments.

(f) [3 pts] Suppose we add another constraint:

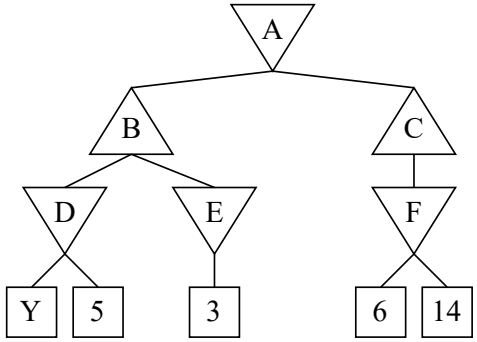
"The West room can contain at most one animal."

Can this constraint be expressed using only binary constraints on the three variables I, J, K?

- ◯ Yes, it can be expressed as 1 binary constraint.
- ◯ Yes, it can be expressed as 2 different binary constraints.
- ◯ Yes, it can be expressed as 3 different binary constraints.
- ◯ No, this is necessarily a higher-order constraint.

Q5. [16 pts] Games

Consider the following game tree where upward triangle nodes are max nodes, and downward triangle nodes are min nodes:



(a) [1 pt] If $Y = 4$, what is the minimax value at the root node A?

(b) [3 pts] What range of values for Y would cause the minimax value at the root A to be equal to Y?

Your answer should be an inequality using $\leq$, such as $Y \leq 188$, or $188 \leq Y \leq 288$, or $388 \leq Y \leq 388$, or "impossible" (if there are no such values for Y).

(c) [4 pts] Assume that $Y = 4$. If we run minimax with alpha-beta pruning, visiting nodes from left to right, which terminal nodes will we **not** visit because of pruning? Select all that apply.

- ☐ Y
- ☐ 5
- ☐ 3
- ☐ 6
- ☐ 14
- ◯ None of the above (we visit all terminal nodes).

(d) [2 pts] When is the statement $A \leq \max(Y, 14)$ true?

◯ Always true. ◯ Sometimes true. ◯ Never true.

(e) [2 pts] The root of this tree is a min node. You would like to represent this same game with a new tree that has a max node at the root.

Select all changes that you would need to make in the new tree.

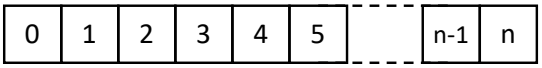
- ☐ Convert all min nodes to max nodes.
- ☐ Convert all max nodes to min nodes.
- ☐ Add a layer of chance nodes to the new tree.
- ☐ Multiply each of the terminal node values by -1.
- ◯ None of the above - it is impossible to represent this game with a new tree with a max node at the root.

(f) [4 pts] Suppose the max nodes B and C are replaced with chance nodes that select actions uniformly at random. What range of values for Y would cause the value at the root A to be equal to Y?

Your answer should be an inequality using $\leq$, such as $Y \leq 188$, or $188 \leq Y \leq 288$, or $388 \leq Y \leq 388$, or "impossible" (if there are no such values for Y).

Q6. [17 pts] MDPs: Flying Pacman

Pacman is in a 1-dimensional grid with squares labeled 0 through n, inclusive, as shown below:



Pacman's goal is to reach square n as cheaply as possible. From state n, there are no more actions or rewards available.

At any given state, if Pacman is not in n, Pacman has two actions to choose from:

- **Run**: Pacman deterministically advances to the next state (i.e. from state i to state $i+1$). This action costs Pacman \$1.
- **Fly**: With probability p, Pacman directly reaches state n. With probability $1-p$, Pacman is stuck in the same state. This action costs Pacman \$2.

(a) [3 pts] Fill in the blank boxes below to define the MDP. i represents an arbitrary state in the range $\{0, \ldots, n-1\}$.

s	a	s'	$T(s,a,s')$	$R(s,a,s')$
i	Run	$i+1$		
i	Fly	i		
i	Fly			

For the next three subparts, assume that $\gamma = 1$.

Let π_R denote the policy of always selecting Run, and π_F denote the policy of always selecting Fly.

Compute the values of these two policies. Your answer should be an expression, possibly in terms of n, p, and/or i.

(b) [2 pts] What is $V^{\pi_R}(i)$?

(c) [2 pts] What is $V^{\pi_F}(i)$?

Hint: Recall that the mean of a geometric distribution with success probability p is $\sum_{k=1}^{\infty} k(1-p)^{k-1}p = 1/p$.

(d) [4 pts] Given the results of the two previous subparts, we can now find the optimal policy for the MDP.

Which of the following are true? Select all that apply. (Hint: consider what value of i makes $V^{\pi_R}(i)$ and $V^{\pi_F}(i)$ equal.)

Note: $\lceil x \rceil$ is the smallest integer greater than or equal to x.

- ☐ If $p < 2/n$, Fly is optimal for all states.
- ☐ If $p < 2/n$, Run is optimal for all states.
- ☐ If $p \geq 2/n$, Fly is optimal for all $i \geq \lceil n - 2/p \rceil$ and Run is optimal for all $i < \lceil n - 2/p \rceil$.
- ☐ If $p \geq 2/n$, Run is optimal for all $i \geq \lceil n - 2/p \rceil$ and Fly is optimal for all $i < \lceil n - 2/p \rceil$.
- ○ None of the above.

Regardless of your answers to the previous parts, consider the following modified transition and reward functions (which may not correspond to the original problem). As before, once Pacman reaches state n, no further actions or rewards are available.

For each modified MDP and discount factor, select whether value iteration will converge to a finite set of values.

(e) [2 pts] $\gamma = 1$

s	a	s'	$T(s, a, s')$	$R(s, a, s')$
i	Run	$i+1$	1.0	+5
i	Fly	$i+1$	1.0	+5

- ◯ Value iteration converges
- ◯ Value iteration does not converge
- ◯ Not enough information to decide

(f) [2 pts] $\gamma = 1$

s	a	s'	$T(s, a, s')$	$R(s, a, s')$
i	Run	$i+1$	1.0	+5
i	Fly	$i-1$	1.0	+5

- ◯ Value iteration converges
- ◯ Value iteration does not converge
- ◯ Not enough information to decide

(g) [2 pts] $\gamma < 1$

s	a	s'	$T(s, a, s')$	$R(s, a, s')$
i	Run	$i+1$	1.0	+5
i	Fly	$i-1$	1.0	+5

- ◯ Value iteration converges
- ◯ Value iteration does not converge
- ◯ Not enough information to decide