# Adversarial Search

## Artificial Intelligence
## Minimax Algorithm

## Game Playing

# Outcomes

- Solve problem using Adversarial Search strategies

- Game Playing
- Alpha Beta Pruning

# Minimax Algorithm

- Minimax is a kind of backtracking algorithm that is used in game theory to find the optimal move for a player .
- It is widely used in two player turn-based games.
  Example: Chess, Checkers, tic-tac-toe.
- In Minimax the two players are called MAX and MIN.
- MAX ⟹ highest value
- MIN ⟹ lowest value
- The minimax algorithm proceeds all the way down to the terminal node of the tree, then backtrack the tree as the recursion.

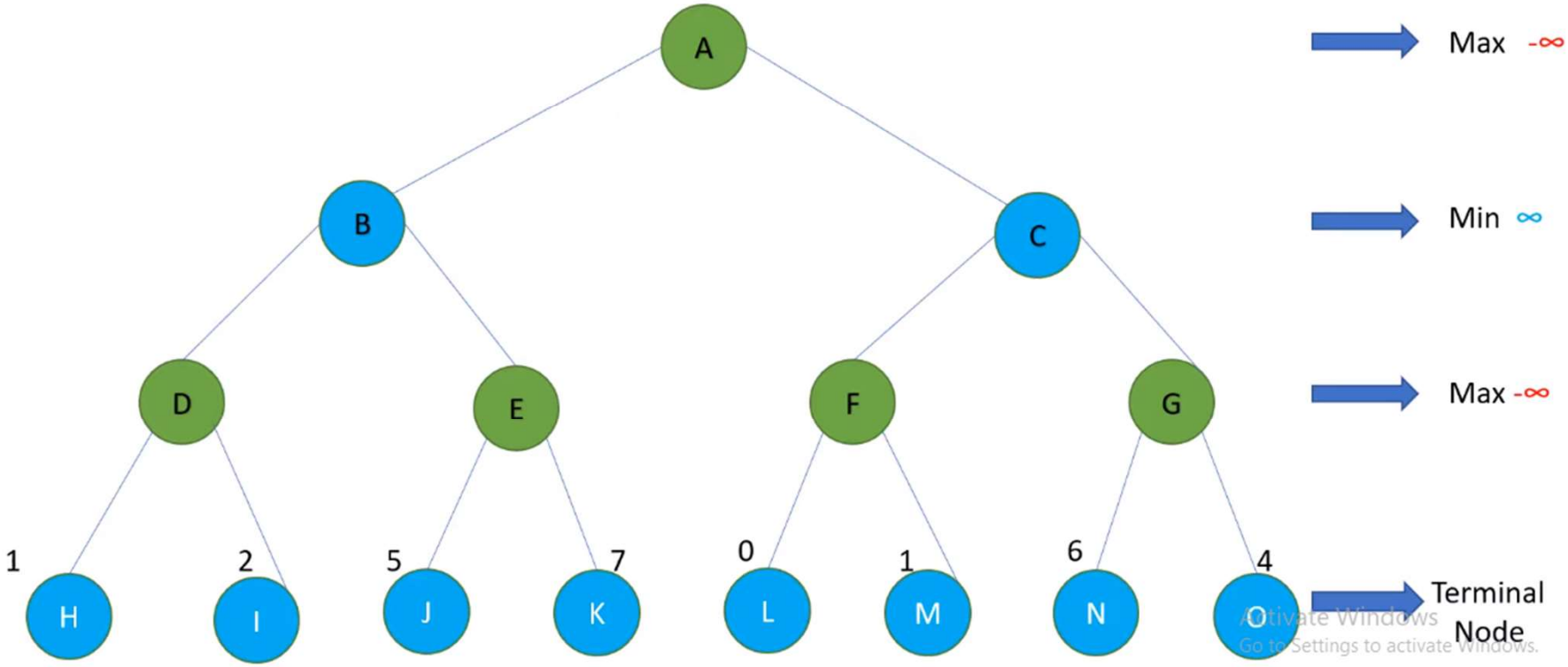
# Minimax Algorithm

- Minimax is a kind of backtracking algorithm that is used in game theory to find the optimal move for a player .
- It is widely used in two player turn-based games.
  Example: Chess, Checkers, tic-tac-toe.
- In Minimax the two players are called MAX and MIN.
- MAX ⟹ highest value
- MIN ⟹ lowest value
- The minimax algorithm proceeds all the way down to the terminal node of the tree, then backtrack the tree as the recursion.

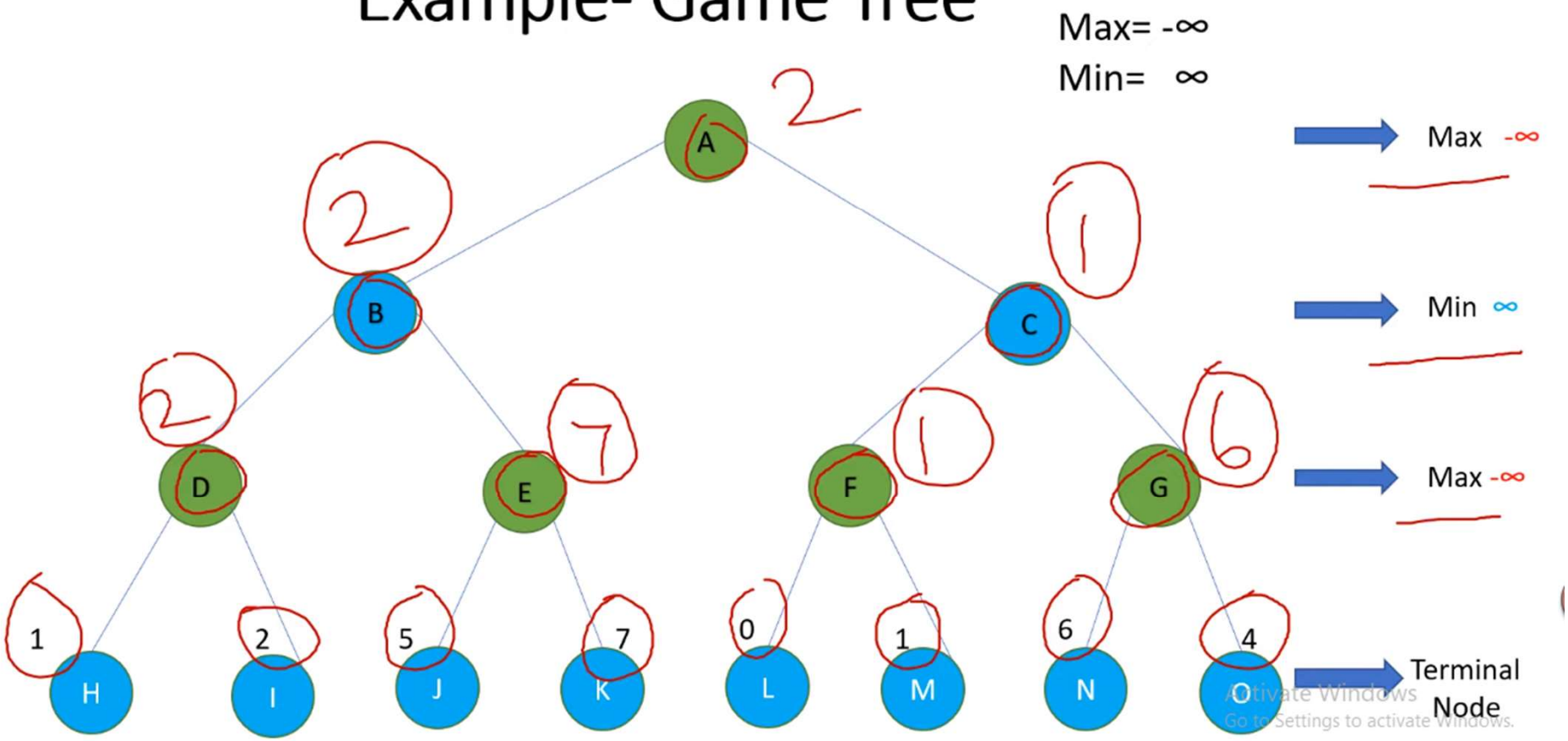
Max – Player 1
Min – Player 2
The value is not score whereas it will be used to make the decision

