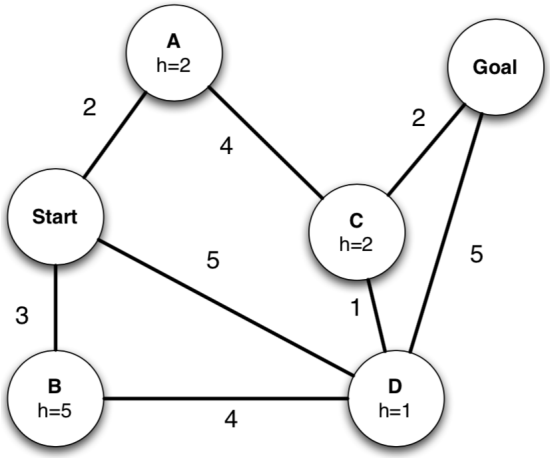
1 Big Picture

Problem	Representation	Stochastic/Deterministic	Goal
Search (all types)	States, Transitions, Start state, Goal state	Deterministic	Finding a path to the goal
CSPs	Domains, Variables, Constraints	Deterministic	Finding a complete assignment that satisfies constraints
LPs, IPs	Constraints, Objective function	Deterministic	Assignment that minimizes objective
Planning	Propositions, Actions, Start state, Goal state	Deterministic	Finding a plan (duh)
MDPs	States, Actions, Transition probabilities, Reward model	Stochastic	Finding a policy
Bayes Nets	CPT's, queries	Stochastic	Finding probabilities of specific events/queries
Game Theory	Choices, Payoffs	Deterministic	Finding a strategy that maximizes utility
Social Choice	Voting rules, candidates	Deterministic	Finding the winner

2 Search



For each of the following graph search strategies, work out the order in which states are explored, as well as the path returned by graph search. In all cases, break ties in alphabetical order. The start and goal state use letter S and G, respectively. Remember that in graph search, a state is explored only once.

(a) Depth-first search.

States Explored: Start, A, C
Path Returned: Start-A-C-Goal

(b) Breadth-first search.

States Explored: Start, A, B, D, C
Path Returned: Start-D-Goal

(c) Uniform cost search.

States Explored: Start, A, B, D, C
Path Returned: Start-A-C-Goal

(d) Greedy search with the heuristic values h shown on the graph.

States Explored: Start, D
Path Returned: Start-D-Goal

(e) $A^\star$ search with the same heuristic.

States Explored: Start, A, D, B, C
Path Returned: Start-A-C-Goal

3 Adversarial Search

Warm up

1. What is the advantage of adding alpha-beta pruning to a minimax algorithm?

   It on average speeds up minimax algorithms by reducing the number of nodes that need to be examined.This is achieved by "pruning" nodes which have been found not to change the result produced by thealgorithm.

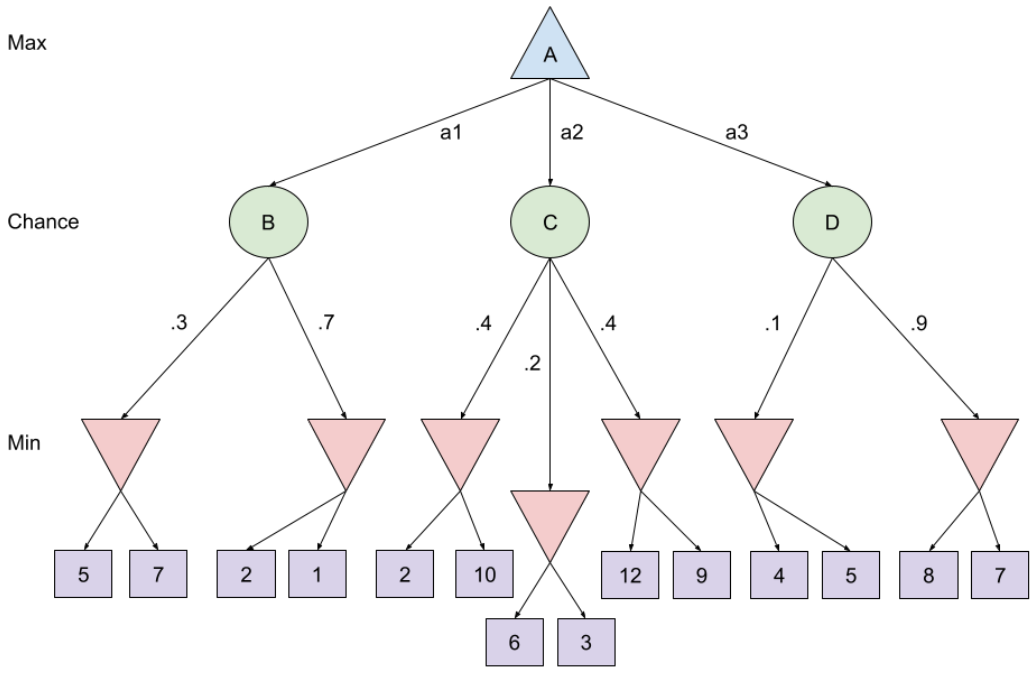
2. Give two advantages of Iterative Deepening minimax algorithms over Depth Limited minimax algorithms.

   I) Solution availability: You always have the solution of the previous iteration available during the execution of the current iteration (this is particularly useful when under a time constraint).

   II) Information gleaned during the current iteration can be employed to increase pruning in successive iterations (Recall HW2 Question 5). Because successive iterations require exponentially more time, and searching at lower depths is typically insignificant while increased pruning at higher depths can be very significant.

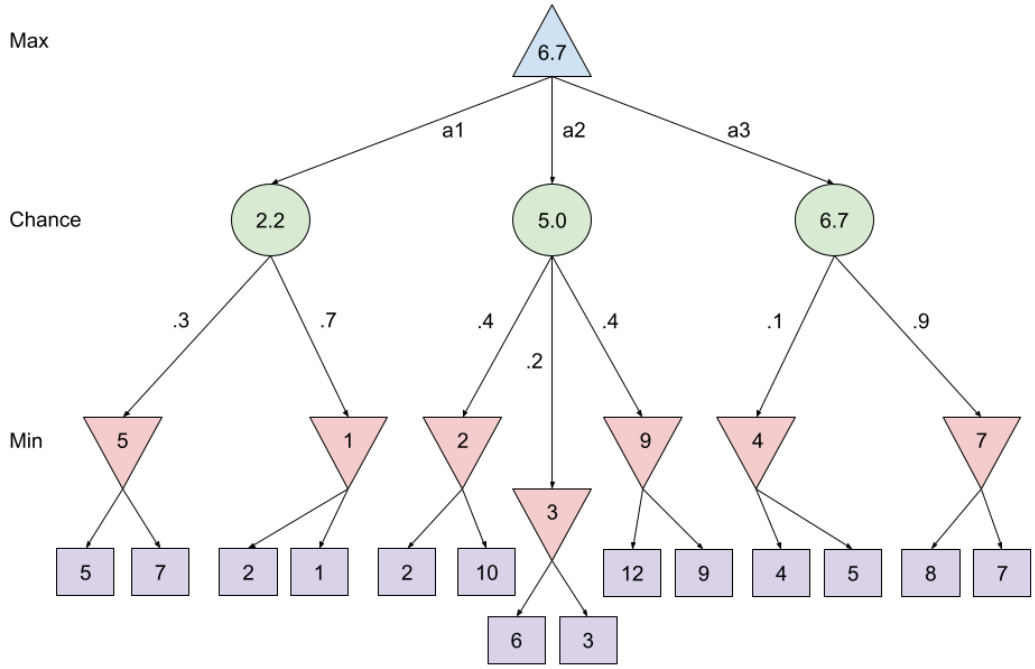
Expectiminimax

The following three questions are about the following adversarial "chance" tree.



