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.

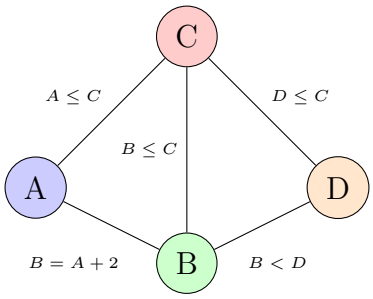


Figure 1: Constraint graph for child ages. Clair is connected to everyone because the constraint says they must all younger or the same age. Bob is constrained to be 2 years older than Ann also also younger than Doug (necessarily different ages).

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 $\mathbf{A}$ to $\mathbf{B}$.

(a) What values on $\mathbf{A}$ remain viable after this operation?

(b) What values on $\mathbf{B}$ 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 $\mathbf{A}$ and $\mathbf{B}$ 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:

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

(b) If we only process the arc from $\mathbf{A}$ to $\mathbf{B}$, the domain of $\mathbf{B}$ does not change. However if we process the arc from $\mathbf{B}$ to $\mathbf{A}$, the domain of $\mathbf{B}$ becomes:

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

(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 $\mathbf{A}$ has changed, we need to reevaluate $\mathbf{B}$ for consistency with the new pruned domain of $\mathbf{A}$. If the rest of the arcs were not enqueued to start with, we would also need to enqueue $\mathbf{C} \geq \mathbf{A}$. We wouldn't need to enqueue this if all of the arcs were originally queued up since it would be there already to account for the change in $\mathbf{A}$.

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 $\mathbf{C}$ and $\mathbf{D}$. These can be done in many different orders but I found it easiest to next prune $\mathbf{D}$ due to the constraint from $\mathbf{B}$:

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

This is because the domain of $\mathbf{B}$ has only two values, 10 and 11, and $\mathbf{D}$ must be greater than (but not equal to) $\mathbf{B}$. This means $\mathbf{D}$ 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:

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

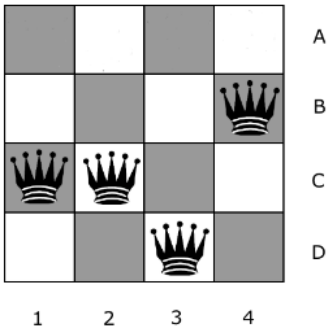
The first four in the queue make no changes to the domains but when we get to $\mathbf{B} = \mathbf{A} + 2$ we further prune $\text{dom}(A)$ to just $\{8\}$. We then add $\mathbf{A} = \mathbf{B} - 2$ to the queue (since the domain of $\mathbf{A}$ changed). The final two arcs that are queued, $\mathbf{B} < \mathbf{D}$ 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?

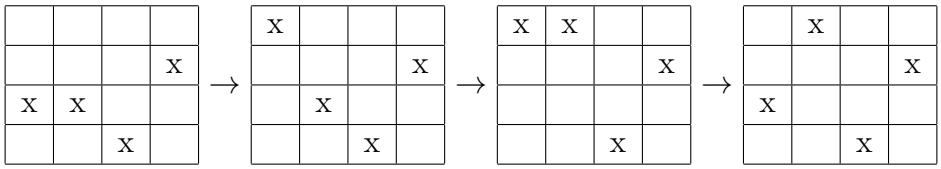
3. Continuing off of Part 2, which queen will be moved, and where will it be moved to?

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.

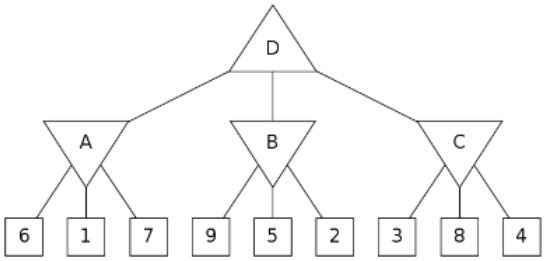
The boards are shown in the figure below.



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.

