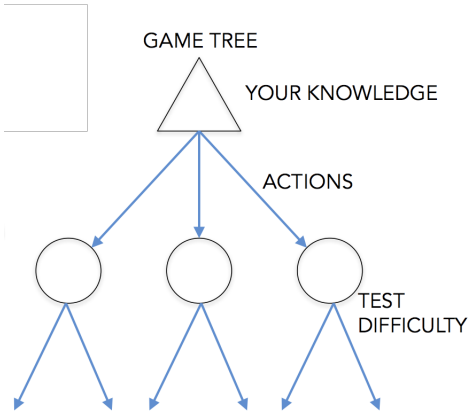
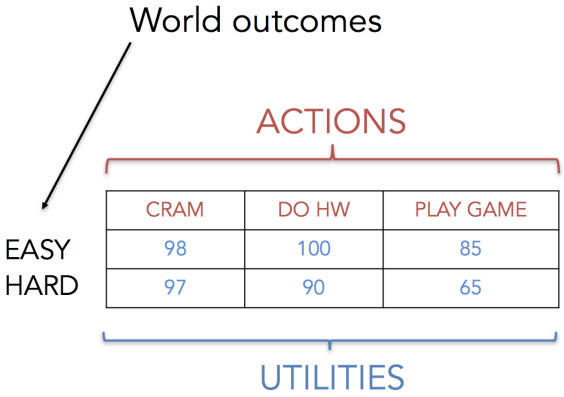
1 Game Theory Search

(a) What are the differences between extensive form and normal form games?

An extensive form game is one where every player is represented, as well as every opportunity that each of those players has to make a move, what moves they can make at that time, what information they have at the time, and the possible payoffs to be received.



On the other hand, a normal form game is one where games are approximated as a single shot. It represents only actions and utilities. This is the table format that we're used to using at this point.



(b) What is a strategy? What is the difference between a pure and a mixed strategy?

A strategy is a probability distribution over actions. A pure strategy is one that is deterministic (e.g. chosen with probability 1), whereas a mixed strategy is one that uses randomized selection.

(c) We can practice computing utilities using this example that was given in lecture:

CRAM	DO HW	PLAY GAME	
98	100	85	P(EASY) = .2
97	90	65	P(HARD) = .8

- What is the utility of the pure strategy: cram?

  98 * 0.2 + 97 * 0.8 = 97.2

- What is the utility of the pure strategy: do HW?

  100 * 0.2 + 90 * 0.8 = 92

- What is the utility of the mixed strategy: $\frac{1}{2}$ cram, $\frac{1}{2}$ do HW?

  $\frac{1}{2} * 97.2 + \frac{1}{2} * 92 = 94.6$

(d) What is a Nash Equilibrium?

A Nash Equilibrium is a strategy profile where none of the participants benefit from unilaterally changing their decision (in other words, their utility would strictly decrease by doing so).

There exist pure and mixed Nash Equilibriums (a pure Nash Equilibrium is built on a pure strategy).

(e) Does a Nash Equilibrium always exist?

If there are a finite number of players and a finite number of actions, then there always exists a Nash Equilibrium (Nash, 1950). This strategy can be either pure or mixed (consider Rock-Paper-Scissors as an example of a game that doesn't have a pure strategy but does have a mixed one).

(f) Consider a two player game, where each player must simultaneously choose a number from $\{2, 3, ..., 99, 100\}$. Let x_1 represent the value chosen by player 1, and x_2 represent the value chosen by player 2. The rules of the game are such that the utility for a player 1 can be given as:

$$u(p_1) = \begin{cases} x_1 & x_1 = x_2 \\ x_2 - 2 & x_1 > x_2 \\ x_2 + 2 & x_1 < x_2 \end{cases}$$

Because the rules of the game for everyone are the same, the utility function for player 2 is symmetric to $u(p1)$. Does there exist a pure Nash Equilibrium for this game? It may help to try to play a few rounds of this game with someone next to you.

