6.034 Final Examination

20 December 2016

Name:		**Email:**	

Indicate which of the 8 sections of the final you are taking. We will grade **only** those sections.

☐ **Quiz 1**			☐ **Quiz 2**			☐ **Quiz 3**		☐ **Quiz 4**	
Problem 1	Problem 2	Problem 3	Problem 1	Problem 2	Problem 3	Problem 1	Problem 2	Problem 1	Problem 2
Quiz 1 Total			Quiz 2 Total			Quiz 3 Total		Quiz 4 Total	
☐ **Bonus SRN**			☐ **SRN 2**			☐ **SRN 3**		☐ **SRN 4**	

Survey

We are curious about possible correlations. This will not affect your final grade. Please indicate:

Number of 6.034 recitations attended (out of 13 weeks):

Number of 6.034 recitation videos watched...

...as a replacement for attending recitation:

...as a supplement to recitation:

There are 52 pages in this exam, not including tear-off sheets. As always, this exam is open book, open notes, open almost everything—including a calculator—but no computers.

(There is no quiz material on this page.)

Quiz 1, Problem 1: A Game of Rules (40 points)

Daenerys of the House Targaryen, a.k.a. Dany, is planning her takeover of Westeros. Dany is questioning the popular opinion that she should ride Drogon, her largest dragon, into battle. To find an answer, she resorts to using black magic in the form of a Rule-Based System, using the rules and assertions shown below.

Rules:

P0	IF OR(AND('**(?a)** will let **(?b)** ride on his back',
	'**(?a)** is terrifying'),
	'**(?a)** has a soul bond with **(?b)**')
	THEN '**(?b)** will ride **(?a)** into battle'
P1	IF AND('**(?y)** is immune to fire',
	'**(?z)** is a dragon',
	'**(?y)** did not lock **(?z)** in a dungeon')
	THEN '**(?z)** will let **(?y)** ride on his back'
P2	IF '**(?x)** is a dragon'
	THEN '**(?x)** is terrifying'

Assertions:

```
A0: Drogon is a dragon
A1: Viserion is a dragon
A2: Rhaegal is a dragon
A3: Dany is immune to fire
A4: Dany did not lock Drogon in a dungeon
```

(A copy of these rules and assertions is available on the tear-off sheet.)

Part A: Backward Chaining (20 points)

Make the following assumptions about backward chaining:

- The backward chainer tries to find a matching assertion in the list of assertions. If no matching assertion is found, the backward chainer tries to find a rule with a matching consequent. In case no matching consequents are found, the backward chainer concludes that the hypothesis is false.
- The backward chainer never alters the list of assertions.
- Rules and antecedents are tried in the order they appear.
- Lazy evaluation/short circuiting is in effect.

A1 (18 points) Using the rules and assertions provided, perform backward chaining starting from the hypothesis:

```
'Dany will ride Drogon into battle'
```

- In the table below, write all the hypotheses that the backward chainer checks, in the order they are checked. (The first line has been filled in for you, and the table has more lines than you should need.)
- You can show your work for partial credit: Use the space on the next page to draw the goal tree that would be created by backward chaining from this hypothesis.

1	Dany will ride Drogon into battle
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	

For partial credit, draw your goal tree here:

Dany will ride Drogon into battle

A2 (2 points) Did backward chaining in Part A1 prove the hypothesis 'Dany will ride Drogon into battle'? Circle one:

YES **NO**

Part B: Forward Chaining (20 points)

Using the rules and assertions provided, run forward chaining and fill in the table below. (There may be more rows than you need.)

- For each iteration, list the rules whose antecedents match the assertions, the rule that fires, and any new assertions that are added.
- If no rules match or fire, or no new assertions are generated, write NONE in the corresponding box, then leave the remaining rows blank.

Make the following assumptions about forward chaining:

- When multiple rules match, rule-ordering determines which rule fires.
- New assertions are added to the bottom of the list of assertions.
- If a particular rule matches in more than one way, the matches are considered in the top-to-bottom order of the matched assertions. Thus, if a particular rule has an antecedent that matches both A1 and A2, the match with A1 is considered first.

	Matched	Fired	New Assertion(s)
1			
2			
3			
4			
5			
6			
7			
8			
9			

Quiz 1, Problem 2: Search (35 points)

Part A: City Search (10 points)

For each situation described below, circle the <u>one</u> search algorithm that is most appropriate. Each algorithm may be used once, more than once, or not at all. Furthermore, assume that:

- All algorithms incorporate backtracking if possible.
- Branch & Bound does *not* use a heuristic or extended set.

1. You are planning a plane trip from Boston to San Francisco; you're looking for a flight that stops at the fewest cities in between.

Depth First	**Breadth First**	**Best First**	**Branch & Bound**	**Hill Climbing**	**British Museum**

2. While vacationing in Paris, you're designing a tour that visits every one of the local landmarks exactly once.

Depth First	**Breadth First**	**Best First**	**Branch & Bound**	**Hill Climbing**	**British Museum**

3. You are descending a cliff face. Your strategy is to reach for the footholds and handholds that are furthest down, because you want to descend as quickly as possible. Whenever you reach a dead end, you backtrack and try a different path down.

Depth First	**Breadth First**	**Best First**	**Branch & Bound**	**Hill Climbing**	**British Museum**

4. You are feeling carefree and whimsical in London. You decide to make your way to your destination by following the first street you see (avoiding loops, of course) at every intersection you encounter. If you get stuck, you'll retrace your steps.

Depth First	**Breadth First**	**Best First**	**Branch & Bound**	**Hill Climbing**	**British Museum**

5. It's freezing cold outside! You know how long it takes to walk between different places, and you'd like to plan the fastest possible route home.

Depth First	**Breadth First**	**Best First**	**Branch & Bound**	**Hill Climbing**	**British Museum**

Part B: Science Fair (15 points)

Long-time rivals Josh and Mindy are competing in a computer science competition at the science fair. The person who can demonstrate better knowledge of search algorithms will be declared the winner!

The following code is a correct implementation of a search algorithm, with two steps missing:

```
function search(start_node, end_node):
     agenda = [path(start_node)]
     while agenda is not empty:
          path = remove next path from agenda
          if path includes goal:
               return path
          else:
               new_paths = extensions(path)  # excluding loops
               (Line 1)  ???
               (Line 2)  ???
     return None  # no paths found
```

And here are five generic actions that you can use to fill in the blanks:

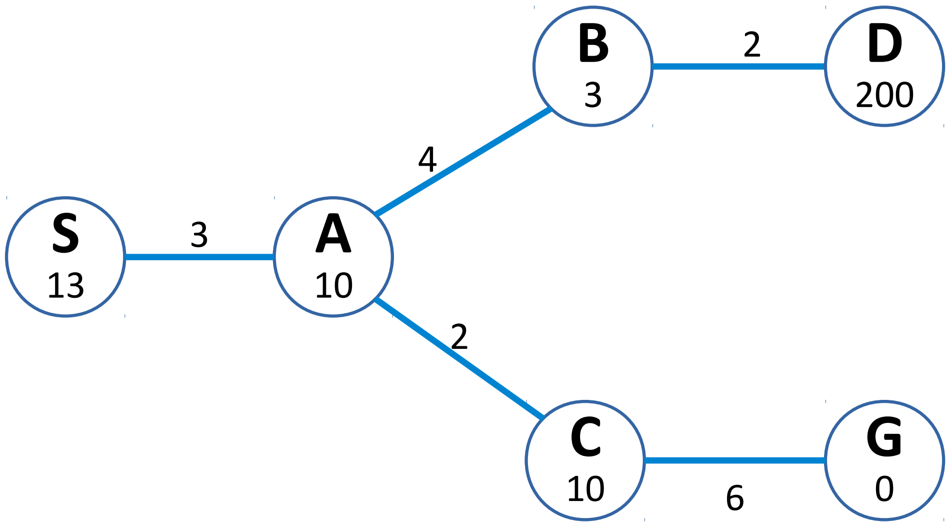
- **PASS**: Do nothing
- **ADD_PATHS**: Add new_paths to the front or back of the agenda
- **OVERWRITE_AGENDA**: Overwrite agenda by setting agenda = new paths
- **SORT_PATHS**: Sort new_paths by path length and/or heuristic value
- **SORT_AGENDA**: Sort entire agenda by path length and/or heuristic value

Which of these generic actions could Lines 1 and 2 perform in order for the pseudocode to exhibit the behavior of each of the algorithms below? **Fill in each cell with one of the five generic actions (PASS, ADD_PATHS, OVERWRITE_AGENDA, SORT_PATHS, SORT_AGENDA).** In each case, you may assume that there are no ties, so you do not need to implement tie-breaking of any kind.

Algorithm	Line 1	Line 2
Hill-climbing		
Hill-climbing **without** backtracking		
Depth-first search		
Branch and Bound		
Best-first search		

Part C: Graph Search (10 points)

Consider the graph below, with edge lengths labeled. Numbers inside each node indicate heuristic estimates of the distance to the goal. Use **beam search with a beam width of one** (w=1) to **find a path from S to G** in the graph. Break ties lexicographically and do not use backtracking.



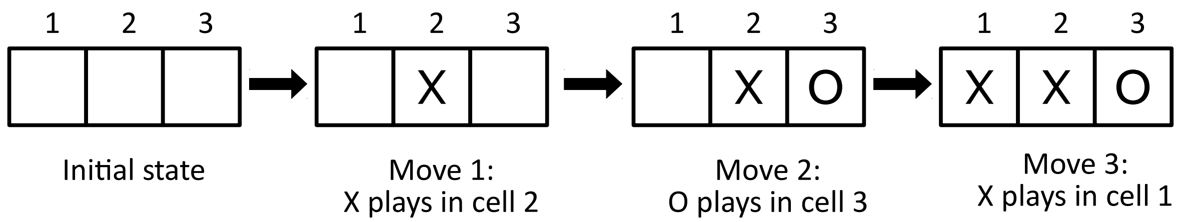
C1 (8 points) Draw the search tree below:

C2 (2 points) What path did your beam search find? Write the path as a list of nodes, including **S** and **G**, or write **NONE** if search terminated before finding a path:

Quiz 1, Problem 3: Games (25 points)

Part A: Learning the Grid Game (4 points)

One of your hallmates has invented a simple new game for you to play together. In the game, Player 1 **(X)** and Player 2 **(O)** take turns placing tokens on a 1×3 grid. A sample game is shown below:


1 2 3
X
X O
X X O
Initial state
Move 1: X plays in cell 2
Move 2: O plays in cell 3
Move 3: X plays in cell 1

The one restriction is that each player must put their token in an unoccupied space. When the board is full, the game ends and the endgame score is determined according to the following chart:

Endgame state: 1	2	3	**Endgame score:**
O	X	X	+5
X	O	X	-4
X	X	O	+9

(A copy of this chart is available on the tear-off sheet.)

