Q1:

Use A\* search by graph to solve this problem.

Initial state: empty list OPEN, empty list CLOSED

Steps:

1. Put start node *n0* in OPEN list
2. Select the first node in OPEN, remove it and put it in CLOSED, call this node *n*
3. Expand node *n*, put its children in set M. Here children means all nodes that node *n* may reach next.
4. Establish a pointer to *n* from each member *m* of M, the pointer means *n* is the parent of *m*. Add these members to OPEN. If member *m* of M has already on OPEN, check if the current path is the best so far, then redirect its pointer to *n*, change the parent of *m*. If member *m* has already on CLOSED, ignore it.
5. Reorder OPEN list by increasing *g(n)+h(n)* values, *f(n)*. *g(n)* means the actual path cost from the start node *n0* to node *n*. *h(n)* means estimated cost, it is never larger than the actual cost.
6. go to step 3.
7. When *n* is the goal node, exist successfully. If OPEN empty at last, then exit with failure.

Operator: *g(n)* needs to add up the cost of traveling from *n0* to *n*. *h(n)* needs to calculate the straight-line distance between *n0* and *n*.

Operator cost: each of them is greater than some positive amount.

Q2:

rows number: m, columns number: n

Variables: <r1, c1, direction>, <r2, c2, direction>, ..., <rn, cn, direction> the position of the first letter of word, and the corresponding direction

Domain: <{1, ..., m}, {1, ..., n}, {horizontal, vertical}>

Constraints:

When direction is horizontal: n - ci + 1 < length of word i, ri-1 <= ri < ri-1 + length of word i-1.

When direction is vertical: m - ri + 1< length of word i, ci-1 <= ci < ci-1 + length of word i-1.

Q3:

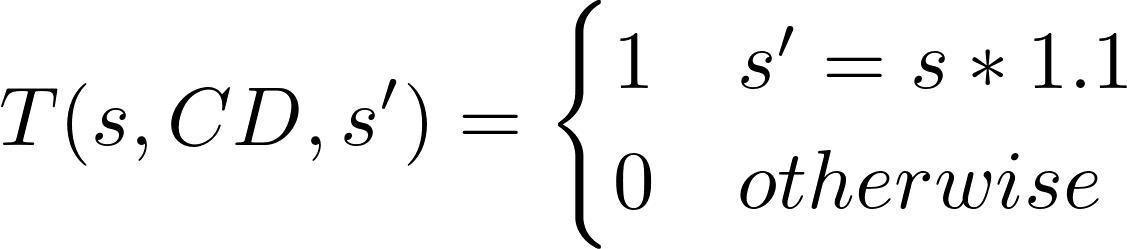
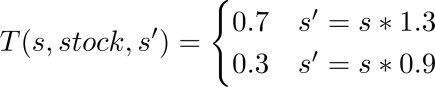
States = Integers

sstart = 0

Terminating state End(s) when s = 18

Actions = {CD, stock}

Transitions:

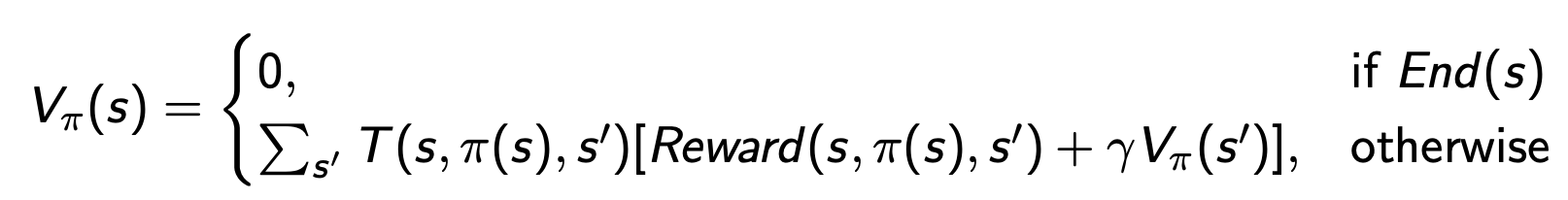
1. 
2. 

Rewards :

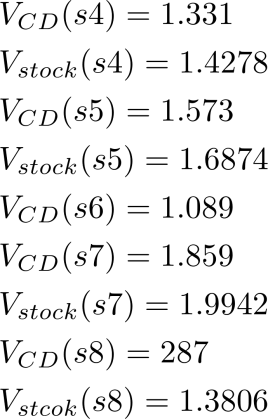
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| year | s | T(s, action, s’) | s’ | Rewards |
| 1 | 0 | 1 | 1 | 1.1 |
| 1 | 0 | 0.7 | 2 | 1.3 |
| 1 | 0 | 0.3 | 3 | 0.9 |
| 2 | 1 | 1 | 4 | 1.21 |
| 2 | 1 | 0.7 | 5 | 1.43 |
| 2 | 1 | 0.3 | 6 | 0.99 |
| 2 | 2 | 1 | 5 | 1.43 |
| 2 | 2 | 0.7 | 7 | 1.69 |
| 2 | 2 | 0.3 | 8 | 1.17 |
| 2 | 3 | 1 | 6 | 0.99 |
| 2 | 3 | 0.7 | 8 | 1.17 |
| 2 | 3 | 0.3 | 9 | 0.81 |
| 3 | 4 | 1 | 10 | 1.331 |
| 3 | 4 | 0.7 | 11 | 1.573 |
| 3 | 4 | 0.3 | 12 | 1.089 |
| 3 | 5 | 1 | 11 | 1.573 |
| 3 | 5 | 0.7 | 13 | 1.859 |
| 3 | 5 | 0.3 | 14 | 1.287 |
| 3 | 6 | 1 | 12 | 1.089 |
| 3 | 6 | 0.7 | 14 | 1.287 |
| 3 | 6 | 0.3 | 15 | 0.891 |
| 3 | 7 | 1 | 13 | 1.859 |
| 3 | 7 | 0.7 | 16 | 2.197 |
| 3 | 7 | 0.3 | 17 | 1.521 |
| 3 | 8 | 1 | 14 | 1.287 |
| 3 | 8 | 0.7 | 17 | 1.521 |
| 3 | 8 | 0.3 | 11 | 1.053 |
| 3 | 9 | 1 | 15 | 0.891 |
| 3 | 9 | 0.7 | 11 | 1.053 |
| 3 | 9 | 0.3 | 18 | 0.729 |

In 3rd year, s6 takes stock, the result might be lower than 0, so do not buy stock in s6, For s9, you can’t buy anything to ensure the result is greater than 0.

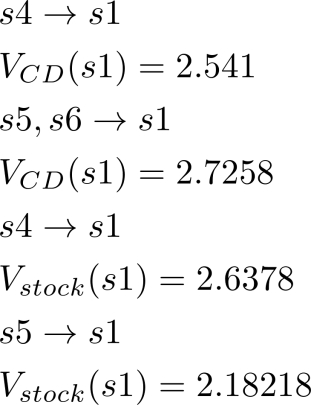
calculate:



first iteration: s4 - s8



second iteration: s1, s2, s3



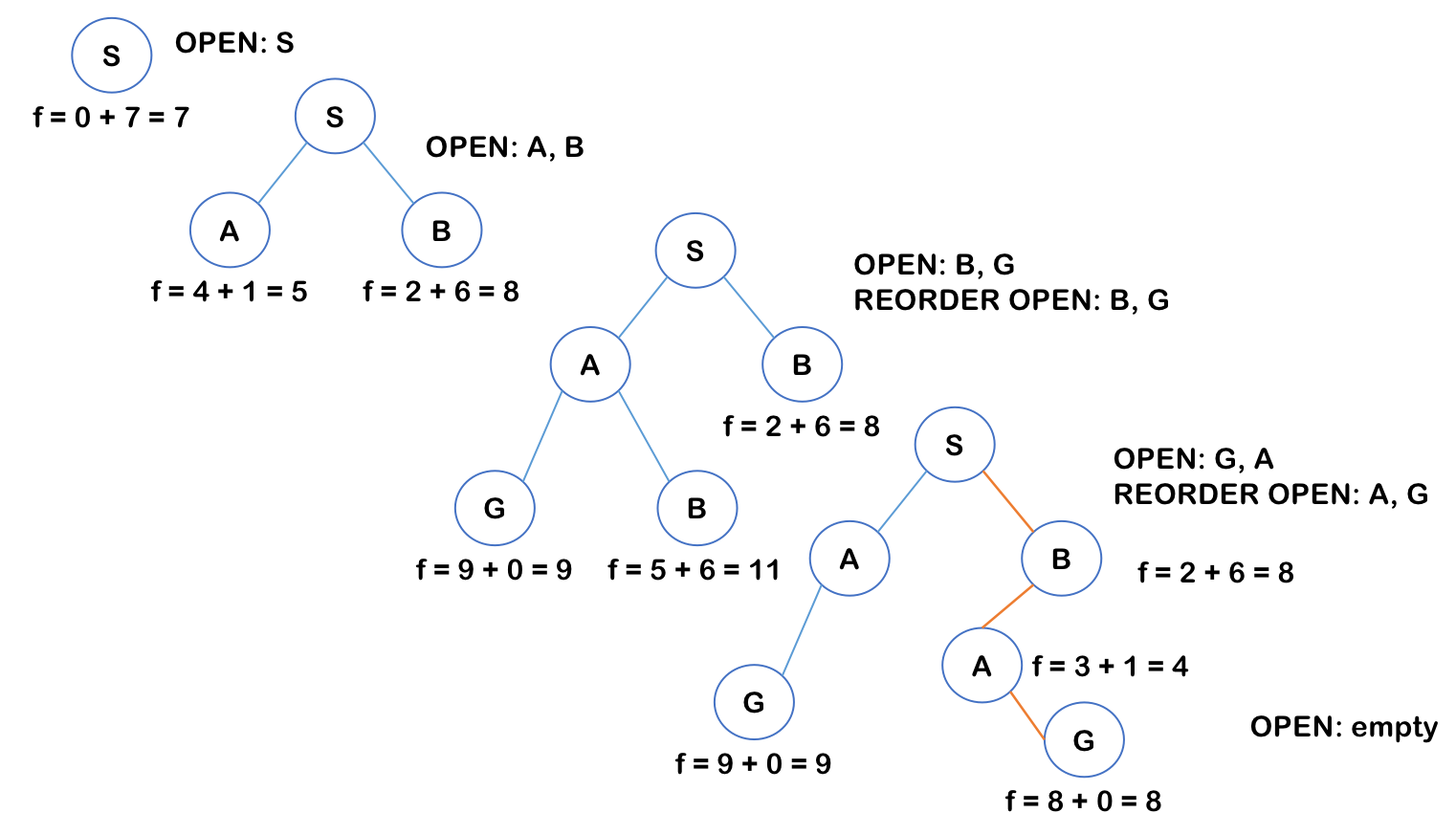
third iteration: s0

the highest value is:



after iteration, through stock, then CD, then CD.

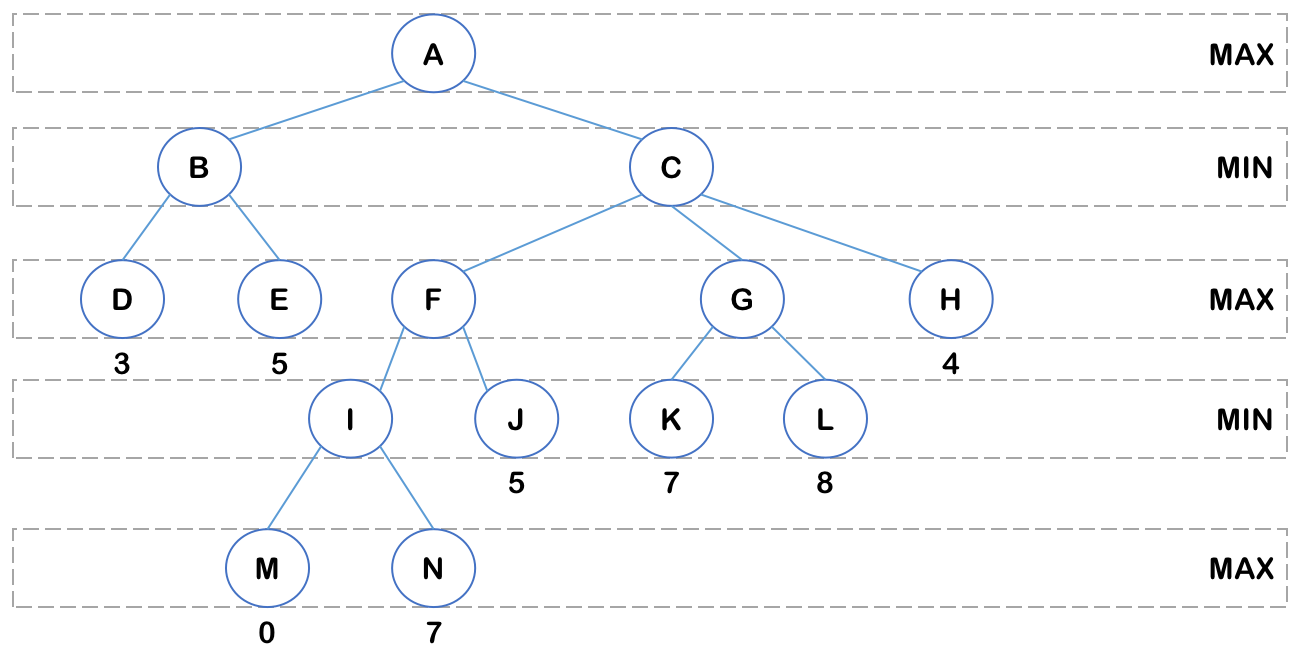
Q4:



Path is: S - A - G

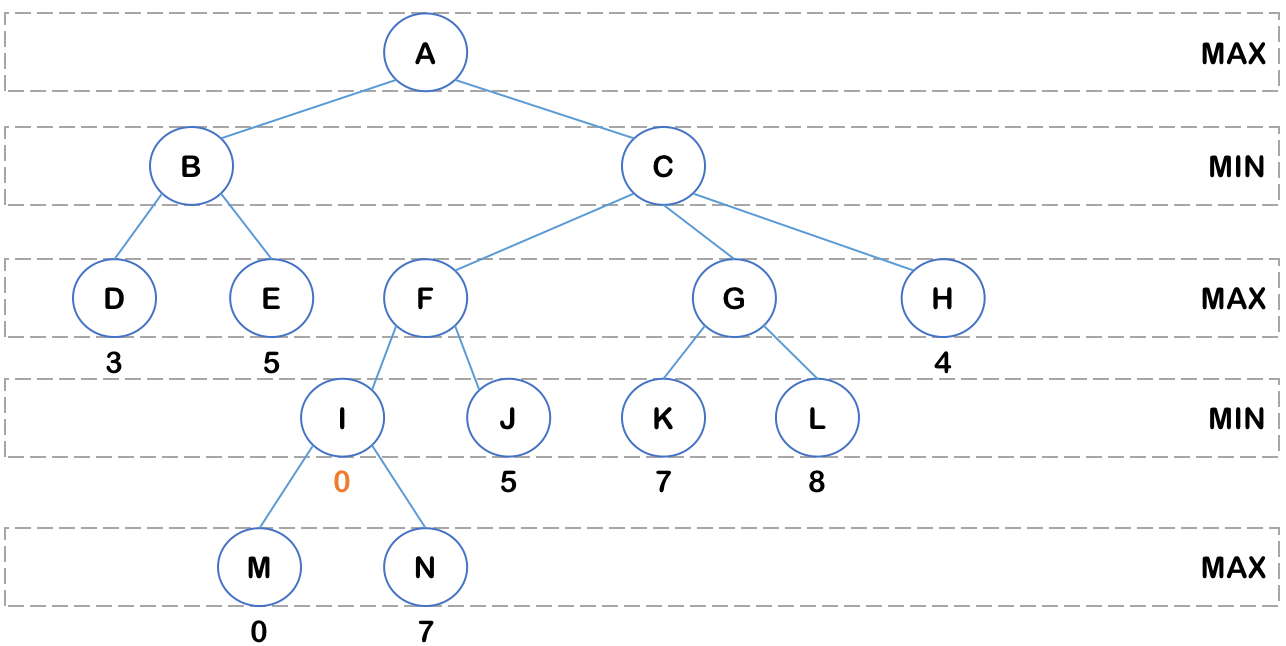
Q5:

**Step 1:**



For node I: min(0, 7) = 0

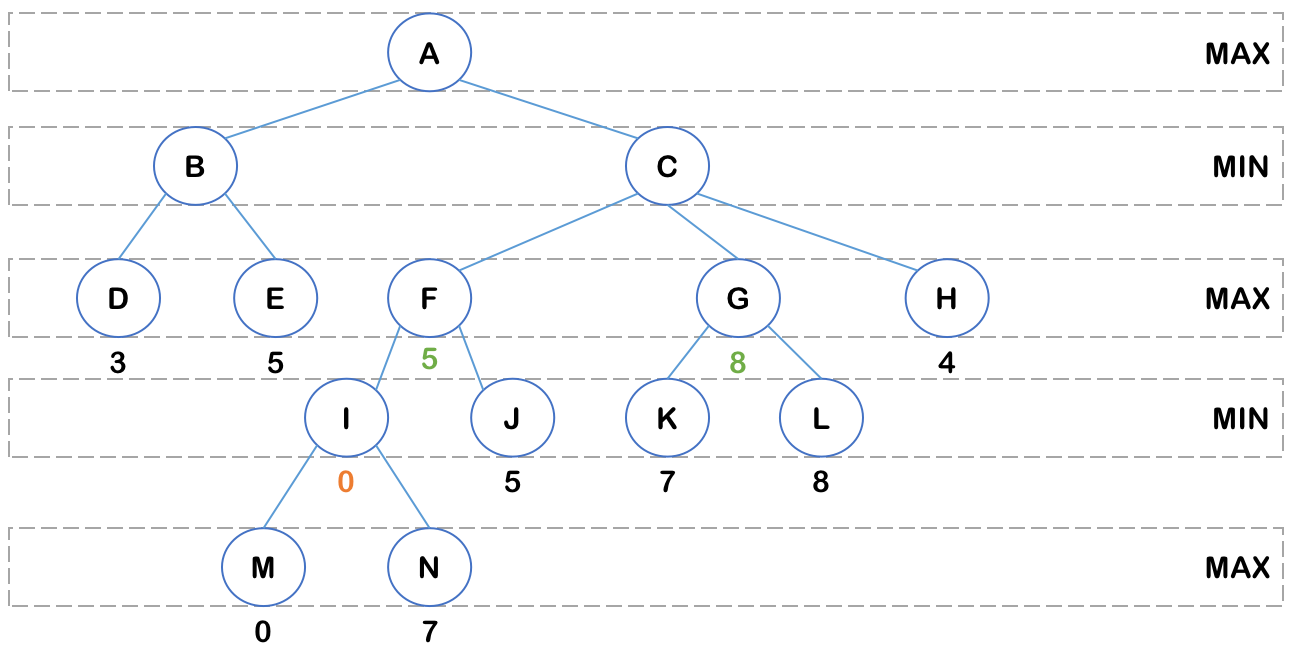
**Step 2:**



For node F: max(0, 5) = 5

For node G: max(7, 8) = 8

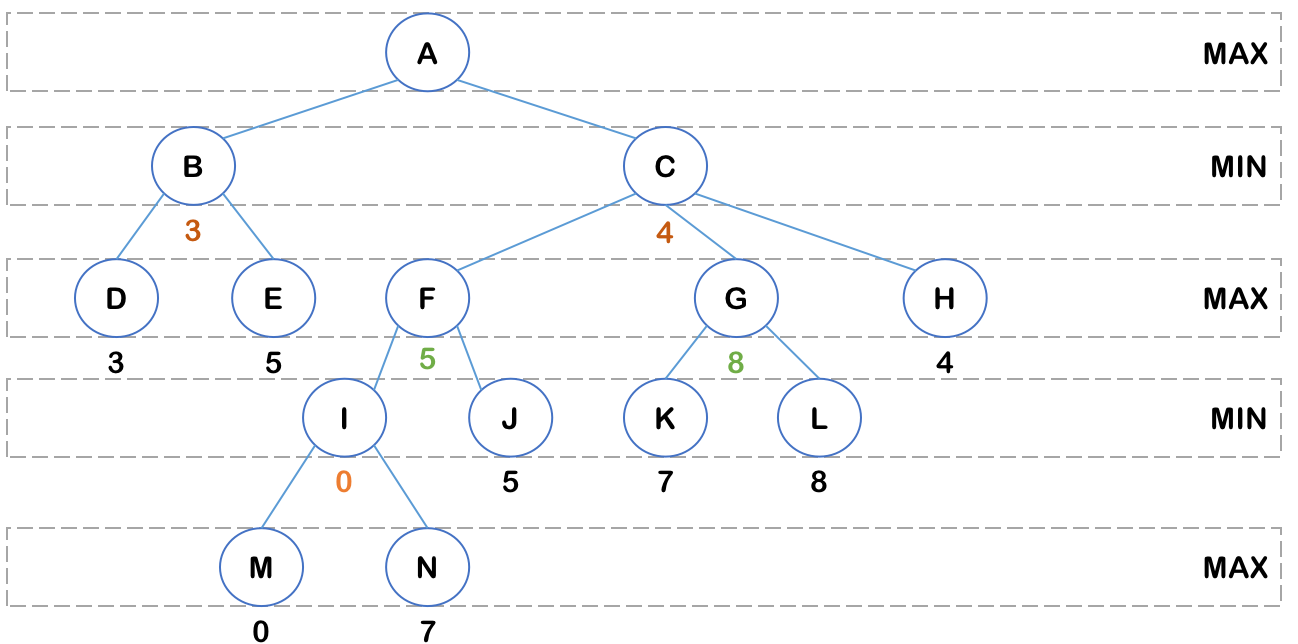
**Step 3:**



For node B: min(3, 5) = 3

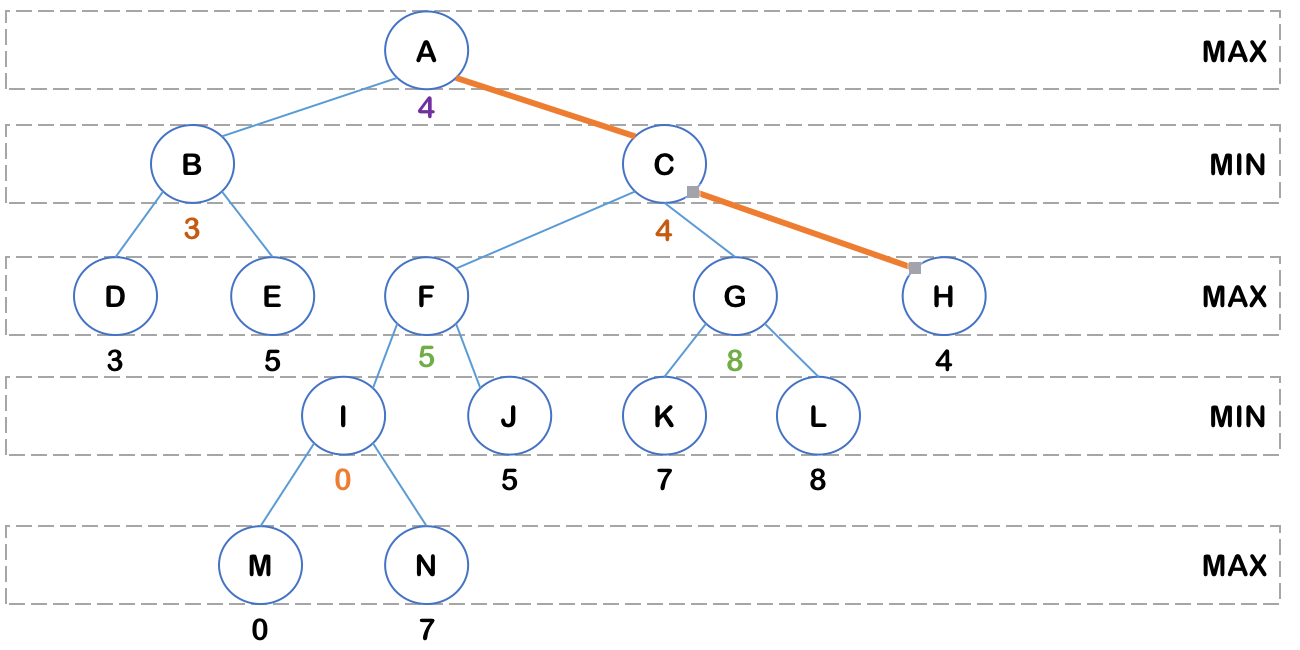
For node C: min(5, 8, 4) = 4

**Step 4:**



For node A: max(3, 4) = 4