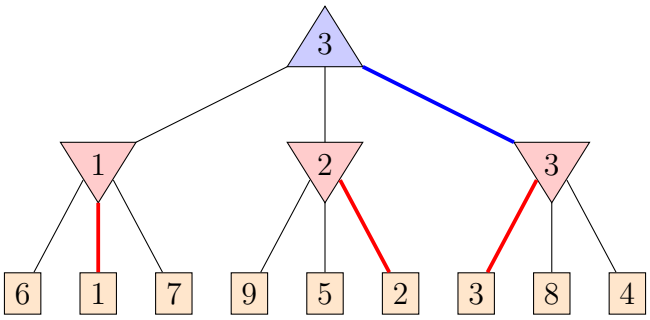
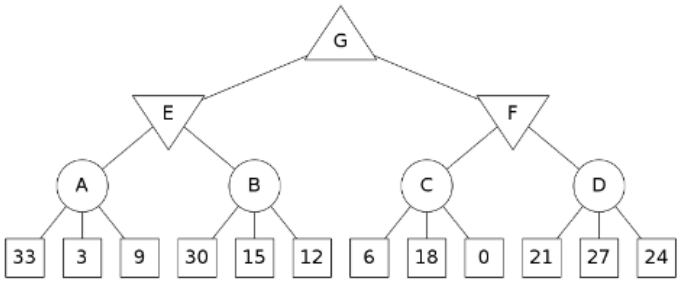


Figure 2: The minimax algorithm applied to the provided zero-sum game tree. A, B, and C take on the values 1, 2, and 3 respectively as they minimize the leaf nodes while D takes on the value 3 as it maximizes.

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. First we'll calculate the expected values to be placed at A, B, C, and D:

$$\mathrm{E}[\mathrm{A}] = \frac{1}{3}[33 + 3 + 9] = 15$$

$$\mathrm{E}[\mathrm{B}] = \frac{1}{3}[30 + 15 + 12] = 19$$

$$\mathrm{E}[\mathrm{C}] = \frac{1}{3}[6 + 18 + 0] = 8$$

$$\mathrm{E}[\mathrm{D}] = \frac{1}{3}[21 + 27 + 24] = 24.$$

From here, E and F minimize the values to 15 and 8 respectively. Finally, G maximizes between the two and chooses 15. This is reflected in the following figure.

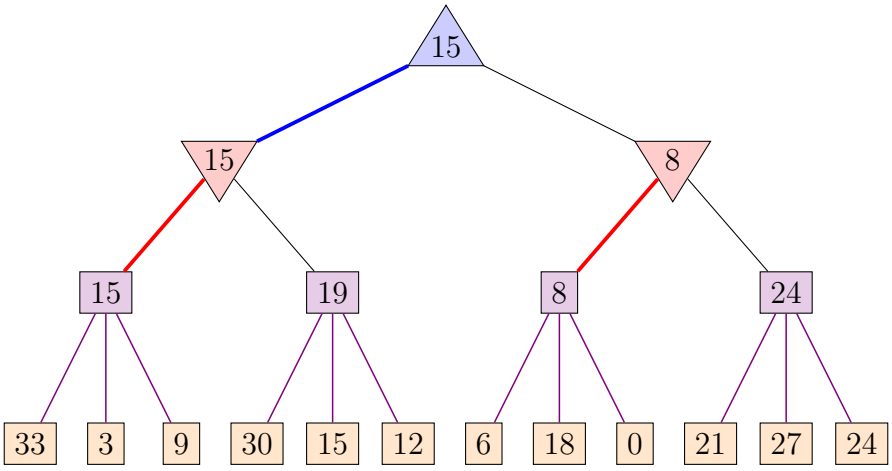
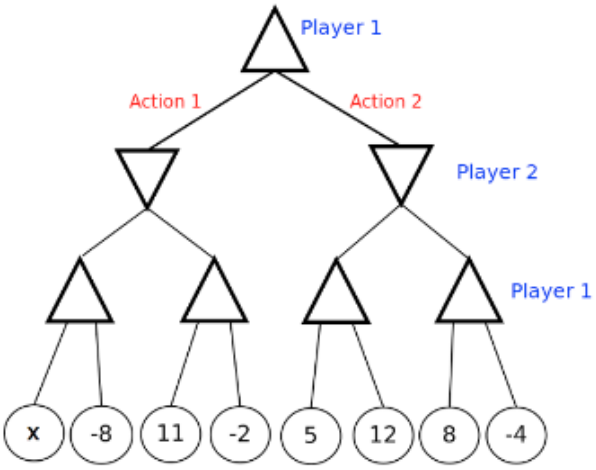