# Example- Game Tree

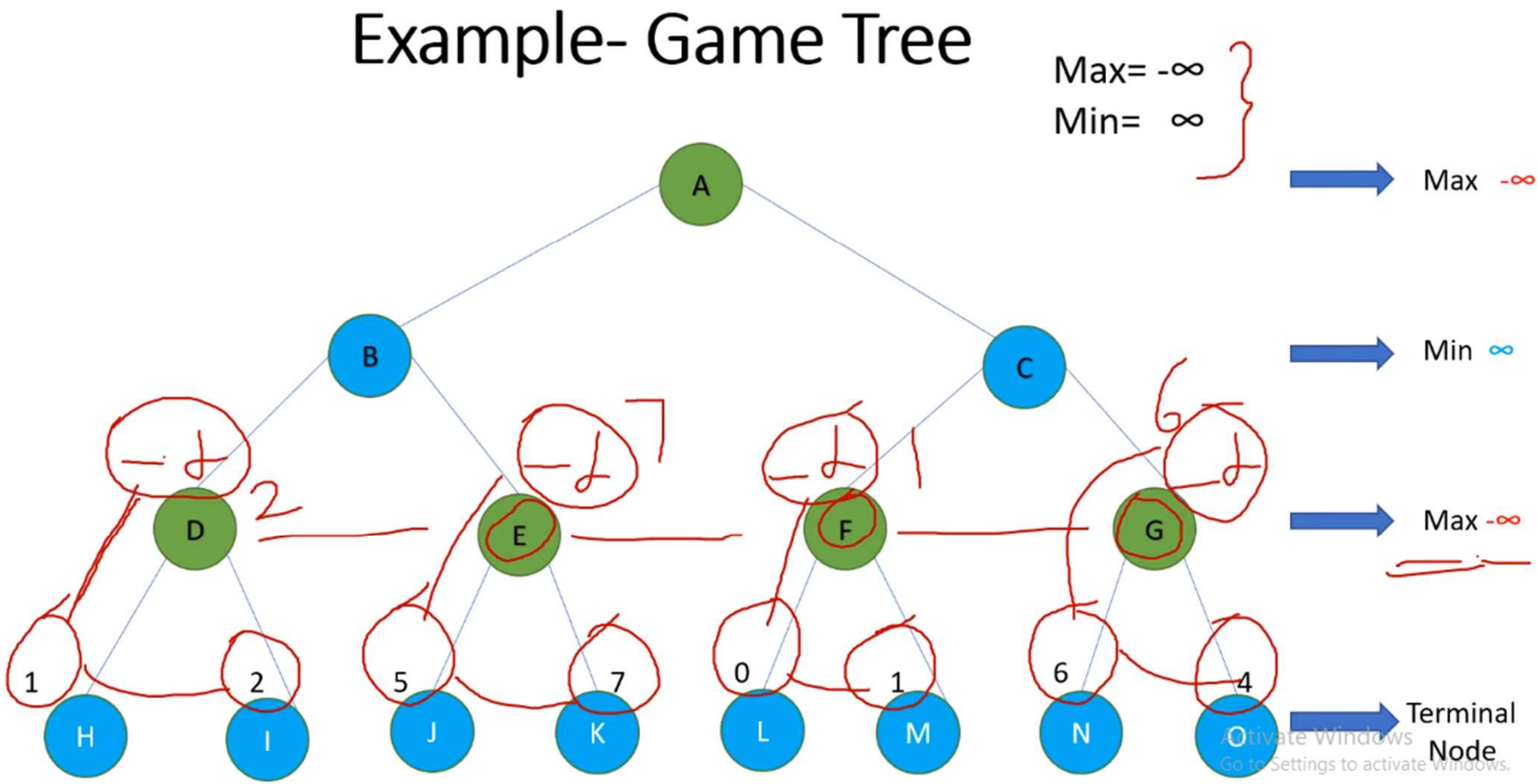
Max= -∞
Min= ∞



# Example- Game Tree



# Let us see how to implement it

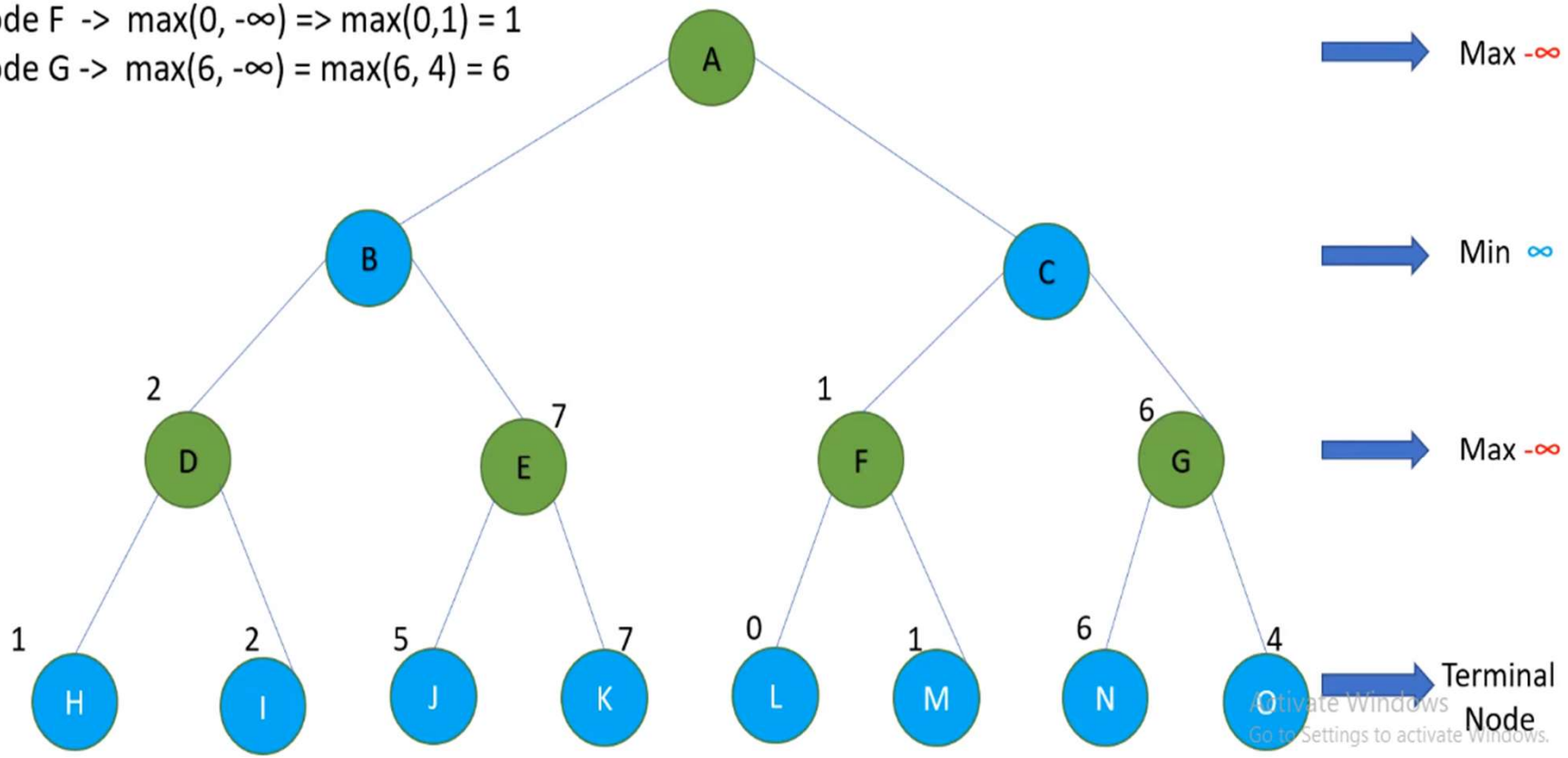


# Example

Max= -∞

Min= ∞

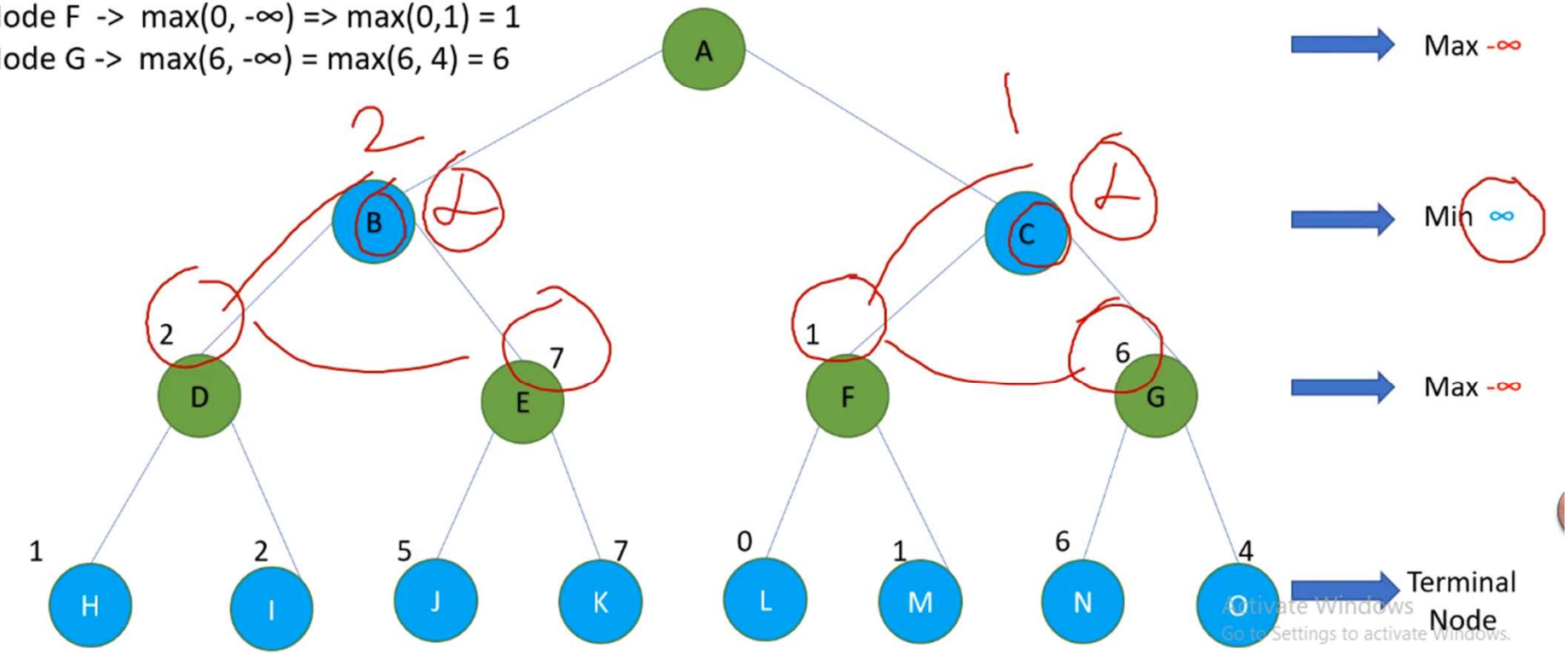
- Node D -> max(1,-∞) => max(1,2)= 2
- Node E -> max(5, -∞) => max(5, 7)= 7
- Node F -> max(0, -∞) => max(0,1) = 1
- Node G -> max(6, -∞) = max(6, 4) = 6



# Example

Max= -∞
Min= ∞

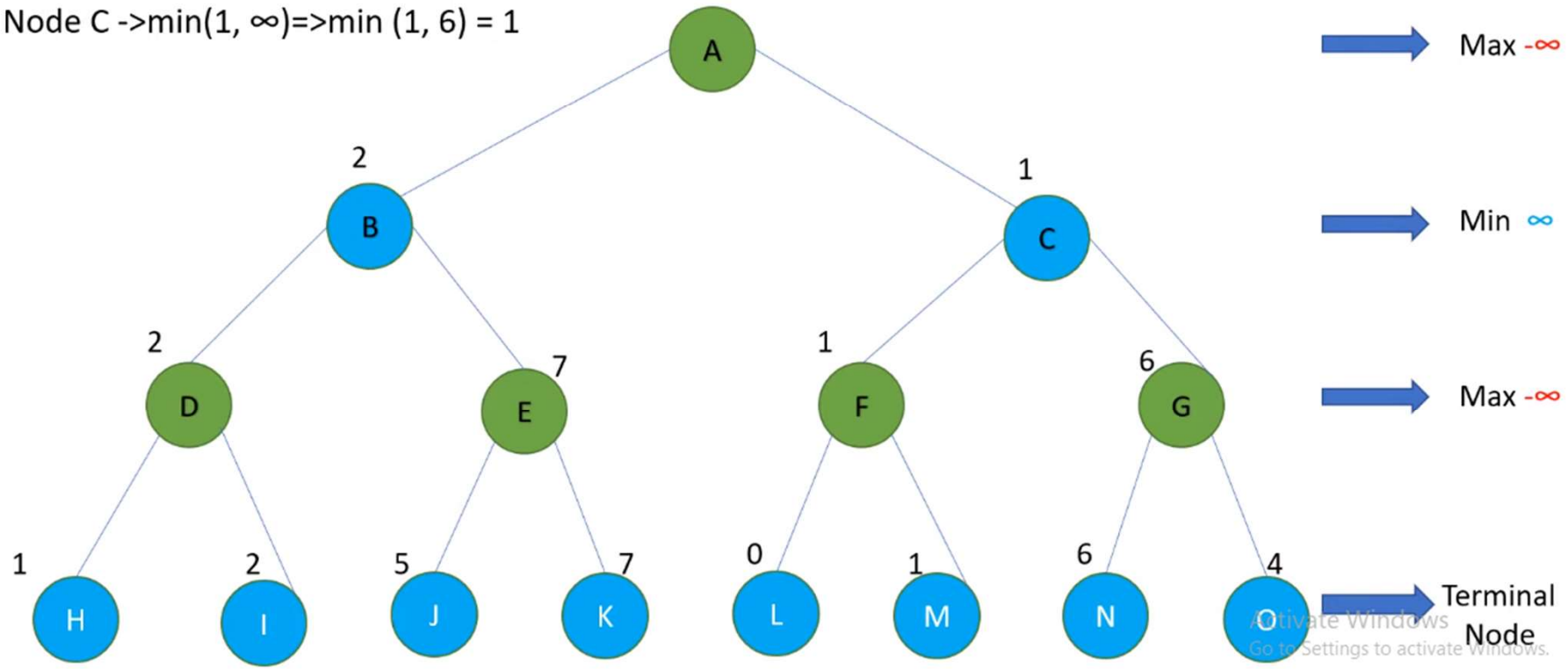
- Node D -> max(1,-∞) => max(1,2)= 2
- Node E -> max(5, -∞) => max(5, 7)= 7
- Node F -> max(0, -∞) => max(0,1) = 1
- Node G -> max(6, -∞) = max(6, 4) = 6