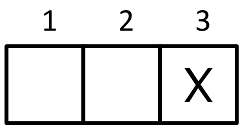
To find out how to win this game, you decide to draw the game as a game tree. To start, you must first figure out how many moves are possible at each step.

A1 (2 points) In this game, what is the branching factor of the **first** move? That is, how many possible first moves are there?

A2 (2 points) What is the branching factor of the **second** move? That is, given any first move, how many possible, legal second moves exist? (Your hallmate reminds you that players must put each token in an unoccupied space.)

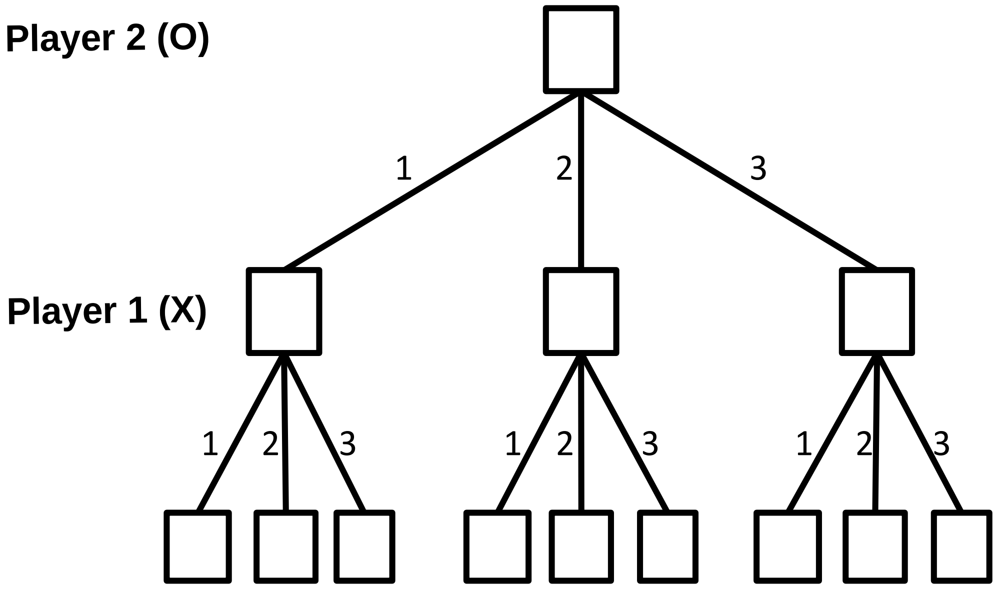
Part B: Game Tree (14 points)

B1 (12 points) You decide to draw a section of the game tree to handle a specific case. Suppose that Player 1 (X) has just played in position 3:

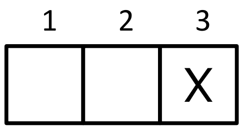


Player 1 (X) wants the *highest possible score* while Player 2 (O) wants the *lowest possible score* – that is, Player 1 will be MAX and Player 2 will be MIN. Fill in the game tree below to show how the remainder of the game will play out.

1. **Cross out** any nodes that are unreachable (impossible game states).
2. **Write the endgame score** in each *reachable* leaf node.
3. **Write the minimax score** in each *reachable non-leaf* node.



Now that you've figured out part of the game tree, you're ready to begin playing against your hallmate! Your hallmate, playing as X (MAX), plays in position 3:



B2 (2 points) Assuming that both players play optimally for the remainder of the game, where should you play the second move to get the best score as O (MIN)? (Circle one)

1 **2** **3**

Part C: Win! (7 points)

To win against your hallmate, you now want to consider the entire game, not just the case where X plays in position 3. Assume that both players play optimally.

You can answer these questions using intuition, or by drawing the complete game tree and performing minimax. For partial credit, show your work below.

C1 (3 points) What position will X (MAX) play in **first**?

1 **2** **3**

C2 (2 points) What will be the endgame state? Fill in the final state:

1	2	3

C3 (2 points) Suppose that you are playing as X (MAX). What is the highest endgame score that O (MIN) will allow you to get?

Show your work for partial credit (Part C):

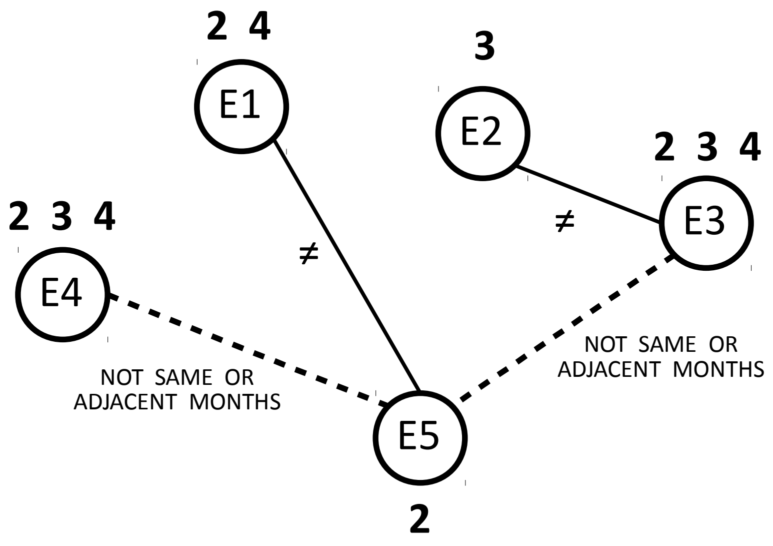
Quiz 2, Problem 1: Constraint Satisfaction (50 points)

Part A: Resident Advisor Events Listing (18 points)

According to your favorite website for electronic music events, the *Resident Advisor*, five of your favorite music artists will be visiting Boston next spring! They will visit during the months of February (2), March (3), and April (4). The table below shows the months when each event is scheduled to take place:

Event (variable)	Performance Months (domain)		
E1 (*Aphex Twin*)	**2**		**4**
E2 (*Boards of Canada*)		**3**	
E3 (*Caribou*)	**2**	**3**	**4**
E4 (*Grimes*)	**2**	**3**	**4**
E5 (*Autechre*)	**2**		

Based on your scheduling constraints and preferences, you come up with the constraint graph below, in which the five events (E1, E2, E3, E4, E5) are variables, and each domain is a subset of the three months (2, 3, 4). The constraints are either *can't be same month* (≠) or *can't be same or adjacent month* (- - -).



(Additional copies of this graph are available on the tear-off sheet.)

Perform **Depth First Search with assignment-checking only** (without Forward Checking, and without Propagation) to find a schedule. Make assignments in lexicographic order: E1-E2-E3-E4-E5. Continue until the search terminates or you've filled all the rows in the table.

★ For credit, show your work on this page and the next page by simultaneously
(1) filling out the domain worksheet and
(2) drawing the search tree.

Fill out this worksheet as you draw your search tree.

1. Every time you **assign a variable** or **remove a variable from the propagation queue** (if applicable), fill out a new row in the table. (There may be more rows than you need.)
2. In that row, indicate **which variable you assigned or de-queued**; write its **assigned value** if it has one (e.g. X=x), otherwise just write its **name** (e.g. X). In the second column, list the **values that were just eliminated from neighboring variables** as a result (or "NONE" or "—" if no values were eliminated). Do not eliminate values from variables that have already been assigned.
3. If your search has to backtrack after assigning or de-queuing a variable: First, **finish listing** all values eliminated from neighboring variables in the current row. Next, check the "backtrack" box in that row. Then, continue with the next assignment in the following row as usual.
4. If you add several variables to your propagation queue at once, break ties by adding variables to your propagation queue in lexicographic order (e.g. E1 before E2). Only add a variable if it is not already on the queue.

	Var assigned or de-queued	List all values just eliminated from neighboring variables	Back track
1			☐
2			☐
3			☐
4			☐
5			☐
6			☐
7			☐
8			☐

Example row showing an assigned variable

ex	X = 3	Y ≠ 3, 4 Z ≠ 3 (example)	☑

Example row showing a de-queued (propagated) variable

ex	X	W ≠ 1, 4 (example)	☐

Draw your search tree on this page.

E1

E2

E3

E4

E5

Part B: Improving efficiency (26 points)

Plain depth-first search takes a long time, so you want to eliminate incompatible months before even starting your search.

B1 (14 points) Perform **<u>Domain Reduction Before Search</u>** to eliminate months from each event's domain (if any). Start by adding all variables to your queue in lexicographic order.

Fill out this worksheet following the instructions given in part A.

	Var assigned or de-queued	**List all values just eliminated from neighboring variables**
1	E1	
2	E2	
3	E3	
4	E4	
5	E5	
6		
7		
8		
9		
10		

B2 (2 points) How many values did Domain Reduction Before Search **eliminate** from your variables in total?

B3 (2 points) How many values did Domain Reduction Before Search **assign** to your variables in total?

B4 (2 points) In the table below, cross out the months that were eliminated from the variables' domains:

Event (variable)	Performance Months (domain)		
E1 (*Aphex Twin*)	**2**		**4**
E2 (*Boards of Canada*)		**3**	
E3 (*Caribou*)	**2**	**3**	**4**
E4 (*Grimes*)	**2**	**3**	**4**
E5 (*Autechre*)	**2**		

B5 (6 points) Perform <u>**Depth First Search with assignment-checking only**</u> (without Forward Checking, and without Propagation) to find a schedule. Make assignments in lexicographic order: E1-E2-E3-E4-E5. Draw your search tree below. (You don't need to fill in a domain worksheet.)