Figure 3: An expectimax algorithm applied to a search tree. First the expected values of the leaves are calculated and supplied as arguments for the minimizing agent. Then, after the minimizing agent picks minimum values, the maximizing agent chooses the highest possible remaining value.

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. Values of x such that $x > 8$. Since Player 2 minimizes and Player 1 maximizes, Action 2 will always correspond to a value of 8. Action 1 will either correspond to a value of x, -8, or 11 (depending on it's value). If $8 < x < 11$, Action 1 will correspond to a value of x and since $x > 8$, Action 1 will be chosen by Player 1. If $x > 11$, Action 1 will correspond to a value of 11 which is fine, since $11 > 8$ so it will still be chosen.

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. I'm not entirely clear on the wording of this problem so I provide 2 interpretations and what they'd yield. I believe the second is most correct.

No values of x. You cannot guarantee that Player 1 will choose Action 1 because of the stochastic nature of the problem. For instance, Suppose x is very large, to attempt to ensure Player 1 would prefer it at Action 1. Given Player 2 chooses randomly with a 50% chance of x or the other child node, it could still choose -2 or 11. Now suppose on the right hand side of the tree, Player 2 has chosen 12. It doesn't matter what x's value was, Player 1 will prefer Action 2 since in this scenario x wasn't chosen.

Although the word expectimax was never used in the context of the problem, let's assume that Player 2 acts as an expectimax agent for this scenario. Then for it's two choices, the expectimax values would be $\frac{1}{2}[\max(x, -8) + 11]$ and $frac12[12 + 8]$. We want Action 1's value to be greater than Action 2's so we set up the following inequality and solve:

$$\begin{aligned}
\frac{1}{2}[\max(x, -8) + 11] &> \frac{1}{2}[12 + 8] \\
\max(x, -8) + 11 &> 20 \\
\max(x, -8) &> 9 \\
x &> 9.
\end{aligned}$$

Therefore, for values of $x > 9$, given that x makes it to Player 2 by random choice, we have a guarantee that Player 1 will choose x.

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. I think we need to deal with this in cases. First we'll consider $x < -8$, then $-8 < x \geq 11$, and finally $x > 11$.

$x < -8$: The minimax value for Action 1 and Action 2 will be -8 and 8 respectively. The expectimax values will be $\frac{1}{2}[-8 + 11] = \frac{3}{2}$ and $\frac{1}{2}[12 + 8] = 10$ again respectively. Therefore the value of the search tree will result in 8 for minimax and 10 for expectimax.

$-8 < x \leq 11$: Actions 1 and 2 will take the values x and 8 for minimax leading to a root value of x (since $x > 8$ by assumption) and $\frac{1}{2}[x + 11]$, 10 for expectimax, respectively. We again want to solve the inequality for x in

$$\frac{1}{2}[x + 11] > 10 \Rightarrow x > 9.$$

Thus, the value of the tree in minimax will be $x > 8$ (but less than 12) and expectimax will be $9 < x \leq 11 \Rightarrow x = 10$ or $x = 11$. Thus, if we choose x to be either 10 or 11, we will always result with them being equal in each case.

