Homework 02

Submission Notices:

- *Conduct your homework by filling answers into the placeholders given in this file (in Microsoft Word format). Questions are shown in black color, instructions/hints are shown in italic and blue color, and your content should use any color that is different from those.*
- *After completing your homework, prepare the file for submission by exporting the Word file (filled with answers) to a PDF file, whose filename follows the following format,*
  <StudentID-1>_<StudentID-2>_HW02.pdf (Student IDs are sorted in ascending order)
  *E.g., **2312001_2312002_HW02.pdf***
  and then submit the file to Moodle directly WITHOUT any kinds of compression (.zip, .rar, .tar, etc.).
- *Note that you will get zero credit for any careless mistake, including, but not limited to, the following things.*
  1. *Wrong file/filename format, e.g., not a pdf file, use "-" instead of "_" for separators, etc.*
  2. *Disorder format of problems and answers*
  3. *Conducted not in English*
  4. *Cheating, i.e., copy other students' works or let the other student(s) copy your work.*

Problem 1 (2pt) Answer the following simple questions.

Please fill your answer in the table below.

Score	Questions	Answers
0.25pt	What defines a neighbor in the state space of a local search algorithm?	*Neighbor of a state is another state that can be reached by using a small, local change (according to the problem's definition to a "move").*
0.25pt	How does the cooling schedule impact the performance of simulated annealing?	*When the temperature decreases too quickly, the algorithm might converge prematurely to a suboptimal solution (local optimum). On the other hand, if the cooling is too slow, the algorithm may take excessively long time to find the optimal solution.*
0.25pt	What is the role of aspiration criteria in overriding tabu restrictions in Tabu search?	*Aspiration criteria are used to override the tabu status of a move if certain conditions are met, typically when the move results in a solution that is better than any previous solution.*
0.25pt	How does the evaluation function influence the Minimax algorithm's decisions?	• *It is used instead of the true utility when the search reaches a pre-defined depth (not the end of the game).* • *It simulated the effect of playing optimally beyond the current depth.* • *A poor evaluation function can mislead the minimax algorithm into*

		picking a suboptimal move, because it misjudges which states are actually better.
0.25pt	What conditions allow alpha-beta pruning to cut off the most branches in Minimax?	*- Optimal move ordering: nodes are evaluated in best-to-worst order => The sooner that is found is the good move, the tighter the α or β bound becomes, allowing more cuts.* *- Deep pruning opportunities: the game tree should have enough depth and branching for pruning to matter. Shallow or small trees don't benefit much.*
0.25pt	How does Expectimax handle probabilistic outcomes compared to Minimax in game trees?	*Intuition, minimax says "What's the worst my opponent can do?", expectimax says "What's the average outcome I can expect, given the probabilities?".* *In more detail, minimax choose action by minimizing the possible loss. Expectimax, at chance node, instead of taking min or max, it computes the expected value.*
0.25pt	What is the benefit of combining MRV with the Least Constraining Value heuristic in CSP?	*MRV will select the variable with the fewest legal values left, LCV based on the variable which is chosen and tries values that rule out the fewest option for its neighbors. So this combination will avoid waiting time on easy variables and reduce the chance of backtracking.*
0.25pt	How does forward checking complement backtracking in solving constraint satisfaction problems?	*Forward checking removes inconsistence values from the domains of unassigned neighboring variables, if any domain becomes empty, the algorithm backtrack immediately.*

Problem 2 – Local Search Problem (2pts) You are tasked with solving a Knapsack Problem using a Genetic Algorithm. The problem is defined as follows:

- Items:
  - Item 1: weight = 1, value = 3
  - Item 2: weight = 2, value = 5
  - Item 3: weight = 3, value = 8
  - Item 4: weight = 5, value = 12
  - Item 5: weight = 8, value = 10
- Knapsack Capacity: 10

Genetic Algorithm Parameters

- Chromosome Representation: Binary string of length 5 (each bit corresponds to an item; 1 = selected, 0 = not selected).
- Fitness Function: Total value of selected items if total weight ≤ 10, else 0.
- Population Size: 4
- Selection Method: Tournament selection with tournament size 2 (randomly pick 2 chromosomes, select the one with higher fitness).
- Crossover Operation: Single-point crossover.
- Mutation Operation: Bit-flip mutation with probability 0.1 per bit.
- Number of Generations: Perform one generation.

