CS6601 – Midterm Exam – Fall 2016

Instructions:

Do not put your name on this exam. They will be graded anonymously.

Fill out this PDF form and upload it to T-Square when you are done, with unlimited resubmissions until the deadline. You can type directly into the form (we suggest using Adobe Reader DC, or Evince on Linux; other programs may not save your answers so **please keep a backup**), or print, hand-write, & scan (or some combination of the two). **Bullet point answers are appreciated over full sentences. Show your work wherever possible and box your final answer (or bold & underline).**

Submit only a single PDF – no phone pictures, please! (You may use an app like Camscanner if you do not have scanner access.) Do not add pages unless absolutely necessary; if you do, please add them at the end of the exam only, and clearly label both the extra page and the original question page.

The exam is open-book, open-note, open video lectures, with no time limit aside from the open period, but no internet use is allowed (with exceptions for e-text versions of the textbook & this semester's CS6601 course materials). Do not discuss the midterm on Piazza, Slack, or any other form of communication. If there is a question for the teaching staff, please make it Private on Piazza and mark it as Midterm in the subject line and tag.

Point breakdown:

Each question sub-part is worth 3 points (with some exceptions). Bonus questions (optional) are 1 point each and applied to the overall grade (not included in the exam points).

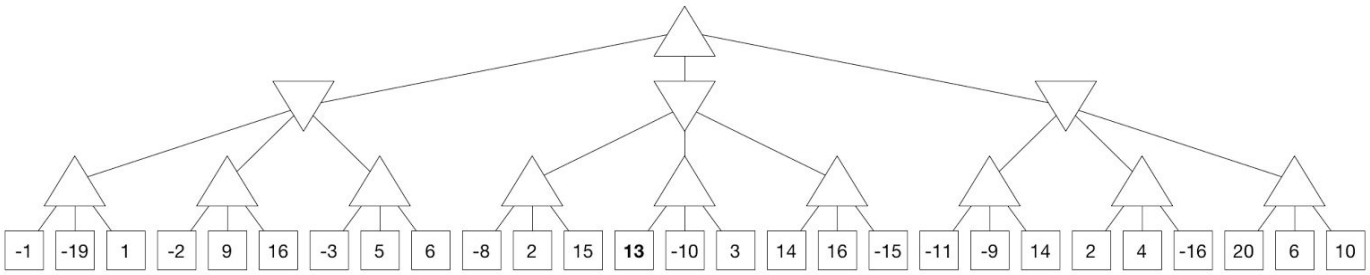
Q	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Total
Points	19	12	6	9	12	12	12	18	**100**

1. Game Playing (19 points)

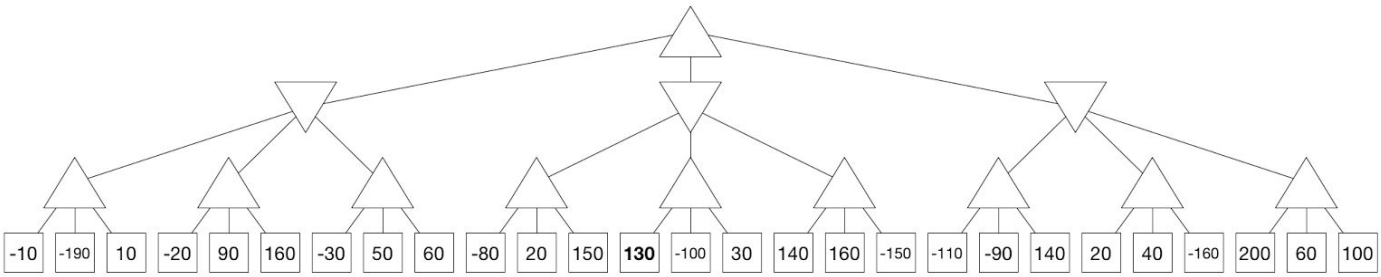
Part 1

Solve each of the game trees using alpha-beta: Check the box over branches that should be pruned, then fill in the remaining values for the tree.