E1

E2

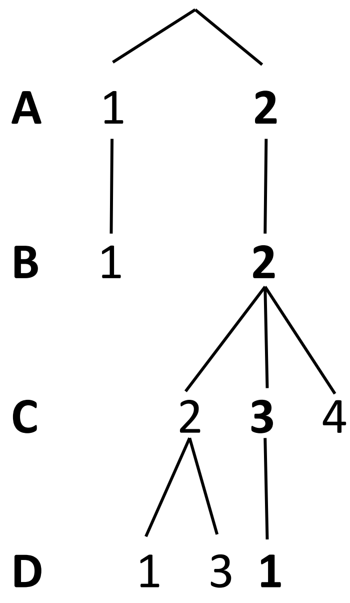
E3

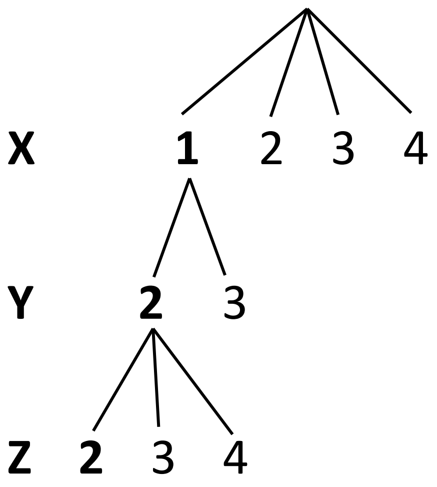
E4

E5

Part C: Backtracking (6 points)

On your way to the first concert, you encounter two wild constraint search trees, with their final assignments listed (and bolded). For each tree, indicate the number of times that the constraint satisfaction algorithm backtracked.





Final assignments: A=2, B=2, C=3, D=1

Number of backtracks: $\boxed{}$

Final assignments: X=1, Y=2, Z=2

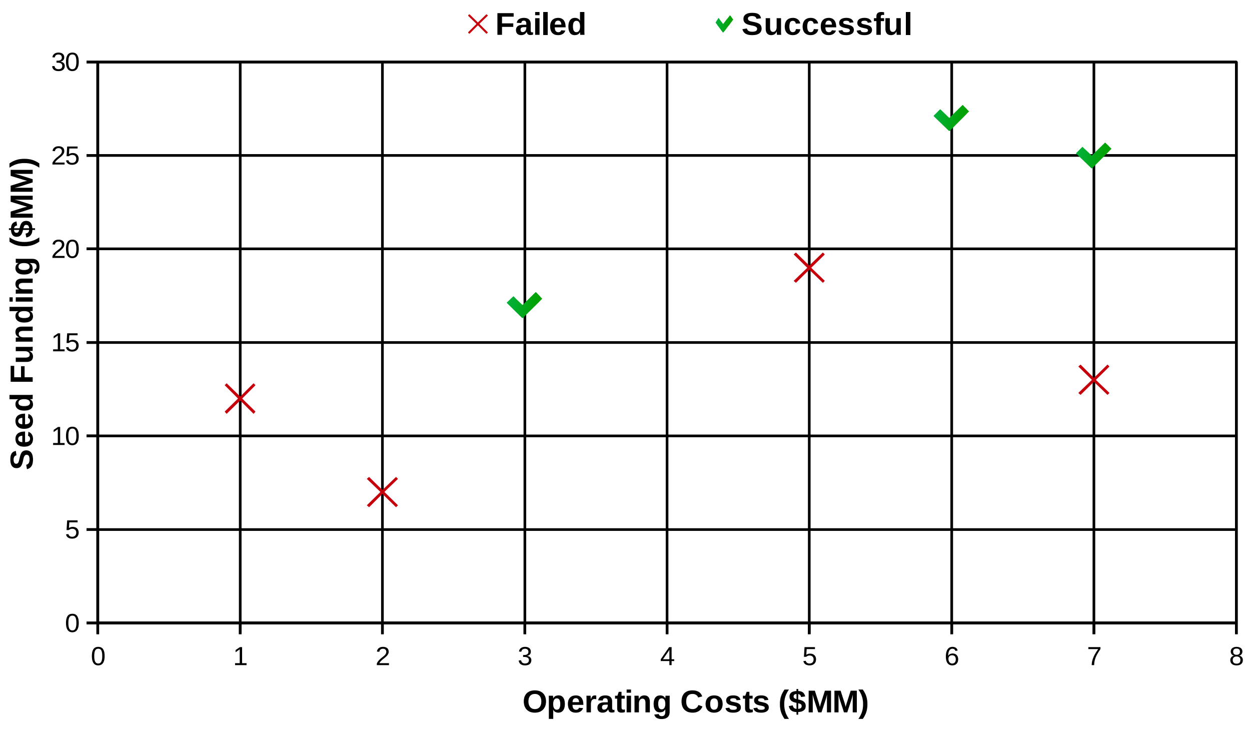
Number of backtracks: $\boxed{}$

Quiz 2, Problem 2: ID Trees (20 points)

Part A: The Start-Up World (10 points)

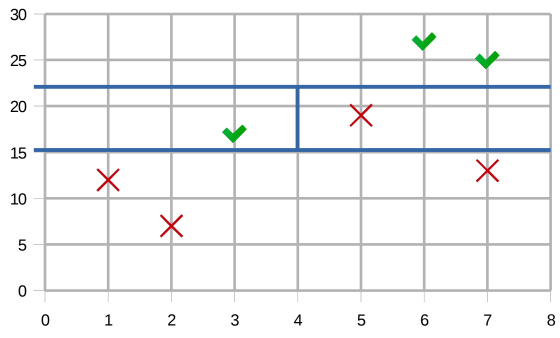
Your friend Nathan's grandmother has just won the lottery and asks for your help in choosing a start-up to fund. In order to convince her that you can predict the next unicorn, you decide to build a model in the form of an ID tree based on data from companies that have already lived through the start-up phase.

The first data set, shown on the graph below, represents a company's success **(✓)** or failure **(✕)** given the amount of seed funding and operating costs, in millions of dollars ($MM), within their first year.

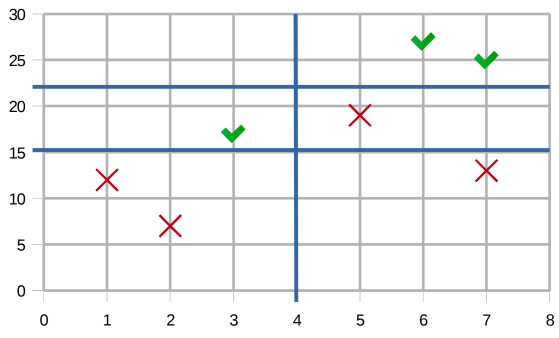


A1 (4 points): Nathan wants to draw the boundary lines representing a greedy disorder-minimizing ID tree that perfectly classifies the data. Each test should only use feature, e.g. 'feature > T', and he wants to break ties by preferring Seed Funding tests over Operating Costs tests. Nathan comes up with six possible graphs representing numeric identification trees, shown on the next page. Which one represents the greedy disorder-minimizing ID tree? (Circle one)

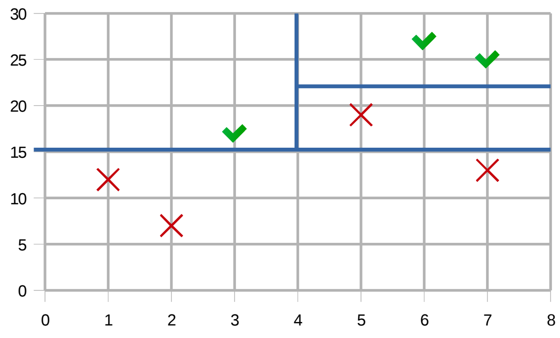
(This problem should not require significant calculation.)



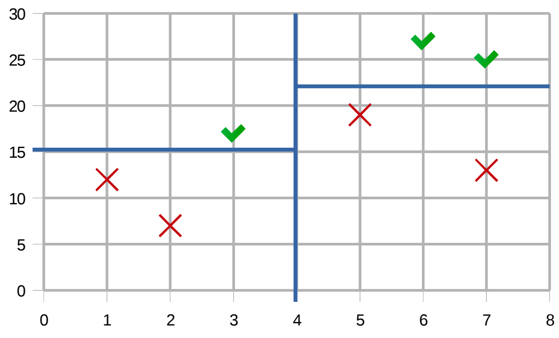
A



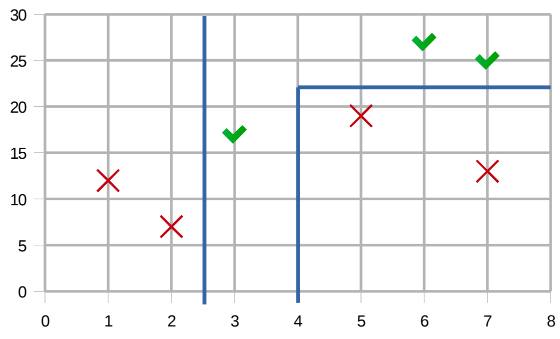
B



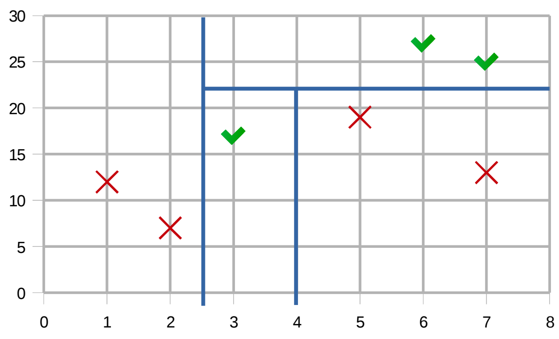
C



D



E



F

A2 (6 points): Draw the greedy, disorder-minimizing identification tree corresponding to the graph that you chose in part A1. (You may approximate the threshold values.)

