Homework 2 [100pts]

General Information

This homework is due by 11:59 PM EST on Wednesday, October 20, 2021. Late homework will be penalized 10% per day (where each day starts at 12:00 AM on the due day). Homework turned in after three days will not be accepted.

Please familiarize yourself with Gradescope if you have never used it before. In particular, when you submit an assignment, Gradescope asks you to select pages of your solution that correspond to each problem. This is to make it easy for graders to find where in your work they should grade. If you mislabel and lose points, you can submit a regrade request, but we will only give you some of your points back!

As a starting point, here is a link to the Google document with this homework. Make a copy of this, add your answers, and convert it to a PDF (with your name). Submit this PDF to the corresponding assignment on Gradescope. You can submit this as many times as you would like before the deadline. For the coding questions, submit your code files (*.py) to the Gradescope autograder. Do not modify any of the imports in these files. You will be graded on both visible and hidden test cases. Thus, you should make sure your solution is always correct (possibly by testing it on other cases that you write) and not just whether it succeeds on the visible test cases. This is an important skill for a practicing computer scientist! You can submit to the autograder as many times as you would like before the deadline. While you will not be graded on style, we encourage you to uphold best-practice style guidelines. For more information about the coding refer to section 2, the coding portion of the homework.

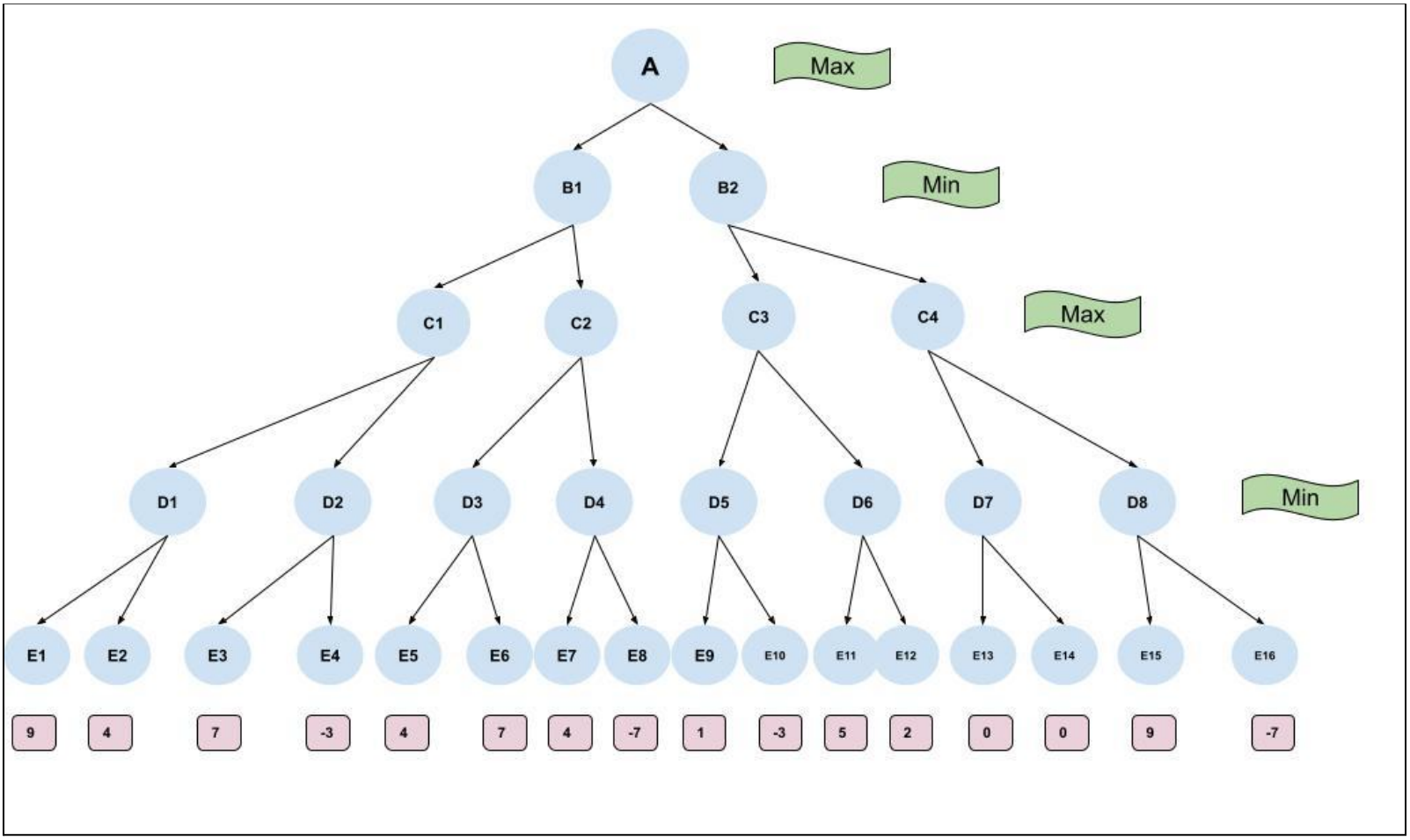
All solutions must be your own.