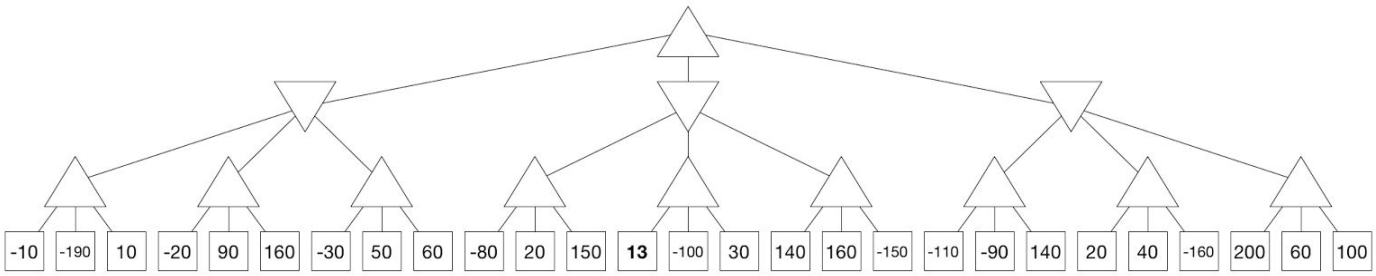
i. 3 points



ii. 3 points



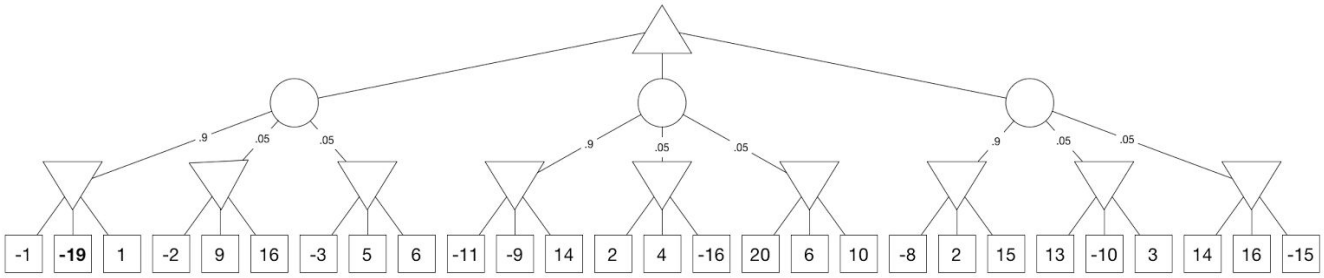
iii. 3 points



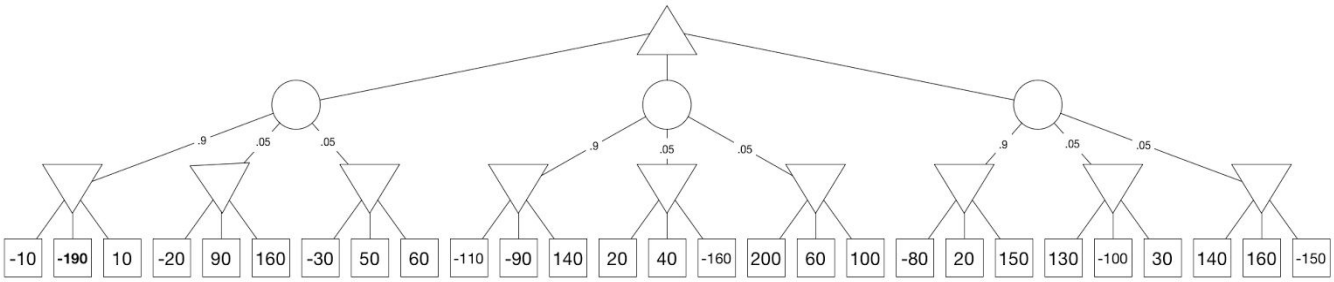
Part 2

Solve each of the game trees using expectimax. Fill in the remaining values for the tree.

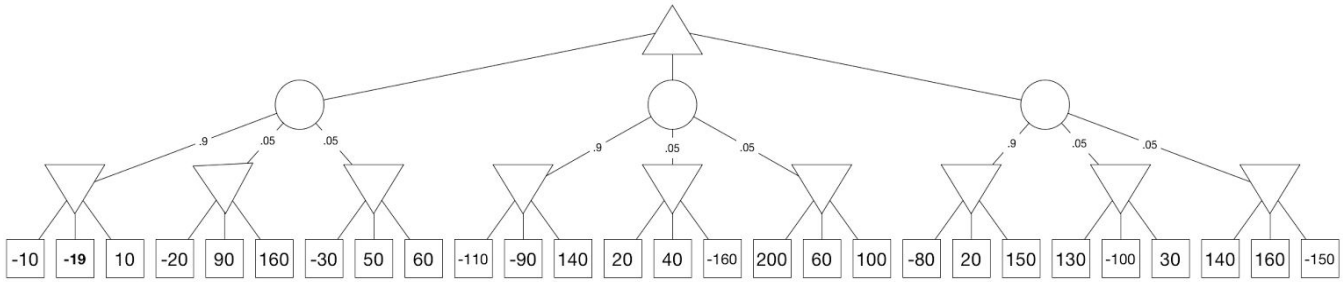
i. 3 points



ii. 3 points



iii. 3 points



Part 3 (1 point)