# Example

Max= -∞
Min= ∞

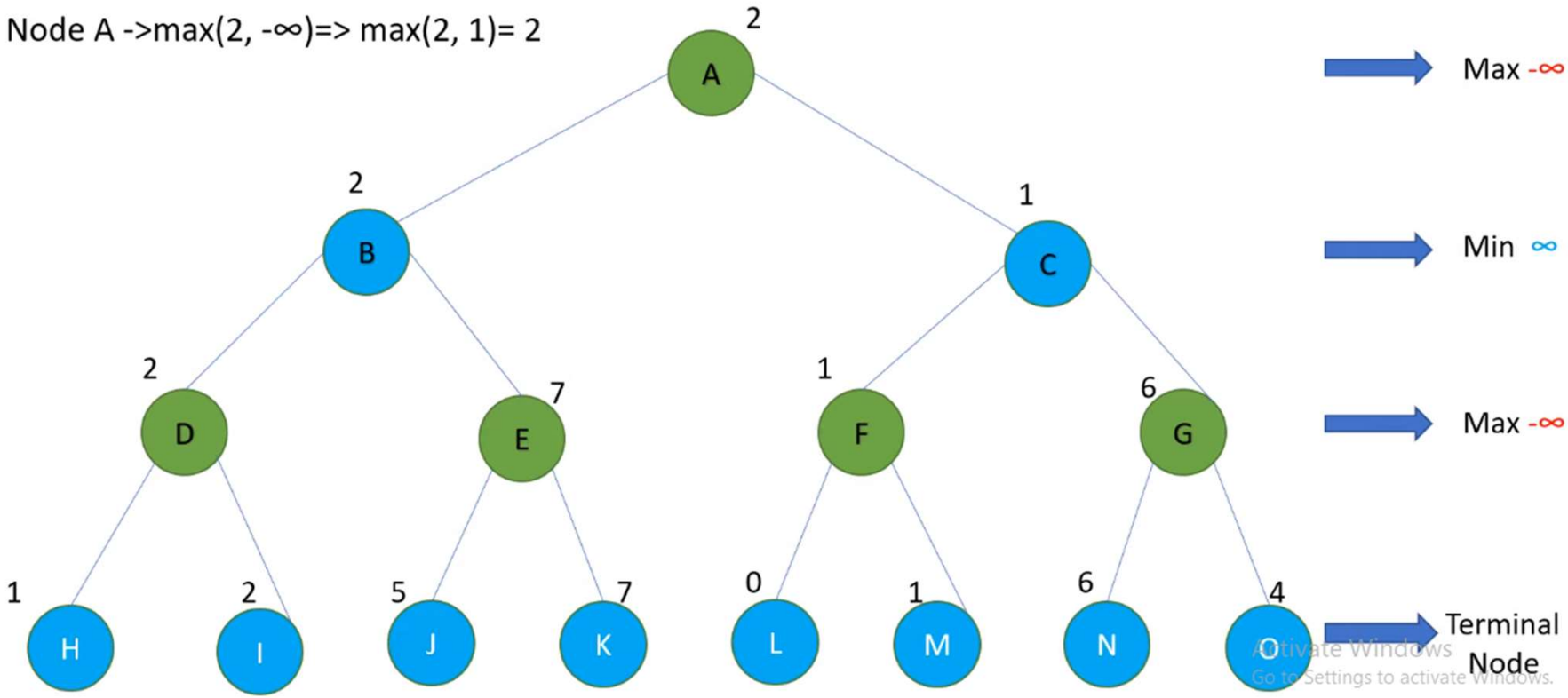
Node B ->min(2, ∞)=> min(2,7) = 2
Node C ->min(1, ∞)=>min (1, 6) = 1



# Example

Max= -∞
Min= ∞

Node A ->max(2, -∞)=> max(2, 1)= 2



# Draw Backs

- Since we are visiting all nodes here, **Time Complexity increases**

- To reduce this, we go for **Alpha-Beta Pruning**

# Alpha Beta Pruning

# Artificial Intelligence

# Alpha Beta Pruning

## Game Playing

# Alpha Beta Pruning

- Alpha-beta pruning is a modified version of the minimax algorithm.
- It is an optimization technique for the minimax algorithm.
- Alpha-beta pruning is the pruning(cutting down) of useless branches in decision trees.
- **Alpha (α) :** highest-value

  Initial value of $\alpha=-\infty$

  Max player will only update the value of alpha
- **Beta (β) :** lowest-value

  Initial value of $\beta=+\infty$

  Min player will only update the value of beta.

Condition

$\alpha >= \beta$

# Alpha Beta Pruning

- Alpha-beta pruning is a modified version of the minimax algorithm.
- It is an optimization technique for the minimax algorithm.
- Alpha-beta pruning is the pruning(cutting down) of useless branches in decision trees.
- **Alpha (α) :** highest-value

  Initial value of $\alpha = -\infty$

  Max player will only update the value of alpha
- **Beta (β) :** lowest-value

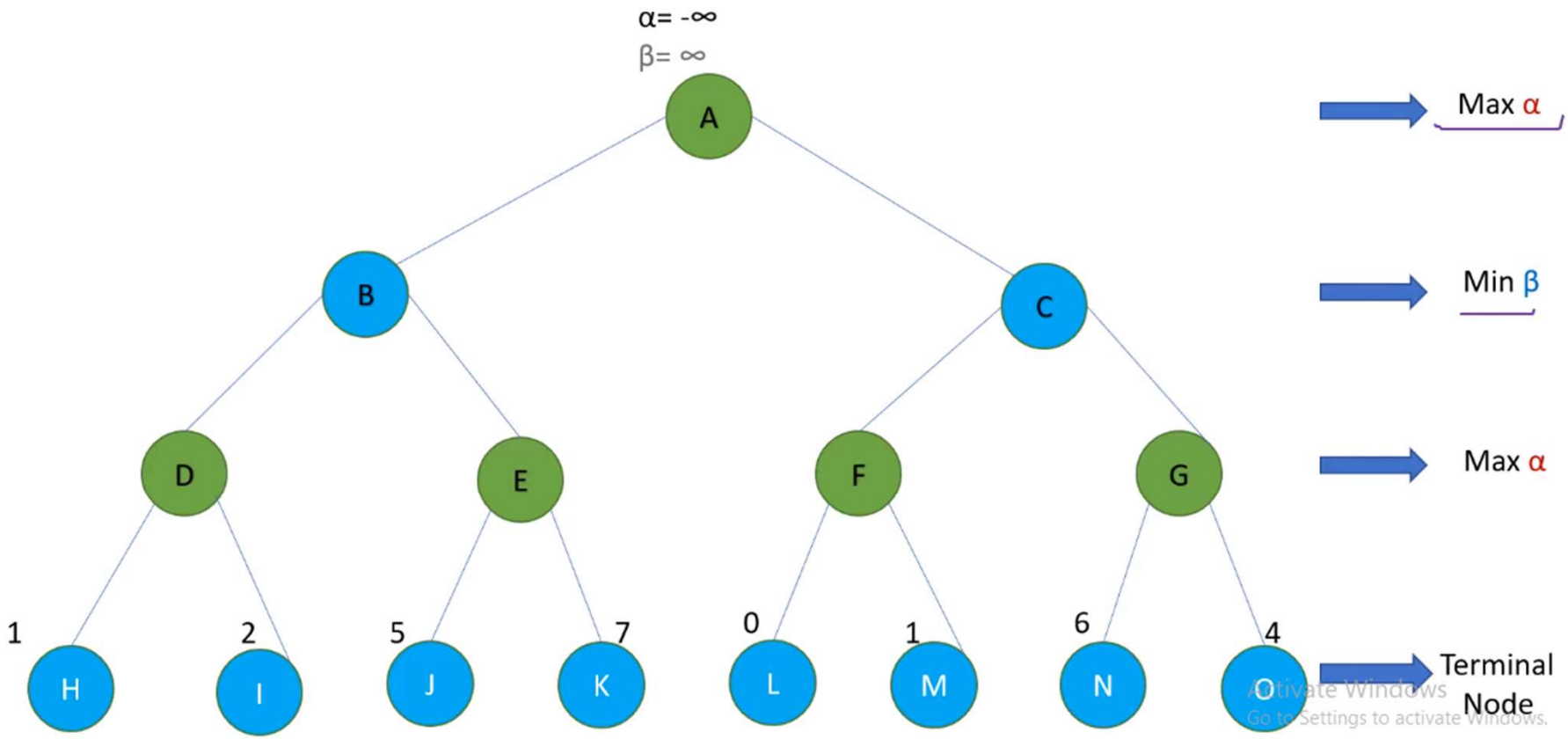
  Initial value of $\beta = +\infty$

  Min player will only update the value of beta.