EECS 492 - Introduction to AI
Fall 2021

Section 1: Written Portion [72pts]

Problem 1: Adversarial Search [10pts]

a. [8pts] Apply minimax to the tree below. Please provide an annotated version of the tree. *You can use either a screenshot of the image provided below or save the image in the homework folder. Add this to your solution document.*



b. [2pts] If we use alpha-beta pruning (from left to right), which nodes could we omit? Please state ***every node*** that would be omitted.

Problem 2: Entailment and Equivalence [10pts]

State whether each of the following statements is correct and provide a justification.
(*Hint: You may find it useful to construct a truth table for each part.)*

a. [2pts] $(A \vee B) \wedge C \models A \vee B$
b. [2pts] $(A \wedge B) \models (A \Leftrightarrow B)$
c. [2pts] False $\models$ True
d. [2pts] True $\models$ False
e. [2pts] $(C \vee (\neg A \wedge \neg B)) \equiv ((A \Rightarrow C) \wedge (B \Rightarrow C))$

Problem 3: Valid, Satisfiable, or Unsatisfiable [10pts]

State whether each of the following sentences is valid, satisfiable but not valid, or unsatisfiable. Provide a justification using truth tables, using a proof, and/or by simplifying the sentences using equivalence rules. If the statement is satisfiable but not valid, please support your reasoning with a counterexample.

a. [2pts] $\neg A \Rightarrow A$
b. [2pts] $(A \Rightarrow B) \Rightarrow (\neg B \Rightarrow \neg A)$
c. [2pts] $\neg[(A \wedge B) \wedge \neg(A \Rightarrow B)]$
d. [2pts] $(\neg A \wedge B) \Rightarrow (A \Rightarrow B)$
e. [2pts] $[(\neg A \Rightarrow B) \wedge (\neg A \wedge B)] \leftrightarrow (A \vee \neg B)$

Problem 4: First Order Logic [10pts]

Consider the following vocabulary:

1. *Cities(c): c is a city, valid constants are {New York, San Francisco, Seattle, Detroit}*
2. *Person(p): p is a person, valid constants are {Barrett, Haoyi}*
3. *Travels(p, c): person p travels to city c*
4. *Neighbor(p1, p2): person p1 follows person p2 on a social network*
5. *Friend(p1, p2): person p1 is a friend of person p2*

Translate the following sentences into first-order logic:

a. [2pts] Haoyi has a friend who travels to New York.
b. [2pts] Every person travels to a city.
c. [2pts] Barrett does not have any friends who travel to Seattle.
d. [2pts] There is a person who travels to San Francisco whose followers all travel to Detroit.
e. [2pts] At most one person travels to San Francisco. (*hint: if there are two variables which both represent someone who travels to* San Francisco*, what must be true about those two variables?*)

Problem 5: Conjunctive Normal Form [12pts]

Convert the following first-order logic statements into Conjunctive Normal Form (CNF). Show your work accompanied with brief comments explaining each step.

a. [2pts] $\exists x\, A(x) \Rightarrow B(x)$
b. [4pts] $\forall x\, [\, [\, \exists y\, A(x, y) \wedge B(y)\,] \Rightarrow C(x)\,]$
c. [6pts] $\exists x\, [\, [\, \exists y\, A(x, y) \wedge B(y)\,] \Rightarrow [\, \forall y\, C(x, y)\,]\,]$

Problem 6: Unification [6pts]

Determine whether or not the following pairs can be unified. If they can be unified, present the most general unifier and show the result. If they cannot be unified, explain why. Use the convention that variables are lowercase letters while constants are capital letters. Functions are represented as either capital letters or as “ManufacturerOf” or “EmployerOf” or “FriendOf” or “GameOf”. Use the unique-names assumption.