What were we trying to show you with these examples? (50 words or less)

2. Search (12 points)

2.1 Algorithms

Consider the problem of morphing one word into another by making one character change at a time. You want to write a computer program that solves this for any start and goal word that is 6 letters long.
(Example: http://www.thinkablepuzzles.com/wordchange)
(Example: http://www.wordplays.com/multi-word-morph)

Let's say you represent every word as a node and all words one character change away as its children. Then, you can use a search algorithm in this word space to go from the start to goal node. Note that every step of the path has to be a valid word. However, not all combinations are valid words, so you have an API call that will tell you if an input is a valid word. Assume no more resources than those explicitly mentioned.

Part A (3 points)
You want to try using an uninformed unidirectional search first. Which method will you use (taking into consideration time and space constraints) under the following conditions?

1. It's guaranteed that a valid solution will be found with exactly 6 changes.

2. It's not guaranteed that a valid solution will be found with exactly 6 changes. (A valid solution is guaranteed.)

Part B (3 points)
Now, you want to try using an informed unidirectional search method.

1. What heuristic will you use? (must be admissible)

Which method will you use (taking into consideration time and space constraints) under the following conditions?

2. It's guaranteed that a valid solution will be found with exactly 6 changes.

3. It's not guaranteed that a valid solution will be found with exactly 6 changes. (A valid solution is guaranteed.)

Part C (50 words, 3 points)
Will a bi-directional search work better? Why or why not? If it works better, what would be the termination condition?

2.2 Heuristics (3 points)

Let's say a given problem has N valid heuristics - h_1 through h_n. Which of the following are valid heuristics for the same problem? Choose multiple; full points will be awarded only if you select all the right answers.

1. $\max(h_1, h_2, ...h_n)$
2. $\min(h_1, h_2, ...h_n)$
3. $\text{sum}(h_1, h_2, ...h_n)$
4. $\text{avg}(h_1, h_2, ...h_n)$

Given a choice between them, which one would you choose and why? (Max three sentences.)

3. Simulated Annealing (6 points)

a. With regards to simulated annealing, what is the probability of accepting the following moves? Assume the problem is trying to maximise the objective function. (Round to 4 decimal places; 3 points)

Current State, E (Evaluation)	**Potential New State, E' (Evaluation)**	**Temperature**	**Probability of Accepting**
100	60	50	_____
200	120	50	_____
100	25	150	_____
200	210	30	_____
100	150	300	_____
200	40	300	_____

b. Explain how you could change the SA algorithm so that it implements hill climbing. (3-4 sentences, 3 points)

4. Probability (9 points)

Suppose you are given a box containing *d* decks of cards. You are told that each of these decks of cards have 52 cards, 26 red and 26 black, except for one fake deck that has all black cards. Show your work for the questions below.

a. You reach into the box, pick out a deck uniformly at random, shuffle it, deal a card at random, and get a black card. What is the (conditional) probability that the deck you chose is the fake deck? (3 points)

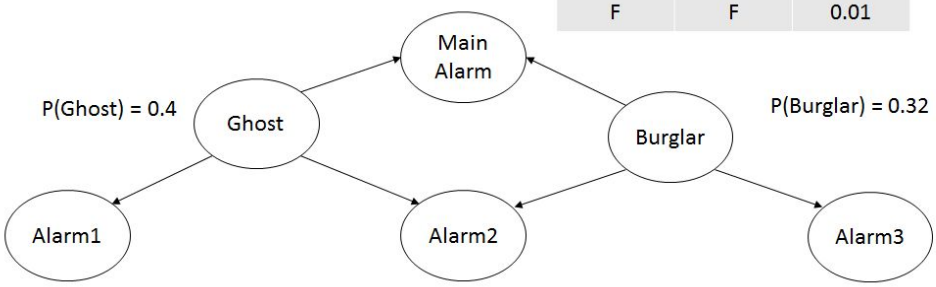
b. Now suppose you continue dealing cards from the deck, replacing the chosen card and shuffling each time to ensure randomness, for a total of k times and see k black cards. Now what is the conditional probability that you picked the fake deck? (3 points)

c. Suppose you wanted to decide whether a chosen deck was the fake one by dealing a random card (replacing it each time) k times. The decision procedure returns fake if all k cards come up black, otherwise it returns normal. What is the (unconditional) probability that this procedure makes an error on the decks drawn from the box? (3 points)