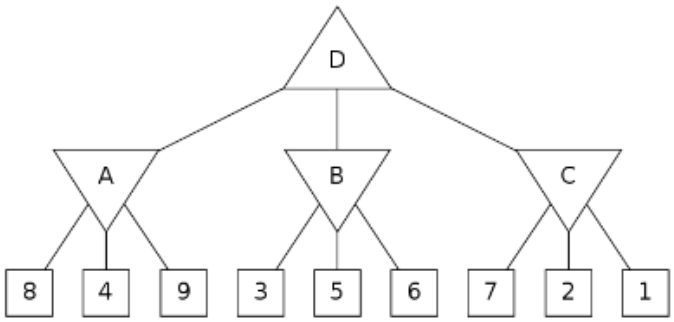
$x > 11$: Actions 1 and 2 take the values 11 and 8, respectively for minimax. For expectimax, we'll get $\frac{1}{2}[x + 11]$ and 10. Since minimax will choose Action 1 valued at 11, we need to find $\frac{1}{2}[x + 11] < 11 \Rightarrow x < 11$ which contradicts our initial assumption.

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

Solution. No, it is not possible. The key here is that we want a scenario where minimax is *strictly* larger than expectimax. While we may come across scenarios where a minimax value is larger than an expectimax, due to the stochastic nature of the problem, there will *always* be a scenario where it will be less than or equal to the expectimax. This is because both players are expected to play optimally.

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.

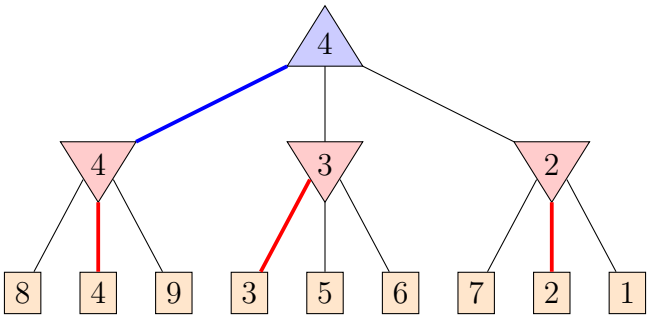


Figure 4: The values that A, B, C, and D take given $\alpha - \beta$ pruning. Leaf 9 is considered but ignored due to 4 being lower than the current β value. Pruning for the other values is described in the next part.

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

Solution. The leaves 5, 6, and 1 are pruned in the $\alpha - \beta$ pruning algorithm. From the left to right search method, when B reaches 3, we know that D won't choose anything but 4

between the two so we can ignore the leaves 5 and 6. Similarly, when C reaches 2, the only options that C will choose over it will be less and we know that D will still choose 4 over anything else. The end result is that the domain of D is $(4, \infty)$.

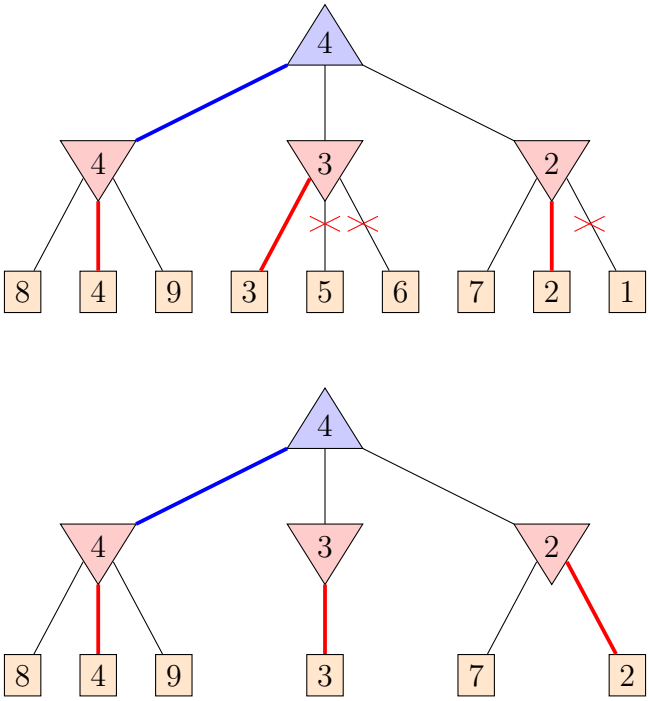


Figure 5: The values for A, B, and C that get pruned. Once B "sees" 3, D knows that it will prioritize A over B so we no longer need to search below B. Similarly, when C reaches 2, we know that A will be prioritized over C as well and the leaf with value 1 doesn't need to be considered.