**Step 5:**

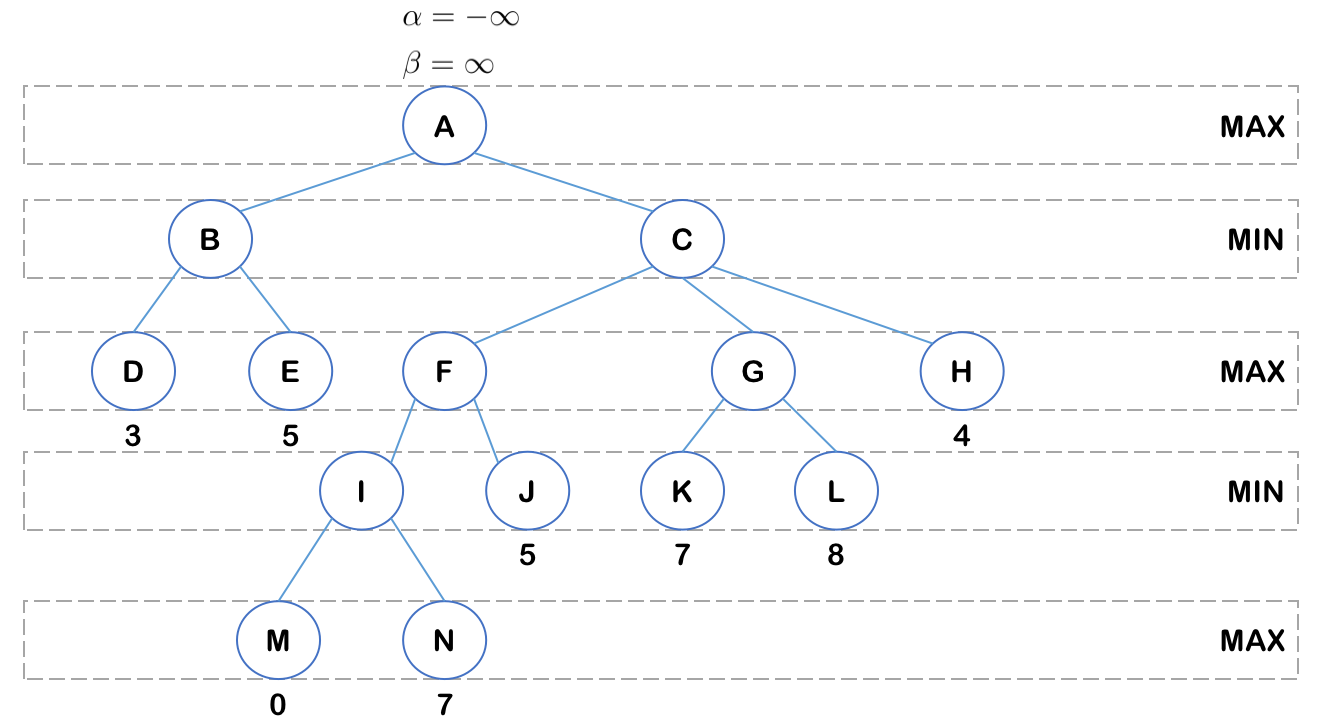


A - C - H

Q6:

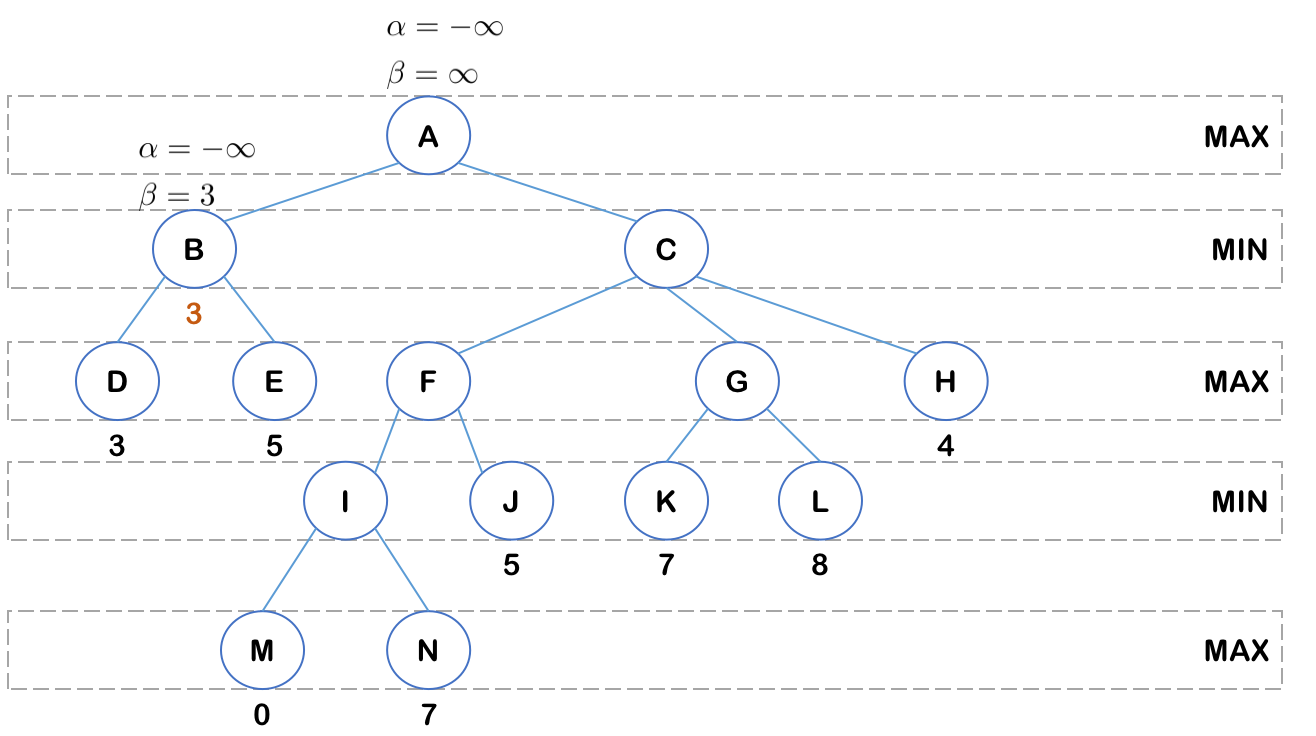
1) Left-to-right alpha-beta prune on the tree

**Step 1:**



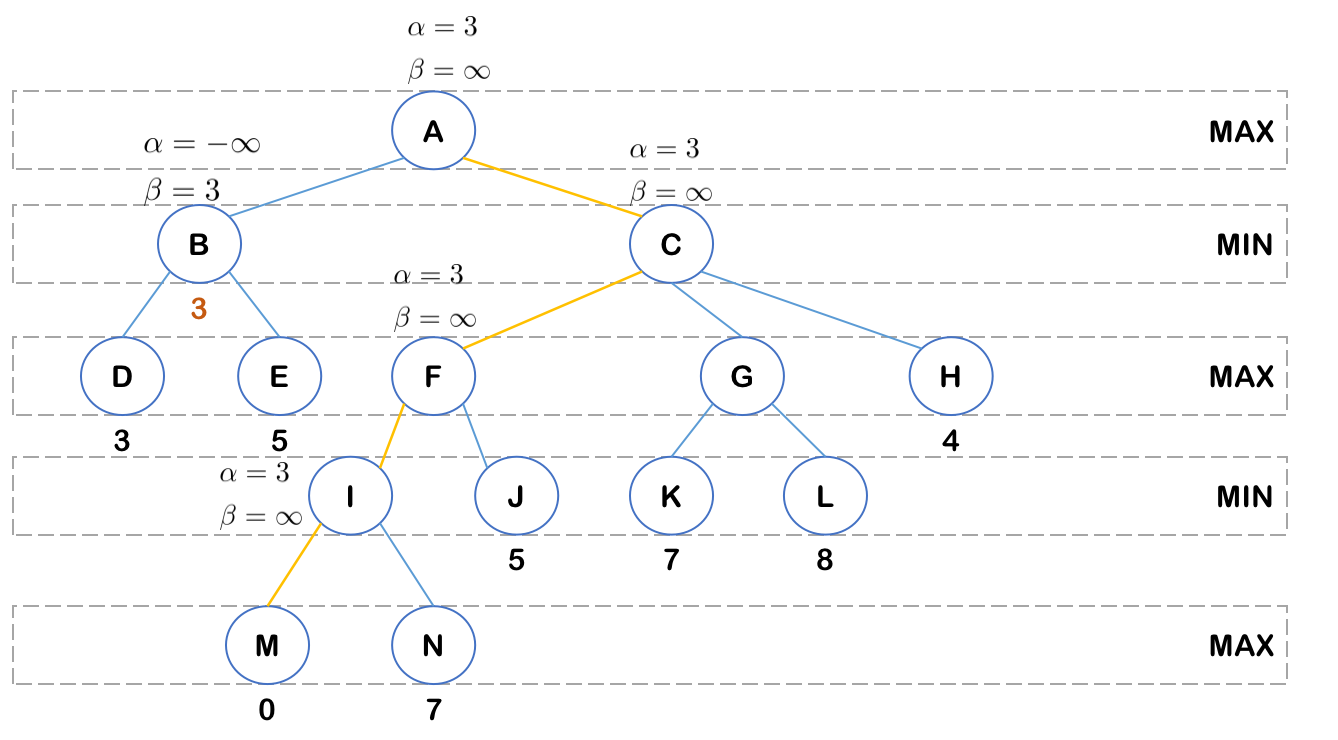
For node B: value of D is 3, update β = min(∞, 3) = 3. value of E is 5, β needs no change, value of B is min(3, 5) = 3

