
Q1. Solving CSPs (25 Points)

In a combined 3^{rd} and 4^{th} grade class, students can be 8, 9, 10, and 11 years old. We are trying to solve for the ages of **A**nn, **B**ob, **C**laire, and **D**oug (abbreviations: **A**, **B**, **C**, **D**). Consider the following constraints:

- No student is older in years than Clair (but may be the same age).
- Bob is two years older than Ann.
- Bob is younger in years than Doug.

Complete the following questions:

1. (5 pts) Draw the constraint graph.

Solution.

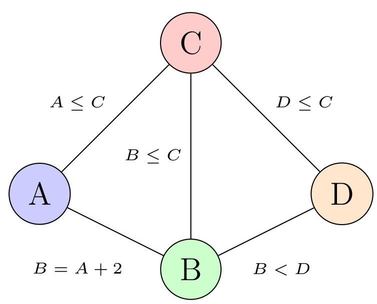


Figure 1: Constraint graph for child ages. Clair is connected to everyone because the constraint says they must all younger or the same age. Bob is constrained to be 2 years older than Ann also also younger than Doug (necessarily different ages).

2. (3 pts) Suppose we are using the AC-3 algorithm for arc consistency. How many total arcs will be enqueued when the algorithm begins execution?

Solution. There will be 10 total arcs when the algorithm *begins* but some will be added during its execution (to ensure consistency). The 10 arcs are

$$
\begin{aligned}
\mathbf{C} &\geq \mathbf{A}, & \mathbf{A} &\leq \mathbf{C}, \\
\mathbf{C} &\geq \mathbf{B}, & \mathbf{B} &\leq \mathbf{C}, \\
\mathbf{C} &\geq \mathbf{D}, & \mathbf{D} &\leq \mathbf{C}, \\
\mathbf{B} &= \mathbf{A} + 2, & \mathbf{A} &= \mathbf{B} - 2, \\
\mathbf{B} &< \mathbf{D}, & \mathbf{D} &> \mathbf{B}.
\end{aligned}
$$

3. (9 pts) Assuming all ages $\{8, 9, 10, 11\}$ are possible for each student before running arc consistency, manually run arc consistency on **only** arc from **A** to **B**.

(a) What values on **A** remain viable after this operation?

(b) What values on **B** remain viable after this operation?

(c) Assuming there are no arcs left in the list of arcs to be processed, which arc(s) would be added to the queue for processing after this operation?

Solution.
(a) Prior to starting the algorithm, the domains of **A** and **B** are $\{8, 9, 10, 11\}$. If we enqueue them in the order $\mathbf{B} = \mathbf{A} + 2, \mathbf{A} = \mathbf{B} - 2$ we get the domain:

$$\mathrm{dom}(\mathbf{A}) = \{8, 9, 10, 11\} \to \{8, 9\}.$$

(b) Similarly, the domain of **B** becomes:

$$\mathrm{dom}(\mathbf{B}) = \{8, 9, 10, 11\} \to \{10, 11\}.$$

(c) Assuming we also enqueued the other arcs and that we processed the arcs in the order from Part (b), after processing $\mathbf{A} = \mathbf{B} - 2$ we would need to re-enqueue $\mathbf{B} = \mathbf{A} + 2$ since the domain of **A** has changed, we need to reevaluate **B** for consistency with the new pruned domain of **A**.

4. (8 pts) Suppose we enforce arc consistency on all arcs. What ages remain in each person's domain?

Solution. Using our result from Part 3, we now need to process the arcs related to **C** and **D**. These can be done in many different orders but I found it easiest to next prune **D** due to the constraint from **B**:

$$\mathrm{dom}(\mathbf{D}) = \{8, 9, 10, 11\} \to \{11\}.$$

This is because the domain of **B** has only two values, 10 and 11, and **D** must be greater than (but not equal to) **B**. This means **D** can only take on the value 11. Checking $\mathbf{B} < \mathbf{D}$ next we find that $\mathbf{B}$ now must be pruned to only include the value 10. Thus, we must re-enqueue not only $\mathbf{A} = \mathbf{B} - 2$ but also $\mathbf{D} > \mathbf{B}$ since the domain of $\mathbf{B}$ has changed.