5. Halloween Bayes Networks! (12 points)

Ghost	Burglar	P(MA)
T	T	0.98
T	F	0.25
F	T	0.64
F	F	0.01

MA : Main Alarm



Ghost	P(Alarm1)
T	0.82
F	0.09

Ghost	Burglar	P(Alarm2)
T	T	0.87
T	F	0.31
F	T	0.63
F	F	0.05

Burglar	P(Alarm3)
T	0.19
F	0.72

Find the following probabilities for this spooky neighborhood house (to 4 decimal places):

1. P (Burglar = T| Alarm2 = T) (3 points)

2. P (Ghost = T| Alarm1 = T, Alarm2 = T) (3 points)

3. P (Alarm3 = T| Alarm2 = F, Burglar = T, Main Alarm = F) (3 points)

4. P (Ghost = F| Alarm2 = F, Alarm3 = T) (3 points)

6. Machine Learning (12 points)

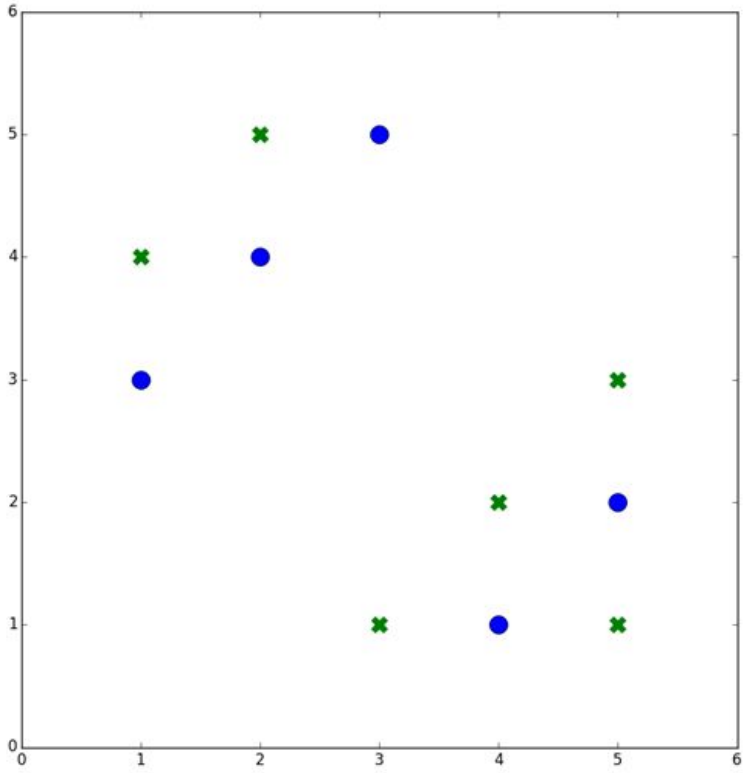
a. What is the No Free Lunch Theorem? (3 points)

b. Compare k-NN (k-nearest neighbors) and Naive Bayes with respect to the No Free Lunch Theorem. Choose multiple; full points will be awarded only if you select all the right answers. (3 points)

1. Naive Bayes is less sensitive to outliers than k-NN.
2. K-NN is a better classifier for real time data compared to Naive Bayes because it can quickly classify the data points into clusters.
3. With a large dataset, k-NN uses less space than Naive Bayes.
4. K-NN is more likely to overfit, which is why you need to test out different k-values to find the best one.
5. Both classifiers work well when the data is not independent.

6.1 Classification

Here's some data. There are two classes, o (blue) and x (green). The actual class is given by the shape of the data point on the graph below:



a. If we use 1-NN to classify these data points, what will be the leave one out cross validation error? (your answer should be between 0 and 1; 3 points)

b. Which of the following values of k leads to the minimum number of validation errors? Which value leads to the maximum error? What was the validation error for each (your answer should be between 0 and 1)? In case of a tie, take circle. (3 points)

- 3
- 5
- 10

7. Constraint Search (12 points + 3 bonus)

Consider the following graphs (called maps from here on out to avoid confusion) where the problem requires you to travel from the start (green triangle) to the goal (orange square) while avoiding obstacles (shaded regions).