**Step 2:**



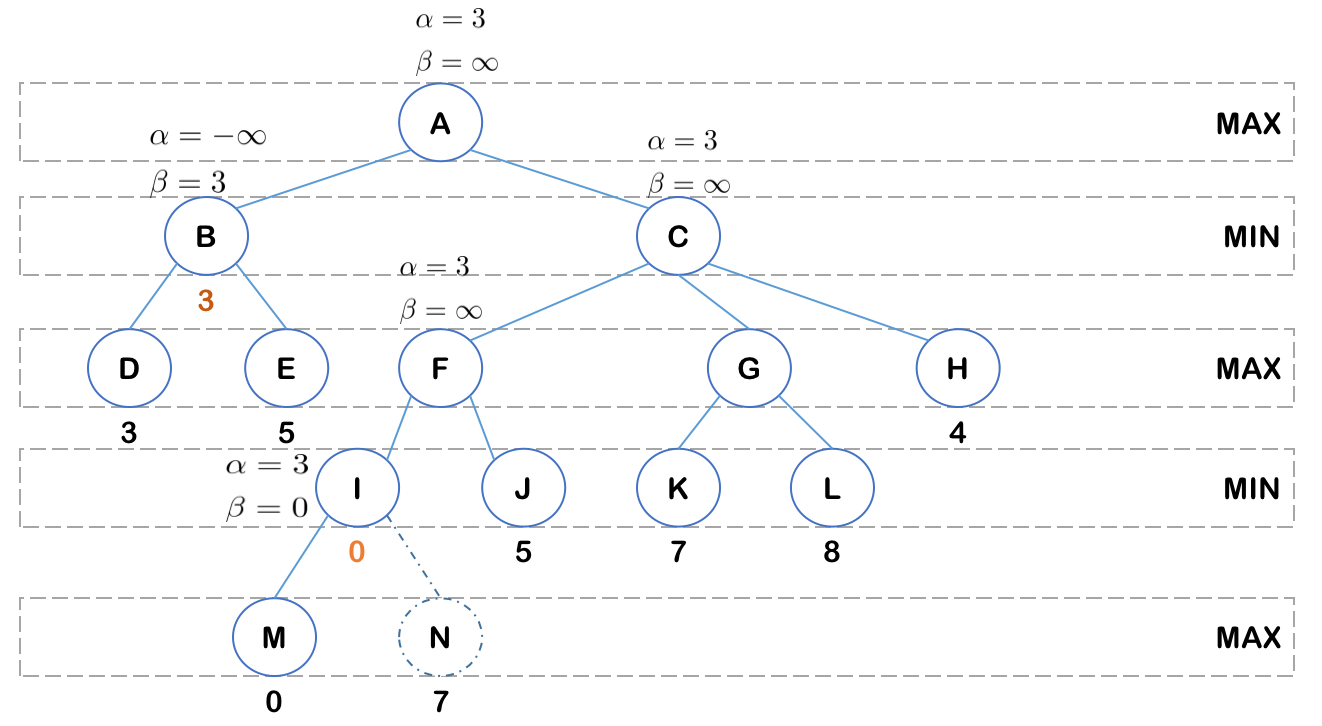
For node A: value of B is 3, update ɑ = max(-∞, 3) = 3

**Step 3:**



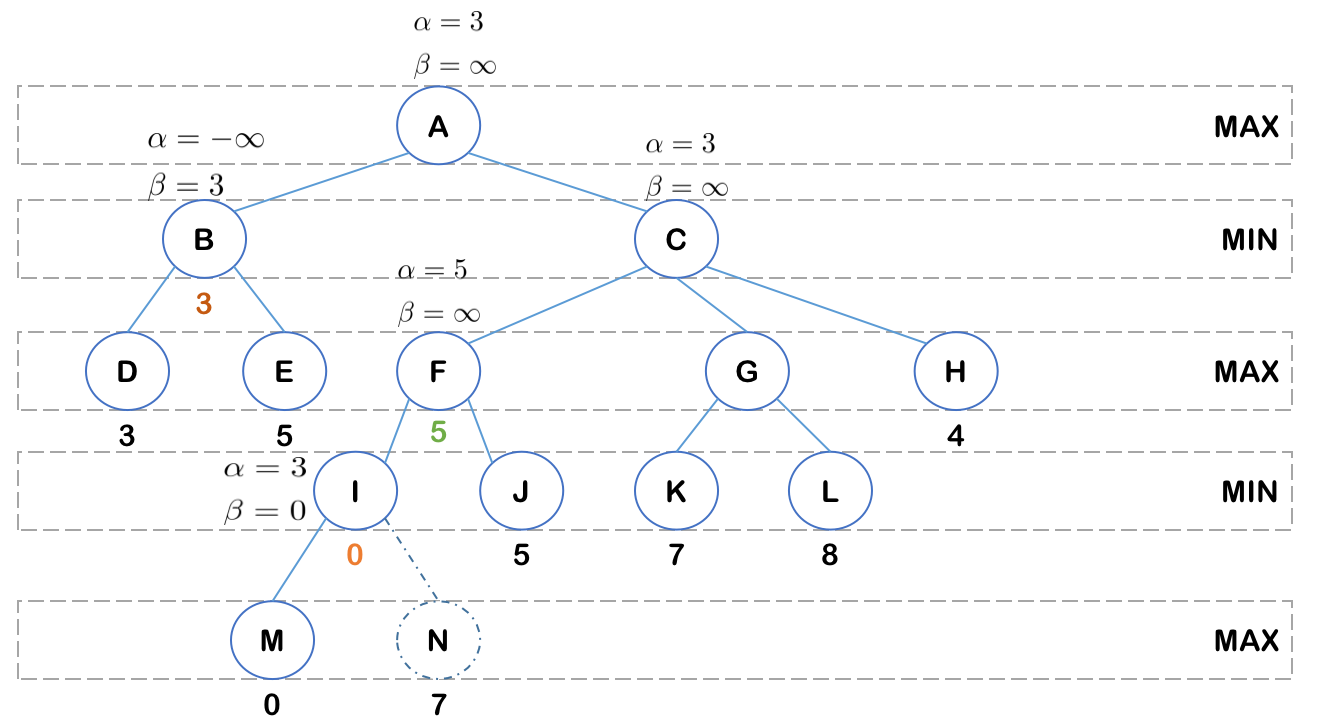
For node I: value of M is 0, update β = min(∞, 0) = 0. ɑ > β, prune node N. value of I is 0.

**Step 4:**



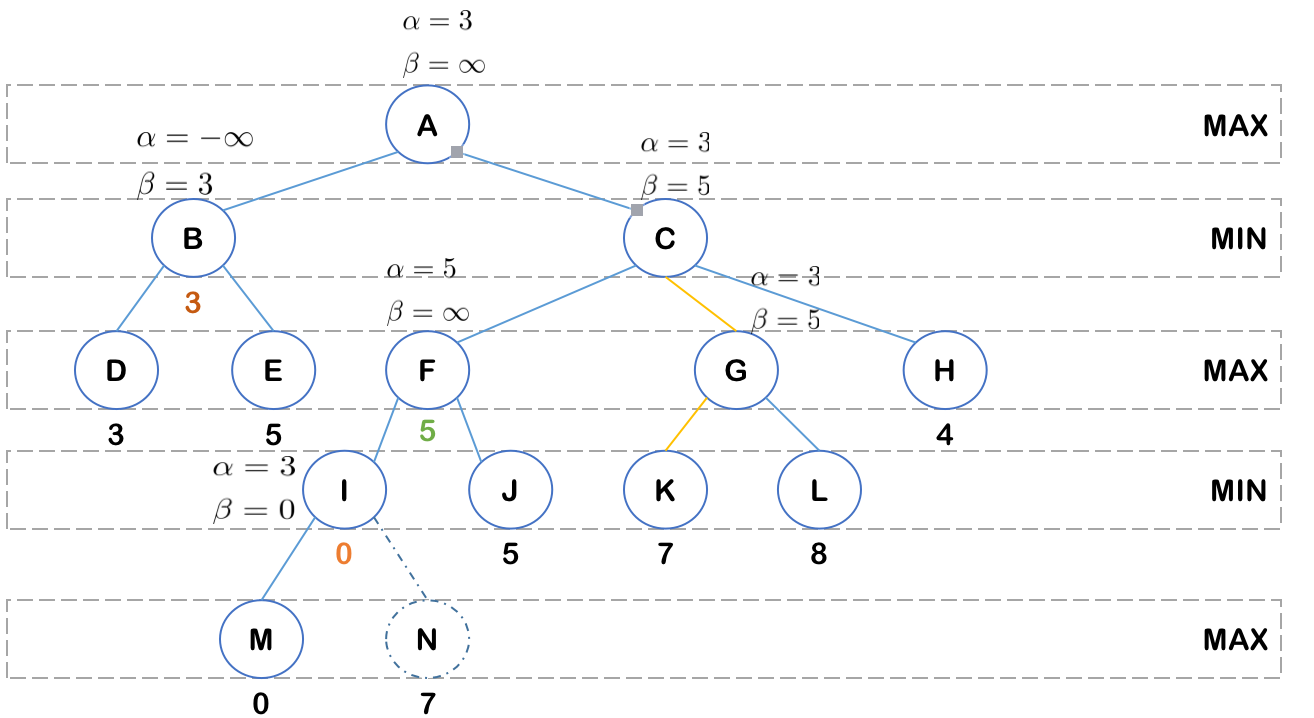
For node F: value of I is 0 < ɑ, ɑ needs no change. value of J is 5 > ɑ, update ɑ = max(3, 5) = 5. value of F is max(0, 5) = 5.