Yes. If $x_1 = x_2 = 2$, then neither player has any incentive to unilaterally change their strategy. There is no option to choose a lower number, and any player who chooses to change their number to anything greater than 2 will then receive a utility of 0, which is strictly worse than their current utility of 2.

(g) What is a Correlated Equilibrium? What is its relationship to a Nash Equilibrium?

A correlated equilibrium is a distribution over action profiles $\vec{a}$ such that after a profile $\vec{a}$ is selected, playing $\vec{a}$ is a best response for player i conditioned on seeing $\vec{a}$, given that everyone else will play according to $\vec{a}$.

In other words, suppose a mediator computes the best joint strategy for p_1 and p_2, and shares a selected a_1 with p_1 and a_2, with p_2. Then, neither p_1 nor p_2 has incentive to unilaterally switch their strategy.

Every Nash Equilibrium is also a Correlated Equilibrium.

(h) Let us consider several voting strategies:

- Plurality

Each voter gets one vote for their top-ranked preference. Alternative with the most votes wins.

- Borda Count

  Each voter awards $m - k$ points to their rank k. Alternative with the most votes wins.

- Single Transferable Vote

  Each voter gets 1 vote per round. In each round, the alternative with the least number of plurality votes is eliminated. Alternative left standing is winner

- Pairwise Elections

  Alternative x beats y in pairwise election if majority of voters prefer x to y.

- Plurality with Runoff

  First Round: Top 2 plurality winners advance to second round. Second Round: Pairwise election between two winners.

- Condorcet Winner

  Alternative x beats y in pairwise election if majority of voters prefer x to y. Condorcet winner x beats every other alternative y in pairwise election. Condorcet paradox is a cycle in majority preferences.

(i) Which voting rule (Plurality, Borda Count, Both, Neither) is Condorcet consistent? Consider the following voting example:

3 voters	2 voters
a	b
b	c
c	a

3 voters	2 voters	2 voters
a	b	c
b	c	b
c	a	a

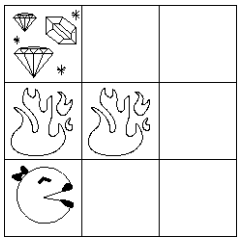
Neither! In the first case, the plurality vote is A; borda count is B; condorcet is A. In the second case, the plurality vote is A; the borda count is B; and the condorcet is B.

(j) Which voting rule is the best?

Hard to say! [Mao, Procaccia, Chen 2013] Compared ranking strategies Plurality, Borda, Condorcet. Found that Borda finds the winners most consistently even with noisy human responses, and Plurality performs the worst.

2 RL: Treasure Hunting

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) How would we compute the Q-values for each state-action pair?

Run Q-value iteration (Q-iteration on the MDP/RL notation sheet) until convergence.

(c) What's the optimal policy?

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 $\pi(s, a)$ (this is similar to a problem seen on midterm 2).

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:

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

(e) If the original map and our assumptions about the transitions and rewards are correct, what relationship will hold for all states s?

$$V^{\pi_0}(s) \leq V^{\pi_\epsilon}(s) \qquad V^{\pi_0}(s) = V^{\pi_\epsilon}(s) \qquad V^{\pi_0}(s) \geq V^{\pi_\epsilon}(s)$$

$V^{\pi_0}(s) \geq V^{\pi_\epsilon}(s)$, because π_0 would be optimal if our map is correct.

As it turns out, Ms. Pacman's map is mostly correct, but some of the fire pits seem to have fizzled out. She observes the following episodes during her Q-learning:
{(0, 0), N, 0}, {(0, 1), N, 0}, {(0, 2), X, 128}, Done
{(0, 0), N, 0}, {(0, 1), N, 0}, {(0, 2), X, 128}, Done
{(0, 0), N, 0}, {(0, 1), E, 0}, {(1, 1), X, -64}, Done

(f) What are her Q-values after observing these episodes? Assume Ms. Pacman initialized her Q-values all to 0 and used a learning rate of 0.1. You may omit any Q-states that were unaffected.

$Q((0, 2), X) = 24.32$
$Q((0, 1), N) = 0.64$
$Q((0, 0), N) = 0.032$
$Q((1, 1), X) = -6.4$

Now let's review a couple problems we've seen before.