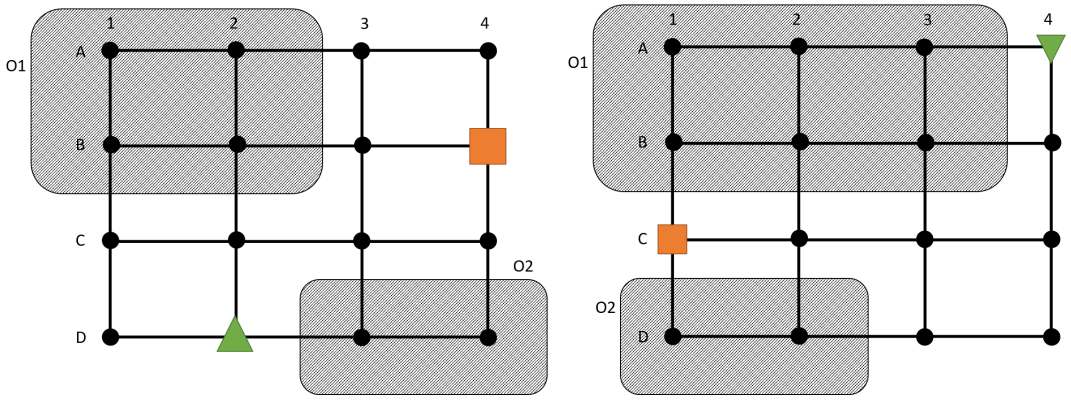


Figure 1 (a,b): Example maps for constrained search. (a) is on the left, (b) is on the right

Formulating the CSP

Your first task is to formulate this problem as a Constraint Satisfaction Problem (CSP). Particularly, assume that we have a naive algorithm that is able to provide you with the following information:

1. The map $M \equiv (V, E)$ is given as a set of vertices, V, and edges, E.
2. Obstacles $O_1, O_2, ..., O_n$ such that $O_i \equiv (V_{O_i})$ is a list of vertices denoting the obstacle.
3. The start vertex $S \in V$, and goal vertex $G \in V$.
4. A set of candidate paths, $P_1, P_2, ..., P_m$ from $S \rightarrow G$ given as a list of edges, $P_i \equiv (E_{P_i})$.

Your CSP must have a valid solution if any of the candidate paths is not obstructed; it must return failure if all the paths are obstructed.

Question 1 (3 points)
Create the Constraint Graphs for the maps given in Figure 1. Assume the following:

a. Figure 1(a):
   $S \equiv D2$
   $G \equiv B4$
   $O_1 \equiv (\{A1, A2, B1, B2\})$
   $O_2 \equiv (\{D3, D4\})$
   $P_1 \equiv (D2C2, C2C3, C3C4, C4B4)$
   $P_2 \equiv (D2D3, D3D4, D4C4, C4B4)$
   $P_3 \equiv (D2C2, C2B2, B2B3, B3B4)$
   $P_4 \equiv (D2D3, D3C3, C3B3, B3B4)$
   $P_5 \equiv (D2C2, C2C3, C3B3, B3B4)$

Clearly label the nodes, mark and describe the constraints, and mention the domains for all the variables in the graph.

b. Figure 1(b):

$S \equiv A4$

$G \equiv C1$

$O_1 \equiv (\{A1, A2, A3, B1, B2, B3\})$

$O_2 \equiv (\{D1, D2\})$

$P_1 \equiv (A4A3, A3A2, A2A1, A1B1, B1C1)$

$P_2 \equiv (A4B4, B4C4, C4C3, C3C2, C2C1)$

$P_3 \equiv (A4B4, B4B3, B3B2, B2B1, B1C1)$

$P_4 \equiv (A4A3, A3A2, A2B2, B2C2, C2C1)$

$P_5 \equiv (A4A3, A3B3, B3B2, B2C2, C2C1)$

Clearly label the nodes, mark and describe the constraints, and mention the domains for all the variables in the graph.

Solving the CSP

Question 2 (3 points)
You now have to solve the CSP that you formulated in the previous section. Which of the following algorithms should you use to solve this CSP? Choose one.

1. AC-3
2. PC-2
3. Min-Conflicts
4. Backtracking Search with Maintaining Arc Consistency
5. Topological Sort followed by Directed Arc Consistency
6. Tree Decomposition followed by AC-3

Question 3 (3 points)
Why did you choose this algorithm? Choose multiple; full points will be awarded only if you select all the right answers.