1. Calculate the EXPECTIMINIMAX values for nodes B, C and D in the above adversarial "chance" tree.

EXPECTIMINIMAX(B) = .3 * 5 + .7 * 1 = 1.5 + .7 = 2.2
EXPECTIMINIMAX(C) = .4 * 2 + .2 * 3 + .4 * 9 = .8 + .6 + 3.6 = 5.0
EXPECTIMINIMAX(D) = .1 * 4 + .9 * 7 = .4 + 6.3 = 6.7



2. Which action will MAX choose, a_1, a_2, or a_3? Why?

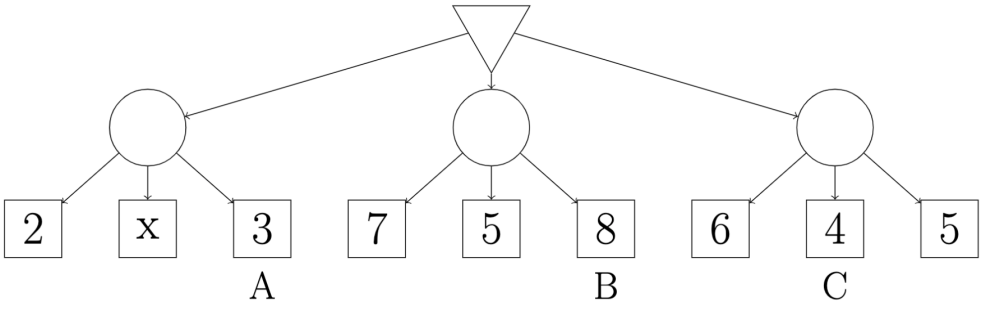
   MAX will choose action a_3 because it has the highest EXPECTIMINIMAX value.

3. If the utility values given for MIN were multiplied with a positive constant c, which action would MAX then choose?

   MAX would still choose action a_3 because multiplying with a positive constant is a positive linear transformation and such transformations do not change decisions made on the basis of EXPECTIMINIMAX values.

Expectimin Pruning

For each of the leaves labeled A, B, and C in the expectimin tree below, determine which values of x would cause the leaf to be pruned, given the information that all values are non-negative and all nodes have 3 children. Assume all children of expectation nodes have equal probability and sibling nodes are visited left to right for all parts of this question. Assume we do not prune on equality.



Below, write your answers as one of (1) an inequality of x with a constant, (2) "none" if no values of x will cause the pruning, or (3) "any" if the node will be pruned for all values of x.

1. A:

   None. No value of x can cause A to be pruned since the minimizer does not have any bound until after A is visited

2. B:

   $x < 7$. To prune B, the value of the first expectation node must be less than the value of the second even if B were 0. $\frac{2+x+3}{3} < \frac{7+5}{3} \implies x < 7$

3. C:

   $x < 1$. To prune C, the value of the first expectation node must be less than the value of the third if both of the last two leaves had value 0. $\frac{2+x+3}{3} < \frac{6}{3} \implies x < 1$

4 CSP Backtracking Search

1. What is the difference between MRV and LCV?

   With MRV we pick the variable with the smallest number of remaining assignable values while for LRV we pick the value that constrains the smallest number of the chosen variable's neighbors.

2. What is the difference between forward checking and AC-3?

   Forward checking only enforces consistency on arcs corresponding to a single variable while AC-3 also enforces arc consistency for any variables whose domains were affected in the process.

3. In this problem, you are given a 3×3 grid with some numbers filled in. The squares can only be filled with the numbers $\{2, 3, \ldots, 10\}$, with each number being used once and only once. The grid must be filled such that adjacent squares (horizontally and vertically adjacent, but not diagonally) are relatively prime.

x_1	x_2	x_3
x_4	x_5	3
4	x_6	2

   We will use backtracking search to solve the CSP with the following heuristics:

   - Use the Minimal Remaining Values (MRV) heuristic when choosing which variable to assign next.
   - Break ties with the Most Constraining Variable (MCV) heuristic.
   - If there are still ties, break ties between variables x_i, x_j with $i < j$ by choosing x_i.
   - Once a variable is chosen, assign the minimal value from the set of feasible values.
   - For any variable x_i, a value v is infeasible if and only if: (i) v already appears elsewhere in the grid, or (ii) a variable in a neighboring square to x_i has been assigned a value u where $\gcd(v, u) > 1$, which is to say, they are not relatively prime.

   Fill out the table below with the appropriate values.

   - Give initial feasible values in set form; x_1 has already been filled out for you.
   - Assignment order refers to the order in which the final value assignments are given. If x_i is the j^{th} variable on the path to the goal state, then the assignment order for x_i is j.
   - In the branching column, write "yes" if the algorithm branches (considers more than one value) at that node in the search tree (for example, x_4 considers more than 1 value), and write "B" if the algorithm backtracks at that node, meaning it is the highest node in its subtree that fails for a value, and has to be chosen again. Also write the values it tried then failed.

Variable	Initial Feasible Values	Assignment Order	Final Value	Branch or Backtrack?
x_1	$\{5, 6, 7, 8, 9, 10\}$	_____	_____	_____
x_2	_____	_____	_____	_____
x_3	_____	_____	_____	_____
x_4	_____	_____	_____	_____
x_5	_____	_____	_____	_____
x_6	_____	_____	_____	_____

Variable	Initial Feasible Values	Assignment Order	Final Value	Branch or Backtrack?
x_1	$\{5, 6, 7, 8, 9, 10\}$	5	6	No
x_2	{5,6,7,8,9,10}	4	7	No
x_3	{5,7,8,10}	6	10	No
x_4	{5,7,9}	1	5	Yes
x_5	{5,7,8,10}	2	8	B: 7
x_6	{5,7,9}	3	9	B: 7

To get started, we can assign 5 to x_4. This forces us to cross 5 off the remaining values for each of the other variables. We additionally have to cross off 10 from x_1 and x_5 since 5 and 10 are not relatively prime. Next, we assign 7 to x_5. Then, we assign 9 to x_6 since this is the only value left in its domain. Next, we can consider the value 6 for x_1. However, that leaves no value for x_2 to take on so instead, we backtrack and consider changing the values of x_5 and x_6 until we settle on 8 for x_5 and 9 for x_6. This leaves the values of 7, 6, and 10 for x_2, x_1, and x_3 respectively (the process is more clearly laid out in the following diagram).

① Assign $x_4 = 5$ (break tie with x_6 using MCV)

x_1	x_2	x_3
x_4 5	x_5	3
4	x_6	2

x_1 : ~~5~~ ~~6~~ 7 8 ~~9~~ ~~10~~
x_2 : ~~5~~ ~~6~~ 7 8 9 10
x_3 : ~~5~~ 7 8 10
x_4 : (5) 7 9
x_5 : ~~5~~ 7 8 10
x_6 : ~~5~~ 7 9

② Assign $x_5 = 7$ (break tie with x_6 using MCV)

x_1	x_2	x_3
x_4 5	x_5 7	3
4	x_6	2

x_1 : ~~5~~ ~~6~~ ~~7~~ 8 9 ~~10~~
x_2 : ~~5~~ ~~6~~ ~~7~~ 8 9 10
x_3 : ~~5~~ ~~7~~ 8 10
x_4 : (5) ~~7~~ 9
x_5 : ~~5~~ (7) 8 ~~10~~
x_6 : ~~5~~ ~~7~~ 9

③ Assign $x_6 = 9$

x_1	x_2	x_3
x_4 5	x_5 7	3
4	x_6 9	2

x_1 : ~~5~~ ~~6~~ ~~7~~ 8 ~~9~~ ~~10~~
x_2 : ~~5~~ ~~6~~ ~~7~~ 8 ~~9~~ 10
x_3 : ~~5~~ ~~7~~ 8 10
x_4 : (5) ~~7~~ ~~9~~
x_5 : ~~5~~ (7) 8 ~~10~~
x_6 : ~~5~~ ~~7~~ (9)