a.	P(x, y, y)	P(A, B, C)
b.	P(B, ManufacturerOf(x))	P(B, EmployerOf(x))
c.	C(B,FriendOf(B), z)	C(B, FriendOf(B), GameOf(FriendOf(B)))
d.	P(x, y)	P(A, FriendOf(B))
e.	Q(B, A)	R(x, FriendOf(y))
f.	Q(B, A)	Q(x, FriendOf(B))

Problem 7: Resolution and Refutation [14pts]

Consider the following sentences in our knowledge base:

1. ***Coco is a puppy.***
2. ***Cristina is a dog lover.***
3. ***All puppies are dogs.***
4. ***If any puppy follows directions, that puppy is good.***
5. ***There is a dog that follows directions.***
6. ***If any dog follows directions, Coco follows directions.***
7. ***Whenever a dog lover finds a dog, if the dog is good, the dog lover adopts the dog.***
8. ***Cristina finds Coco.***

a. [4pts] Translate the above knowledge base into Conjunctive Normal Form, and give each sentence a number from 1 to 8.
   Use the following conventions:
   1. *Puppy(x) – x is a puppy.*

2. *Dog(x) – x is a dog.*
3. *Good(x) – x is good.*
4. *DogLover(x) – x is a dog lover.*
5. *FollowsDirections(x) – x follows directions.*
6. *Finds(x, y) – x finds y*
7. *Adopts(x, y) – x adopts y*

b. [10pts] Use resolution-refutation to **prove that Cristina adopts Coco** by filling in the rest of the table below. In the Number column, write the number of the sentence in the KB. In the Sentence column, write the resulting sentence you are adding to the KB. In the Resolution and Substitution column, write which two sentences you are resolving together, and what you are substituting for the variables in the two sentences, if necessary.
Note: You may require fewer or greater rows than in the table provided below.

Number	Sentence	Resolution and Substitution
9	¬Adopts(Cristina, Coco)	Assume negation.

EECS 492 - Introduction to AI
Fall 2021

Section 2: Coding Portion [28pts]

Constraint Satisfaction

In this problem, you will write an algorithm to solve the popular logic puzzle Sudoku (https://en.wikipedia.org/wiki/Sudoku). You will implement the AC-3 algorithm to enforce arc consistency, and pair it with our implementation of backtracking search.

Consider the Sudoku puzzle as pictured below. Each variable is named by both its row and column. For example, the variable in the upper left corner is A1.

	1	2	3	4	5	6	7	8	9
A			3		2		6		
B	9			3		5			1
C			1	8		6	4		
D			8	1		2	9		
E	7								8
F			6	7		8	2		
G			2	6		9	5		
H	8			2		3			9
I			5		1		3		

	1	2	3	4	5	6	7	8	9
A	4	8	3	9	2	1	6	5	7
B	9	6	7	3	4	5	8	2	1
C	2	5	1	8	7	6	4	9	3
D	5	4	8	1	3	2	9	7	6
E	7	2	9	5	6	4	1	3	8
F	1	3	6	7	9	8	2	4	5
G	3	7	2	6	8	9	5	1	4
H	8	1	4	2	5	3	7	6	9
I	6	9	5	4	1	7	3	8	2

Each variable must be assigned a value from 1 to 9 subject to the constraint that no two cells in the same row/column/box may contain the same digit.

Before you begin, please complete these written questions:

1. [1pt] Consider the image above on the left. There are two types of variables, those with numbers listed and those without. Assume that variables with the number shown on the left can only have that value in their domain.
   List all of the variables (e.g., G4) in the bottom center box (the one that has a red border) and their corresponding domains after **only unary constraints** have been enforced. Do not enforce arc constraints yet.

2. [1pt] Reduce the domain for variables in this box by enforcing the arc consistency for these domains with the entire puzzle. Then, perform AC-3 over arcs within the red box, only adding arcs that are within the red box. List the new domains for the variables in this box.

Next, the coding portion:

3. [24pts, graded by autograder] Carefully read the documentation in the provided starter code to see how we will represent the Sudoku game in code. Use the provided starter code to implement the AC-3 algorithm and integrate it into our provided backtracking search. Test your code on the provided set of puzzles (sudokus.txt). Submit your algorithm to the autograder on Gradescope. Don't print anything in your solution- you will be graded on your program's output.

Reflection:

4. [1pt] Run sudoku.py on the puzzle in sudoku_comparison.txt without using any additional inference (so, the unmodified sudoku.py file) and using AC-3. How many backtracks were made without using AC-3? How many were made when using AC-3?
5. [1pt] What is the impact of this difference?