1. All the constraints are binary
2. Arc consistency is guaranteed when there is an unobstructed path
3. Value assignments will be invalid only when no unobstructed path exists
4. Path consistency is guaranteed when there is an unobstructed path
5. We need to know of at least one valid unobstructed path from S to G
6. Checking for inconsistencies can be done in roughly constant time per iteration, assuming the same number of constraints
7. If a valid path exists, we are guaranteed to find it in finite time
8. We are guaranteed to learn of the lack of paths in finite time
9. The constraint graph is fully connected
10. There are disconnected subcomponents
11. Runtime is linear in number of variables

Question 4 (3 points)
Do you think we should use the CSP we formulated for collision checking, or should our initial search algorithm have been smarter to avoid the obstacles altogether when providing candidates? Justify, especially in terms of runtime and space complexity, in one sentence (points will be deducted for more than $min(1\ sentence,\ 35\ words)$ - be concise!).

Bonus: Extending the CSP

Now consider the following map:

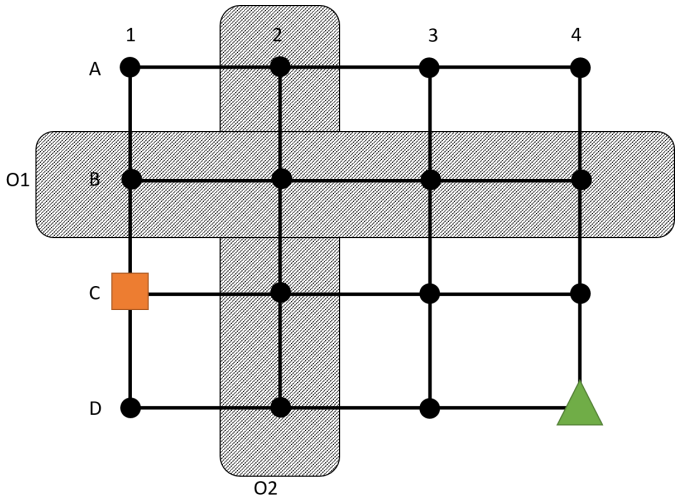


Figure 2: Example map where CSP should return false

Clearly an obstacle-free path from the start to the goal does not exist. However, instead of returning a failure, your CSP must now return a valid list of constraints that are preventing execution along any of the candidate paths. Of course, a list of all constraints is valid, but we want to be more parsimonious than that!

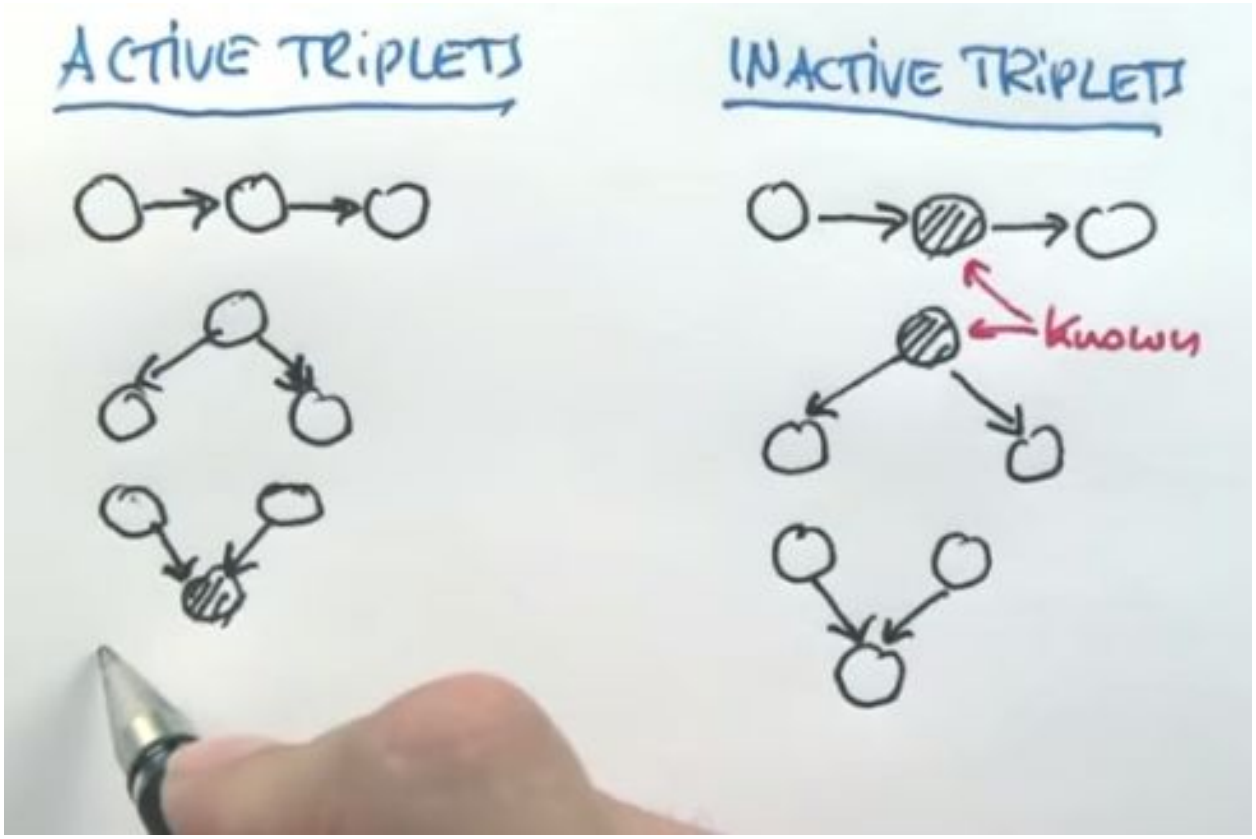
For this section, you will need to write pseudo-code. When writing out this pseudo-code, try to follow and reference the patterns of known algorithms in the textbook. In addition, include comments wherever you might have made crucial changes to the known algorithms.