(g) 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).

(iii) $h(s, a) = \sum_{s'} T(s, \pi(s), s')[R(s) + \gamma h(s', a)]$

None. Looking at the types, we can first rule out any functions computing V because h takes an action as a parameter. This also wouldn't be a policy because policy functions require taking an argmax over a.
Looking at what h is trying to compute, we are summing over successor states (presumably of s, after taking the action dictated by policy π) and computing the reward in state s + future discounted rewards. The last term $h(s', a)$ (which, in the usual Bellman equations, computes the discounted future rewards), doesn't make sense because we're passing into the recursive call our current action with the successor state.

(h) 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:

$Q(s, a) = Q(s, a) + \alpha[r + \gamma max_{a'} Q(s', a') - Q(s, a)]$ becomes $Q(s, a)$.
We put 0 weight on newly observed samples, never updating the Q-values we already have.

(ii) γ:

$Q(s, a) = Q(s, a) + \alpha[r + \gamma max_{a'} Q(s', a') - Q(s, a)]$ becomes $(1 - \alpha)Q(s, a) + \alpha r$.
Our valuation of reward becomes short-sighted, as we weight Q-values of successor states with 0. It is interesting to note setting $\alpha = 0$ has a similar effect.

(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.

(i) Why can't we use the MDP policy extraction formula to extract a policy in TD-learning or Q-learning?

We TD- or Q-learn in the context of online learning, meaning we don't have access to the transition or reward functions.

3 CSP Backtracking Search

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.

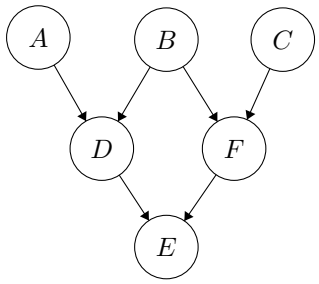
- Give initial feasible values in set form; x_1 has already been filled out for you.
- Assignment order refers to the order in which the final value assignments are given. If x_i is the j^{th} variable on the path to the goal state, then the assignment order for x_i is j.
- In the branching column, write "yes" if the algorithm branches (considers more than one value) at that node in the search tree, 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	$\{5, 6, 7, 8, 9, 10\}$	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

4 Bayes Nets

(a) For the following graphs, explicitly state the minimum size set of edges that must be removed such that the corresponding independence relations are guaranteed to be true.
Mark the removed edges with an 'X' on the graphs.
Note: this question relies on a method using *d-separation*, which is out of scope for this class. If you're curious, you can read about d-separation here. For that reason, this problem would be out of scope for the final exam. However, you are still responsible for understanding the conditional independence relationships outlined in lecture and practiced in homework assignments.

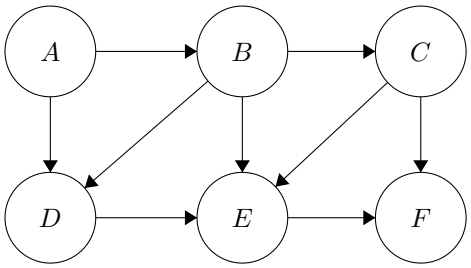
(i)



$A \perp\!\!\!\perp B \mid F$
$A \perp\!\!\!\perp F \mid D$
$B \perp\!\!\!\perp C$

1 edge: AD, BD, or BF.
$A \perp\!\!\!\perp B \mid F$ and $B \perp\!\!\!\perp C$ already hold in the original BN. B and C are (unconditionally) independent because they are both parents (not of each other). A and B are also unconditionally independent of one another because of this same reason, and since F is not their shared child, they are still independent even given the value of F.
$A \perp\!\!\!\perp F \mid D$ does not initially hold, because knowing the value of D allows us to make inferences about B based on A (makes A and B conditionally dependent), which further allows us to make inferences about F.
Thus we need to eliminate any edge(s) which sever all paths of inference from A to F that goes through D. This can done by removing any one of the above edges.