**Step 5:**



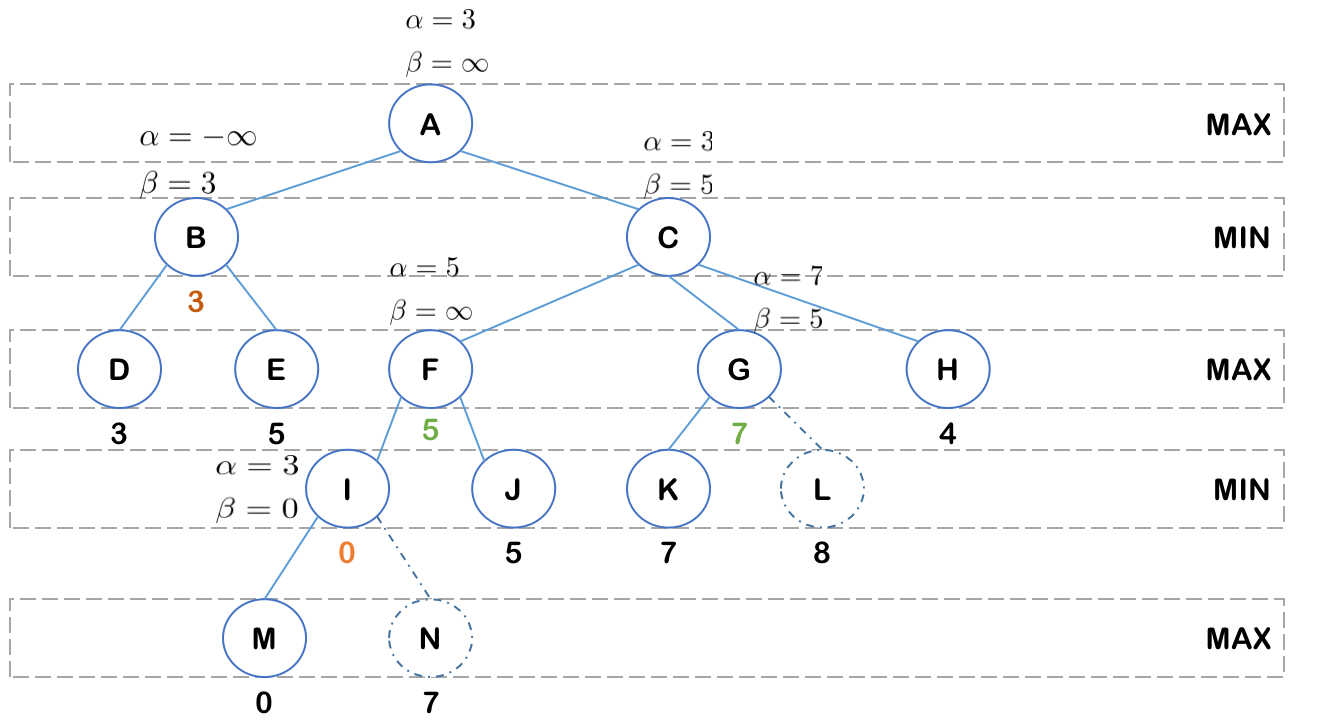
For node C: value of F is 5, update β = min(∞, 5) = 5

**Step 6:**



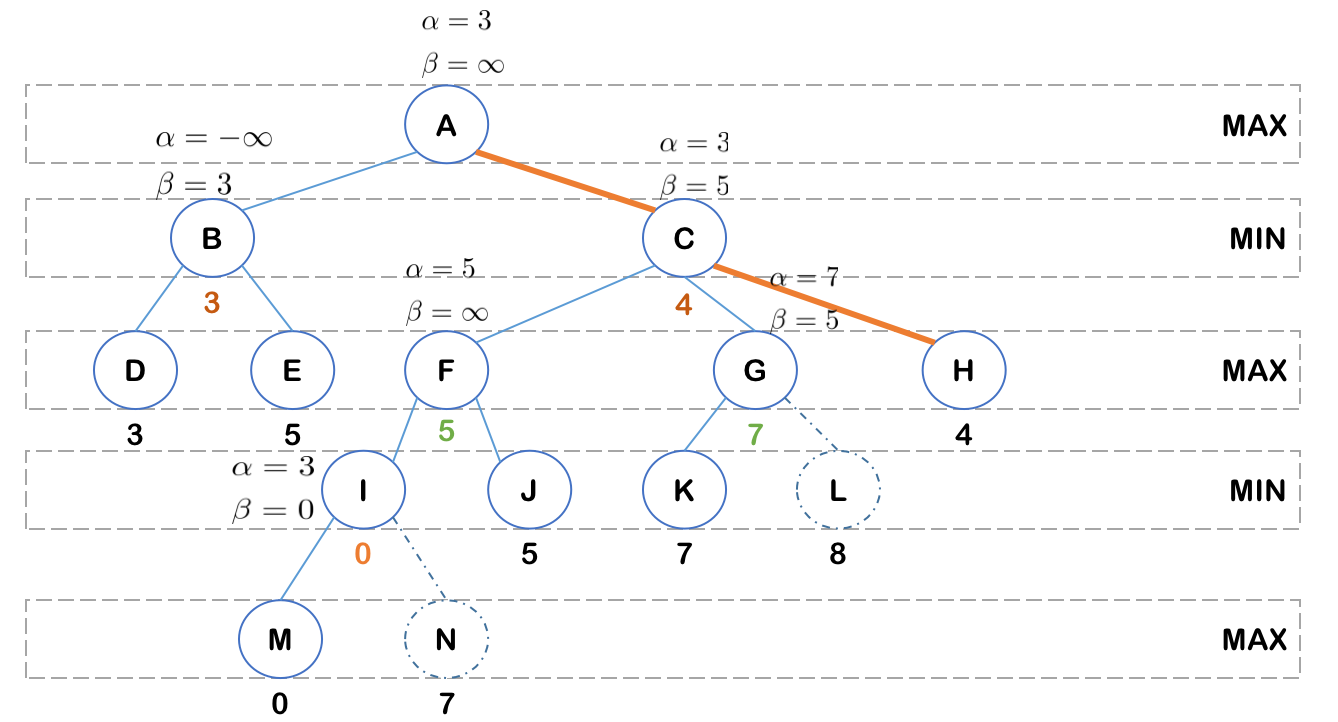
For node G: value of K is 7, update ɑ = max(3, 7) = 7, ɑ > β, prune L, value of G is 5.

**Step 7:**



For node C: value of C is min(5, 7, 4) = 4.