Condition

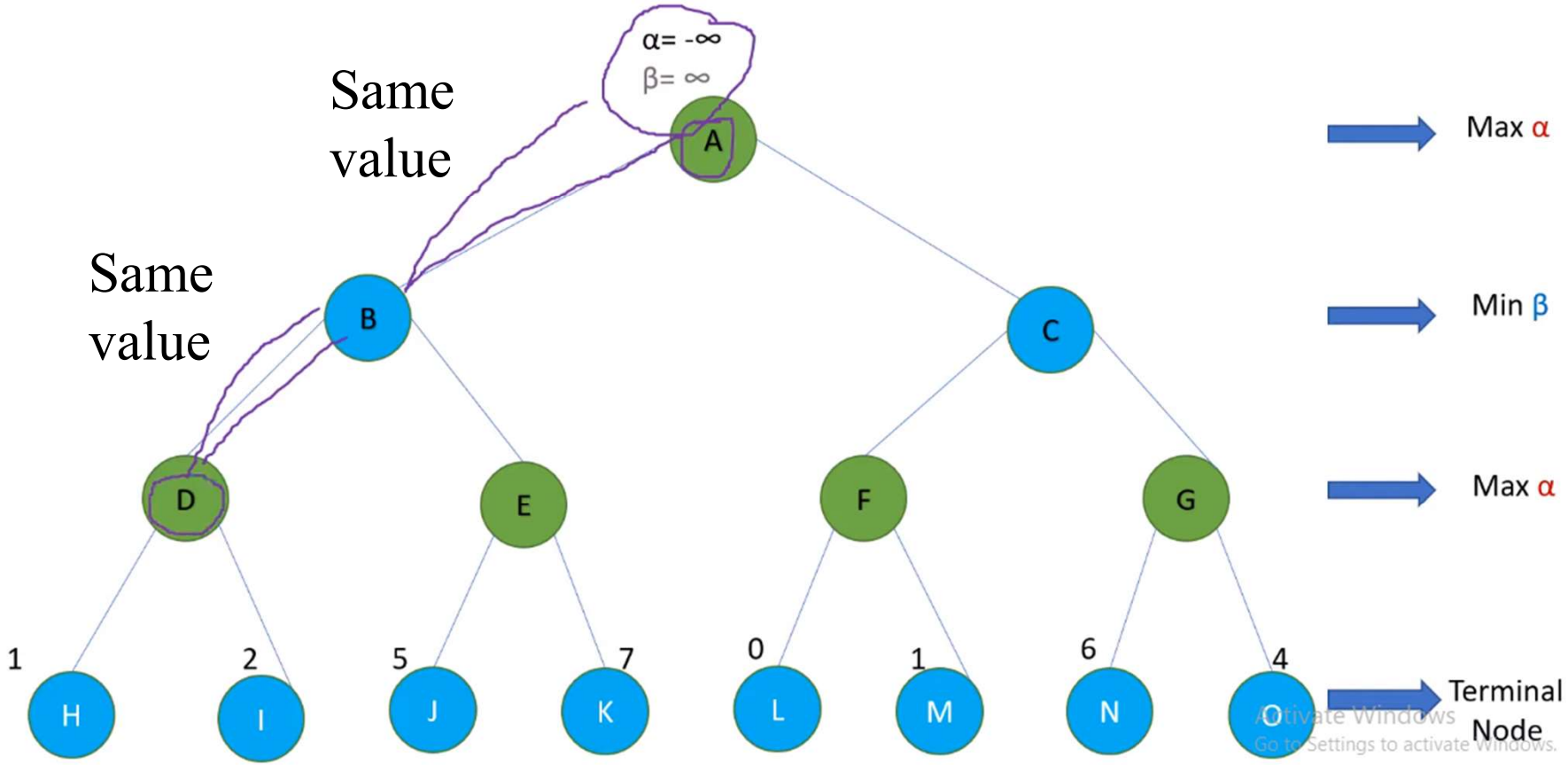
$\alpha >= \beta$

# Example



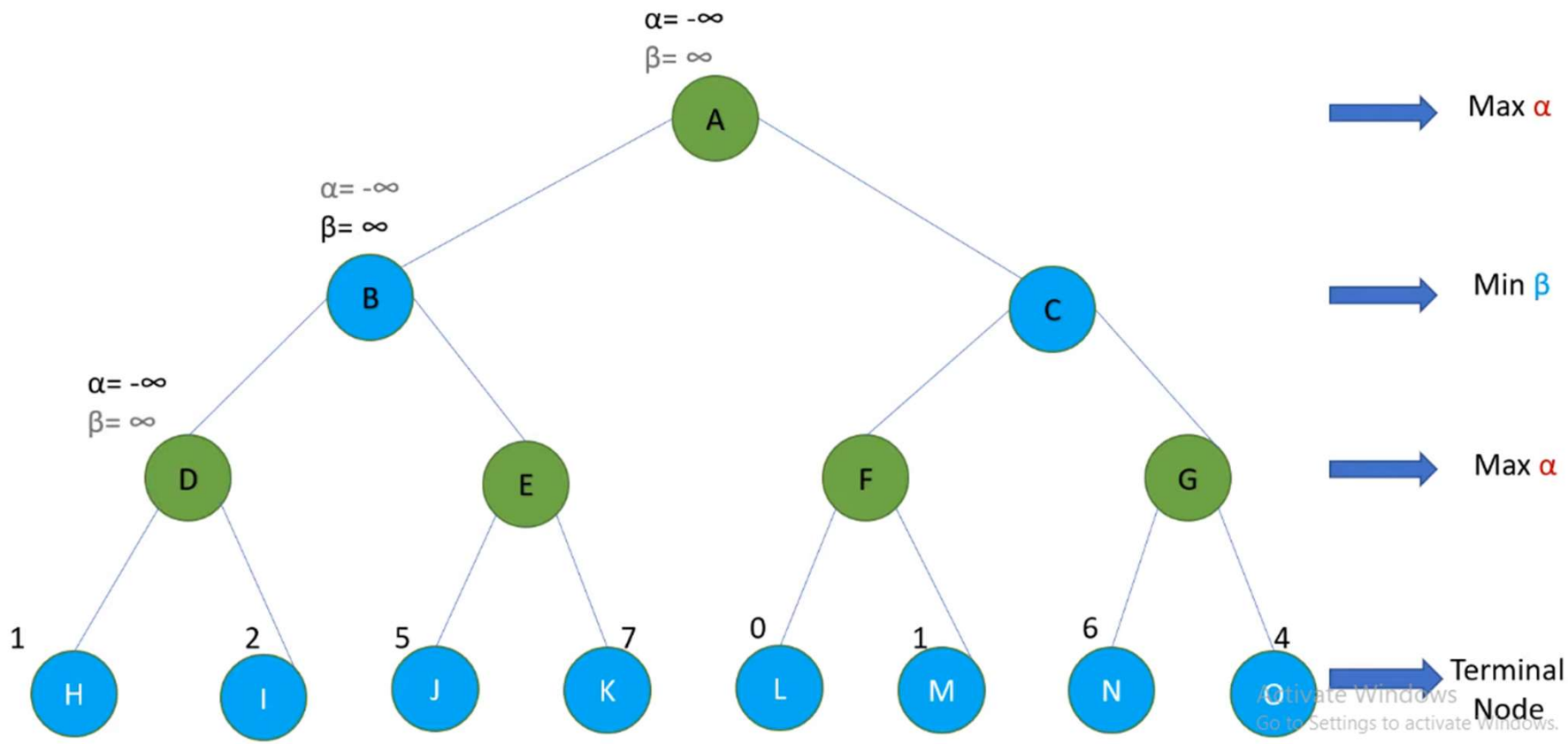

Apply DFS

# Example

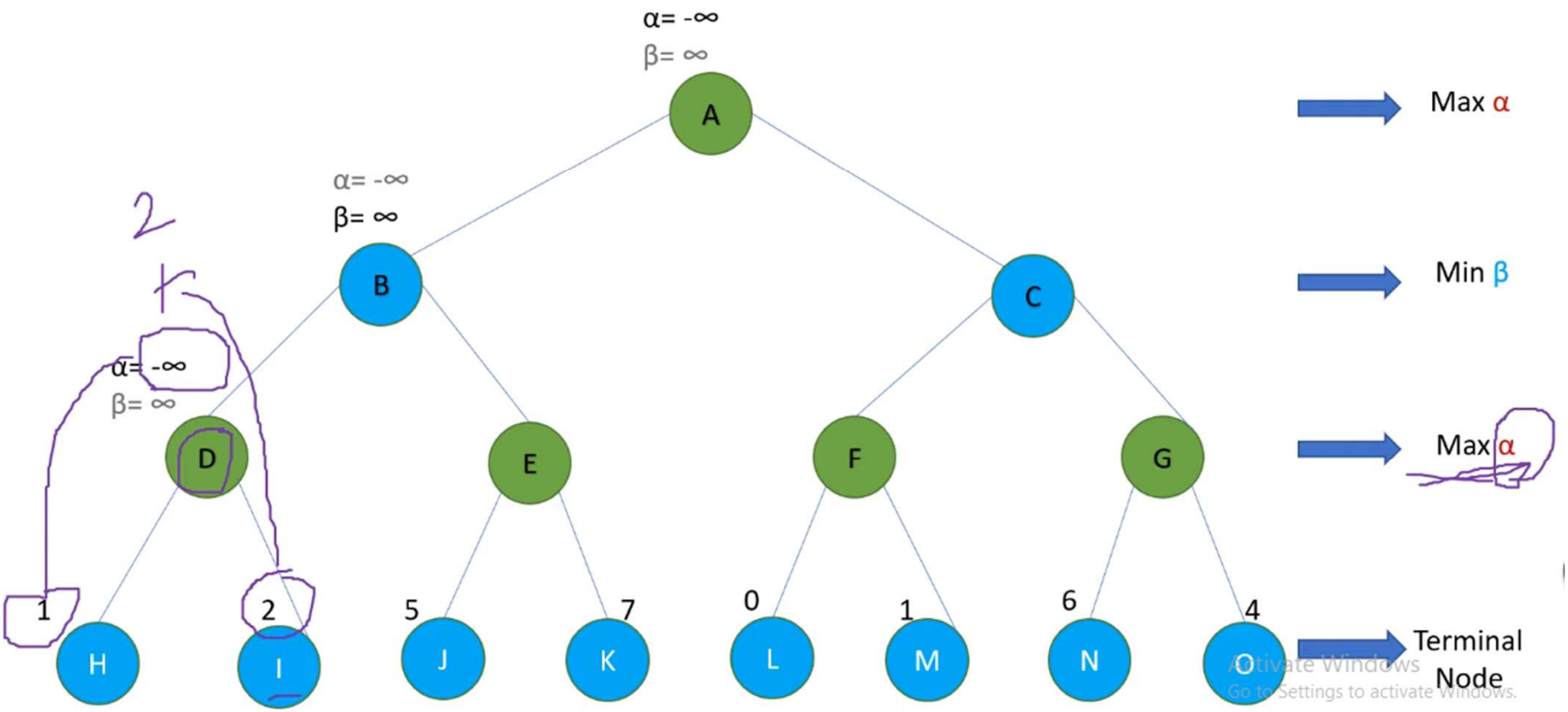



Start from A

# Example



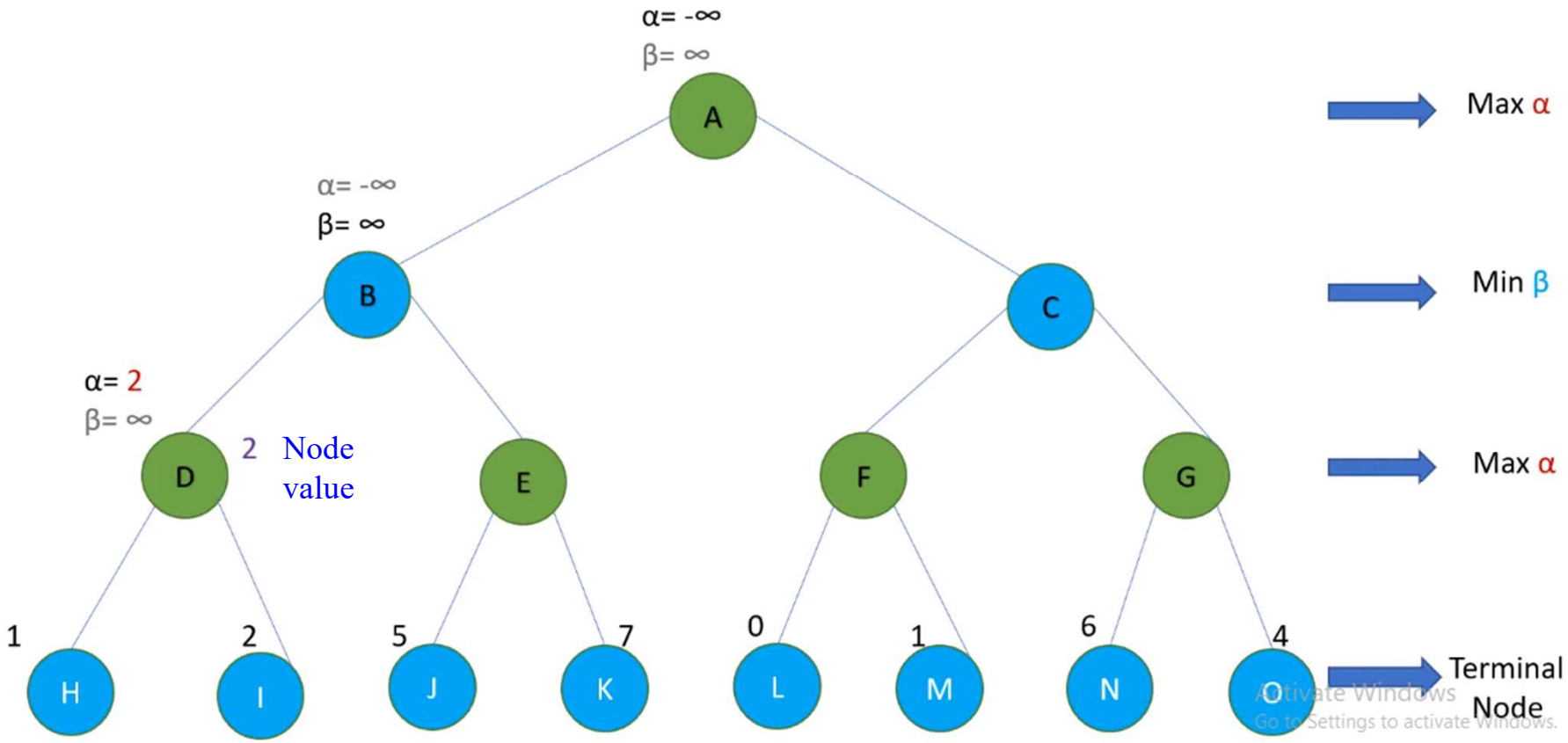
# Example



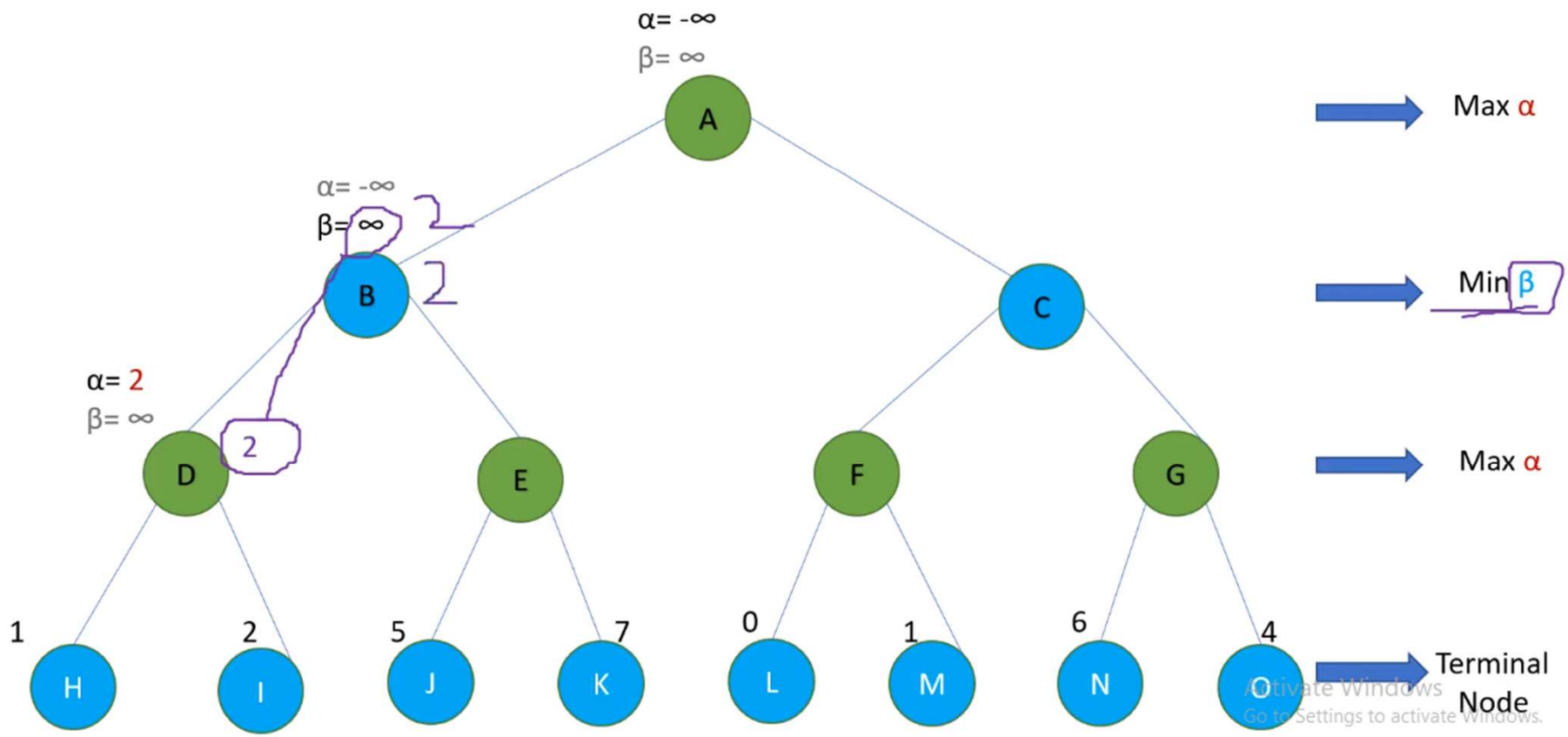