Your Task:

1. (0.5pt) Define an initial population of 4 chromosomes (you can choose any valid binary strings of length 5).
2. (0.5pt) Apply the genetic algorithm operations:
    - Use tournament selection to choose parents.
    - Perform single-point crossover on the selected parents to create offspring.
    - Apply mutation to the offspring with a probability of 0.1 per bit.
3. (0.5pt) Calculate the fitness of each chromosome in the new population.
4. (0.5pt) Identify the best solution in the new population after one generation.

Note: You may assume specific outcomes for random events (e.g., crossover points, mutation occurrences) to simplify calculations.

Guidelines:

- *First, define four arbitrary chromosomes according to the specifications above.*
- *Then, for each of the following steps, use these four chromosomes to demonstrate the effect of the step. Clearly indicate the input, output, and key operations (where applicable).*
- *Programming code is not recommended, as it is less intuitive for understanding.*
- *Demonstration figures are highly encouraged to improve clarity.*

Answer:

1. Initial population chromosomes

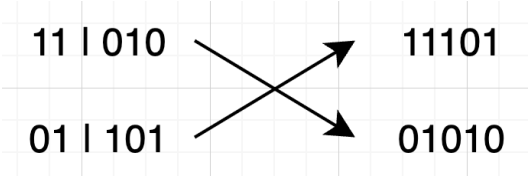
Chromosome	Binary	Total weight	Total value	Fitness function
C1	11010	8	20	20
C2	01101	5	13	13
C3	10010	6	15	15
C4	11001	11	18	0

2. Selection, crossover, mutation

Randomly choose two pairs (because applying the tournament selection with k = 2) {C1, C3} and {C2, C4}:

- With {C1, C3}, we have Fit(C1) = 20 > Fit(C3) = 15 so we choose C1 as the first parent.
- With {C2, C4}, we have Fit(C2) = 13 > Fit(C4) = 0 so we choose C2 as the second parent.

Randomly pick the crossover point at the middle of 2nd and 3rd bits, we have:



We call two offsprings are O1 = 11101 and O2 = 01010, after that we have the mutation operation is bit-flip with probability of 0.1 per bit, the number of bits in O1 and O2 are 10 thus lead to exactly 1 bit in 10 bits of O1 and O2 will be flipped. We randomly choose the 4th bit of O2 and flip it. So we have O1 = 11100 and O2 = 01000

3. The new population

Chromosome	Binary	Total weight	Total value	Fitness function
O1	11101	13	26	0
O2	01000	2	5	5
C3	10010	6	15	15
C4	11001	11	18	0

4. The best solution

We choose the best solution follow up the best fitness function value (the biggest), so the best solution is the chromosome C3 = 10010.

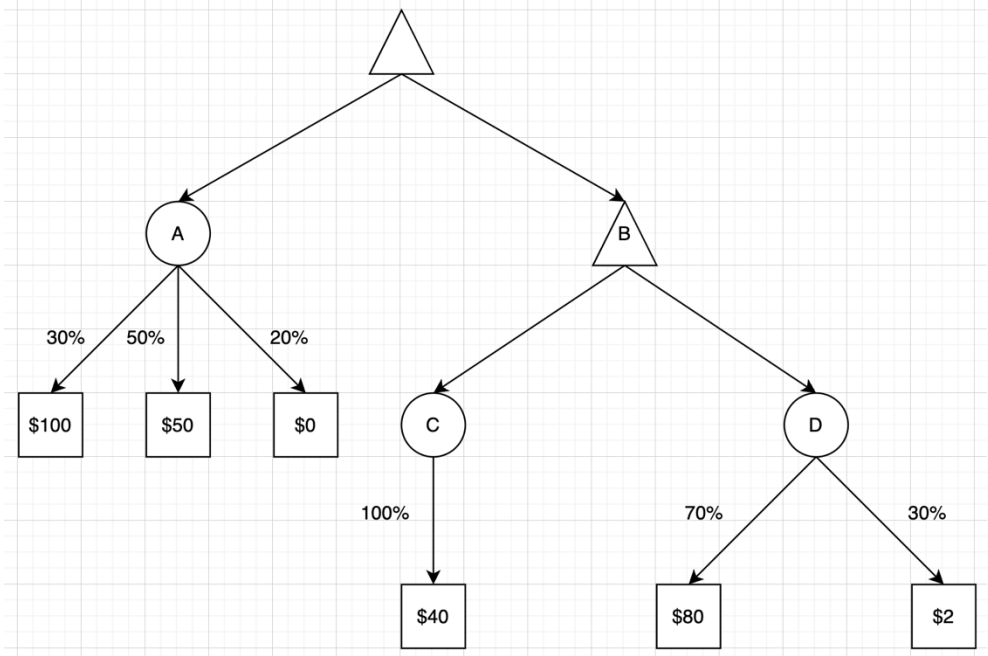
Problem 3 – Adversarial Search - Expectimax (2pts) You are at an initial decision node with two options**: Option A** and **Option B**:

- **Option A** leads to a chance node with the following outcomes:
  - 30% chance of winning $100
  - 50% chance of winning $50
  - 20% chance of winning $0
- **Option B** leads to another decision node with two choices:
  - **Option C**: Win $40 for sure
  - **Option D**: Leads to a chance node with:
    - 70% chance of winning $80
    - 30% chance of winning $2

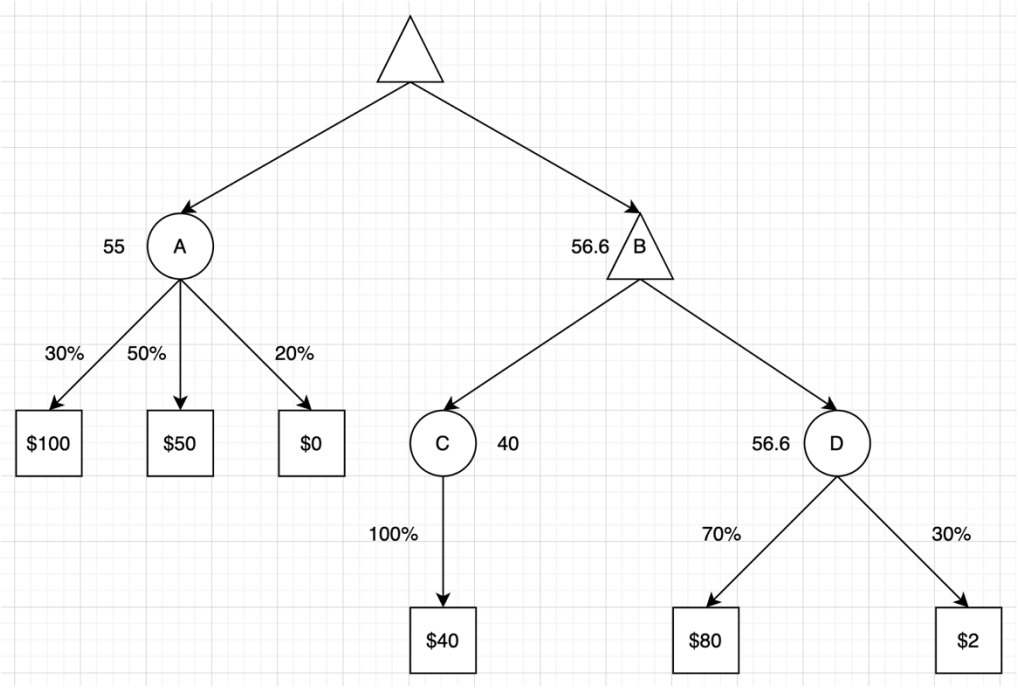
Your task: Using the Expectimax algorithm, determine the best choice at the initial decision node and calculate its expected value, showing your calculations step-by-step.

Answer:

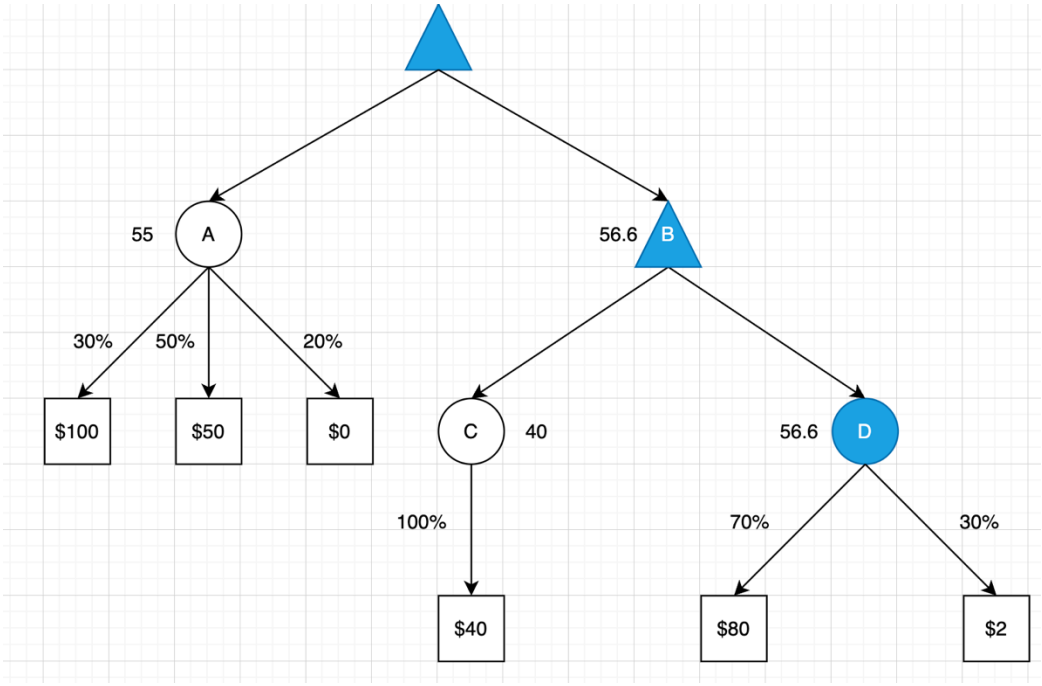
The initial tree is in the following figure:



First, for each A, B, C, D chance nodes, we must calculate the expected value for each of them.
For A, we have Exp(A) = 0.3 * 100 + 0.5 * 50 + 0.2 * 0 = 55
For C, we have Exp(C) = 1 * 40 = 40
For D, we have Exp(D) = 0.7 * 80 + 0.3 * 2 = 56.6
For B, we have Exp(B) = max(Exp(C), Exp(D)) = 56.6



So Max at root node will choose B as the better move, the final path is the following path:

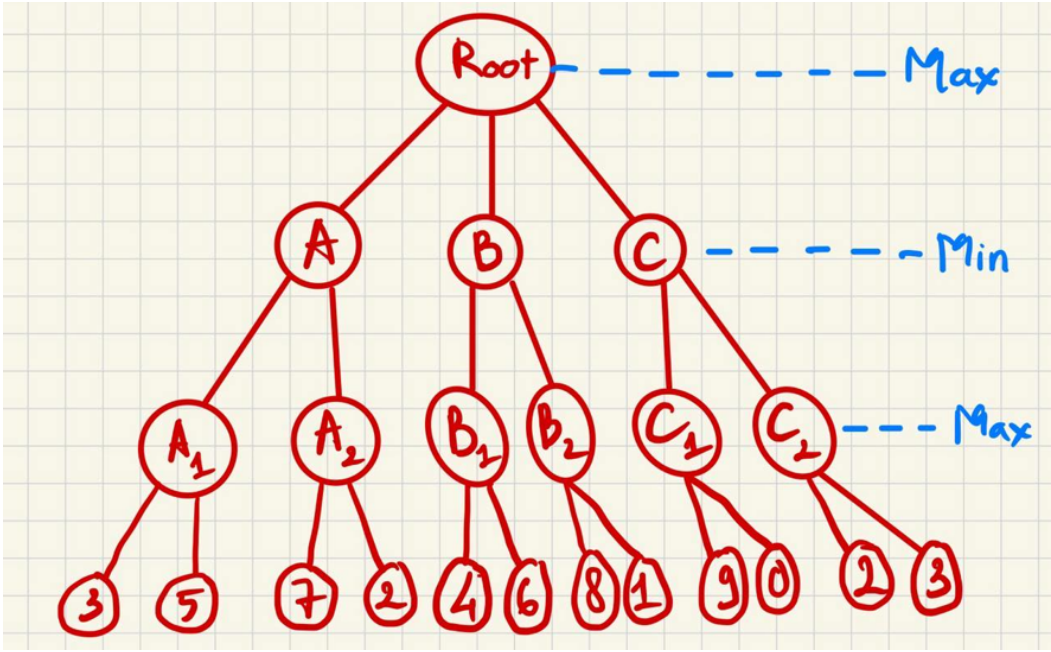


Problem 4 – Minimax (2pts) Consider a two-player game where players Max and Min take turns making moves. The game tree for this problem is structured as follows:

- The root node is a Max node with three choices: A, B, or C.

