1 SATurdays are for everyone

1. Determine whether the sentences below are satisfiable or unsatisfiable (using any method you like).

   (a) $(\neg(y \vee \neg y) \vee x) \wedge (x \vee (z \iff \neg z))$

   (b) $\neg(x \vee \neg(x \wedge (z \vee T))) \implies \neg(y \wedge (\neg y \vee (\top \implies \bot)))$

   (c) $((\top \iff \neg(x \vee \neg x)) \vee z) \vee z) \wedge \neg(z \wedge ((z \wedge \neg z) \implies x))$

2. Suppose $A \models B$. Which of the following statements must be true for all truth assignments to A and B?

   (a) $A \wedge B$
   (b) $A \Rightarrow B$
   (c) $B \Rightarrow A$
   (d) $A \vee B$
   (e) B

3. How would we formulate the SAT problem as a CSP? What are the variables? Domains? Constraints?

4. Suppose we have an algorithm which determines whether a sentence is satisfiable or not. Given two sentences A, B, how could we determine whether $A \models B$?

5. Determine whether the sentence below is satisfiable or unsatisfiable using DPLL. Break ties by assigning variables in alphabetical order, starting with **false**. If satisfiable, what model does the algorithm find?

$$(A \vee B) \wedge (B \vee C \vee D) \wedge (\neg A \vee \neg B \vee C) \wedge (\neg A \vee \neg C \vee \neg D) \wedge A \wedge (C \vee \neg D)$$

2 All About Logic

1. Propositional Logic

   (a) Vocab check: are you familiar with the following terms?

      i. Symbols
      ii. Operators
      iii. Sentences
      iv. Equivalence
      v. Literals
      vi. Knowledge Base
      vii. Entailment
      viii. Query
      ix. Satisfiable
      x. Valid
      xi. Clause - Definite, Horn clauses
      xii. Model Checking
      xiii. Theorem Proving
      xiv. Modus Ponens

2. **Chipotle?**

   (a) Sean is a student in class. In his knowledge base, he admits that if he doesnt get bored, he will pass the class no matter what he does later on. However, if he gets bored and he ends up going directly to Chipotle, he believes he can always pass the class after getting food. He doesn't really use Facebook. Represent Sean's knowledge base in propositional logic. Let B be the symbol representing if Sean gets bored, C representing going to Chipotle, F representing using Facebook, and P representing passing the class. (Select all that can represent his Knowledge Base)

      i. $\neg B \Rightarrow P; B \wedge C \Rightarrow P$
      ii. $\neg B \Rightarrow P; \neg P \Rightarrow \neg C \vee F$
      iii. $B \vee P; (\neg B \wedge \neg C \wedge F) \vee P$
      iv. $B \vee P; (\neg B \vee \neg C) \vee P$

3. Given the following propositional logic clauses, show R must be true and using only the resolution inference rule to derive a contradiction. Your answer should be in the form of a graph, where each resolvent is connected by lines to its two parent clauses. Use the clauses below as the initial set of nodes in the graph.

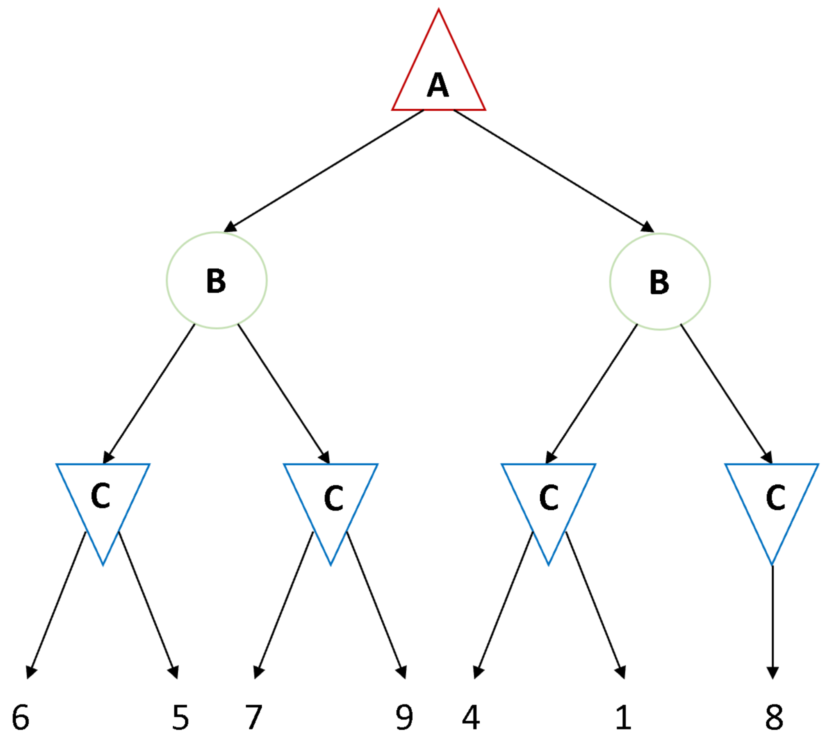
   Note: You do not need to use all the nodes, and you may use a node more than once.