④ Assign $x_1 = 6$. This leaves no possible values in the domain of x_2 so we must backtrack!

x_1	x_2	x_3
6		
x_4 5	x_5 7	3
4	x_6 9	2

x_1 : ~~5~~ (6) ~~7~~ 8 ~~9~~ ~~10~~
x_2 : ~~5~~ ~~6~~ ~~7~~ ~~8~~ ~~9~~ ~~10~~
x_3 : ~~5~~ ~~7~~ 8 10
x_4 : (5) ~~7~~ ~~9~~
x_5 : ~~5~~ (7) 8 ~~10~~
x_6 : ~~5~~ ~~7~~ (9)

⑤ After trying out combos of values for x_5 and x_6 we settle on $x_5 = 8$, $x_6 = 9$

x_1	x_2	x_3
x_4 5	x_5 8	3
4	x_6 9	2

x_1 : ~~5~~ ~~6~~ 7 ~~8~~ ~~9~~ ~~10~~
x_2 : ~~5~~ ~~6~~ 7 ~~8~~ ~~9~~ ~~10~~
x_3 : ~~5~~ 7 ~~8~~ 10
x_4 : (5) 7 ~~9~~
x_5 : ~~5~~ 7 (8) ~~10~~
x_6 : ~~5~~ 7 (9)

⑥ Assign $x_2 = 7$

x_1	x_2	x_3
	7	
x_4 5	x_5 8	3
4	x_6 9	2

x_1 : ~~5~~ ~~6~~ 7 ~~8~~ ~~9~~ ~~10~~
x_2 : ~~5~~ ~~6~~ (7) ~~8~~ ~~9~~ ~~10~~
x_3 : ~~5~~ 7 ~~8~~ 10
x_4 : (5) 7 ~~9~~
x_5 : ~~5~~ 7 (8) ~~10~~
x_6 : ~~5~~ 7 (9)

⑦ Assign $x_1 = 6$

x_1	x_2	x_3
6	7	
x_4 5	x_5 8	3
4	x_6 9	2

x_1 : ~~5~~ (6) ~~7~~ ~~8~~ ~~9~~ ~~10~~
x_2 : ~~5~~ ~~6~~ (7) ~~8~~ ~~9~~ ~~10~~
x_3 : ~~5~~ ~~7~~ ~~8~~ 10
x_4 : (5) 7 ~~9~~
x_5 : ~~5~~ 7 (8) ~~10~~
x_6 : ~~5~~ 7 (9)

⑧ Assign $x_3 = 10$

x_1	x_2	x_3
6	7	10
x_4 5	x_5 8	3
4	x_6 9	2

x_1 : ~~5~~ (6) ~~7~~ ~~8~~ ~~9~~ ~~10~~
x_2 : ~~5~~ ~~6~~ (7) ~~8~~ ~~9~~ ~~10~~
x_3 : ~~5~~ ~~7~~ ~~8~~ (10)
x_4 : (5) 7 ~~9~~
x_5 : ~~5~~ 7 (8) ~~10~~
x_6 : ~~5~~ 7 (9)

5 Local Search

1. Warm-up Questions

   (a) In what ways are genetic algorithms and local beam search similar? How are they different?

   Both start with k randomly generated states and choose a set of "best" successors. Genetic algorithms use a fitness function to rank the states, which influences which states get paired in selection. Additionally, genetic algorithms contain crossover, which combines two parents, and random mutation for each child state.

   (b) What is the difference between first-choice hill climbing and random-restart hill climbing?

   First-choice hill climbing: randomly generate successors until a better one is found
   Random-restart hill climbing: randomly generate a start state and perform hill climbing from there

   (c) What are some pros and cons of local beam search?

   Pros: quickly abandons unfruitful searches and moves resources to where the most progress is being made
   Cons: can suffer from a lack of diversity among the k states. States can quickly become concentrated in a small region of the state space, making the search little more than an expensive version of hill climbing

2. Of the local search algorithms we have discussed, which one(s) would perform best in a continuous state space and why?

   First-choice hill climbing and simulated annealing: both of these search algorithms do not have infinite branching factors, so they would be able to handle continuous state spaces. AIMA Page 129 includes more discussion on local search in continuous spaces.

3. What are the disadvantages and advantages of allowing sideways moves? How can we modify our search algorithm to address the disadvantages?

   Advantage: the algorithm can find a better state if we make a sideways move along a shoulder.
   Disadvantage: allowing sideways moves could result in an infinite loop if we are at a local maximum.
   One potential modification to address this disadvantages would be to limit the number of consecutive sideways moves taken.

6 LP/IP

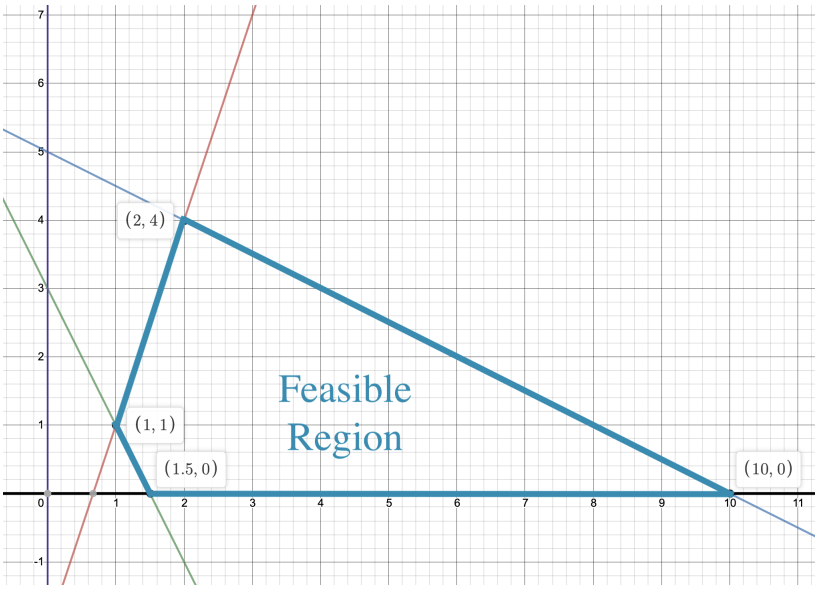
It's the holiday season and Santa Claus is getting ready to deliver lots of presents this year. However, he is struggling to pack everything, especially since there are a lot of perishable and fragile items. He needs help in determining how much dry ice and packing peanuts to use.

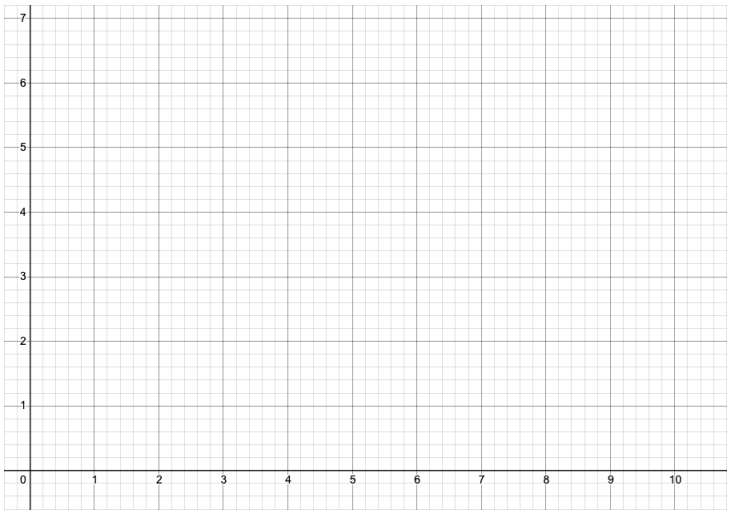
Let x_1 represent **pounds of dry ice** and let x_2 represent **pounds of packing peanuts**. We assume we can use a fraction of a pound of dry ice or packing peanuts. Santa has given us the following requirements:

- Santa only has room to spare for 10 bags of these materials. One pound of dry ice fits in one bag while one pound of packing peanuts takes up two bags.
- Santa needs to ensure that the temperature of the storage compartment of his sleigh is at most -2 degrees. One pound of dry ice **decreases** the temperature by 3 degrees while one pound of packing peanuts **increases** the temperature by 1 degree.
- Santa needs to ensure that the gifts are properly cushioned by these packaging materials. A pound of dry ice provides 2 units of "protectiveness" and a pound of packing peanuts provides 1 unit of "protectiveness". Santa wants there to be at least 3 units of "protectiveness".
- Unfortunately, Santa is on a tight budget this year so he is trying to minimize costs. One pound of dry ice costs \$0.5 and one pound of packing peanuts costs \$1.