(ii)



$A \perp\!\!\!\perp D \mid B$
$A \perp\!\!\!\perp F \mid C$
$C \perp\!\!\!\perp D \mid B$

2 edges: AD and (EF or AB).
In the original BN, A and D are not independent, even given B, since D is a child of A.
Similarly, A and F are not independent because F is a descendant of A (even given C, since the paths of inference $ADEF, ABDEF, ABEF$ still exist).
Since C and D are children of the same parent, knowing the value of that parent (B) leaves them independent (see common cause slide in Lecture 19) - so $C \perp\!\!\!\perp D \mid B$ already holds.
Thus to satisfy all 3 relationships, we need to cut the edge between A and D to remove their parent-child relationship, and we need to sever all paths of inference between A and F. The most efficient way to do this is by removing EF or AB (since we've already removed AD).

(b) You're performing variable elimination over a Bayes Net with variables A, B, C, D, E. So far, you've finished joining over (multiplying all factors containing C into one factor), but not summing out C, when you realize you've lost the original Bayes Net!

Your current factors are $f(A), f(B), f(B, D), f(A, B, C, D, E)$. Note: these are generic factors, NOT joint distributions. You don't know which variables are conditioned or unconditioned.

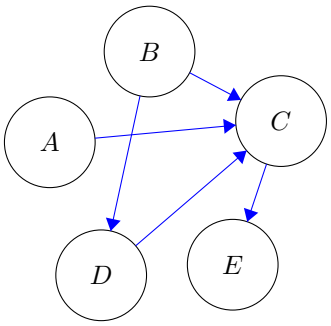
(i) What's the smallest number of edges that could have been in the original Bayes Net? Draw out one such Bayes Net below.

5 edges.
The original Bayes net must have had 5 factors, 1 for each node.
$f(A)$ and $f(B)$ must have corresponded to nodes A and B, and indicate that neither A nor B have any parents (otherwise A's parent would appear in A's factor, and likewise for B).

$f(B, D)$, then, must correspond to node D, and indicates that D has only B as a parent.

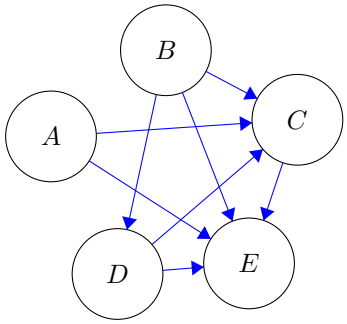
Since there is only one factor left, $f(A, B, C, D, E)$, for the nodes C and E, those two nodes must have been joined while you were joining C. From this we can infer two things:
1) E must have had C as a parent. (If E didn't have C as a parent, then there would be a separate factor containing E but not C.)
2) If A, B, D all appear in this big factor containing C and E, they must have either been in a probability table of the form $P(C \mid \dots)$ or $P(C \mid E, \dots)$. Thus each of A, B, and D must have been a parent of either C or E.

The below figure is one possible solution that uses the fewest possible edges to satisfy the above.



(c) What's the largest number of edges that could have been in the original Bayes Net? Draw out one such Bayes Net below.

8 edges.
The constraints are the same as outlined in part b.i. To maximize the number of edges, we make each of A, B, and D a parent of both C and E, as opposed to a parent of one of them. The below figure is the only possible solution.