$$P \vee Q \qquad\qquad \neg P \vee R \qquad\qquad \neg Q \vee R$$

3 Adversarial Search (Minimax+Expectimax Pruning)

So we've done pruning on minimax trees, but what happens when we introduce chance nodes? Recall that a chance node has the expected value of its children, and let each child have an equal probability of being chosen.

Perform pruning on the following game tree and fill in the values at letter nodes. A is a maximizer, B is a chance node, and C is a minimizer. Assume that values can only be in the range 0-9.



4 CSPs

You've generously saved a row of 6 seats in Rashid for 6 of your 15-281 classmates (A-F[1]), and are now trying to figure out where each person will be seated. You know the following pairs of people have some kind of binary constraint between them:

- A, B
- A, C
- A, D
- B, C
- B, E
- C, F

(a) Draw the constraint graph to represent this CSP.

(b) Some value is assigned to A. Which domains could change as a result of running forward checking for A?

(c) Some value is assigned to A, and then forward checking is run for A. Then some value is assigned to B. Which domains could change as a result of running forward checking for B?

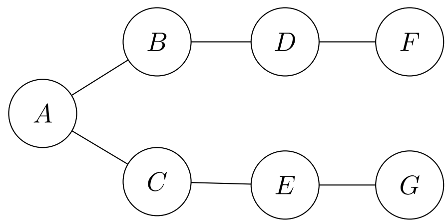
(d) Some value is assigned to A. Which domains could change as a result of running AC-3 after this assignment?

(e) Some value is assigned to A, and then arc consistency is enforced via AC-3. Then some value is assigned to B. Which domains will and will not change as a result of enforcing arc consistency after B's assignment?

[1] not correlated with their grades

(f) You are now trying a brand new algorithm to solving CSPs by enforcing arc consistency via AC-3 initially, then **after every even-numbered assignment** of variables (after assigning 2 variables, then after 4, etc.).

You have to backtrack if, after assigning a value to variable X, there are no constraint-satisfying solutions. Mathematically, for a single variable with d values remaining, it is possible to backtrack d times in the worst case. For the following constraint graph, assume each variable has domain of size d. How many times would you have to backtrack in the worst case for the specified orderings of assignments?



i. ABCDEFG:

ii. GECABDF:

iii. GFEDCBA:

(g) Recall the CSP from part (a). The actual constraints are as follows:
Both C and E want to sit next to B.
A wants to sit next to D, but not next to B or C.
F and C had a falling out over whether AI or blockchain was cooler, so there needs to be at least 2 seats between them.

B gets to class first and sits down in seat 3. Run AC-3 to determine the final seating arrangement.

5 Search (Algorithms & Properties)

(a) When can we guarantee completeness and optimality (if ever) for each of the following search algorithms we've seen in class? For each algorithm, indicate under what conditions it is complete and/or optimal.

Algorithm	**Complete**	**Optimal**
Breadth-First		
Depth-First		
Iterative Deepening		
A*		

(b) Consider a dynamic A* search where after running A* graph search and finding an optimal path from start to goal (assuming there's only one goal), the cost of one of the edges $X \rightarrow Y$ in the graph changes. Instead of re-running the entire search, you want to find a more efficient way of returning the optimal path for this new search problem.

For each of the following changes, describe how the optimal path cost would change (if at all). If the optimal path itself changes, describe how to find the new optimal path. Denote c as the original cost of $X \rightarrow Y$, and assume $n > 0$.

i. c is increased by n, $X \rightarrow Y$ is on the optimal path, and was explored by the initial search.

ii. c is decreased by n, $X \rightarrow Y$ is on the optimal path, and was explored by the initial search.