1. Represent the following problem as an LP and graph the constraints in the provided graph.

$$\min_x c^T x \text{ st } Ax \leq b \text{ where A} = \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ -3 & 1 \\ 1 & 2 \\ -2 & -1 \end{bmatrix}, \text{ b} = \begin{bmatrix} 0 \\ 0 \\ -2 \\ 10 \\ -3 \end{bmatrix}, c = \begin{bmatrix} 0.5 \\ 1 \end{bmatrix}$$





2. Which rows in A correspond to the lines in the graph?

   ip_sol.pdf

3. What would the optimal solution be?

   The optimal point would be (1.5, 0).

4. If Santa didn't know the cost of dry ice and packing peanuts, what would be a cost vector that makes the optimal solution (1, 1)?

   A possible cost vector would be [1, 0].

5. List four cost vectors that will lead to an infinite number of solutions.

   $\begin{bmatrix}2\\1\end{bmatrix}, \begin{bmatrix}-1\\-2\end{bmatrix}, \begin{bmatrix}3\\-1\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}$

6. If Santa can only add dry ice and packing peanuts in increments of 1 pound (ie. he cannot divide a pound up) which constraints (and corresponding priorities) get pushed onto the priority queue in the first iteration of Branch and Bound?

   $x_1 \leq 1$ with priority 1.5
   $x_1 \geq 2$ with priority 1

7. What is the IP solution to this problem?

   (2, 0) with cost 1

7 Propositional Logic

1. Warm Up: Are you familiar with these terms?
   - Symbols

     Variables that can be T/F (capital letter)
   - Operators

     and, or, not, implies, equivalent
   - Sentences

     Symbols connected with operators, can be T/F
   - Literals

     atomic sentence
   - Knowledge Base

     Sentences agents know to be true
   - Entailment

     α entails β iff $\forall$ models that causes α to be true, β is also true
   - Query

     A sentence we want to know whether it's true (usually we want to know whether KB entails q)
   - Satisfiable

     At least one model makes the sentence true
   - Valid

     True for all models
   - Clause - Definite, Horn clauses

     Clause - disjunction of literals; definite - clause with exactly one positive literal, horn - clause with at most one positive literal
   - Model Checking

     Enumerating all possible models and using that to check entailment/satisfiability
   - Theorem Proving

     Search for a sequence of proof steps. (e.g. Forward Chaining)
   - Modus Ponens

     From $P, (P \longrightarrow Q)$, infer Q

2. Assume we have the following propositions: $BatteryDead, RadioWorks, OutOfGas$, and $CarStarts$ (You may use the abbreviations B, R, O, C in your answer).

   (a) What is the total number of possible models?

4 variables, hence $2^4 = 16$ models

(b) How many models are there in which the following sentence is $false$?
$(RadioWorks \wedge CarStarts) \implies (\neg OutOfGas \wedge \neg BatteryDead)$

An implication is false if the premise is true and the conclusion is false. There are 4 models where $R \wedge C$ is true, The negated conclusion is $\neg(\neg O \wedge \neg B)$ which is just $O \vee B$, and this is true in 3 of the 4 models

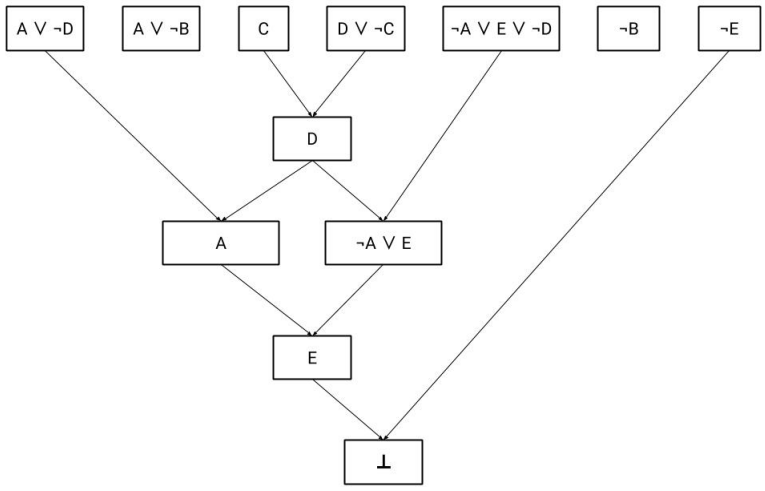
(c) Is the above sentence equivalent to a set of Horn clauses? Explain.

Yes. It is equivalent to $R \wedge C \neg O$ and $R \wedge C \implies \neg B$. In clause form these become $\neg R \vee \neg C \vee \neg O$ and $\neg R \vee \neg C \vee \neg B$. These clauses have zero positive literals and hence are Horn.

(d) Prove that the above sentence is *not* entailed by the sentence $RadioWorkds \implies \neg BatteryDead$.

To prove that A does not entail B, one simply has to provide a model where A is true and B is false. The model is $R, C, \neg B, O$.

3. From the knowledge base below, prove E.



The resolution algorithm involves iteratively applying the general resolution rule to derive new clauses which will (hopefully) lead to a contradiction.
The general resolution rule has the following format:

$a_1 \vee \cdots \vee a_m \vee b \; \neg b \vee c_1 \vee \ldots c_n \; a_1 \vee \cdots \vee a_m \vee c_1 \vee \ldots c_n$ We can show that this works by considering the unit clause case - resolving $(a \vee b) \wedge (\neg b \vee c)$. By the law of excluded middle, exactly one of the following cases must hold:

Case 1: b is true
In this case, c must be true for the clause $(\neg b \vee c)$ to evaluate to true.

Case 2: b is false
In this case, a must be true for the clause $(a \vee b)$ to evaluate to true.

Thus, a or c must be true for our original conjunction to be true.
We can inductively use this logic to show that the general resolution rule applies for arbitrary $m, n \in \mathbb{N}$.

A couple things to note:

- Applying the resolution rule isn't "reducing" or removing anything from our current knowledge base - rather, we're generating more clauses and adding them to our KB.
- It's not necessary to resolve all possible pairs of clauses to reach a contradiction. As we can see above, we only needed to derive $\neg P_{1,2}$ to reach a contradiction.
- We can only eliminate one pair of literals at a time. To demonstrate this, consider the following pair of clauses: $(A \vee P \vee Q) \wedge (B \vee \neg P \vee \neg Q)$. We can only resolve this pair to get $(A \vee P \vee B \vee \neg P)$ and $(A \vee Q \vee B \vee \neg Q)$ separately (which both end up evaluating to True).
  What we cannot do is derive $(A \vee B)$. Consider the model A = False, B = False, P = True, Q = False.