We can quickly process the constraints for $\mathbf{C} \geq \mathbf{D}$ as it's apparent that $\text{dom}(\mathbf{C}) = \{11\}$. To check in, we currently have the following domains and queue:

$$\begin{aligned} \text{dom}(\mathbf{A}) &= \{8, 9\} \\ \text{dom}(\mathbf{B}) &= \{10\} \\ \text{dom}(\mathbf{C}) &= \{11\} \\ \text{dom}(\mathbf{D}) &= \{11\} \end{aligned} \qquad\qquad \begin{aligned} \mathbf{C} &\geq \mathbf{A} \\ \mathbf{A} &\leq \mathbf{C} \\ \mathbf{C} &\geq \mathbf{B} \\ \mathbf{B} &\leq \mathbf{C} \\ \mathbf{B} &= \mathbf{A} + 2 \\ \mathbf{B} &< \mathbf{D} \end{aligned}$$

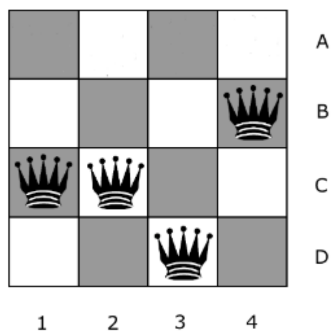
The first four in the queue make no changes to the domains but when we get to $\mathbf{B} = \mathbf{A} + 2$ we further prune $\text{dom}(A)$ to just $\{8\}$. We then add $\mathbf{A} = \mathbf{B} - 2$ to the queue (since the domain of $\mathbf{A}$ changed). The final two arcs that are queued, $\mathbf{B} < \mathbf{D}$ and $\mathbf{A} = \mathbf{B} - 2$, do not make any further changes to our domains and our final (consistent) domains are then

$$\begin{aligned} \text{dom}(\mathbf{A}) &= \{8\} \\ \text{dom}(\mathbf{B}) &= \{10\} \\ \text{dom}(\mathbf{C}) &= \{11\} \\ \text{dom}(\mathbf{D}) &= \{11\}\,. \end{aligned}$$

Q2. 4-Queens (12 points)

The min-conflicts algorithm attempts to solve CSPs iteratively. It starts by assigning some value to each of the variables, ignoring the constraints when doing so. Then, while at least one constraint is violated, it repeats the following: (1) randomly choose a variable that is currenly violating a constraint, (2) assign to it the value in its domain such that after the assignment the total number of constraints violated is minimized (among all possible selections of values in its domain).

In this question, you are asked to execute the min-conflicts algorithm on a simple problem: the 4-queens problem in the figure shown below. Each queen is dedicated to its own column (i.e. we have variables Q_1, Q_2, Q_3, and Q_4 and the domain for each one of them is $\{A, B, C, D\}$). In the configuration shown below, we have $Q_1 = C, Q_2 = D, Q_3 = D, Q_4 = B$. Two queens are in conflict if they share the same row, diagonal, or column (though in this setting they can never share the same column.



You will execute min-conflicts for this problem three times, starting with the state shown in the figure above. When selecting a variable to reassign, min-conflicts chooses a conflicted variable at random. For this problem, assume that your random number generator always chooses the leftmost conflicted queen. When moving a queen, move it to the square in its column that leads to the fewest conflicts with other queens. If there are ties, choose the topmost square among them.

1. Starting with the queens in the configuration shown in the above figure, which queen will be moved, and where will it be moved to?