Part B: The Engine (10 points)

Nathan’s grandmother hears from one of her poker friends that MIT has started *The Engine*, a new venture fund which has contributed to several new start-ups. You decide to use the feature test **Type of Start-up** to predict the status of unknown start-ups.

B1 (4 points) Compute the disorder of the test **Type of Start-up** on the dataset below. You do not need to simplify the logarithms.

Name	Status	Type of Start-up
Teslr	Successful	Electric Cars
Oculure	Successful	Virtual Reality
Prie-US	Failed	Electric Cars

Show your work for partial credit.

B2 (6 points) Compute the disorder of the test **Type of Start-up** on the dataset below. You do not need to simplify the logarithms. Note that there are now four types of start-ups.

Name	Status	Type of Start-up
Oculure	Successful	Virtual Reality
VRealest	Failed	Electric Cars
NightVishawn	Failed	Virtual Reality
Eelleeff	Failed	Electric Cars
Cool-Hip Start-Whip	Successful	Alternative Nourishments
LowKeyz	Unknown	Alternative Nourishments
Power H. W.	Unknown	Virtual Reality

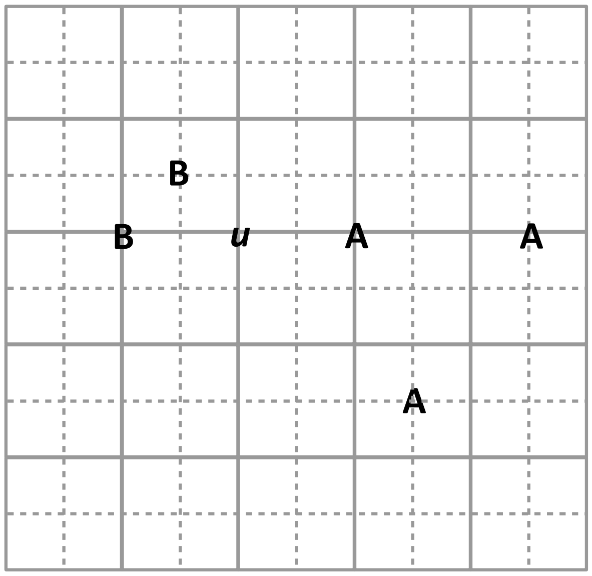
Show your work for partial credit.

Quiz 2, Problem 3: k-Nearest Neighbors (30 points)

Jon Snow knows nothing. He comes to you for help because he has a 6.034 final the next day and is terrified because he has no understanding of any K-nearest neighbors concepts. As the resident 6.034 expert, you decide to help him by solving some practice problems that he is struggling with.

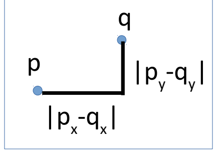
Part A: k-Nearest Neighbors (12 points)

Using two different distance metrics, **Manhattan distance** and **Euclidean distance** (both defined below, as a reminder), and differing values of k, help Jon fill out the table below indicating the classification of ***u*** as one of the following: **A, B,** or **UNKNOWN.**



	Manhattan Distance	**Euclidean Distance**
1-Nearest Neighbors		
3-Nearest Neighbors		
5-Nearest Neighbors		

The **Manhattan distance** between two points is defined as $D(\vec{p},\vec{q}) = |p_x - q_x| + |p_y - q_y|$, so the distance between two points is measured by adding up two line lengths as shown in the figure to the right.

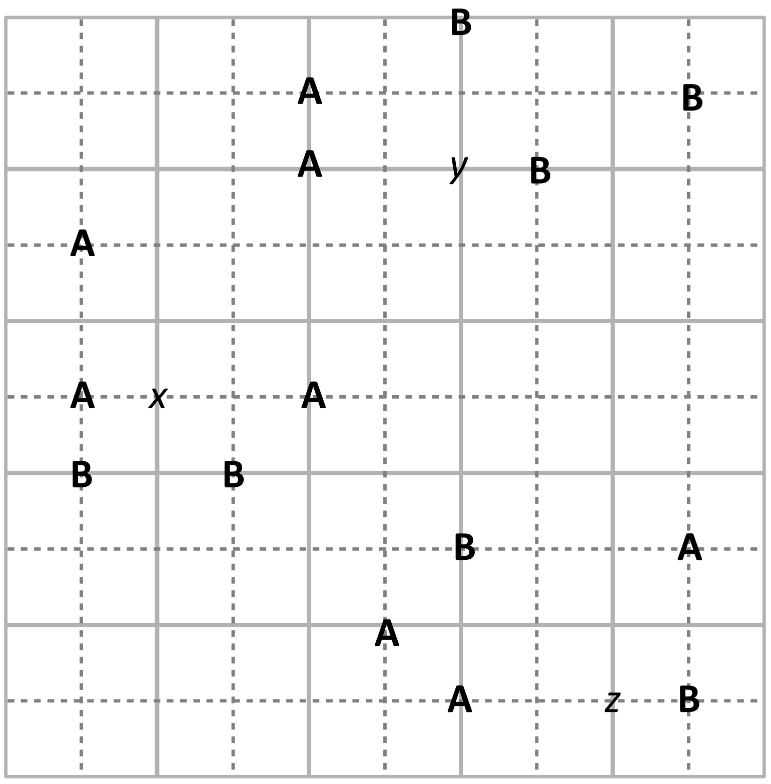


The **Euclidean distance** between two points is defined as $D(\vec{p},\vec{q}) = \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2}$.

Part B: Cross Validation (8 points)

Good work! Jon is already feeling more prepared for his exam. However, he still needs your help to find the best value of **k** using cross-validation with the three points ***x***, ***y***, and ***z*** as the test set. The actual classifications of the three points are:

x: A, *y*: B, *z*: A



B1 (6 points) For each value of **k**, indicate which of the three points (***x***, ***y***, ***z***) are misclassified, or write **NONE** if none of the three points were misclassified. Then, write the error rate (fraction of points misclassified) for each **k**. As usual, use Euclidean distance.

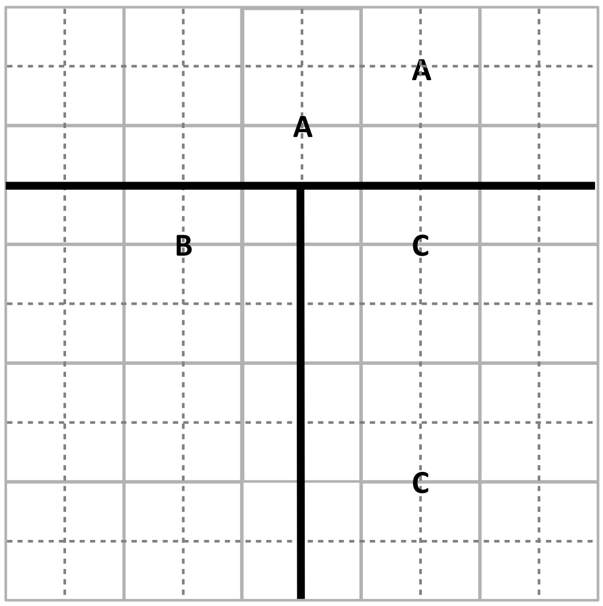
	Misclassified	**Error rate**
k = 1		
k = 3		
k = 5		

B2 (2 points) Based on your cross-validation results, what is the best value of **k**? (Circle one)

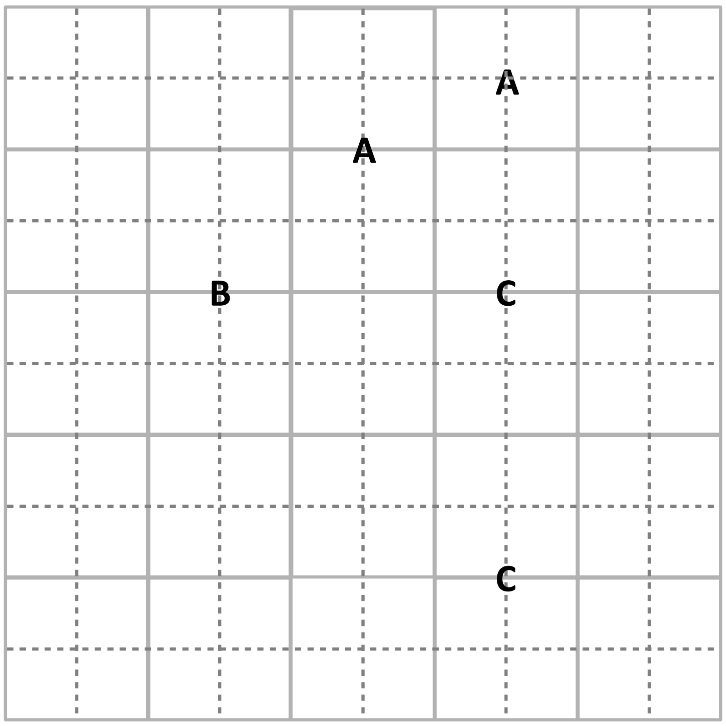
k = 1   **k = 3**   **k = 5**

Part C: Jon Snow Knows No Boundaries (10 points)

Jon sees a problem about drawing a decision boundary and immediately draws what he thinks is the correct boundary:



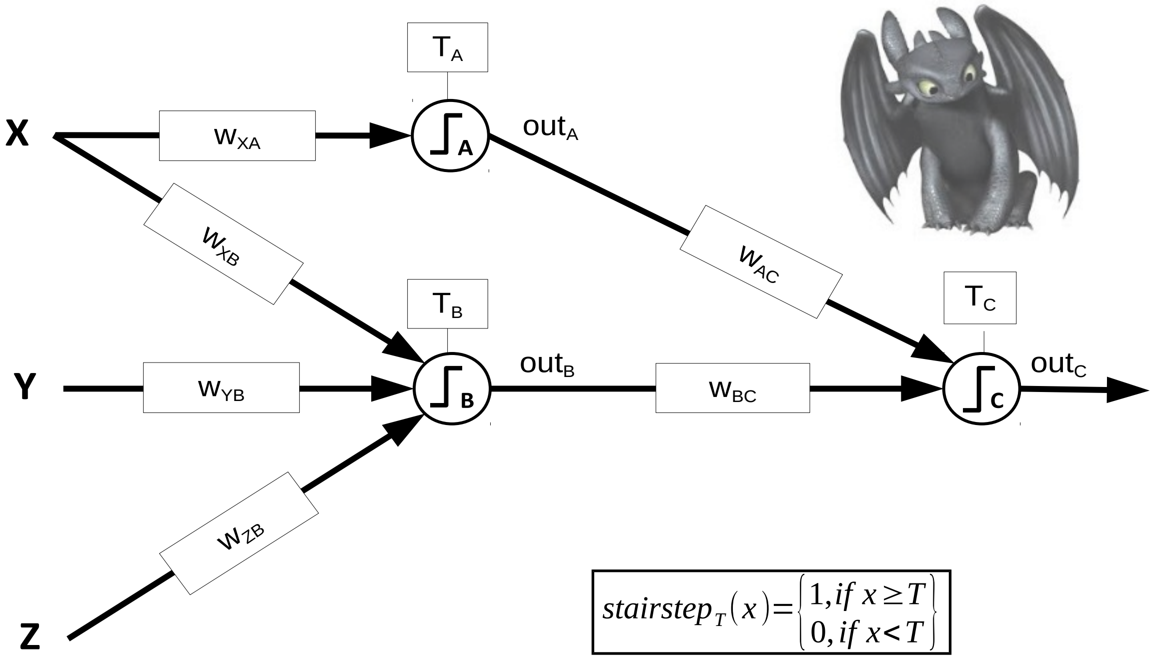
However, as we all know, Jon Snow knows nothing. As the decision-boundary expert, help Jon by drawing the actual **1-nearest neighbor decision boundary.** As usual, use Euclidean distance.



Quiz 3, Problem 1: Neural Networks (50 points)

Part A. Forward Propagation (40 points)

Major film company Disni is thrilled about the resounding success of its feature film *How to Train Your Network*, starring Hiccup, the adventurous machine learning enthusiast. Disni has hired you to help them produce a sequel! Currently, they are trying to characterize the following model:



$$stairstep_T(x) = \begin{cases} 1, if\ x \geq T \\ 0, if\ x < T \end{cases}$$

A1 (6 points) Which of the following six (6) logical functions can **neuron C** perform on its inputs **out$_A$** and **out$_B$**? Circle **ALL** that apply. Truth tables have been provided for your reference.

AND **OR** **XOR**

NAND **NOR** **NOT**

p	q	AND(p, q)
1	1	1
1	0	0
0	1	0
0	0	0

p	q	OR(p, q)
1	1	1
1	0	1
0	1	1
0	0	0

p	q	XOR(p,q)
1	1	0
1	0	1
0	1	1
0	0	0

p	q	NAND(p, q)
1	1	0
1	0	1
0	1	1
0	0	1

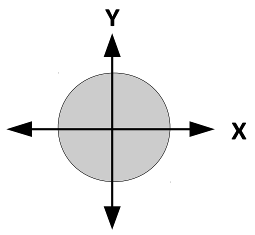
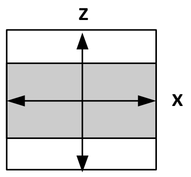
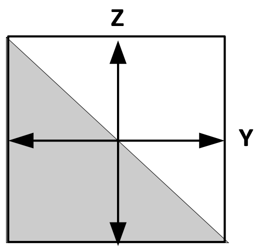
p	q	NOR(p, q)
1	1	0
1	0	0
0	1	0
0	0	1

p	NOT(p)
1	0
0	1

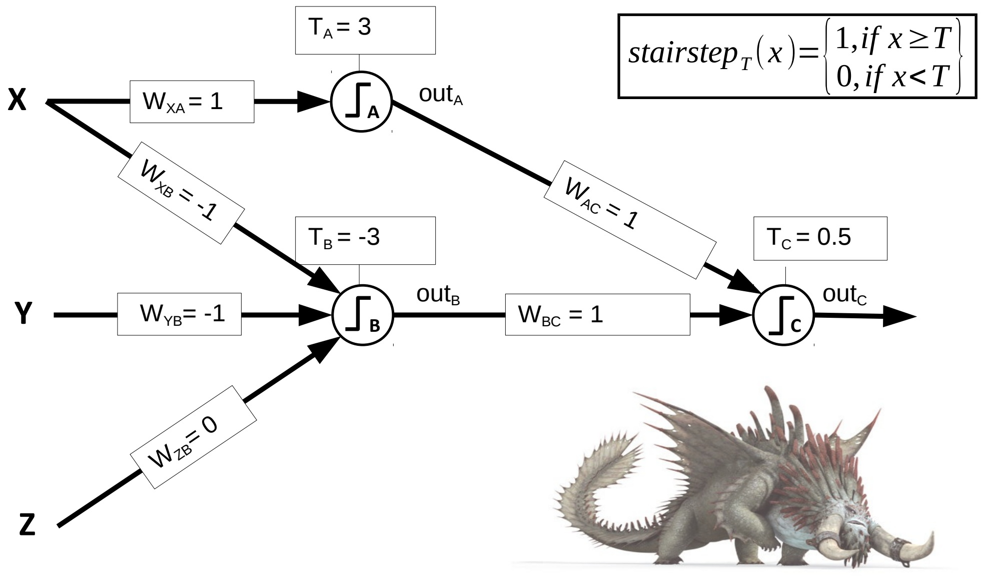
A2 (10 points) Consider **neuron B**. For *each* of the three pictures below, decide whether there exists some assignment of weights and thresholds which allow **neuron B** to draw the picture, in two different, unrelated scenarios:

- when $\mathbf{w}_{XB}$ can have any value you choose
- when $\mathbf{w}_{XB}$ must be equal to zero

In the images below, the shaded regions represent where neuron B outputs a 1. Non-shaded regions represent where neuron B outputs a 0. In each cell, *CLEARLY* write **YES** or **NO** to indicate whether or not **neuron B** could possibly draw the given picture with the indicated constraint on $\mathbf{W}_{XB}$.

		It is possible for neuron B to draw this picture…	
Description	**Picture**	**…when there is no restriction on W_{XB}.**	**…when W_{XB} = 0.**
A circle in the X-Y plane that shades everything inside it.			
Two horizontal lines in the X-Z plane that shade everything between them.			
A line in the Y-Z plane that shades everything below it.			

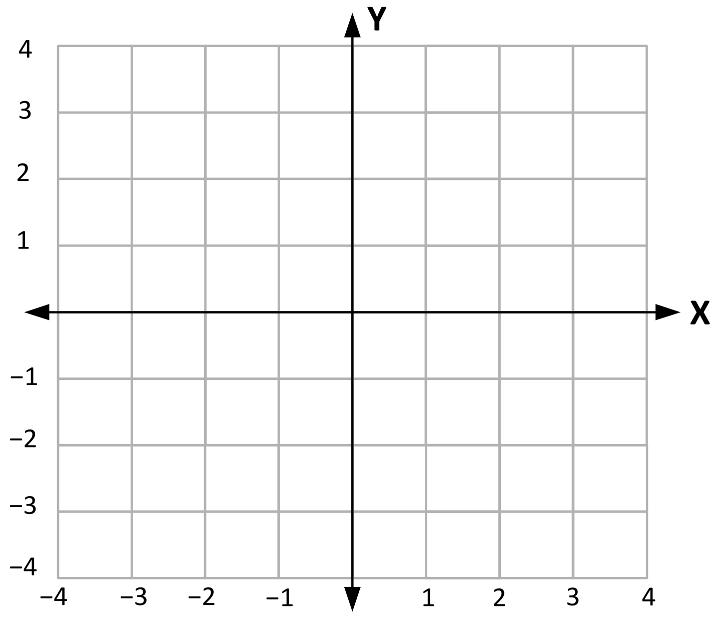
A3 (24 points) In the final scene of the film, Hiccup's trusty neural network has its weights scrambled by a malicious adversary! To rescue his network, Hiccup must understand its new output. The new weights are shown below.



$$stairstep_T(x)=\begin{Bmatrix}1, if\ x\geq T\\ 0, if\ x<T\end{Bmatrix}$$

On the grid below, draw how the neural network would divide up the space. **Shade the space where the output of the neural network is 1.** Do not shade the space where the output is 0. If you are having trouble, for partial credit you may discuss which logic function neuron C emulates.

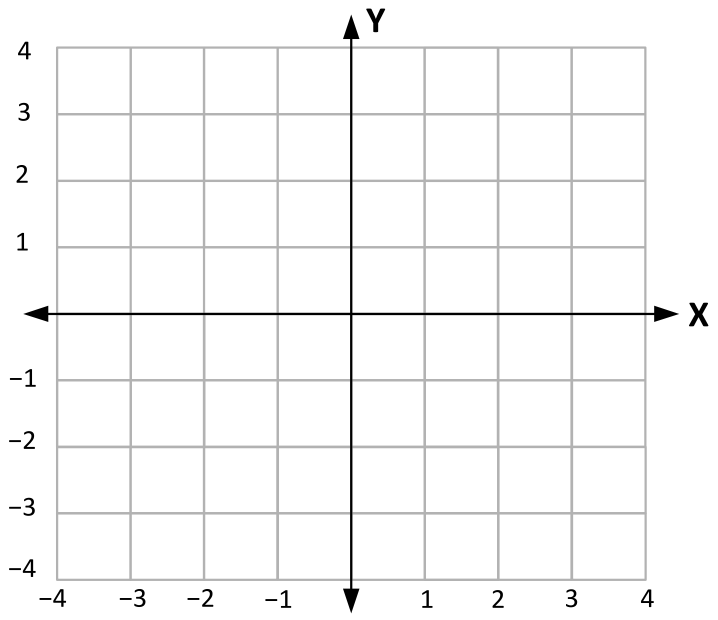
Space to show your work for partial credit is provided on the next page. If you want to start over, we have provided an extra copy of the grid on the next page.