4. Indicate whether the following sentence is *valid*, *satisfiable*, or *unsatisfiable*. If satisfiable, give a model such that the sentence is satisfied. Prove your answer by reducing the sentence to its simplest form. Remember to **show all the steps and write down an explanation of each step.** Let T stand for the atomic sentence *True* and F for the atomic sentence *False*.
   $((T \Leftrightarrow \neg(x \vee \neg x)) \vee z) \wedge \neg(z \wedge ((z \wedge \neg z) \Rightarrow x))$

   Unsatisfiable

$((T \Leftrightarrow \neg(x \vee \neg x)) \vee z) \wedge \neg(z \wedge ((z \wedge \neg z) \Rightarrow x))$	
$((T \Leftrightarrow (\neg x \wedge x)) \vee z) \wedge \neg(z \wedge ((z \wedge \neg z) \Rightarrow x))$	De Morgan's Law
$((T \Leftrightarrow F) \vee z) \wedge \neg(z \wedge (F \Rightarrow x))$	$a \wedge \neg a$ is equivalent to False
$(((T \Rightarrow F) \wedge (F \Rightarrow T)) \vee z) \wedge \neg(z \wedge (F \Rightarrow x))$	Biconditional Elimination
$(((\neg T \vee F) \wedge (\neg F \vee T)) \vee z) \wedge \neg(z \wedge (\neg F \vee x))$	Implication Elimination
$(((F \vee F) \wedge (T \vee T)) \vee z) \wedge \neg(z \wedge (T \vee x))$	Negation
$((F \wedge T) \vee z) \wedge \neg(z \wedge T)$	$F \vee F$ is $F, T \vee T$ is T, $T \vee a$ is T
$(F \vee z) \wedge \neg(z \wedge T)$	$F \wedge T$ is False
$(F \vee z) \wedge \neg z \vee \neg T)$	De Morgan's Law
$(F \vee z) \wedge (\neg z \vee F))$	Negation
$z \wedge \neg z$	$F \vee a$ is a
F	$a \wedge \neg a$ is F

5. Indicate whether the following sentence is *valid*, *satisfiable*, or *unsatisfiable*. If satisfiable, give a model such that the sentence is satisfied. Prove your answer by reducing the sentence to its simplest form. Remember to **show all the steps and write down an explanation of each step.** Let T stand for the atomic sentence *True* and F for the atomic sentence *False*.
   $(\neg(x \vee \neg x) \wedge y) \vee ((x \vee (z \Rightarrow \neg z)) \wedge ((x \Rightarrow z) \vee \neg(F \Rightarrow T)))$

Satisfiable. {x: T, z: F}

$(\neg(x \vee \neg x) \wedge y) \vee ((x \vee (z \Rightarrow \neg z)) \wedge ((z \Rightarrow x) \vee \neg(F \Rightarrow T)))$	
$(\neg(T) \wedge y) \vee ((x \vee (z \Rightarrow \neg z)) \wedge ((z \Rightarrow x) \vee \neg(F \Rightarrow T)))$	$x \vee \neg x$ is True
$(F \wedge y) \vee ((x \vee (z \Rightarrow \neg z)) \wedge ((z \Rightarrow x) \vee \neg(F \Rightarrow T)))$	Negation
$F \vee ((x \vee (z \Rightarrow \neg z)) \wedge ((z \Rightarrow x) \vee \neg(F \Rightarrow T)))$	$F \wedge T$ is False
$(x \vee (z \Rightarrow \neg z)) \wedge ((z \Rightarrow x) \vee \neg(F \Rightarrow T))$	$F \vee a$ is a
$(x \vee (z \Rightarrow \neg z)) \wedge ((z \Rightarrow x) \vee \neg(T))$	$F \Rightarrow T$ is True
$(x \vee (z \Rightarrow \neg z)) \wedge ((z \Rightarrow x) \vee F)$	Negation
$(x \vee (z \Rightarrow \neg z)) \wedge (z \Rightarrow x)$	$a \vee F$ is a
$(x \vee (\neg z \vee \neg z)) \wedge (z \Rightarrow x)$	Implication Elimination
$(x \vee \neg z) \wedge (z \Rightarrow x)$	$a \vee a$ is a
$(x \vee \neg z) \wedge (\neg z \vee x)$	Implication Elimination
$x \vee \neg z$	$a \wedge a$ is a

8 Satisfiability and Planning

Up until now we have assumed that the plans we create always make sure that an actions preconditions are satisfied. Let us now investigate what propositional successor-state axioms such as $HaveArrow^{t+1} \iff (HaveArrow^t \wedge \neg Shoot^t)$ have to say about actions whose preconditions are not satisfied.

(a) First, let us consider what successor-state axioms are. How do they differ from action axioms, and why might we choose to use them?

Action axioms give us the conditions needed at the current state for an action to be carried out at the current state. Successor state axioms are axioms outlining what preconditions in the current state need to be true in order to ensure that the state at the next timestep will be as specified. By definition, it is an axiom that sets the truth value of F^{t+1} (where F is some fluent, or changeable variable in an environment) in one of two ways:

- The action at time t causes F to be true at $t+1$ (which refers to $ActionCausesF^t$)
- F was already true at time t and the action at time t does not cause it to be false.

It has the following schema: $F^{t+1} \iff ActionCausesF^t \vee (F^t \wedge \neg ActionCausesNotF^t)$.

We use successor state axioms to ensure that each state we compute is the result of legal action. Also, writing only in terms of action axioms can become overwhelming, if there are a lot of time-dependent variables to keep track of. Successor state axioms are what help us do this.

(b) Show that the axioms predict that nothing will happen when an action is executed in a state where its preconditions are not satisfied.

We can illustrate the basic idea using the axiom given. Suppose that $Shoot^t$ is true but $HaveArrow^t$ is false. Then the RHS of the axiom is false, so $HaveArrow^{t+1}$ is false, as we would hope. More generally, if an action precondition is violated, then both $ActionCausesF^t$ and $ActionCausesNotF^t$ are false, so the generic successor-state axiom reduces to $F^{t+1} \iff False \vee (F^t \wedge True)$ which is the same as saying $F^{t+1} \iff F^t$, i.e., nothing happens.

(c) Consider a plan p that contains the actions required to achieve a goal but also includes illegal actions. Is it the case that successor-state axiom will allow the actions?

We recommend that you write a truth table and ask yourself the following question when looking at the truth table:

- Can I shoot if I don't have an arrow?

Yes, the plan plus the axioms will allow the actions and the axiom to evaluate to true; the axioms will copy every fluent across an illegal action and the rest of the plan will still work. Note that goal entailment is trivially achieved if we add precondition axioms, because then the plan is logically inconsistent with the axioms and every sentence is entailed by a contradiction. Precondition axioms are a way to prevent illegal actions in satisfiability-based planning methods (this will become clearer once we go over First Order Logic and Planning lectures). Refer to the truth table below where $A1$ refers to $HaveArrow^1$, $A2$ refers to $HaveArrow^2$ and $S1$ refers to $Shoot^1$.

A1	S1	A2	$A1 \wedge \neg S1$	$A2 \iff A1 \wedge \neg S1$	Reference
F	T	F	F	T	1
F	T	T	F	F	
T	T	F	F	T	
T	T	T	F	F	

Here, 1 refer to "Can I shoot if I don't have an arrow?"