2. Continuing off of Part 1, which queen will be moved, and where will it be moved to?

3. Continuing off of Part 2, which queen will be moved, and where will it be moved to?

Solution. 1. The first queen to be moved will be Q_1 since it is in conflict and is leftmost with 1 conflict. If she were to be moved to D1, that would increase conflicts to 2. If she were to be moved to B1, we would have 3 conflicts (assuming we can count Q_3 as a conflict even though she's behind Q_2). If moved to position A1, there are zero conflicts, therefor Q_1 will be moved first to A1.

2. The next queen to be moved will be Q_2 as it is in conflict with Q_3 and leftmost in conflict. Q_2 will be moved to A2 as that leads to just 1 conflict with Q_1 whereas the other space options lead to 2 conflicts.

3. Lastly, Q_1 is leftmost in conflict again and will be moved back to C1 as that has zero conflicts whereas the other two positions have at least 1 conflict.

The boards are shown in the figure below.

			x
x	x		
		x	

→

x			
			x
	x		
		x	

→

x	x		
			x
		x	

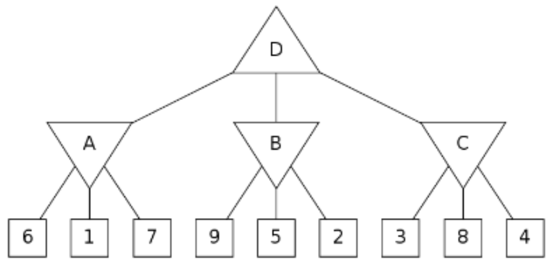
→

	x		
			x
x			
		x	

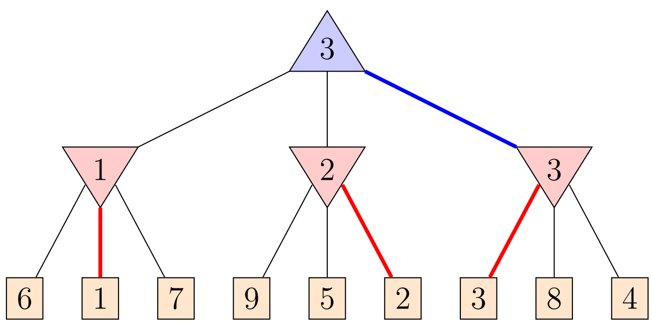
Q3. Minimax and Expectimax (15 points)

Q3.1. Minimax (7.5 points)

Consider the zero-sum game tree shown below. Triangles that point up, such as at the top node (root), represent choices for the maximizing player; triangles that point down represent choices for the minimizing player. Outcome values for the maximizing player are listed for each leaf node, represented by the values in squares at the bottom of the tree. Assuming both players act optimally, carry out the minimax search algorithm. Enter the values for the letter nodes.



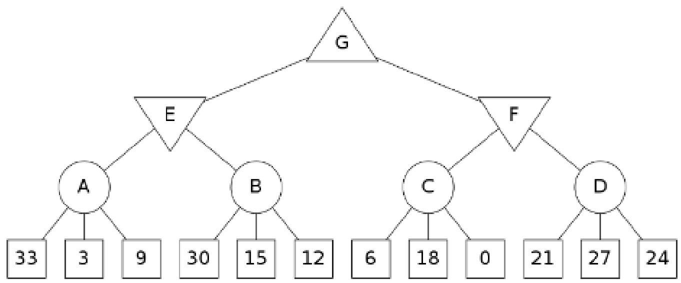
Solution. For nodes A, B, and C the minimization portion of the algorithm will choose 1, 2, and 3 respectively. The maximization will then choose the maximum of those 3 nodes and thus D will choose the value 3, shown in the following figure.



Q3.2. Expectimax (7.5 points)

Consider the game tree shown below. As in the previous problem, triangles that point up, such as the top node (root), represent choices for the maximizing player; triangles that point down represent choices for the minimizing player. The circular nodes represent chance nodes in which each of the possible actions may be taken with equal probability. The square nodes at the bottom represent leaf nodes. Assuming both players act optimally, carry out the expectiminimax search algorithm. Enter the values for the letter nodes.