Show your work for partial credit:

This is a duplicate copy of the grid on the previous page. If you want to have this copy graded instead, check the box:

☐ **I want to start over; grade this copy.**



Part B. Backward-Propagation (10 points)

The producers want to create a scene featuring deep neural network training, so they ask for your input on a few more concerns.

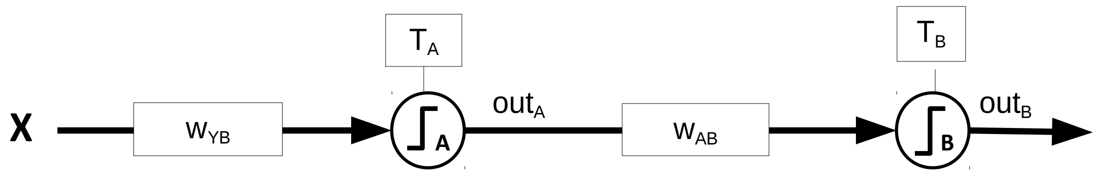
B1 (4 points) Disni is worried that training a deep neural network on screen will take up too much time, because back-propagation is complex. “Not to fear,” you tell them. “The weight-update algorithm is actually not that slow, because...” (circle the **one** best explanation):

1. It takes advantage of redundancies in the update equations so that downstream values are only computed once.
2. In most cases, the threshold trick speeds up the algorithm by an order of magnitude.
3. The step function is faster to compute and differentiate than the sigmoid which was used by scientists previously.
4. Nowadays, even a standard laptop computer can quickly compute 60,000,000 parameters.

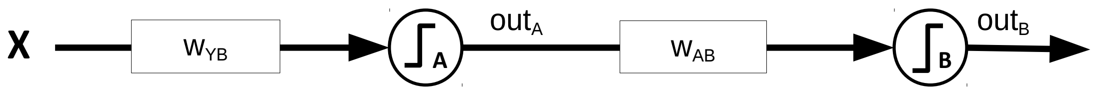
B2 (6 points) Disni doesn’t want to deal with updating both weights *and* thresholds in their back-propagation algorithm. However, you tell them that there is a natural way to transform a neural net with thresholds into an equivalent neural net without thresholds (this is known as the “threshold trick”).

There are *two* neural networks shown below. The first is a simple neural net with two threshold parameters; the second is a *skeleton* you should augment to result in new neural network equivalent to the first after applying the threshold trick. **Augment the skeleton network with neural network elements (wires, weights, thresholds, neurons, and/or outputs) as appropriate to demonstrate the threshold trick**:

Original Network (do not draw on this one):



Skeleton Network (augment this one by drawing things on it to make it equivalent to above):



(There is no quiz material on this page.)

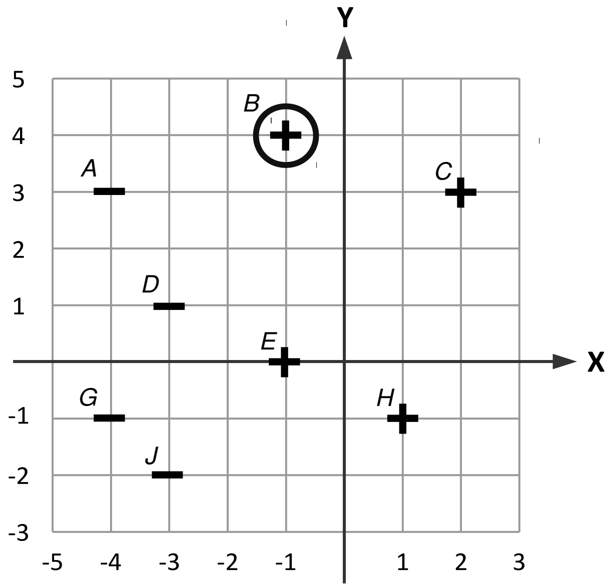
Quiz 3, Problem 2: Support Vector Machines (50 points)

Eve is on her way to the Olaf's Winter Wonderland, a Christmas tree lot, to pick out a tree for her living room! Eve knows the tree lot has two types of trees, LIVE trees and FAKE trees, and she wants to use her recently-mastered knowledge of 6.034 to create SVMs that can distinguish between the two types of trees.

Part A: Eve's Christmas Classifier (29 points)

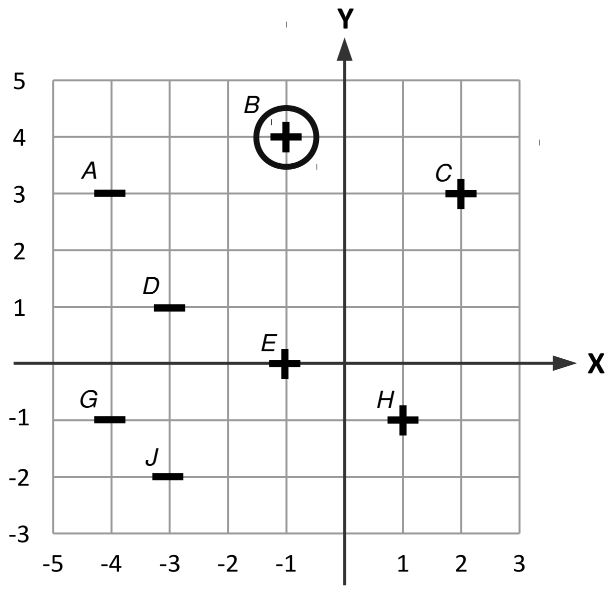
Upon arriving at the tree lot, Eve is relieved to see that the Christmas trees are perfectly separable by a linear SVM!

A1 (14 points) The diagram below shows a map of the trees on the lot. The trees are arranged on a grid, and each tree is labeled for future convenience. LIVE trees are **positive (+)** samples and FAKE trees are **negative (–)** samples. Furthermore, we have circled **one** of the support vectors (sample **B**) for you.



If you want to have this copy graded instead, check the box:

☐ **I want to start over; grade this copy.**

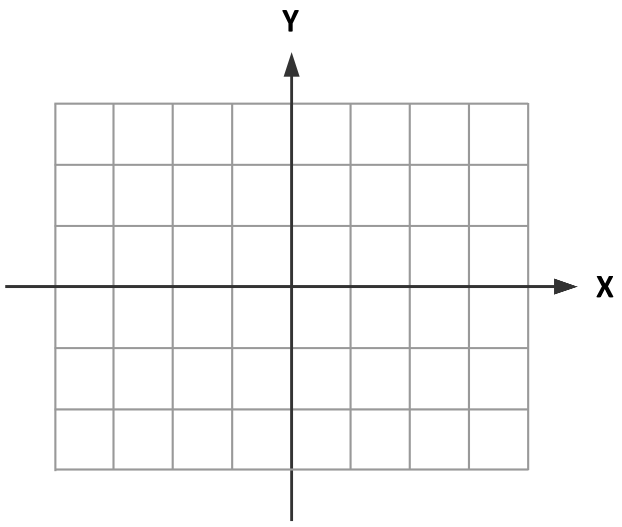


On the diagram above, with the knowledge that point B is a support vector,

- draw the linear SVM boundary with a **solid line**,
- draw the positive and negative gutters with **dashed lines**,
- and **circle** all other support vectors.

We have provided a duplicate copy of the graph for you above. If applicable, please ***clearly*** indicate which copy of the graph you would like us to grade.

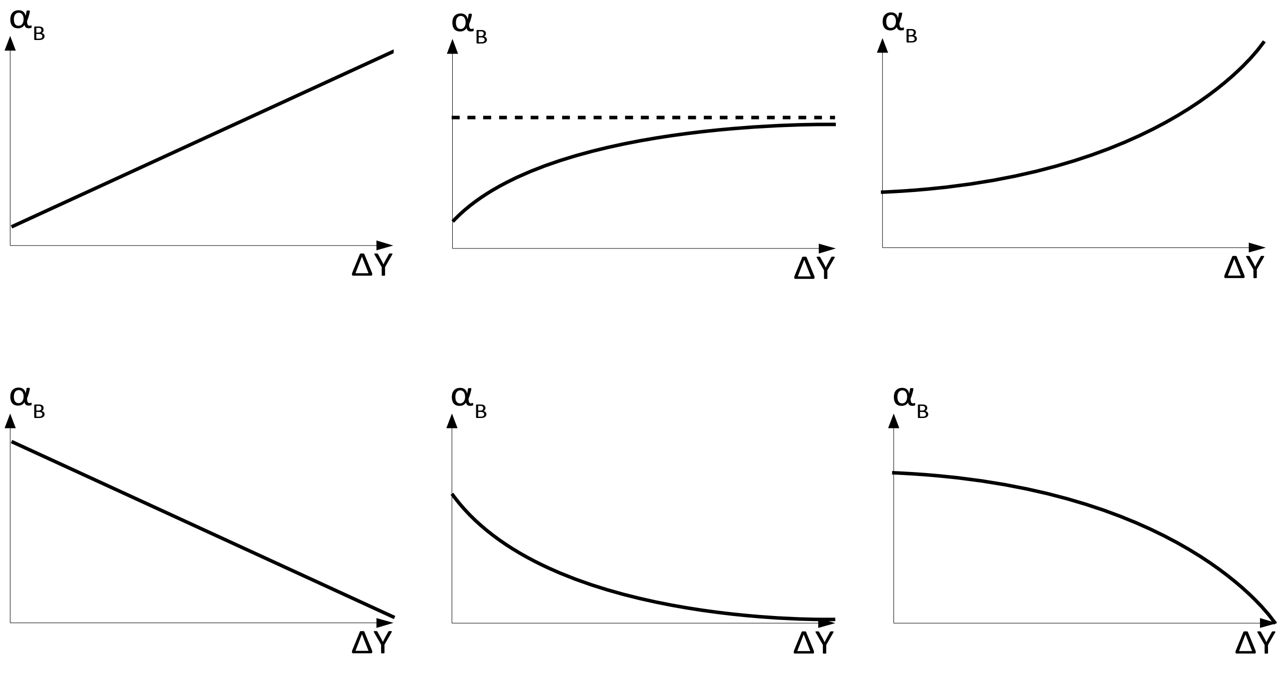
A2 (3 points) Based on the boundary you drew in A1, in what **direction** does the vector $\vec{w}$ point? On the graph below, **draw a vector from the origin pointing in the correct direction**. You do not need to compute or illustrate the length of $\vec{w}$. *You should not need to do any calculations to answer this question.*