Suppose we are tasked with making a plan to deliver N-95 masks around the U.S.
We use the following propositions below in a GraphPlan approach:

- $at(loc)$: our cargo plane is at location loc
- $fuel(x)$: the fuel level is at x , $x \in [0, 5]$.
- $hasFuel(loc)$: location loc has fuel (to re-fuel the plane with)
- $hasMasks(loc)$: location loc has masks

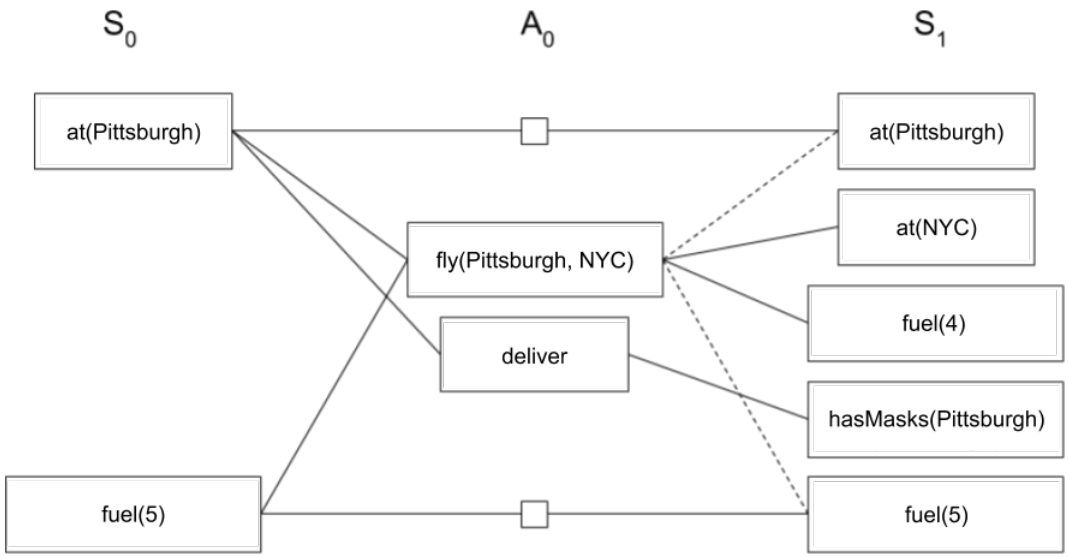
Our starting state is $at(Pittsburgh) \wedge fuel(5)$.

1. Define each action as an operator in the following table (note that we can drop the t parameter from each predicate and action):

	$refuel$	$fly(o, d)$	$deliver$
Precondition			
Add			
Delete			

	$refuel$	$fly(o, d)$	$deliver$
Precondition	$at(loc)$, $hasFuel(loc)$, $fuel(x)$	$at(o)$, $fuel(x)$, $x > 0$, $o \neq d$	$at(loc)$
Add	$fuel(5)$	$fuel(x - 1)$, $at(d)$	$hasMasks(loc)$
Delete	$fuel(x)$	$fuel(x)$, $at(o)$	

2. Now draw the GraphPlan graph up to proposition level S_1. Suppose NYC is the only other location besides Pittsburgh.



3. Which operators are mutually exclusive in A_0? Which propositions are mutually exclusive in S_1?

   noop/persist on both propositions (at(Pittsburgh) and fuel(5)) and fly(Pittsburgh, NYC) have an inconsistent effects mutex relation. fly(Pittsburgh, NYC) and deliver have an interference mutex relation. at(NYC) and at(Pittsburgh); fuel(4) and fuel(5); fuel(4) and at(Pittsburgh); fuel(5) and at(NYC) are mutex since all possible pairs of actions leading to these propositions are mutex.

4. In general, when does GraphPlan stop extending the graph?

   GraphPlan stops extending the graph once it reaches a proposition level where all goal propositions are present and a solution is found, or when the graph levels off, i.e., two consecutive proposition levels are identical.

5. Is GraphPlan sound? complete? optimal (with respect to the number of actions in the plan returned)?

   GraphPlan is sound, complete, but not optimal.

9 Probability

1. Write all the possible chain rule expansions of the joint probability $P(a, b, c)$. No conditional independence assumptions are made.

   $P(a)P(b|a)P(c|a, b), P(a)P(c|a)P(b|a, c),$
   $P(b)P(a|b)P(c|a, b), P(b)P(c|b)P(a|b, c),$
   $P(c)P(b|c)P(a|b, c), P(c)P(a|c)P(b|a, c)$

2. Determine if the following statements are true or false. No independence assumptions are made.

   (a) $P(A, B) = P(A|B)P(A)$

   False. This is not a valid application of the product rule. $P(A, B) = P(A|B)P(B)$ would be a valid example.

   (b) $P(A|B)P(C|B) = P(A, C|B)$

   False. This assumes that A and C are conditionally independent given B

   (c) $P(B, C) = \sum_{a \in A} P(B, C|A)$

   False. $P(B, C) = \sum_{a \in A} P(A, B, C)$ would be a valid example.

   (d) $P(A, B, C, D) = P(C)P(D|C)P(A|C, D)P(B|A, C, D)$

   True. This is a valid application of the chain rule.

3. Let A be a random variable representing the choice of protein in the sandwich with three possible values, $\{mutton, bacon, egg\}$, let B be a random variable representing the choice of bread with two possible values, $\{toast, naan\}$, and let K be a random variable representing the presence of ketchup or not, $\{+k, k\}$.

   How many values are in each of the probability tables and what do the entries sum to?
   Write **'?'** if there is not enough information given.

Table	num	sum
$P(A, B)$		
$P(A, B, +k)$		
$P(A, B \mid +k)$		
$P(B \mid +k, A)$		

Table	num	sum
$P(A, B)$	6	1
$P(A, B, +k)$	6	?
$P(A, B \mid +k)$	6	1
$P(B \mid +k, A)$	6	3

   $P(A, B)$ sums to 1 through marginalizing out both A and B ($\sum_a \sum_b P(A = a, B = b) = \sum_a P(A = a) = 1$).
   We do not know $P(A, B, +k)$ because the resulting sum is dependent on $+k$ ($\sum_a \sum_b P(A = a, B = b, +k) = P(+k)$).
   $P(A, B| + k)$ sums to 1 through marginalizing out both A and B ($\sum_a \sum_b P(A = a, B = b| + k) = \sum_a P(A = a| + k) = 1$).
   $P(B| + k, A)$ sums to 3 through casing on values of A and marginalizing out B ($\sum_a \sum_b P(B = b, | + k, A = a) = \sum_b P(B = b, | + k, A = mutton) + \sum_b P(B = b, | + k, A = bacon) + \sum_b P(B = b, | + k, A = egg) = 1 + 1 + 1 = 3$).

4. Consider the following probability tables:

A	$P(A)$
+a	0.8
−a	0.2

A	B	$P(B\|A)$
+a	+b	0.9
+a	−b	0.1
−a	+b	0.6
−a	−b	0.4

B	C	$P(C\|B)$
+b	+c	0.8
+b	−c	0.2
−b	+c	0.8
−b	−c	0.2

C	D	$P(D\|C)$
+c	+d	0.25
+c	−d	0.75
−c	+d	0.5
−c	−d	0.5

(a) Using the table above and the assumptions per subquestion, calculate the following probabilities given no independence assumptions. If it is impossible to calculate without more independence assumptions, specify the least number of independence assumptions that would allow you to answer the question (don't do any computation in this case).

(i) $P(+a, -b)$

.8 * .1 = .08

(ii) $P(-a, -b, +c)$

Given that C is independent of A given B

(iii) Now assume C is independent of A given B and D is independent of everything else given C. Calculate $P(+a, -b, +c, +d)$ or say what other independence assumptions are necessary.

.8*.1*.8*.25 = .016

(b) Which of the following expressions indicate that X is independent of Y given Z?

(i) $P(X, Y|Z) = P(X|Z)P(Y|Z)$
(ii) $P(X|Y, Z) = P(X|Z)$
(iii) $P(X, Y, Z) = P(X, Z)P(Y)$
(iv) None of the above

(i), (ii)

(c) Which of the following expressions are equal to $P(R, S, T)$ given no independence assumptions?

(i) $P(R|S, T)P(S|T)P(T)$
(ii) $P(T, S|R)P(R)$
(iii) $P(T|R, S)P(R)P(S)$
(iv) $P(T|R, S)P(R, S)$
(v) $P(R|S)P(S|T)P(T)$
(vi) $P(R|S, T)P(S|R, T)P(T|R, S)$
(vii) None of the above

(i),(ii),(iv)

10 MDPs/RL

1. Warm Up

   - What does the Markov Property state?

     Markov Property states that action outcomes depend on the current state only. It states that action outcomes do not depend on the past.

   - What are the Bellman Equations, and when are they used?