Bonus Question 1 (1 point)
Return a list with the minimum number of constraints that are stopping us from completing our path(s).

Bonus Question 2 (1 point)
Assume that each constraint is associated with a weight ($w_1, w_2, ..., w_n$) which signifies some cost to removing the constraint. Return a list with the minimum total cost.

Bonus Question 3 (1 point)
Could a search algorithm have accomplished the same goal of returning the constraints in violation? Why or why not? You have 50 words.

8. D-Separation (18 points)



(Note: question has parts a-f)

In the lectures and in the challenge exercise on Piazza, we saw how d-separation could be used to simplify Bayes Nets calculations. In particular, Sebastian discusses “active triplets” versus “inactive triplets.” With inactive triples, the middle node separated the two other nodes and made them (conditionally) independent.

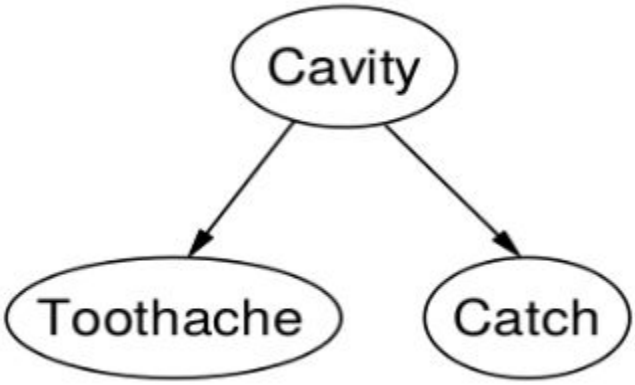
As an instructor, you've decided to work out one of these triplets to help students remember d-separation rules and get an intuition for the concept. You want to provide an example of full joint distribution table and show conditional independence through Bayes Rule and explicit calculations.

Using the middle case (B<-A->C), you extend the cavity, catch, toothache example from the book. Here is the full joint distribution table for that example:

	toothache		¬*toothache*	
	catch	¬*catch*	*catch*	¬*catch*
cavity	0.108	0.012	0.072	0.008
¬*cavity*	0.016	0.064	0.144	0.576

Figure 13.3 A full joint distribution for the *Toothache*, *Cavity*, *Catch* world.

Here is the corresponding Bayes Net for the problem.



You show the students the table and the network topology and demonstrate that when there is a Cavity, Toothache is independent of Catch, and Catch is independent of Toothache. In other words, you show

P(TOOTHACHE / CAVITY) = P(TOOTHACHE / CAVITY, CATCH) =
P(TOOTHACHE / CAVITY, ~CATCH)