A3 (4 points) Eve wonders about what the relative values of α_B and α_E mean conceptually, given the boundary you drew in A1. Her curiosity leads her to think about the relative *importance* of points B and E in determining the SVM boundary. In the sentence below, there are three possible ways to fill in the blank. Help Eve by circling the **one** best entry indicating how points B and E compare:

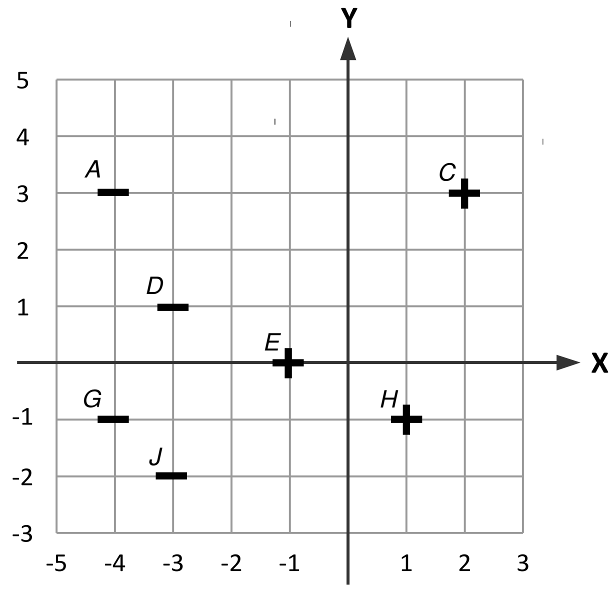
	more important than	
Point B is...	**equally important as**	...point E in determining the boundary and gutters of the SVM.
	less important than	

A4 (4 points) Suppose that point B were **moved in the positive Y-direction**. As a function of the displacement in the positive Y-direction (ΔY), how does B's supportiveness value (α_B) change, assuming that B continues to be a support vector? Circle the **one** graph that *best* illustrates the relationship between ΔY and α_B .



A5 (4 points) Oh no! A customer just bought the tree located at point B, and is removing it from the lot (in other words, sample B has been **removed** from the training data). Eve is worried that she will have to re-do all of her SVM calculations. Supposing that Eve re-trains her SVM **without sample B**, will the SVM decision boundary or gutters change? Circle the **one** best answer below.

For your convenience, we have printed the new graph below.



YES, the SVM decision boundary or gutters (or both) will change.

NO, neither the SVM decision boundary nor the gutters will change.

Part B: Kris Kringle's Kernels (9 points)

As Eve pays for her tree at the cash register, she sees a printed advertisement to win a Kris Kringle figurine from Olaf's Winter Wonderland. Drawn on the advertisement is a beautiful arrangement of Balsam Fir **(F)** and Scotch Pine **(S)** Christmas trees! She decides she wants to sketch an SVM that can classify this complex arrangement of trees.

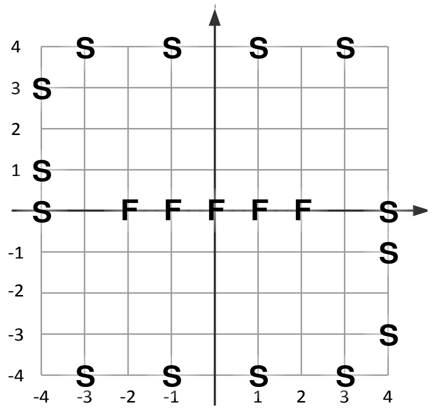
For each of the three types of kernels listed below, circle either **YES** or **NO** indicating whether Eve could train an SVM with a kernel of that type to *perfectly* classify the trees.

If YES, <u>sketch a decision boundary</u> that such a classifier could produce.

Linear

YES

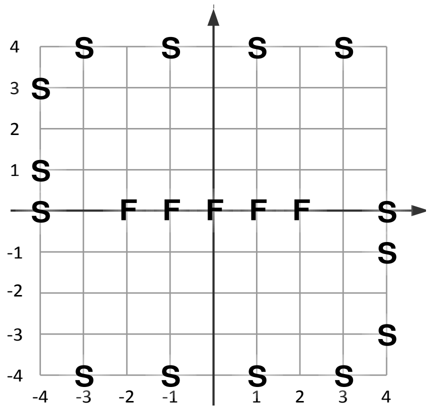
NO



Quadratic

YES

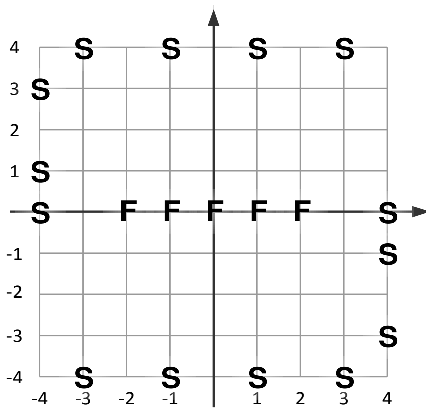
NO



Radial Basis Function (RBF)

YES

NO



Part C: Jake's Parting Present (12 points)

Your caring TA has decided to leave you one final gift: the gift of miscellaneous SVM questions that can't fit into another category. You can use these to impress your peers, professors, and even pets! (These questions are independent from, and do not rely on, the previous sections.)

C1 (4 points) Consider a linear SVM that classifies points on a number line (i.e. in 1 dimension). Assuming there is at least one positive sample and one negative sample, what are the *possible* numbers of support vectors for this linear SVM **after training has terminated**? What if the linear SVM classifies points on a plane (i.e. in 2 dimensions)? For 1D and 2D, **circle ALL possible numbers of support vectors for a fully-trained SVM.**

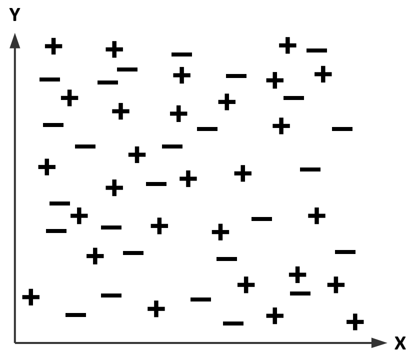
1D (line):	1	2	3	4	5	6	7	8
2D (plane):	1	2	3	4	5	6	7	8

C2 (4 points) This question asks about SVM kernels *in general*, not about any particular SVM architecture or application. For each of the two functions below, explain why that function would **not** be a good choice as a kernel function:

Function	**Explanation (fewer than 15 words)**
$K(\vec{u},\vec{v}) = 1$	
$K(\vec{u},\vec{v}) = \|\vec{u}\|$	

C3 (4 points) As part of a Secret Santa gift exchange, Professor Winston gives you a challenging puzzle:

Given the following graph, can you design an SVM that perfectly classifies the data?



After weeks of crunching numbers, you successfully create an **SVM model with a Radial Basis Function (RBF) kernel** that perfectly classifies all of the data! However, when you plug in a *new* test point to be classified by the model, you find that the model incorrectly classifies it. Indeed, you find that your model incorrectly classifies roughly half of the new test vectors that are input into the model! Of the options below, circle the **one** that best describes what likely went wrong in this situation:

1. RBF kernels are notorious for causing SVMs to underfit to their training data.
2. The training algorithm did not run for long enough; the **α** values have not fully been trained, which means the relative weights of the support vectors are invalid.
3. There are not enough support vectors in the system: because **σ** (the RBF parameter) is so small, the tight boundaries preclude there being sufficient support vectors.
4. The trained model doesn't fully taken into account the 2D features of the training points.
5. The SVM trained on data in which there there is no regularity; it presumes order from otherwise random training points.

Bonus Question (1 extra point)

This question is extra credit. Do not spend too much time on it. Getting this question wrong will not negatively impact your grade.

Recall that a kernel function $K(\vec{u},\vec{v})$ is defined as the dot product between two transformed vectors $\Phi(\vec{u})$ and $\Phi(\vec{v})$. Consider some three-dimensional vector $\vec{x}=\langle x_1, x_2, x_3\rangle$, and suppose the feature transformation $\Phi(\vec{x})$ is defined as $\Phi(\vec{x}) = \langle x_1, x_2^2, x_3\rangle$. What is the kernel function $K(\vec{u},\vec{v})$ associated with this feature transformation?

$$K(\vec{u},\vec{v}) = \boxed{}$$

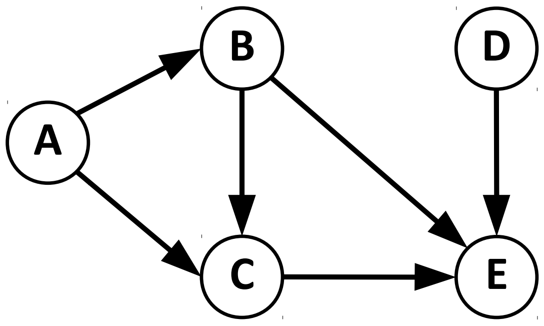
Quiz 4, Problem 1: Bayesian Inference (50 points)

Part A: Parameters in Bayes Nets (12 points)

For each Bayes net described below, answer this question: **Assuming all of the variables are boolean, what is the minimum number of parameters in the Bayes net?** (For each net, the number of parameters is the total number of entries in all conditional probability tables.)

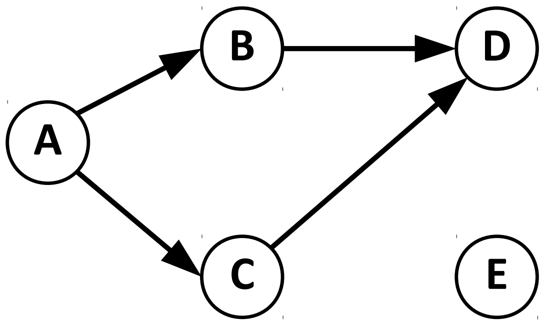
You may show your scratch work for partial credit.