Max(α = -∞, 1) => α = 1

Max(α = 1, 2 ) => α = 2

# Example

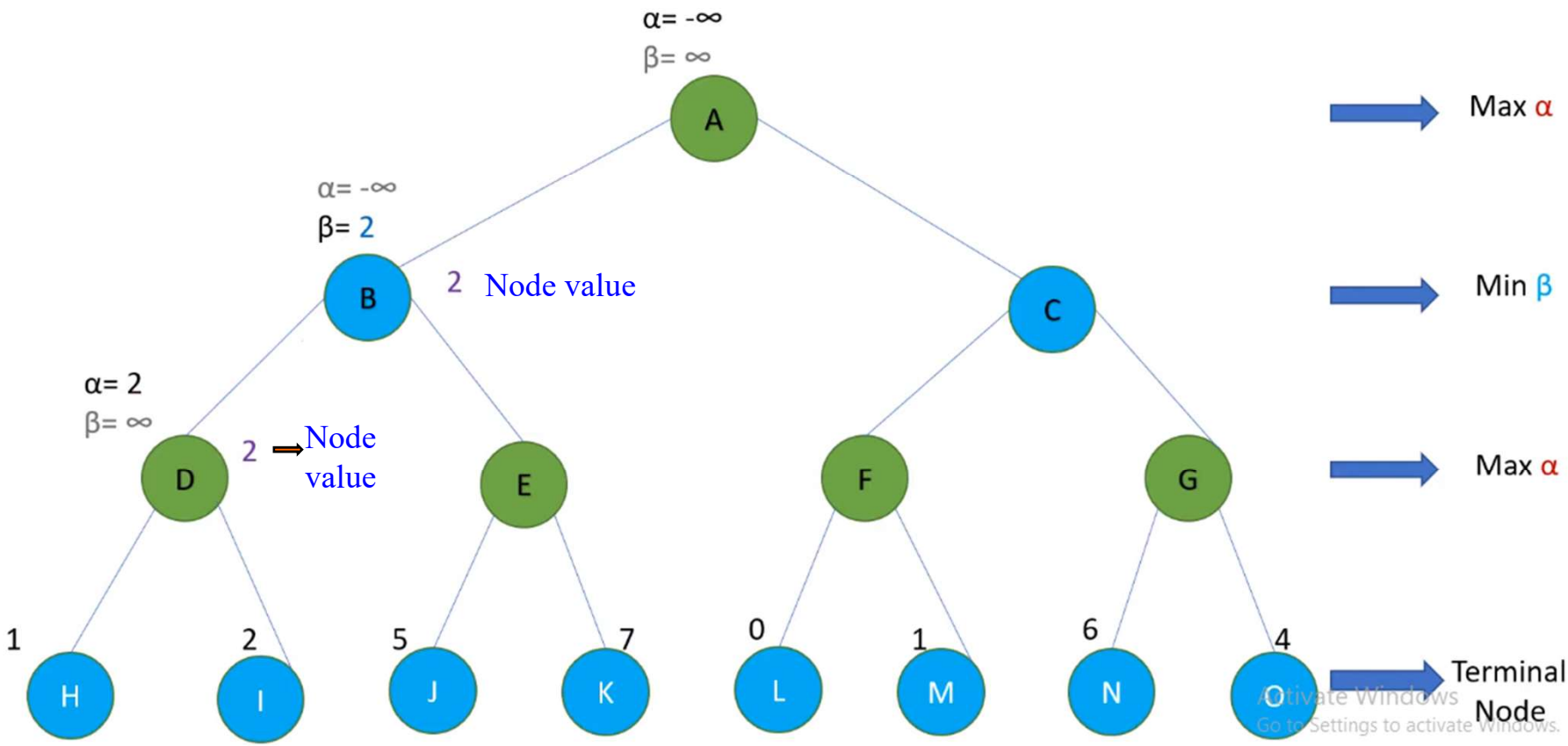


# Example

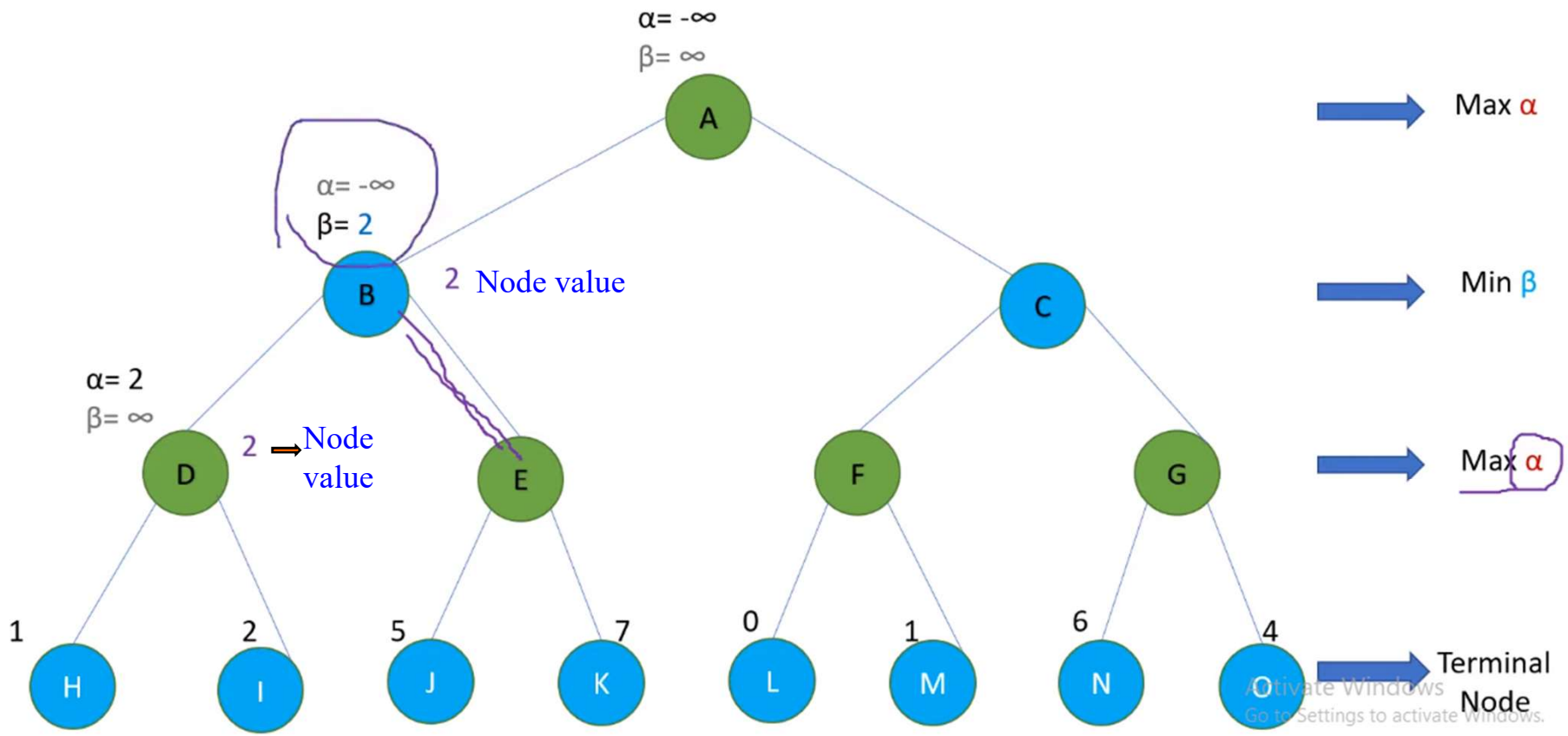



$\text{Min}(\beta = \infty, 2) \Rightarrow \beta = 2$

# Example

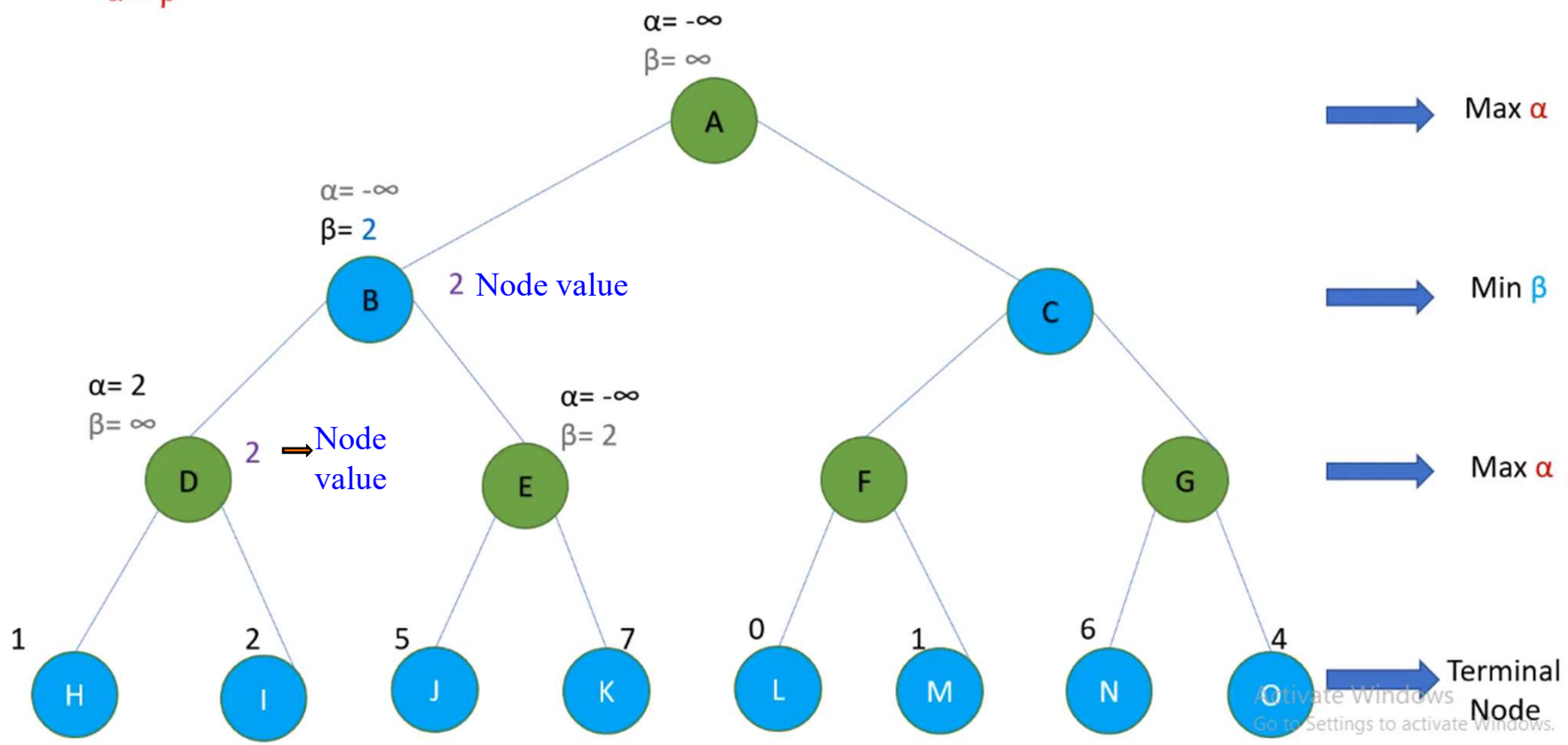


# Example

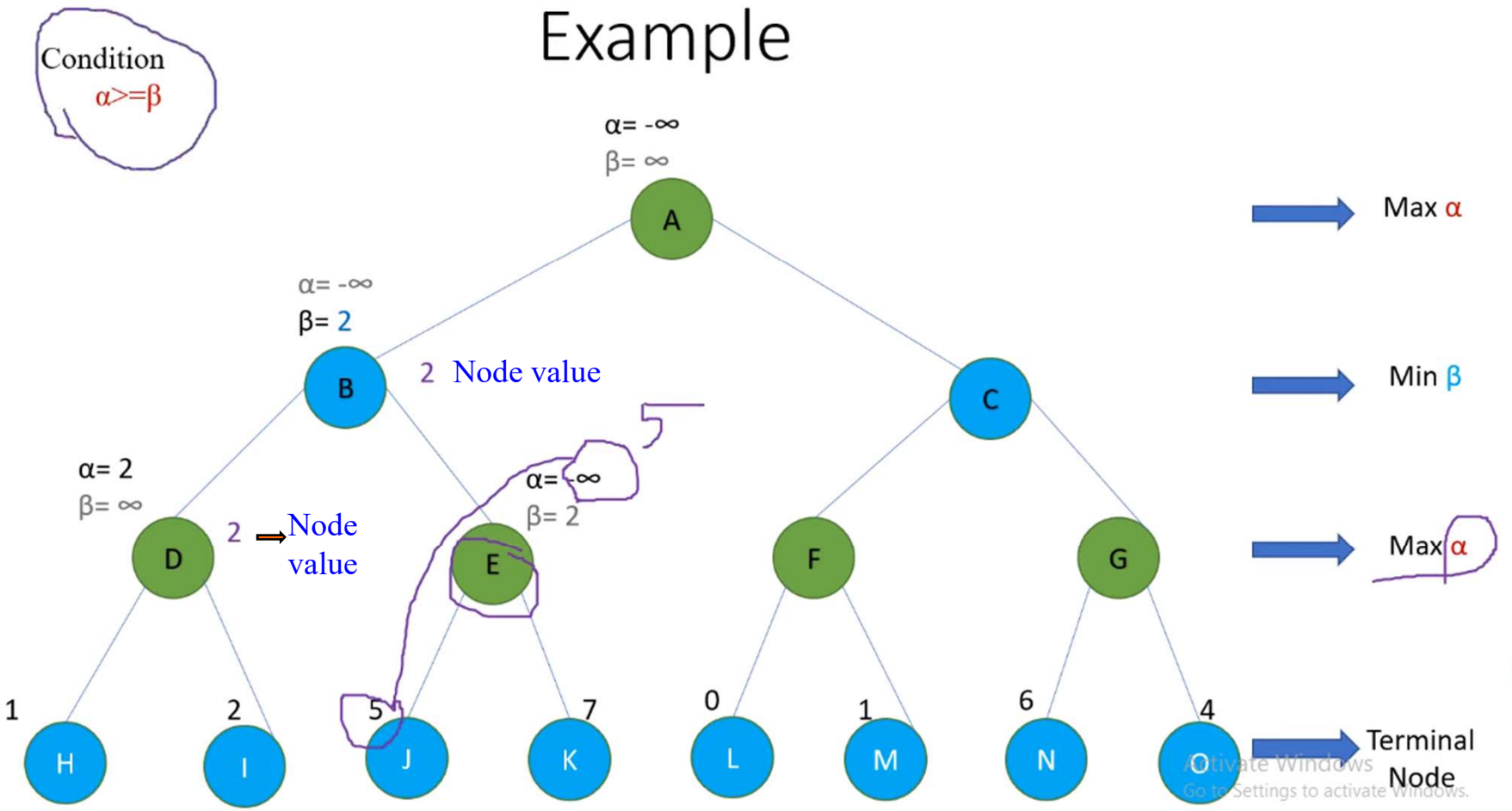