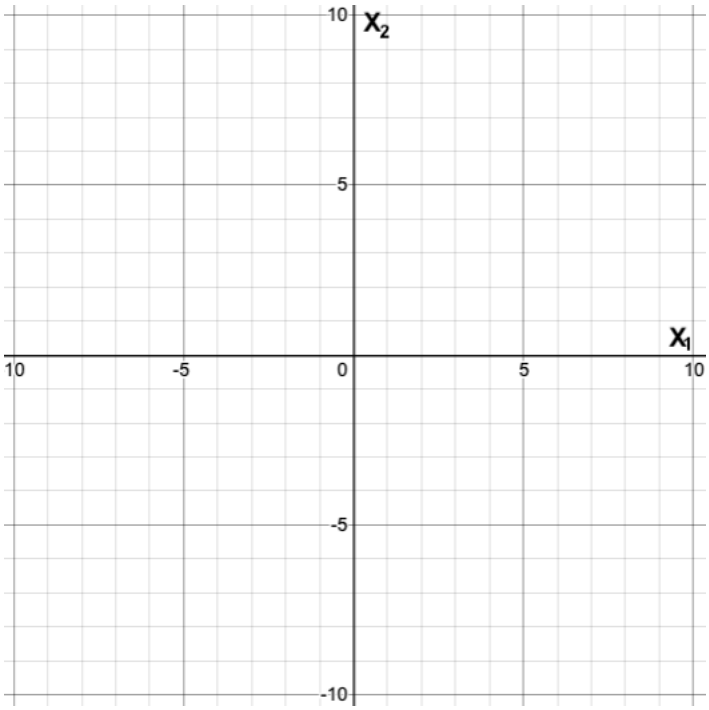
5 Linear Programming & Integer Programming

(a) In the following optimization problem, plot the boundary lines for the three inequality constraints.

$$\min_x c^T x$$

$$\text{s.t. } Ax \leq b$$

$$A = \begin{bmatrix} 1 & 4 \\ 2 & -1 \\ -4 & -1 \end{bmatrix}, b = \begin{bmatrix} 5 \\ 8 \\ 8 \end{bmatrix}$$

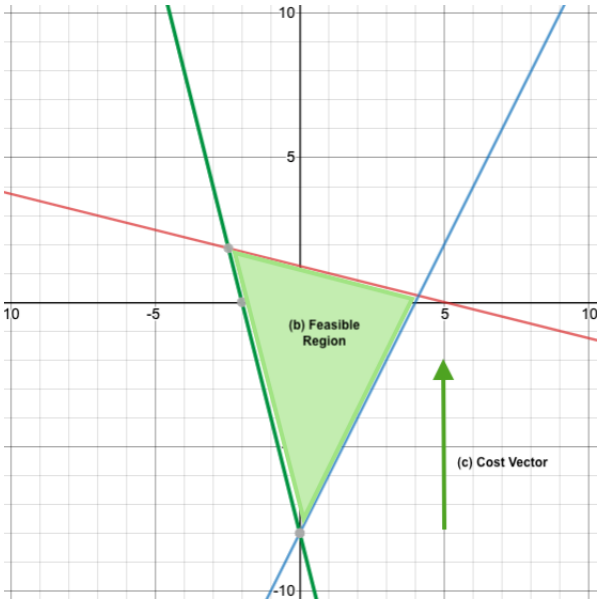


(b) In your graph from part (a), mark the feasible region with an 'F'

See graph on the following page.

(c) In your graph from part (a), draw in a cost vector such that the optimal solution is the point (0, -8).

See graph on the following page. (0, -8) should correspond to the minimum cost point within the feasible region.



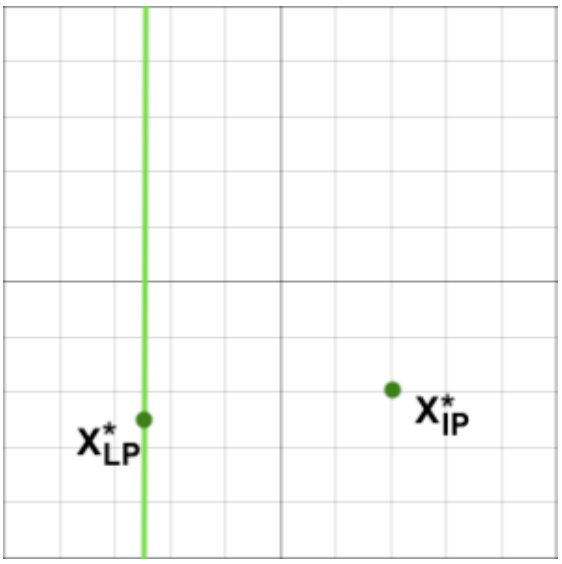
(d) List three cost vectors (that are not scaled versions of each other) that will lead to an infinite number of solutions.