A1 (4 points)



Number of parameters = $\boxed{}$

A2 (4 points)



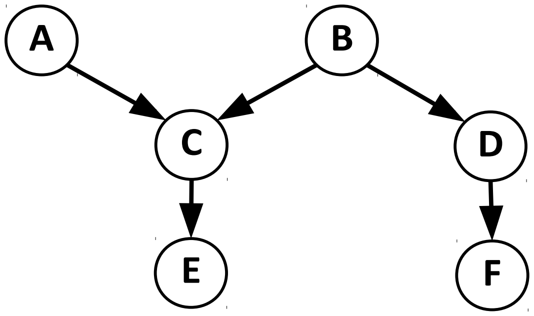
Number of parameters = $\boxed{}$

A3 (4 points) A Bayes net with five variables, in which ALL variables are assumed to be independent.

Number of parameters = $\boxed{}$

Part B: Independence by d-Separation (12 points)

Consider the Bayes net below, containing 8 variables. Assume that the only independence relations that hold are exactly the ones enforced by the shape of the Bayes network. For each question below, show your work for partial credit.



B1 (4 points) Is A marginally independent of D? (Circle one)

YES

NO

Show your work for partial credit.

B2 (4 points) Is P(A|CD) = P(A|C)? (Circle one)

YES

NO

Show your work for partial credit.

B3 (4 points) Is P(E|D) = P(E)? (Circle one)

YES

NO

Show your work for partial credit.

Part C: Probability in Bayes Nets (8 points)

Here is a Bayes net with 4 boolean variables and their associated probability tables. Each probability is represented by a lowercase variable.

P(A)
a

$$\begin{tikzcd}
A \arrow[r] \arrow[d] & C \arrow[d] \\
B \arrow[r] & D
\end{tikzcd}$$

A	P(C|A)
T	p
F	q

A	P(B|A)
T	f
F	g

B	C	P(D|BC)
T	T	w
T	F	x
F	T	y
F	F	z

Write an expression for $P(AB\overline{C}\,\overline{D})$ in terms of the variables specified in the Bayes net.

$$P(AB\overline{C}\,\overline{D}) = \boxed{}$$

Show your work for partial credit:

Part D: Specificity and Sensitivity (18 points)

Last week, MIT freshman Bryce SoCal saw and felt snow for the first time! It was an awe-inspiring experience for him, but his roommate Yuri Siberia didn't even seem to notice the snow. Intrigued, Bryce develops a model to predict whether an MIT freshman has seen snow before coming to college, based on their hometown.

Bryce determines that:

- 75% of MIT freshmen have seen snow before coming to college,
- his model is 80% specific (of the students who have seen snow, the model accurately predicts that 80% of the time), and
- his model is 99% sensitivity (of the students who haven't seen snow, the model accurately predicts that 99% of the time).

For each freshman, let **S** be the event that they have seen snow before, and let **M** be the event that Bryce's model predicts they have seen snow. Then:

- $P(S)=0.75$
- $P(M \mid S)=0.80$
- $P(\overline{M} \mid \overline{S})=0.99$

D1 (9 points) Out of 100 randomly selected freshmen, how many will Bryce's model predict as having seen snow, regardless of whether each freshman has actually seen snow?

The expected number of freshmen predicted to have seen snow is *approximately* (circle one):

25 30 35 40 45 50 55 60 65 70 75

Show your work for partial credit:

D2 (9 points) Bryce's model predicts that Martin has never seen snow before college. Given that result, the probability that Martin actually had seen snow is *most nearly* (circle one):

0% 10% 20% 30% 40% 50% 60% 70% 80% 90% 100%

Show your work for partial credit:

Quiz 4, Problem 2: Adaboost (50 points)

You are collaborating with the MIT biology lab to help improve diagnosis of Traumatic Brain Injury (TBI). The biology lab has heard rumors of your 6.034 prowess and has requested your assistance in making a machine learning model.

Part A: Brain Boosting (23 points)

The biology lab has given you data from a pilot study of 6 different TBI tests, each run on 6 different human participants. Each test (a binary classifier) was found to have misclassified the presence of TBI in at least one of the participants.

Your goal is to use the data to create a machine learning model for improved diagnosis of TBI. Due to the nature of the data, you quickly realize Adaboost would be an ideal method.

To ensure that you come up with a model as objectively as possible, the researchers have blinded the blood test names with identifiers T1-T6, and the participant names with identifiers A-F. The table of data is given below. The lab emphasizes to you that the table contains **one row per weak classifier** and **one column per training point**, and **cells shaded in black represent points that the classifier misclassifies**.

Weak Classifiers	Selected Participants (Training Points)					
	A	B	C	D	E	F
T1		■			■	
T2	■	■	■	■	■	
T3	■			■		
T4	■					■
T5			■	■		
T6			■			

On the next page, perform three rounds of boosting with these classifiers and training data. In each round, **pick the classifier with the error rate furthest from 1/2**. Break ties by picking the classifier that comes first numerically.

In any round, if Adaboost would terminate instead of choosing a classifier, write **NONE** for the weak classifier (**h**) and for the voting power (**α**). Then, leave all remaining spaces blank.

Space to show work is provided on the next page.

	Round 1	Round 2	Round 3
weight A	1/6		
weight B			
weight C			
weight D			
weight E			
weight F			
Error rate of T1			
Error rate of T2			
Error rate of T3			
Error rate of T4			
Error rate of T5			
Error rate of T6			
weak classifier chosen (**h**)			
weak classifier error (**ε**)			
voting power (**α**)			

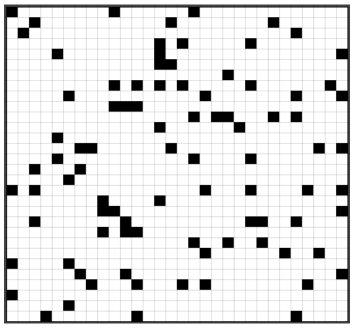
Show your work for partial credit:

Part B: Big-Picture Boosting (12 points)

After you develop your ensemble classifier in part A, the head researcher excitedly tells you that they have just finished compiling a massive dataset with the results of *hundreds* of different TBI tests and many different trial participants.

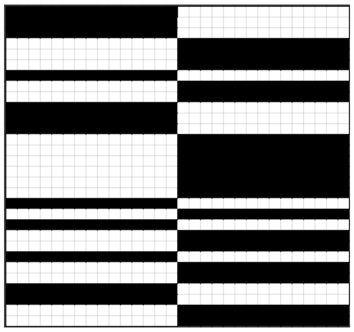
The lab gives you several tables of these results. **These tables are of the same format as the one in part A**, but are much bigger. Again, in every table, each row represents a particular binary test for TBI, and each column represents a trial participant. As before, a *shaded* cell means that the classifier **misclassifies** the presence of TBI in the subject.

For each of the below tables of test misclassifications, circle **YES** or **NO** indicating whether or not the weak classifiers (TBI tests) presented in the table **could be used in boosting to somehow create a *perfect* ensemble classifier H**:



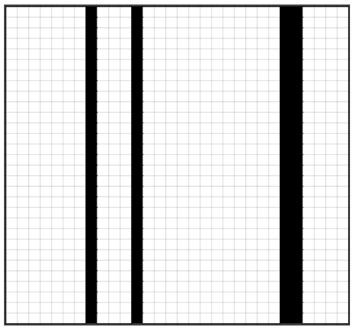
YES

NO



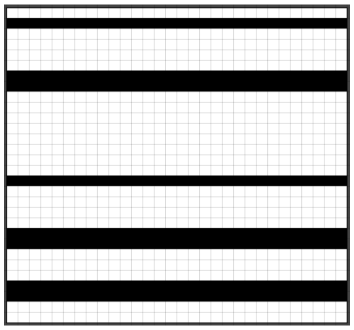
YES

NO



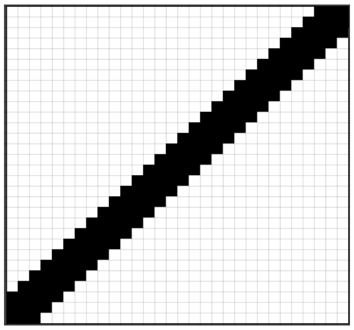
YES

NO



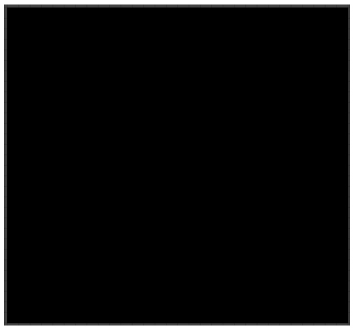
YES

NO



YES

NO



YES

NO

Part C: Conceptual Questions (15 points)

This section asks questions about Adaboost *in general*—these questions do not rely on the preceding section. For True/False questions, circle the **one** best answer. For short-answer questions, answer in 15 words or fewer.

C1 (3 points) If two imperfect classifiers misclassify *disjoint* sets of points, it is possible to assign voting powers to them in a way to create a perfect ensemble classifier.

TRUE **FALSE**

C2 (3 points) A weak classifier that misclassifies exactly half of the training points will never be chosen in the first round of boosting.

TRUE **FALSE**

C3 (3 points) The same weak classifier can be chosen in more than one round while performing boosting.

TRUE **FALSE**

C4 (3 points) The following assignments to weights is **not possible** in any round of boosting. Explain why.

Training Points	**J**	**K**	**L**	**M**	**N**
Weights	3/16	1/8	1/8	7/16	1/8

Explanation (15 words or fewer):

C5 (3 points) The following assignments to weights is **not possible** in round 2 of boosting. Explain why.

Training Points	**W**	**X**	**Y**	**Z**
Weights	1/4	1/8	1/8	1/2

Explanation (15 words or fewer):