- Bring down (α and β ) of node B to node E

# Example

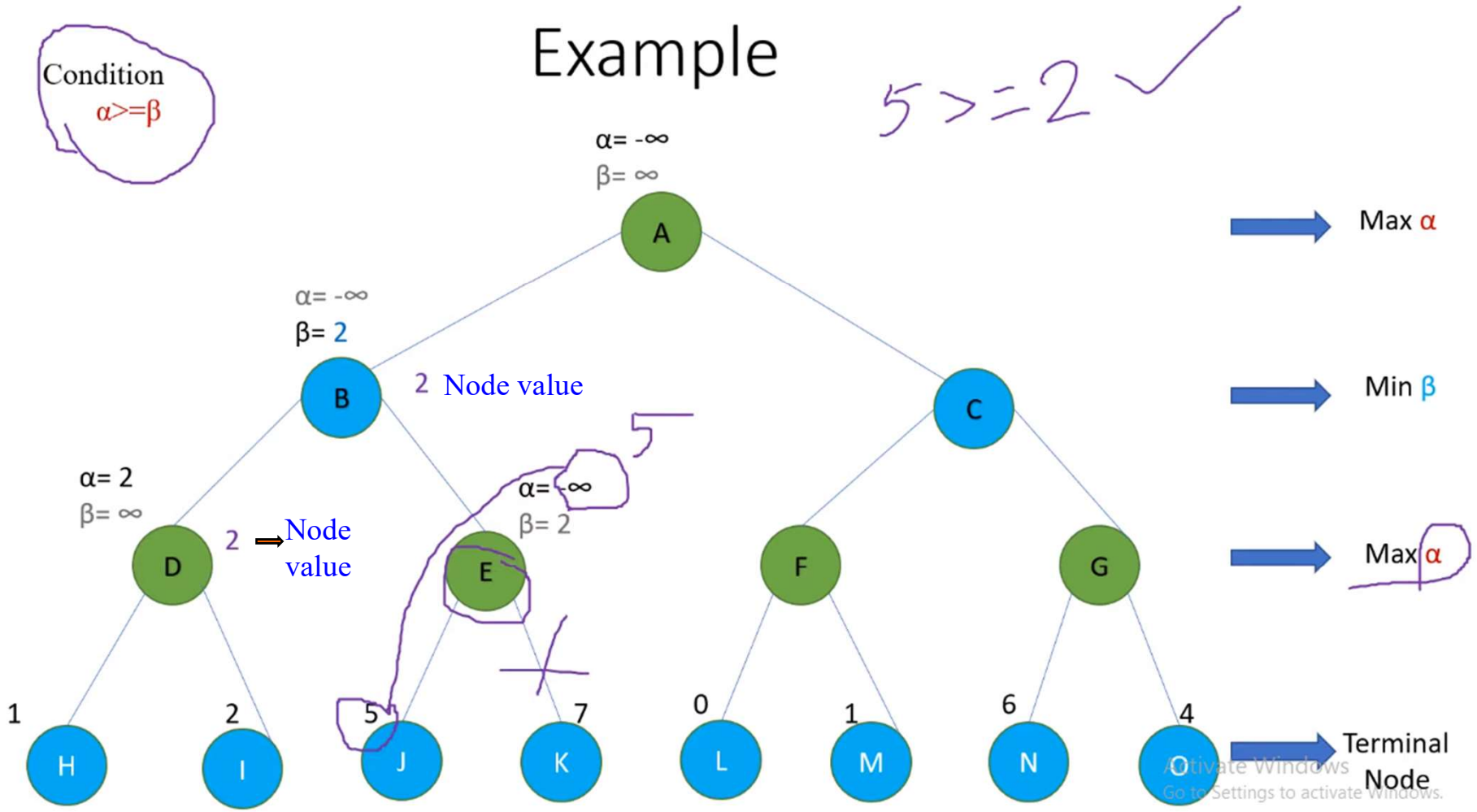



- Bring down (α and β ) of node B to node E
- Find max at the node E




Max(α = -∞, 5) => α = 5

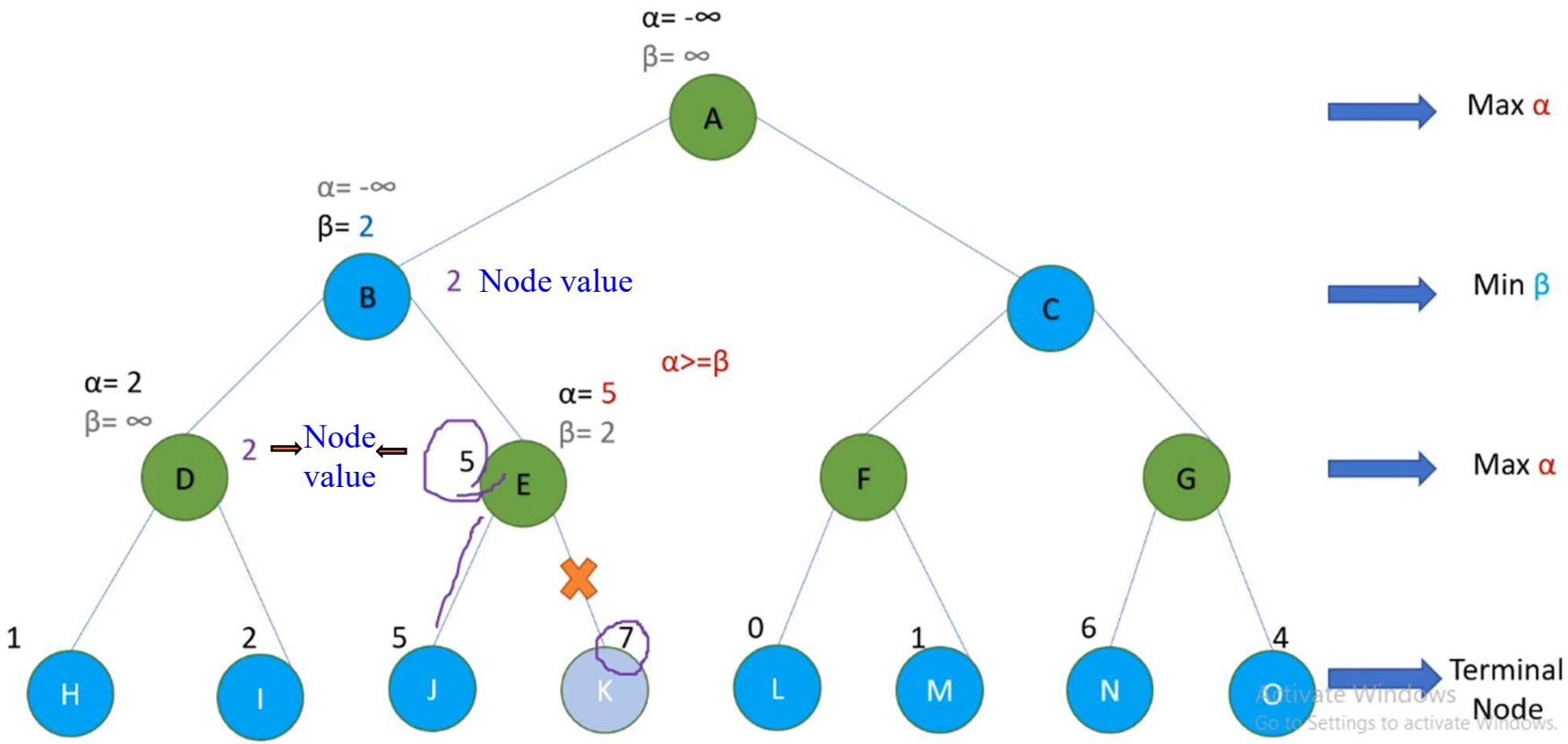
Check condition α > β



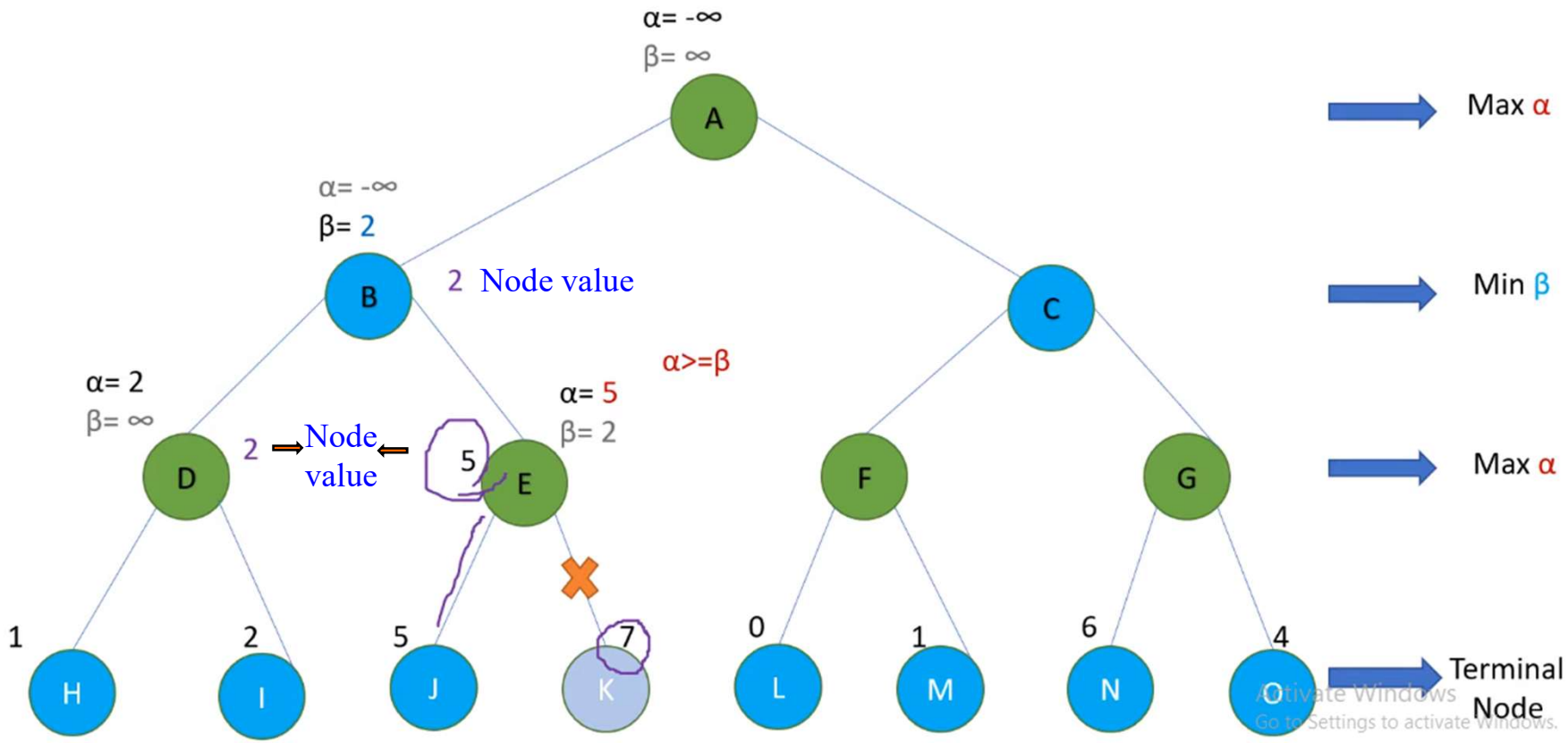

Max(α = -∞, 5) => α = 5