**Step 8:**

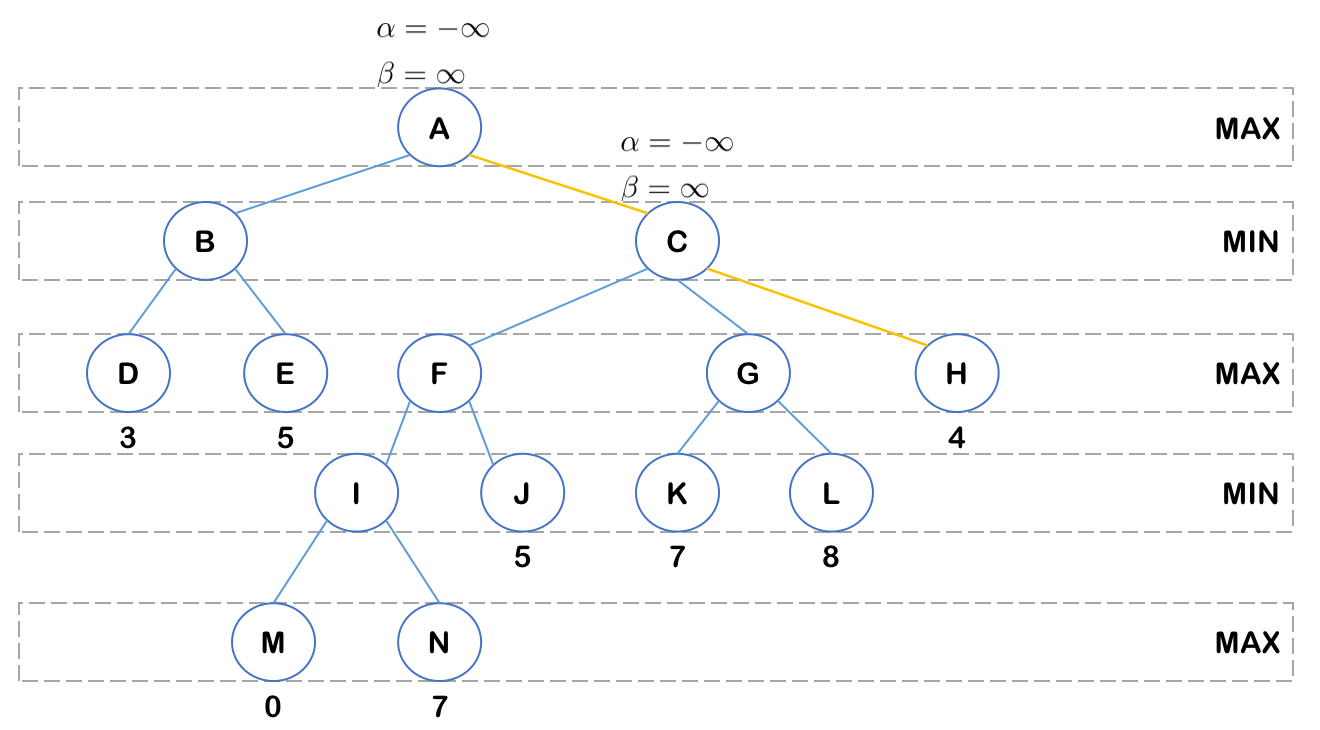


For node A: max(3, 4) = 4

A - C - H

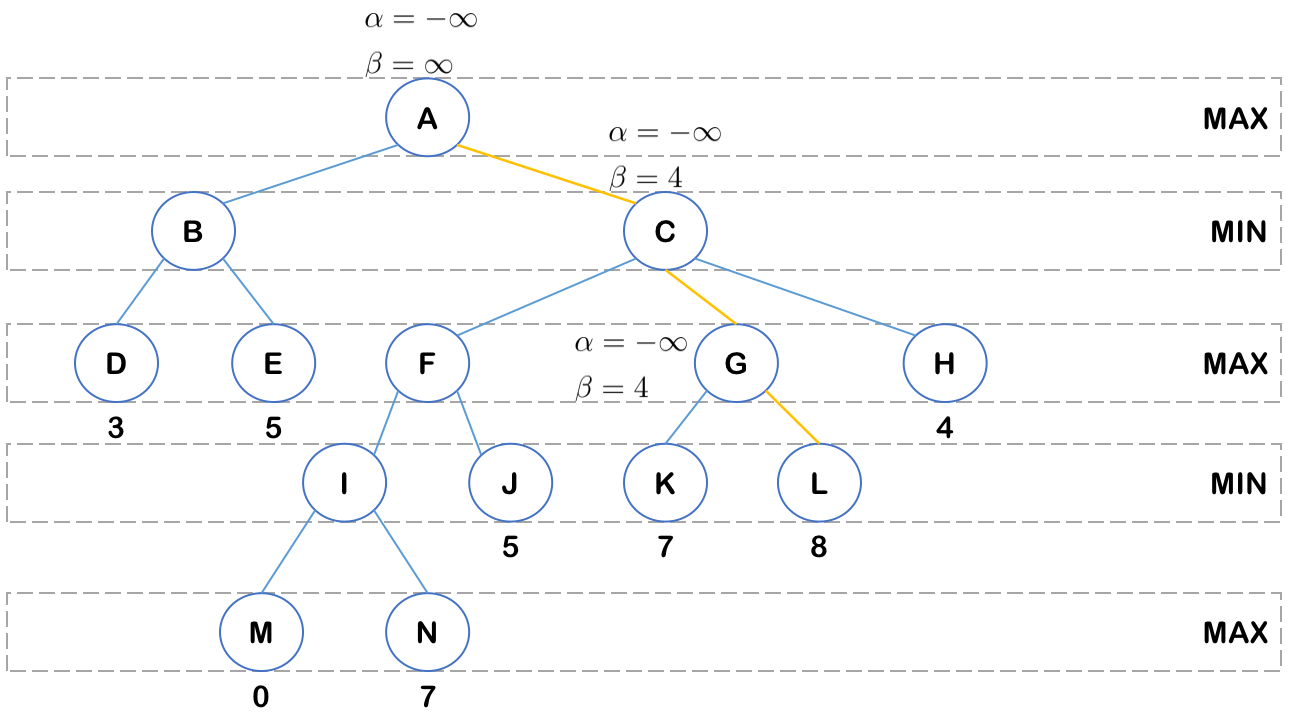
2) Right-to-left prune

**Step 1:**



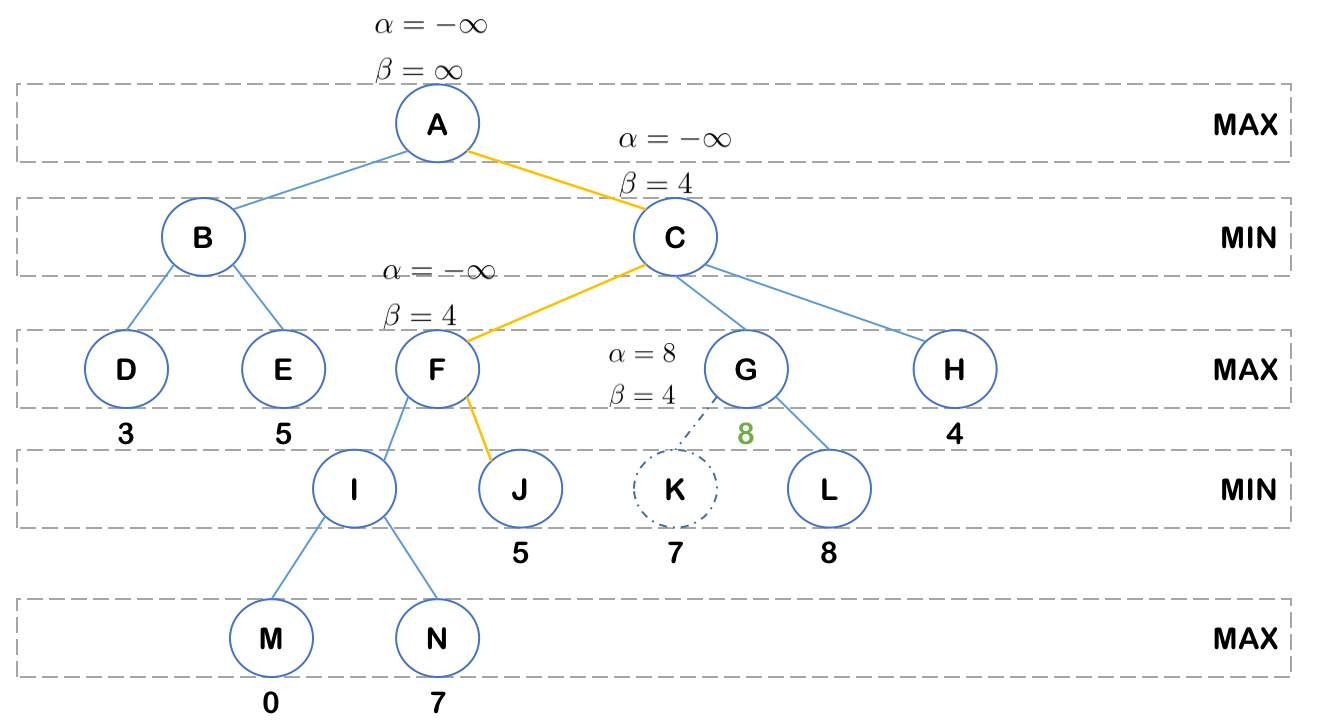
For node C: value of H is 4, update β = min(∞, 4) = 4