     The Bellman Equations give a definition of "optimal utility" via expectimax recurrence. They give a simple one-step lookahead relationship amongst optimal utility values.

   - What is a policy? What is an optimal policy?

     A policy is a function that maps states to actions. $\pi(s)$ gives an action for state s. An optimal policy is a policy that maximizes the expected utility if an agent follows it.

   - How does the discount factor γ affect the agent's policy search? Why is it important?

     γ determines how much the value of a state should take into account future states. The higher the discount factor, the more one state would value distant states. Having $0 < \gamma < 1$ also helps our algorithms converge.

   - What are the two steps to Policy Iteration?

     Policy evaluation and policy improvement.

   - What is the relationship between $V^*(s)$ and $Q(s, a)$?

     $V^*(s) = \max_a Q(s, a)$

   - Exploration, exploitation, and the difference between them? Why are they both useful?

     Exploration: trying out unknown actions; Exploitation: Following the known policy.
     Exploration allows the agent to see if there are any other actions that lead to a better reward by taking random actions. Exploitation guarantees that the agents get some reward at least.

   - What is the difference between on-policy and off-policy learning?

     For on-policy learning, it attempts to evaluate and improve the policy that is being used to make decisions. For off-policy learning, it attempts to evaluate or improve a policy different from the one that is being used to generate the data.

   - What is the difference between model-based and model-free learning?

     For model-based learning, the agent learns an approximate model based on experiences and solves for values as if the learned model were correct. A model-free learning is an algorithm which does not use the transition probability distribution.

   - We are given a pre-existing table of Q-values (and its corresponding policy), and asked to perform ϵ-greedy Q-learning. Individually, what effect does setting each of the following constants to 0 have on this process?

     (i) α:

     α is the the learning rate. It determines by how much the q-values should change each iteration given the new information found. The smaller α, the slower the policy will approach a solution, but the more accurate the solution would be; thus,
     $Q(s, a) = Q(s, a) + \alpha[r + \gamma max_{a'} Q(s', a') - Q(s, a)]$ becomes $Q(s, a)$.
     We put 0 weight on newly observed samples, never updating the Q-values we already have.

(iii) ϵ:

By definition of an ϵ-greedy policy, we randomly select actions with probability 0 and select our policy's recommended action with probability 1; we exclusively exploit the policy we already have.

- For each of the following functions, write which MDP/RL value the function computes, or none if none apply. We are given an MDP (S, A, T, γ, R), where R is only a function of the current state s. We are also given an arbitrary policy π.

  Possible choices: $V^*, Q^*, \pi^*, V^\pi, Q^\pi$.

  (i) $f(s) = R(s) + \sum_{s'} \gamma T(s, \pi(s), s') f(s')$

  $f = V^\pi$. This is only different from the given formula for $V^\pi(s)$ on the formula sheet in that the reward function only depends on s here. Thus, we consider $R(s)$ outside the summation over s' - and do not discount it (because the reward is wrt. our current state).

  (ii) $g(s) = \max_a \sum_{s'} T(s, a, s')[R(s) + \gamma \max_{a'} Q^*(s', a')]$

  $g = V^*$. What this function does is essentially extract optimal values from optimal Q-values. Of our possible actions, we take the actions that yields the max sum of reward + future discounted rewards given by Q^*, summed over all possible successor states (each weighted by the successor's probability).

2. MDPs - Micro-Blackjack: In micro-blackjack, you repeatedly draw a card (with replacement) that is equally likely to be a 2, 3, or 4. You can either Draw or Stop if the total score of the cards you have drawn is less than 6. Otherwise, you must Stop. When you Stop, your utility is equal to your total score (up to 5), or zero if you get a total of 6 or higher. When you Draw, you receive no utility. There is no discount ($\gamma = 1$).

   (a) What are the states and the actions for this MDP?

   The state is the current sum of your cards, plus a terminal state:

   $$0, 2, 3, 4, 5, Done$$

   The actions are $\{Draw, Stop\}$.

   (b) What is the transition function and the reward function for this MDP?

   The transition function is

   $$T(s, Stop, Done) = 1$$

   $$T(s, Draw, s') = \begin{cases} 1/3 \text{ if } s' - s \in \{2, 3, 4\} \\ 1/3 \text{ if } s = 2 \text{ and } s' = Done \\ 2/3 \text{ if } s = 3 \text{ and } s' = Done \\ 1 \text{ if } s \in \{4, 5\} \text{ and } s' = Done \\ 0 \text{ otherwise} \end{cases}$$

   The reward function is

   $$R(s, Stop, s') = s, s \leq 5$$

   $$R(s, a, s') = 0 \text{ otherwise}$$

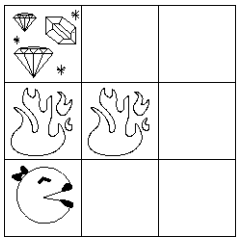
(c) Fill out the value iteration table below. We have filled out the first row for you. (Recall that we always initialize $V_0(s)$ to 0 for all states s.) Then, perform policy extraction and give the optimal policy for this MDP.

V	0	2	3	4	5	Done
V_0	0	0	0	0	0	0
V_1						
V_2						
V_3						
Policy Extraction						

V	0	2	3	4	5	Done
V_0	0	0	0	0	0	0
V_1	0	2	3	4	5	0
V_2	3	3	3	4	5	0
V_3	10/3	3	3	4	5	0
Policy Extraction	$10/3_{Draw}$	3_{Draw}	3_{Stop}	4_{Stop}	5_{Stop}	0_{Stop}

The optimal policy is **Draw if $s \leq 2$, Stop otherwise.** The optimal policy is given by taking the *argmax* instead of *max*, after the final iteration of value iteration.

3. Ms.Pacman: While Pacman is busting ghosts, Ms. Pacman goes treasure hunting on GridWorld Island. She has a map showing where the hazards are, and where the treasure is. From any unmarked square, Ms. Pacman can take any of the deterministic actions (N, S, E, W) that doesn't lead off the island. If she lands in a hazard square or a treasure square, her only action is to call for an airlift (X), which takes her to the terminal *Done* state; this results in a reward of -64 if she's escaping a hazard, or +128 if she reached the treasure. There is no living reward.



(a) Let $\gamma = 0.5$. What are the optimal values V^* of each state in the grid above?

128	64	32
-64	-64	16
2	4	8

(b) Assuming you have V*, how would we compute the Q-values for each state-action pair?

Bellman equation without taking the max: $Q(s,a) = \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s)]$.

(c) What's the optimal policy? You may use the grid below to fill in the optimal action for each state.

X	W	W
X	X	N
E	E	N

Call this policy π_0.

Ms. Pacman realizes that her map might be out of date, so she uses Q-learning to see what the island is really like. She believes π_0 is close to correct, so she follows an ϵ-random policy, ie., with probability ϵ she picks a legal action uniformly at random (otherwise, she does what π_0 recommends). Call this policy π_ϵ.

π_ϵ is known as a *stochastic* policy, which assigns probabilities to actions rather than recommending a single one. A stochastic policy can be defined with $\pi(s, a)$, the probability of taking action a when the agent is in state s.

(d) Write a modified Bellman update equation for policy evaluation when using a stochastic policy $\pi(s, a)$.

We'll keep most of the original evaluation formula, but additionally sum over all possible actions recommended by the policy, each weighted by the probability of taking that action via the policy:

$$V^{\pi}_{k+1}(s) = \sum_{a} \pi(s, a) \sum_{s'} P(s' \mid s, a)[R(s, a, s') + \gamma V^{\pi}_{k}(s')]$$

11 Bayes' Nets: Representation, Independence

1. For this problem, any answers that require division can be left written as a fraction.
PacLabs has just created a new type of mini power pellet that is small enough for Pacman to carry around with him when he's running around mazes. Unfortunately, these mini-pellets don't guarantee that Pacman will win all his fights with ghosts, and they look just like the regular dots Pacman carried around to snack on.
Pacman (P) just ate a snack , which was either a mini-pellet ($+p$), or a regular dot ($-p$), and is about to get into a fight (W), which he can win ($+w$) or lose ($-w$). Both these variables are unknown, but fortunately, Pacman is a master of probability. He knows that his bag of snacks has 5 mini-pellets and 15 regular dots. He also knows that if he ate a mini-pellet, he has a 70% chance of winning, but if he ate a regular dot, he only has a 20% chance.