Check condition α > β = 5 >=2 => TRUE. So prune the right side tree

# Example

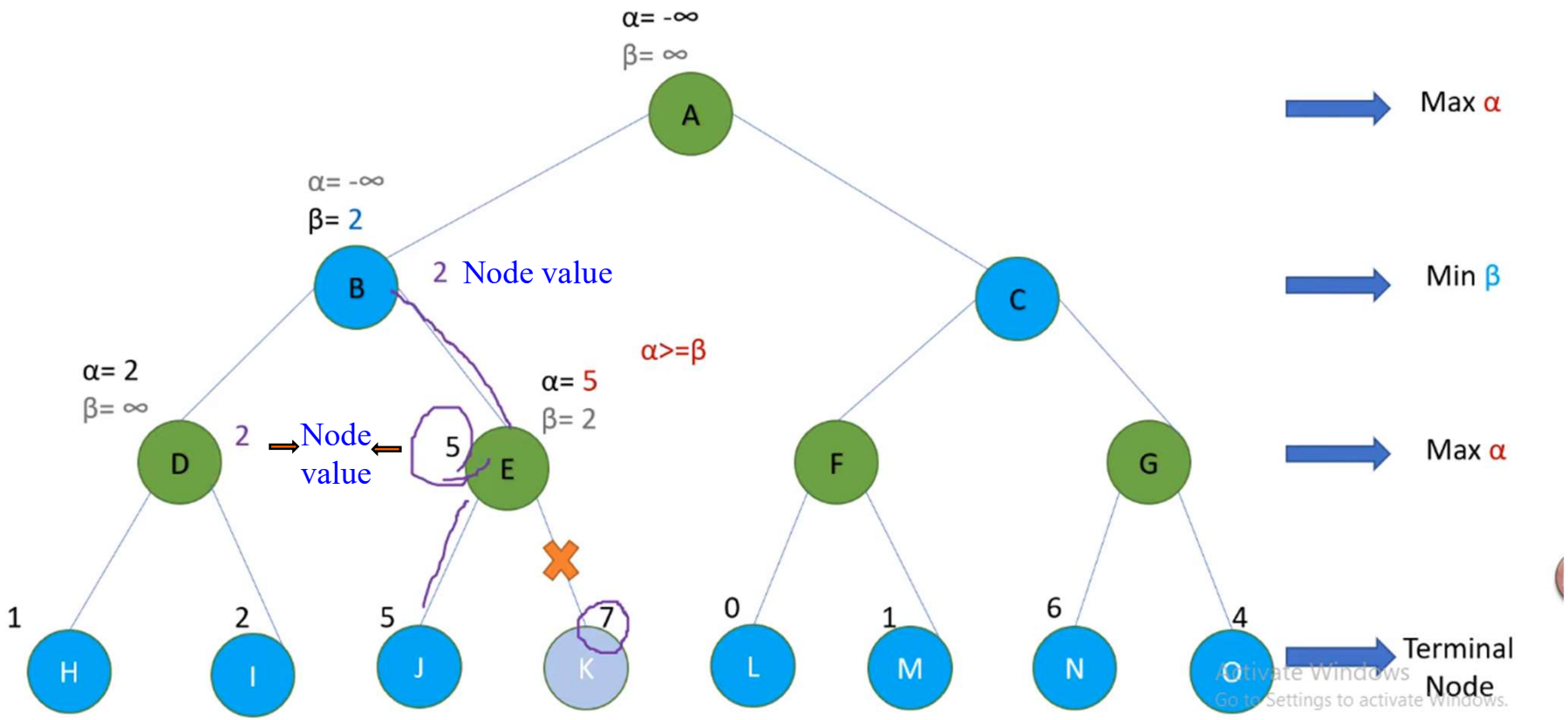

α= -∞
β= ∞
A
Max α
α= -∞
β= 2
B
2 Node value
C
Min β
α>=β
α= 2
β= ∞
α= 5
β= 2
D
2
Node value
5
E
F
G
Max α
1
2
5
7
0
1
6
4
H
I
J
K
L
M
N
O
Terminal Node

# Example

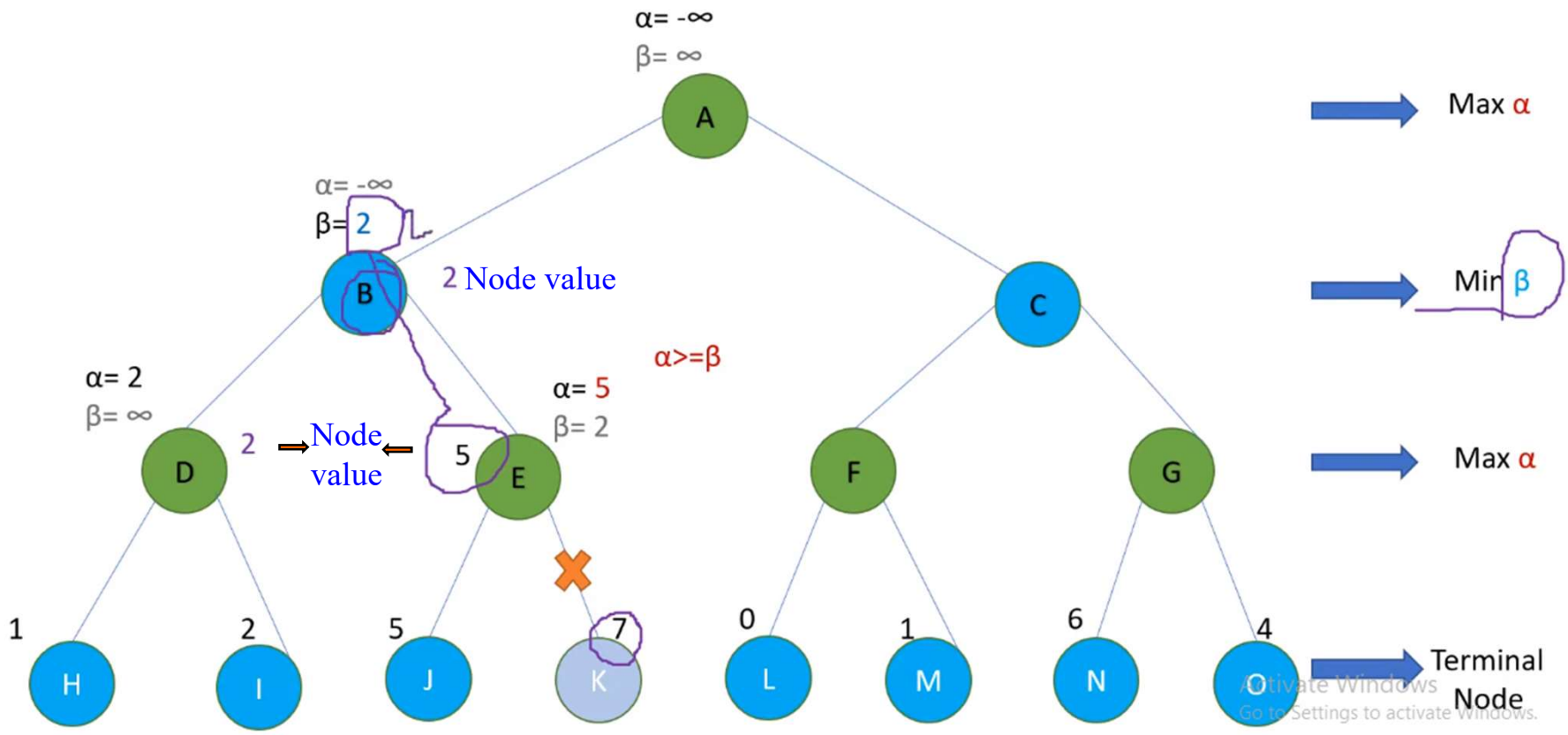