**Step 2:**



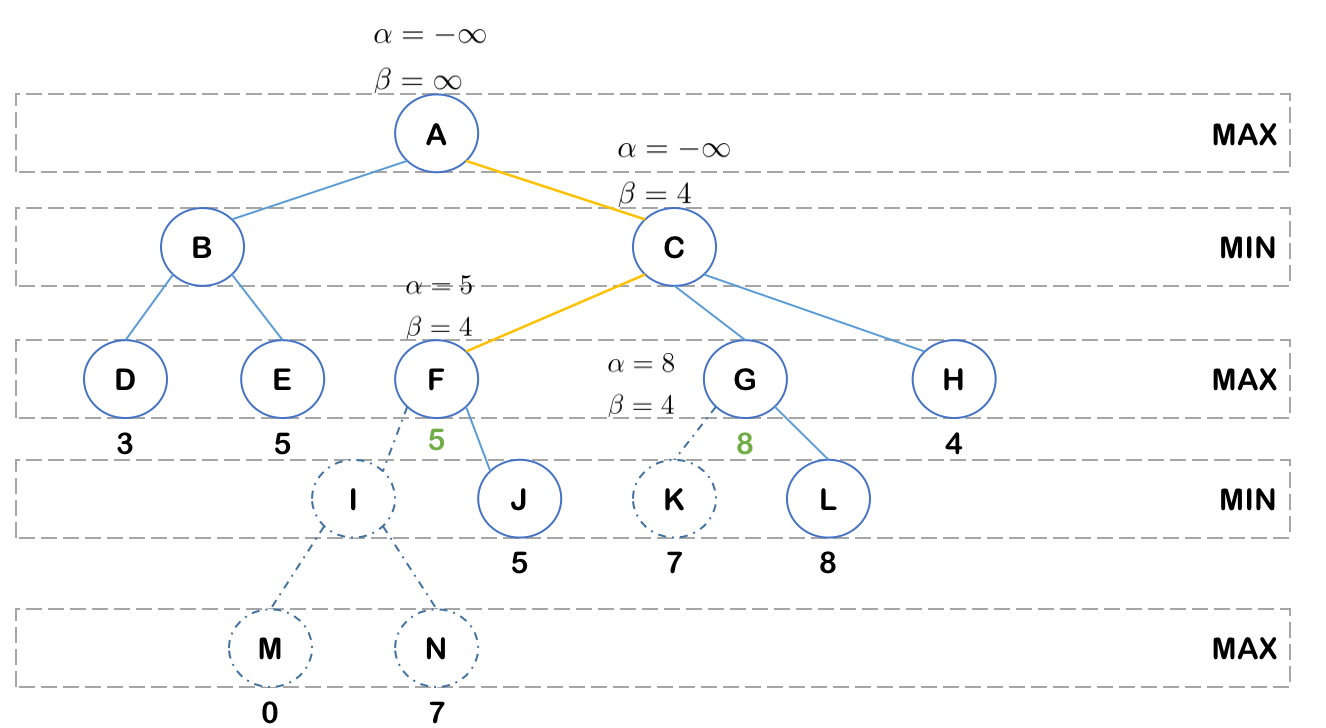
For node G: value of L is 8, update ɑ = max(-∞, 8) = 8. ɑ > β, prune node K. value of G is 8.

**Step 3:**



For node F: value of J is 5, update ɑ = max(-∞, 5) = 5, ɑ > β, prune node I. value of F is 5.

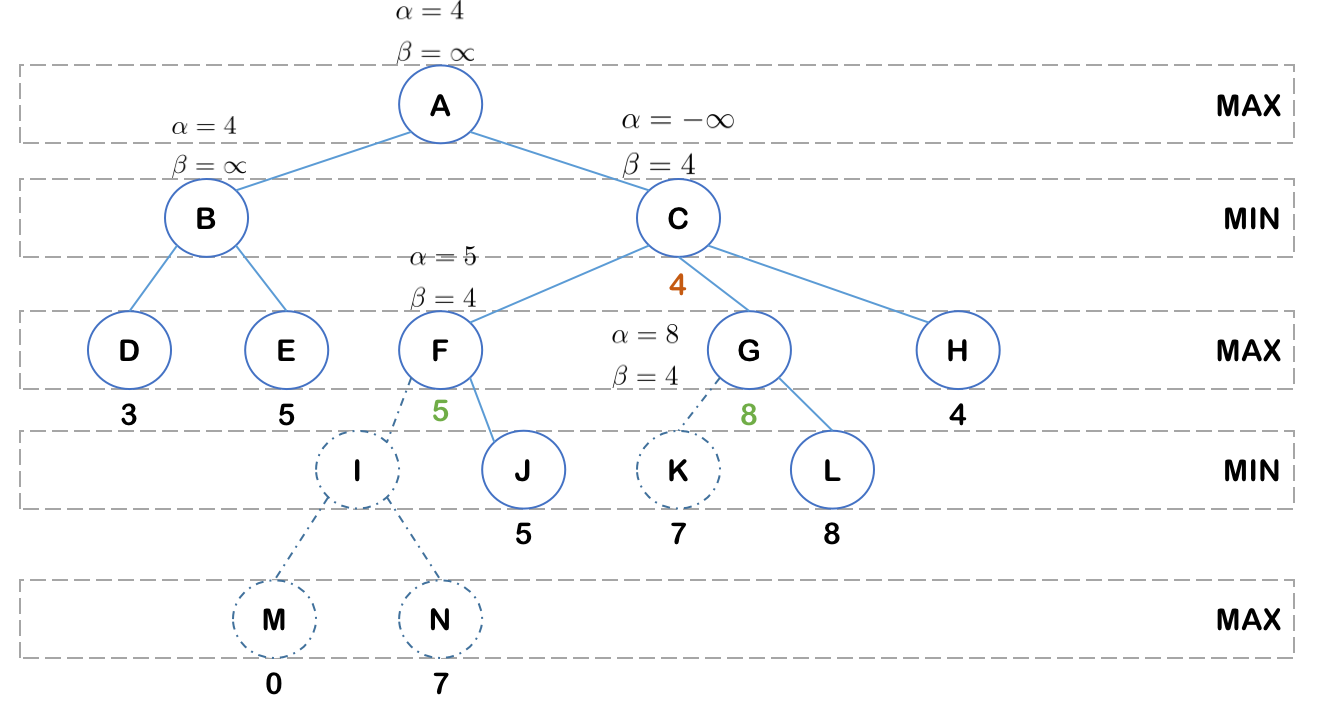
**Step 4:**



For node C: value is min(5, 8, 4) = 4

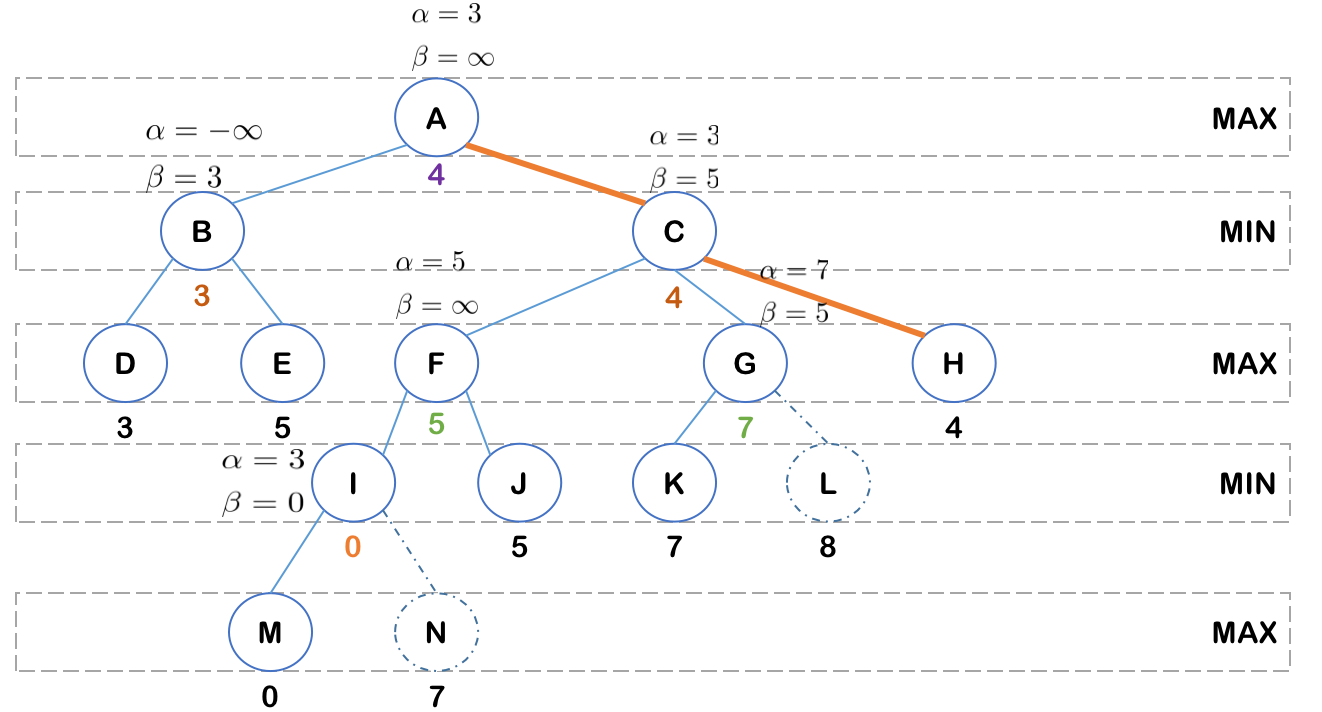
For node A: value of C is 4, update ɑ = max(-∞, 4) = 4

**Step 5:**



For node B: value of 5 is 5, update β = min(∞, 5) = 5. value of D is 3, update β = min(5, 3) = 3. value of B is 3.

**Step 6:**



For node A: value is max(3, 4) = 4

A - C - H