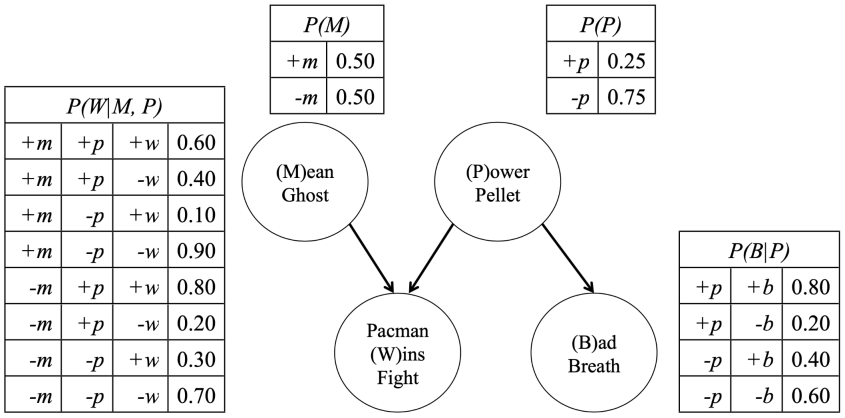
   (a) What is $P(+w)$, the marginal probability that Pacman will win?

$$\begin{aligned} P(+w) &= \sum_p P(+w|p)P(p) \\ &= \frac{7}{10} \times \frac{1}{4} + \frac{2}{10} \times \frac{3}{4} = \frac{13}{40} = 0.325 \end{aligned}$$

   (b) Pacman won! Hooray! What is the conditional probability $P(+p \mid +w)$ that the food he ate was a mini-pellet, given that he won?

$$\begin{aligned} P(+p|+w) &= \frac{P(+w,+p)}{P(+w)} = \frac{P(+w|+p)P(+p)}{P(+w)} \\ &= \frac{\frac{7}{10} \times \frac{1}{4}}{\frac{13}{40}} = \frac{7}{13} \approx 0.538 \end{aligned}$$

   Pacman can make better probability estimates if he takes more information into account. First, Pacman's breath, B, can be bad ($+b$) or fresh ($-b$). Second, there are two types of ghost (M): mean ($+m$) and nice ($-m$). Pacman has encoded his knowledge about the situation in the following Bayes' Net:



(c) What is the probability of the event $(-m, +p, +w, -b)$, where Pacman eats a mini-pellet and has fresh breath before winning a fight against a nice ghost?

$P(-m, +p, +w, -b) = P(-m)P(+p)P(+w| - m, +p)P(-b| + p) = \frac{1}{2} \times \frac{1}{4} \times \frac{4}{5} \times \frac{1}{5} = \frac{1}{50} = 0.02$

For the remaining of this question, use the half of the joint probability table that has been computed for you below:

$P(M, P, W, B)$				
$+m$	$+p$	$+w$	$+b$	0.0800
$+m$	$+p$	$+w$	$-b$	0.0150
$+m$	$+p$	$-w$	$+b$	0.0400
$+m$	$+p$	$-w$	$-b$	0.0100
$+m$	$-p$	$+w$	$+b$	0.0150
$+m$	$-p$	$+w$	$-b$	0.0225
$+m$	$-p$	$-w$	$+b$	0.1350
$+m$	$-p$	$-w$	$-b$	0.2025

(d) What is the marginal probability, $P(+m, +b)$ that Pacman encounters a mean ghost and has bad breath?

$P(+m, +b) = 0.08 + 0.04 + 0.015 + 0.135 = 0.27$

(e) Pacman observes that he has bad breath and that the ghost he's facing is mean. What is the conditional probability, $P(+w \mid +m, +b)$, that he will win the fight, given his observations?

$P(+w| + m, +b) = \frac{P(+w,+m,+b)}{P(+m,+b)} = \frac{0.08+0.015}{0.27} = \frac{19}{54} \approx 0.352$

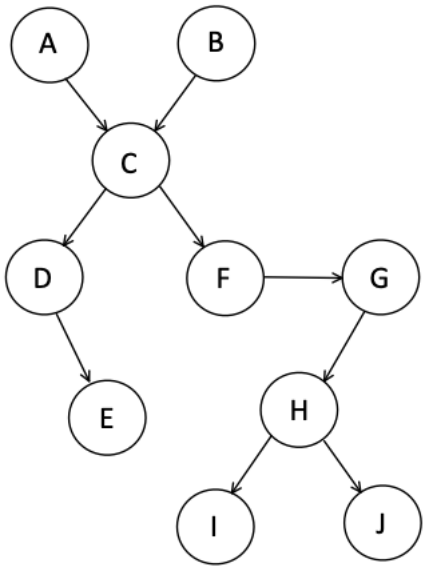
(f) Pacman's utility is +10 for winning a fight, -5 for losing a fight, and -1 for running away from a fight. Pacman wants to maximize his expected utility. Given that he has bad breath and is facing a mean ghost, should he stay and fight, or run away? Justify your answer.

Let U_f be the utility of fighting and U_r be the utility of running.

$$\begin{aligned} E(U_f| + m, +b) &= 10 \times P(+w| + m, +b) + (-5) \times P(-w| + m, +b) \\ &\approx 10 \times 0.352 - 5 \times 0.648 \\ &= 0.28 > -1 = U_r \end{aligned}$$

Since $E(U_f| + m, +b) > E(U_r| + m, +b)$, Pacman should stay and fight.

2. Consider the following Bayes Net:



Determine whether the following statements hold given this Bayes Net

(a) $P(A, B) = P(A)P(B)$

True. This statement is true if $A \perp\!\!\!\perp B$. The only path from A to B is through C. For the triple ACB, the common effect C is unobserved (and none of its descendants are observed) so this triple is inactive. Therefore, the path ACB is inactive.

(b) $A \perp\!\!\!\perp E|C$

True. The only path from A to E is ACDE. Because C is observed in the causal chain triple ACD, this triple is inactive. Therefore, the path ACDE is inactive and thus $A \perp\!\!\!\perp E|C$

(c) $D \perp\!\!\!\perp J|I$

False. The only path from D to J is DCFGHJ. For the triple ACF, the common cause C is unobserved, so this triple is active. Every other triple on this path is a causal chain with none of the variables being observed, so they are all active. Since every triple on this path is active, that means the path is active as well.

(d) $A \perp\!\!\!\perp B|H$

False. The only path from A to B is through C. For the triple ACB, the common effect C is unobserved, but its descendant H is observed, so this triple is active. Since all triples are active, the path ACB is active.

(e) $P(E|C) = P(E|C, G)$

True. This statement is true if $E \perp\!\!\!\perp G|C$. The only path from E to G is EDCFG. The triple EDC is a causal chain with none of the variables being observed, so it is an active triple. The triple DCF is a common cause triple with C observed, so this triple is inactive. Therefore, the path EDCFG is inactive as well.

(f) $I \perp\!\!\!\perp J|G$

False. The only path from I to J is IHJ. For the triple IHJ, the common cause H is unobserved, so this triple is active. Since all triples along this path are active, that means the path is active as well.

(g) $P(B|F) = P(B|E, F)$

False. This statement is true if $B \perp\!\!\!\perp E|F$. If we consider the path BCDE, all of the triples along this path are causal chain triples with none of the variables observed, so they are all active. Since all triples along this path are active, that means the path must be active as well.