Where ^ is intersection, V is union, ~ is negation. Working it out:

$$P(\text{TOOTHACHE} / \text{CAVITY}) = \frac{P(\text{CAVITY} \wedge \text{TOOTHACHE})}{P(\text{CAVITY})} = \frac{0.108 + 0.012}{0.108 + 0.012 + 0.072 + 0.008} = 0.6$$

$$P(\text{TOOTHACHE} / \text{CAVITY}, \text{CATCH}) = \frac{P(\text{TOOTHACHE} \wedge (\text{CAVITY} \wedge \text{CATCH}))}{P(\text{CAVITY} \wedge \text{CATCH})} = \frac{0.108}{0.108 + 0.072} = 0.6$$

$$P(\text{TOOTHACHE} / \text{CAVITY}, \sim\text{CATCH}) = \frac{P(\text{TOOTHACHE} \wedge (\text{CAVITY} \wedge \sim\text{CATCH}))}{P(\text{CAVITY} \wedge \sim\text{CATCH})} = \frac{0.012}{0.012 + 0.008} =$$

$$= 0.6$$

For symmetry, you also work out

P(CATCH / CAVITY) = P(CATCH / CAVITY, TOOTHACHE) =
P(CATCH / CAVITY, ~TOOTHACHE)

But one of your students asks a question - “What if we know there is NOT a cavity - is it still independent?”

Using the same tables, you then show

P(TOOTHACHE / ~CAVITY) = P(TOOTHACHE / ~CAVITY, CATCH) = P(TOOTHACHE / ~CAVITY, ~CATCH)

$$P(\text{TOOTHACHE} / \sim\text{CAVITY}) = \frac{P(\sim\text{CAVITY} \wedge \text{TOOTHACHE})}{P(\sim\text{CAVITY})} = \frac{0.016 + 0.064}{0.016 + 0.064 + 0.144 + 0.576} = 0.1$$

$$P(\text{TOOTHACHE} / \sim\text{CAVITY}, \text{CATCH}) = \frac{P(\text{TOOTHACHE} \wedge (\sim\text{CAVITY} \wedge \text{CATCH}))}{P(\sim\text{CAVITY} \wedge \text{CATCH})} = \frac{0.016}{0.016 + 0.144} = 0.1$$

$$P(\text{TOOTHACHE} / \sim\text{CAVITY}, \sim\text{CATCH}) = \frac{P(\text{TOOTHACHE} \wedge (\sim\text{CAVITY} \wedge \sim\text{CATCH}))}{P(\sim\text{CAVITY} \wedge \sim\text{CATCH})} = \frac{0.064}{0.064 + 0.576} = 0.1$$

And

P(CATCH / ~CAVITY) = P(CATCH / ~CAVITY, TOOTHACHE) = P(CATCH / ~CAVITY, ~TOOTHACHE)

That convinces your students that Toothache and Catch are truly conditionally independent when the state of Cavity is known.

But perhaps Toothache and Catch are always independent! To show that is not true, you calculate
P(TOOTHACHE / CATCH) ~= P(TOOTHACHE / ~CATCH) ~= P(TOOTHACHE)

P(CATCH / TOOTHACHE) ~= P(CATCH / ~TOOTHACHE) ~= P(CATCH)

The exercise really seemed to help students understand Bayes Nets in general, not just d-separation, so you decide to continue it.

a) Now it gets more interesting. You need to calculate an appropriate full joint distribution for the d-separated inactive triplet A->B->C such that your students can perform calculations that demonstrate the independence of the node C from the node A when the node B is known to be true. You can use A->B->C , but it is better to draw a diagram with more creative node labels. Remember, you are trying to use fun and intuitive examples so that the students feel good about learning! Enter the full joint distribution for your example in the space below. (3 points)

b) Like in the cavity example above
(e.g., P(TOOTHACHE / CAVITY) = P(TOOTHACHE / CAVITY, CATCH) = P(TOOTHACHE / CAVITY, ~CATCH) , …. , P(CATCH / TOOTHACHE) ~= P(CATCH / ~TOOTHACHE)...)
write the equations we want your students to calculate to prove to themselves that d-separation works for A->B->C . (3 points)

c) Now, like the cavity example above, work out the series of calculations yourself to provide a key for your TAs to grade the students. This exercise will help you make sure that your full joint distribution is correct! (3 points)

d) Now let's make an example for the last three node d-separation example from Sebastian's lecture. Create an appropriate full joint distribution to correspond with the A->C<-B d-separated inactive triplet in the picture above and enter the full joint distribution in the space below. You can use A->C<-B , but again it is better to draw a diagram with more creative and intuitive node labels. Enter the full joint distribution for your example in the space below. (3 points)

e) Like in the cavity example above, write the equations you need your students to calculate to prove to themselves that d-separation works for A->C<-B . (3 points)

f) Finally, like the cavity example above, work out the series of calculations yourself to provide a key for your TAs to grade the students. Hopefully, the exercise you just designed will really help your students get an intuition for d-separation so they can solve Bayes Nets problems more quickly! (3 points)