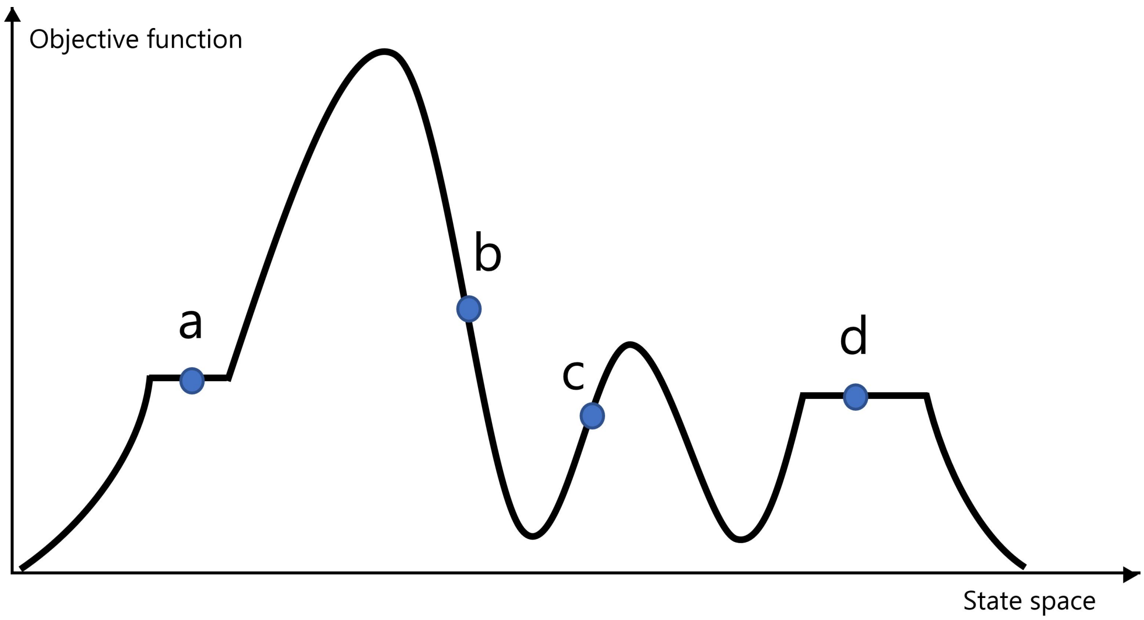
iii. c is increased by n, $X \rightarrow Y$ is not on the optimal path, and was explored by the initial search.

iv. c is decreased by n, $X \rightarrow Y$ is not on the optimal path, and was explored by the initial search.

v. c is increased by n, $X \rightarrow Y$ is not on the optimal path, and was not explored by the initial search.

vi. c is decreased by n, $X \rightarrow Y$ is not on the optimal path, and was not explored by the initial search (assuming the edge weights c can't go negative.)

6 Local Search



Consider how each of the following searches performs in state space above. Recall that in the context of local search, our goal is to find the state that optimizes the objective function.

(a) Hill-climbing search with start state c

Does it terminate? If so, where?

Is it complete?

(b) Random-restart hill climbing with randomly generated initial states a, d, and b

Does it terminate? If so, where?

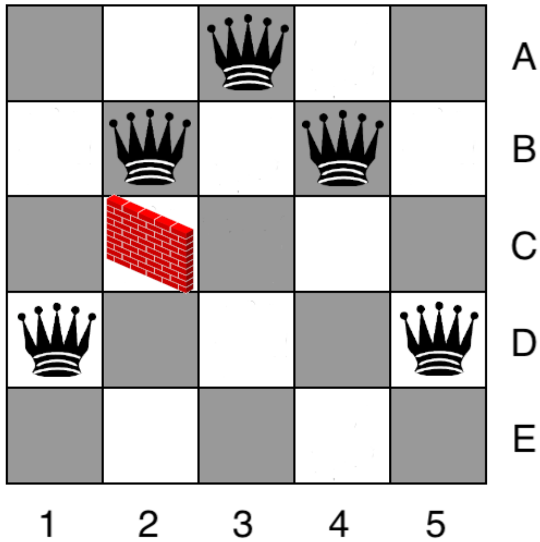
Is it complete?

(c) Stochastic hill-climbing that allows sideways moves with start state d

Does it terminate? If so, where?

Is it complete?

Now, let's revisit the 5-QUEENS problem. Our goal is to try to place 5 queens on a 5x5 chessboard with no conflict between any two queens. However, this time, there is a twist: our chessboard has a wall!



If the wall blocks the path between two queens (horizontal, vertical, or diagonal) then those two queens will not be in conflict. The wall cannot be moved, and a queen cannot pass through the wall.

Below is some pseudocode to try to solve the 5-QUEENS problem with hill climbing.

```
function 5-QUEENS-HILL-CLIMBING(problem)
    current_state = problem.INITIAL-STATE
    loop do
        neighbor = state with lowest # of conflicts between queens obtained by
                   making one legal chess move from current_state
        if neighbor.num_conflicts >= current_state.num_conflicts
            return current_state
        current_state = neighbor
```

Using the given starting state, pseudocode, and the rules regarding walls, answer the following questions.

(a) Run 5-QUEENS-HILL-CLIMBING on the given start state until completion. Where do the queens end up? Is this a goal state?

(b) If we remove the brick wall, will 5-QUEENS-HILL-CLIMBING end up in a global optima?