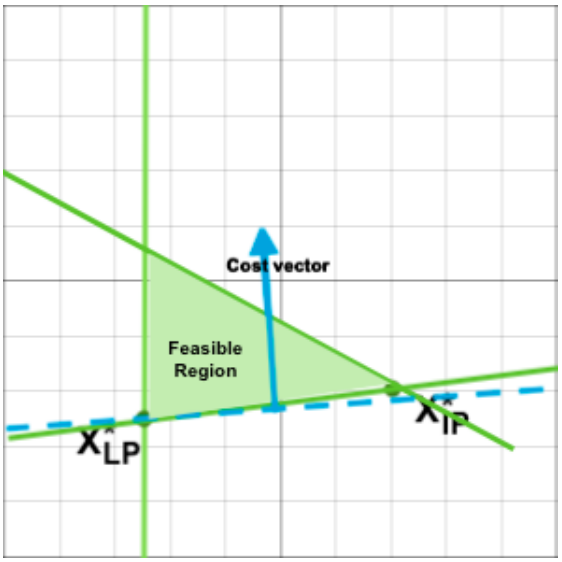
$\begin{bmatrix}-2\\1\end{bmatrix}, \begin{bmatrix}-1\\-4\end{bmatrix}, \begin{bmatrix}4\\1\end{bmatrix}$

(e) Now, let us look at at the following constraint graphs. One inequality constraint has been drawn in.

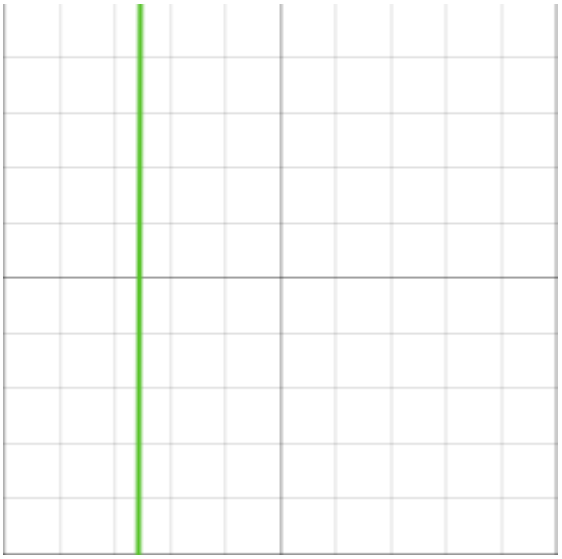
For each:
-Choose a feasible region and mark it with an 'F'
-Draw two additional constraint boundaries such that the given conditions are met
-Draw a cost vector such that the given conditions are met

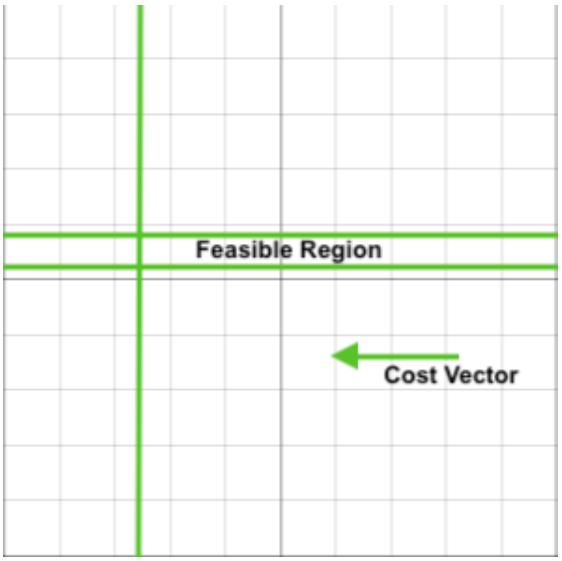
(i) The point x^*_{LP} is the linear programming solution and x^*_{IP} is the integer programming solution.





(ii) The minimum objective for the linear program is $-\infty$ and the integer program is infeasible.





(f) What is the outcome of running the branch and bound algorithm on each of the graphs from part (e)?

(i)

Returns X^*_{IP}.

(ii)

Throws an error, return infeasible, or never returns (implementation dependent).