- Each of these choices leads to a Min node, where each Min node has two choices.
- Each Min choice then leads to a Max node, where Max has two choices.
- Finally, each of these Max choices leads to leaf nodes with the following payoff values (higher values are better for Max):

Game tree structure and Payoffs:



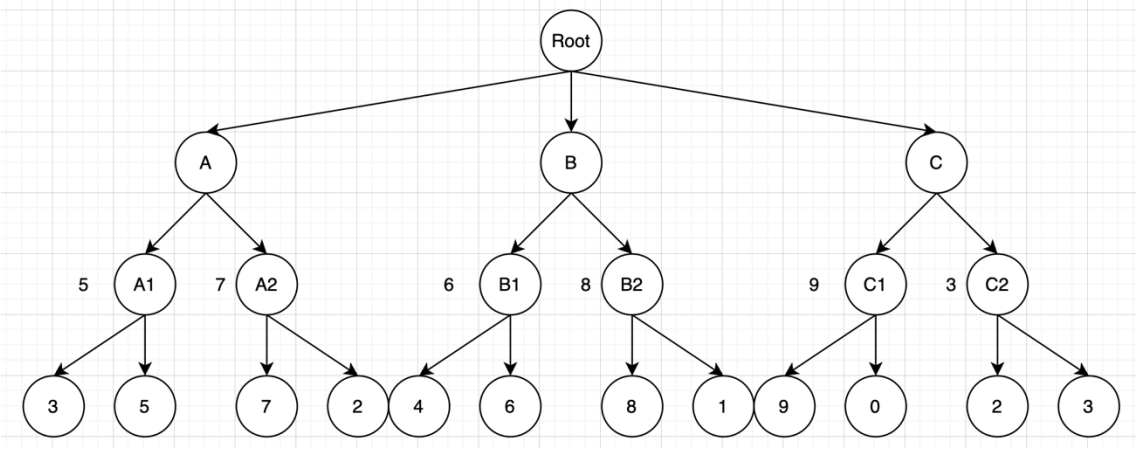
Evaluation order: Leaves are evaluated from left to right as listed (e.g., for A1, evaluate 3 then 5; for A2, evaluate 7 then 2, etc.).

Your task:

a) (1pt) **Minimax Algorithm:** Determine the best move for Max at the root and compute the Minimax value at each node in the tree step-by-step (0.75pt). Then, illustrate your results with a tree (0.25pt).

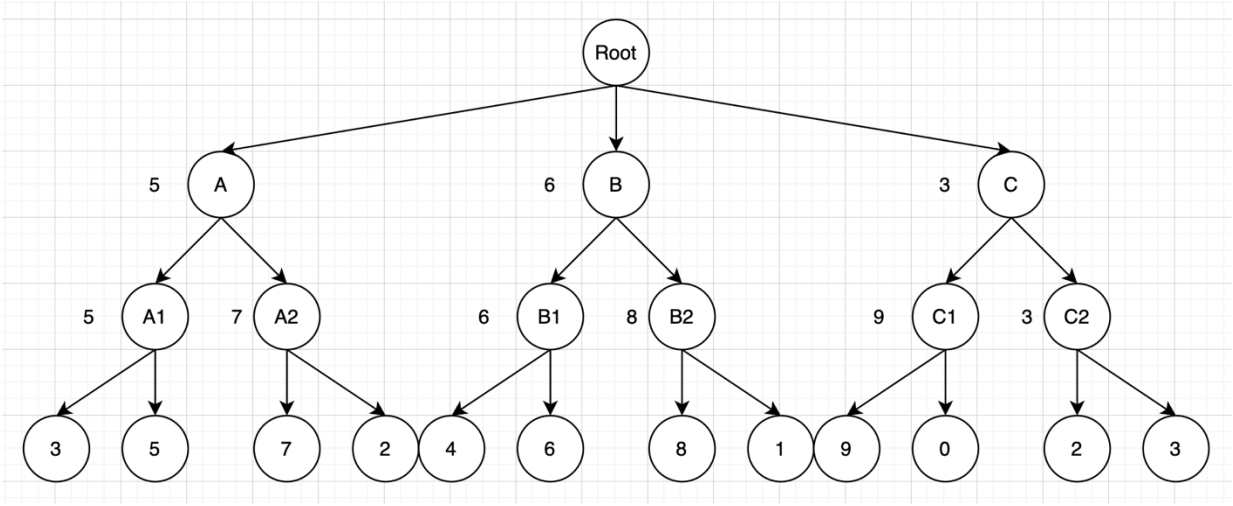
Answer:

For A1, Max will go through 3 first and then is 5, finally it choose 5 as the Minimax value.
For A2, Max will go through 7 first and then is 2, finally it choose 7 as the Minimax value.
For B1, Max will go through 4 first and then is 6, finally it choose 6 as the Minimax value.
For B2, Max will go through 8 first and then is 1, finally it choose 8 as the Minimax value.
For C1, Max will go through 9 first and then is 0, finally it choose 9 as the Minimax value.
For C2, Max will go through 2 first and then is 3, finally it choose 3 as the Minimax value.
After the above steps, we will have the following tree:

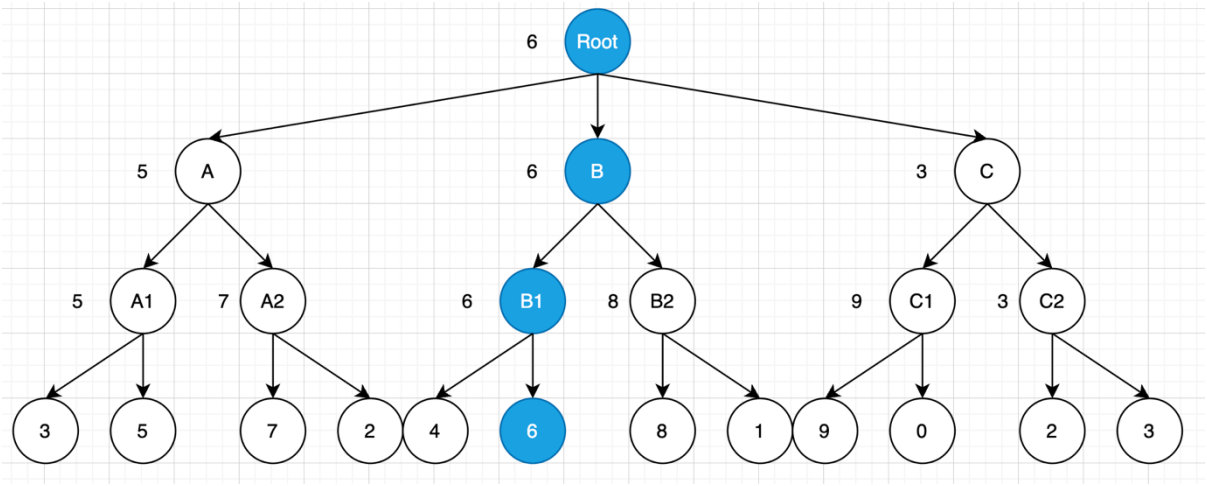


Then, Min at the children of root node will choose the Minimax for its nodes.
For A, Min choose 5 in 5 and 7 as the Minimax value.

For B, Min choose 6 in 6 and 8 as the Minimax value.
For C, Min choose 3 in 9 and 3 as the Minimax value.

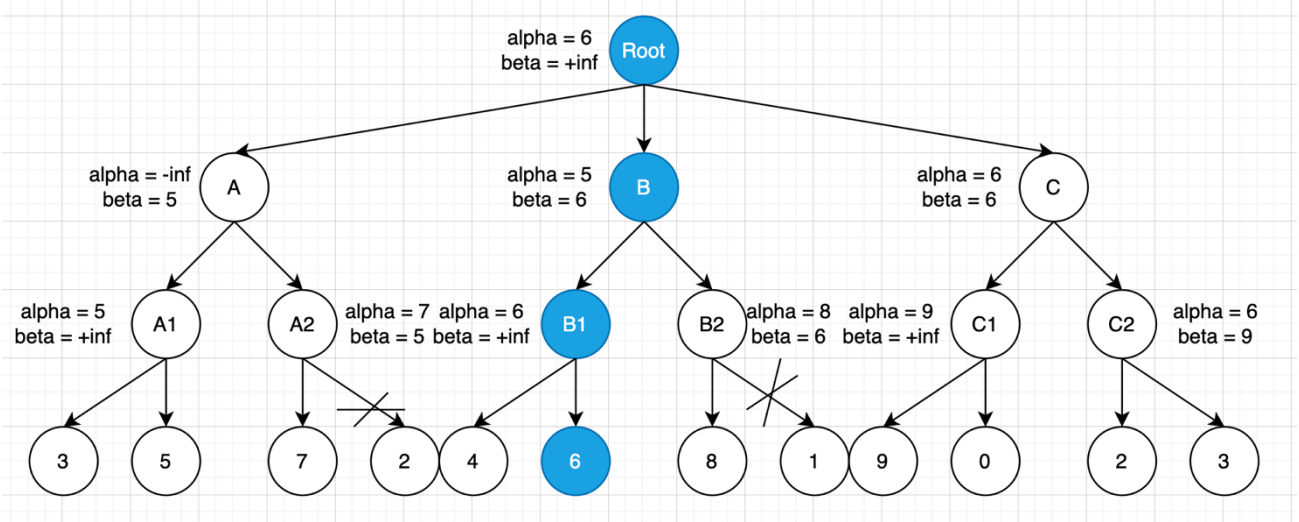


Finally, Max at root node will choose the Minimax value for root. Obviously, Max choose 6 in 5, 6 and 3 as the Minimax value for root.
So the best move is the following path:



b) (1pt) **Alpha-Beta Pruning:** Apply alpha-beta pruning with left-to-right evaluation (0.5pt). Identify which branches (leaves) are pruned and provide a brief explanation for each pruning decision (0.25pt). Also put a mark 'X' on the branch(es) of the tree (0.25pt).

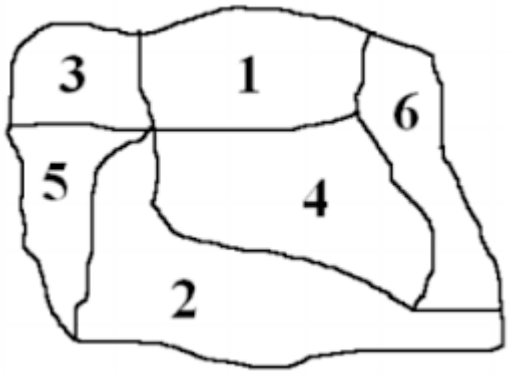
Answer:



At node A2, from A we have β = 5 and Max choose α = 7 thus make α >= β, so we can prune the remaining branch.

At node B2, from B we have α = 5 and β = 6, Max choose the new α = 8 thus make α >= β, so we can prune the remaining branch.

Problem 5 – CSP (2pts) Consider coloring the six-region map shown to the right with three colors: R, G, B so that no two adjacent regions that share a border have the same color. Regions that only meet at a corner are not considered adjacent (e.g., 5 and 1).



a) (0.5pt) Identify the variables that should be used to set this up as a CSP problem and the domain of possible values for each variable.

Answer:

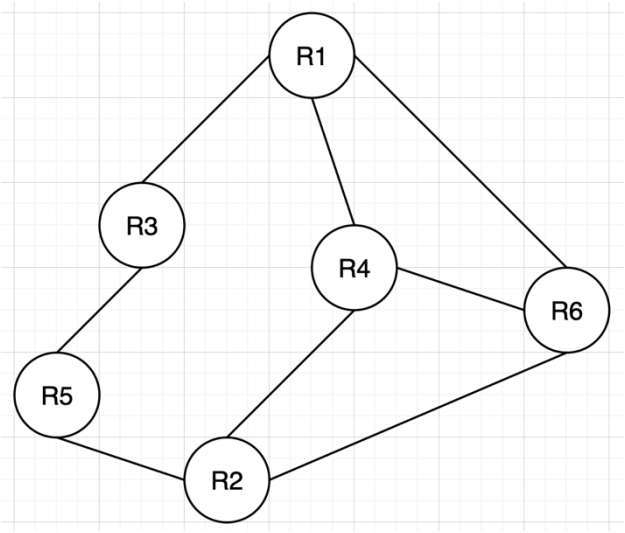
With i = {1, 2, 3, 4, 5, 6}:

- Let Ri is the variable identify the color for the region i.
- Let Di is the domain of color values for the region i. So D_i can be in {R, G, B} for all i.

b) (0.5pt) Draw an abstract constraint graph with a node for each variable and an edge between two variables if there is a constraint between them

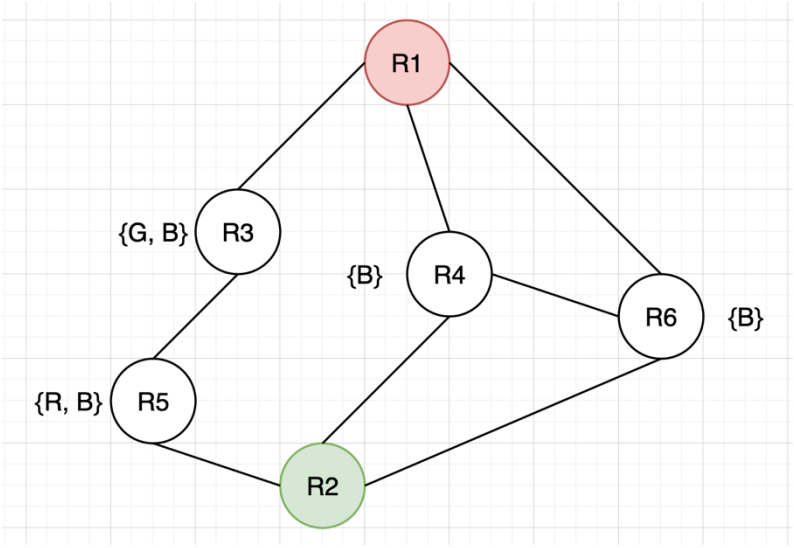
Answer:

The constraint: the adjacent regions do not have the same color.



c) (0.5pt) Assume that we assign **region 1 to be R** and **region 2 to be G**. Perform **forward checking** and show the variables and their possible values for all six regions. Recall that when a value is assigned to a variable V, forward checking eliminates inconsistent values in variables connected to V in the constraint graph. Is a solution possible from this state?

Answer:



When R1 is assigned R value, so its neighbors are R3, R4, R6 just get G or B. But R2 has been assigned G value, so its neighbors are R5 (just get R or B), R4 and R6 just get B => R4 and R6 have a connection, so we can not get any solutions in this state.

d) (0.5pt) Assume the initial domains of the regions in the map above are given as **1 = {R,G,B}**, **2 = {R,G}**, **3 = {R,G,B}**, **4 = {R}**, **5 = {R,G,B}**, and **6 = {R}**. Perform the **Arc Consistency algorithm (AC-3)** step by step to show how the sets of possible values for all six regions change throughout the algorithm. Is a solution possible from this state?

Answer:

The initial queue: Q = {(R1, R3), (R1, R4), (R1, R6), (R2, R4), (R2, R5), (R2, R6), (R3, R1), (R3, R5), (R4, R1), (R4, R2), (R4, R6), (R5, R3), (R5, R2), (R6, R1), (R6, R4), (R6, R2)}.

Pop (R1, R3) and check all values of R1:

- R1 = R, we have R3 = G or R3 = B that satisfy R1 != R3.
- R1 = G, we have R3 = R or R3 = B that satisfy R1 != R3.
- R1 = B, we have R3 = R or R3 = G that satisfy R1 != R3.

=> No value is removed from D1.

Pop (R1, R4) and check all values of R1:

- R1 = R, we don't have any value of R4 that satisfy R1 != R4. So we remove R in D1.
- R1 = G, we have R4 = R that satisfy R1 != R4.
- R1 = B, we have R4 = R that satisfy R1 != R4.

=> D1 is updated to {G, B}

Pop (R1, R6) and check all values of R1:

- R1 = G, we have R6 = R that satisfy R1 != R6.
- R1 = B, we have R6 = R that satisfy R1 != R6.

=> No value is removed from D1.

Pop (R2, R4) and check all values of R2:

- R2 = R, we don't have any value of R4 that satisfy R2 != R4. So we remove R in D2.
- R2 = G, we have R4 = R that satisfy R2 != R4.

=> D2 is updated to {G}.

Pop (R2, R5) and check all values of R2:

- R2 = G, we have R5 = R or R5 = B that satisfy R2 != R5.

=> No value is removed from D2.

Pop (R2, R6) and check all values of R2:

- R2 = G, we have R6 = R that satisfy R2 != R6.

=> No value is removed from D2.

Pop (R3, R1) and check all values of R3:

- R3 = R, we have R1 = G or R1 = B that satisfy R3 != R1.
- R3 = G, we have R1 = B that satisfy R3 != R1.
- R3 = B, we have R1 = G that satisfy R3 != R1.

=> No value is removed from D3.

Pop (R3, R5) and check all values of R3:

- R3 = R, we have R5 = G or R5 = B that satisfy R3 != R5.
- R3 = G, we have R5 = R or R5 = B that satisfy R3 != R5.
- R3 = B, we have R5 = R or R5 = G that satisfy R3 != R5.

=> No value is removed from D3.

Pop (R4, R1) and check all values of R4:

- R4 = R, we have R1 = G or R1 = B that satisfy R4 != R1.

=> No value is removed from R4.

Pop (R4, R2) and check all values of R4:

- R4 = R, we have R2 = G that satisfy R4 != R2.

Pop (R4, R6) and check all values of R4:

- R4 = R, we don’t have any value of R6 that satisfy R4 != R6. So we remove R in D4. It leads to D4 is empty set and make the state to be impossible to solve.

We can not find any possible solution for this state.