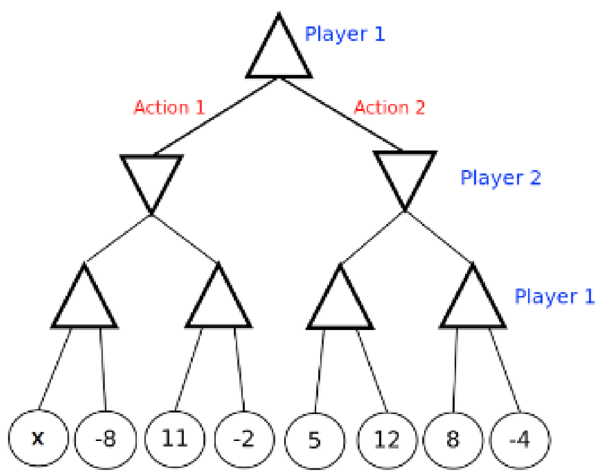
Solution. HERE



Q4. Unknown Leaf Values (28 points)

Consider the following game tree, where one of the leaves has an unknown payoff, x. Player 1 moves first, and attempts to maximize the value of the game. Each of the next 3 questions asks you to write a constraint on x specifying the set of values it can take. Please specify your answer in one of the following forms:

- Write All if x can take on all values.
- Write None if x has no possible values.
- Use an inequality in the form $x < \{value\}$, $x > \{value\}$, or $\{value1\} < x < \{value2\}$ to specify an interval of values. As an example, if you think x can take on all values larger than 16, you should enter $x > 16$.



Q4.1. Assume Player 2 is a minimizing agent (and Player 1 knows this). For what values of x is Player 1 guaranteed to choose Action 1 for their first move?

Solution. HERE

Q4.2. Assume Player 2 chooses actions at random with each action having equal probability (and Player 1 knows this). For what values of x is Player 1 guaranteed to choose Action 1?

Solution. HERE

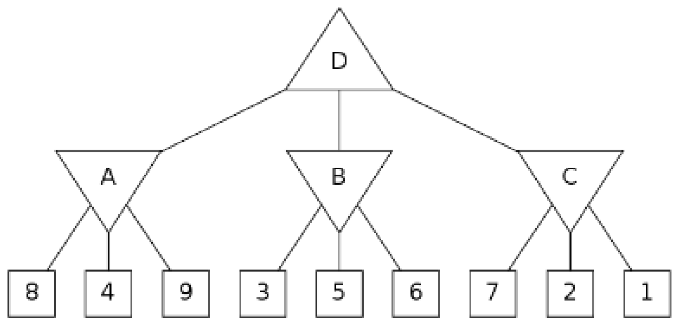
Q4.3. Denote the minimax value of the tree as the value of the root when Player 1 is the maximizer and Player 2 is the minimizer. Denote the expectimax value of the tree as the value of the root when Player 1 is the maximizer and Player 2 chooses actions at random (with equal probability). For what values of x is the minimax value of the tree worth more than the expectimax value of the tree?

Solution. HERE

Q4.4. Is it possible to have a game, where the minimax value is strictly larger than the expectimax value? *Solution.* HERE

Q5. Alpha-Beta Pruning (20 points)

Consider the game tree shown below. Triangles that point up, such as at the top node (root), represent choices for the maximizing player; triangles that point down represent choices for the minimizing player. Assuming both players act optimally, use alpha-beta pruning to find the value of the root node. The search goes from left to right; when choosing which child to visit first, choose the left-most unvisited child.



Hint: Note that the value of a node where pruning occurs is not necessarily the maximum or minimum (depending on which node) of its children. When you prune on conditions, $V > \beta$ or $V < \alpha$, assume that the value of the node is V.

Q5.1 (10 points) Enter the values of the labeled nodes.

Solution. HERE

Q5.2 (10 points) Select the leaf nodes that don't get visited due to pruning.

Solution. HERE