Next Step : Backtrack to B from E

# Example

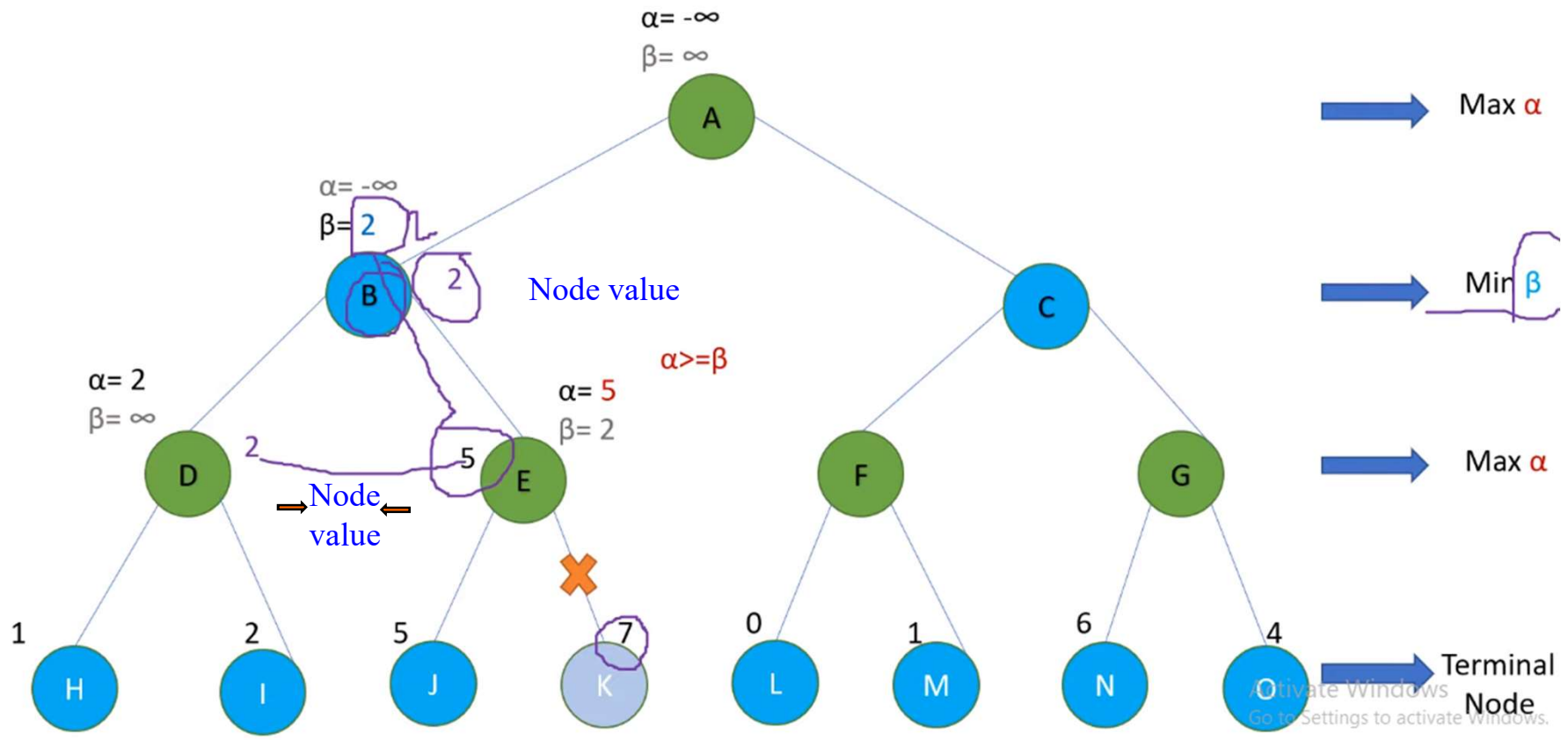

α= -∞
β= ∞
A
α= -∞
β= 2
B
2 Node value
C
α>=β
α= 2
β= ∞
D
2
Node value
5
α= 5
β= 2
E
F
G
1
H
2
I
5
J
7
K
0
L
1
M
6
N
4
O
Max α
Min β
Max α
Terminal Node

# Example




Update β by Min (β=2, 5) = 2  i.e. No change

# Example

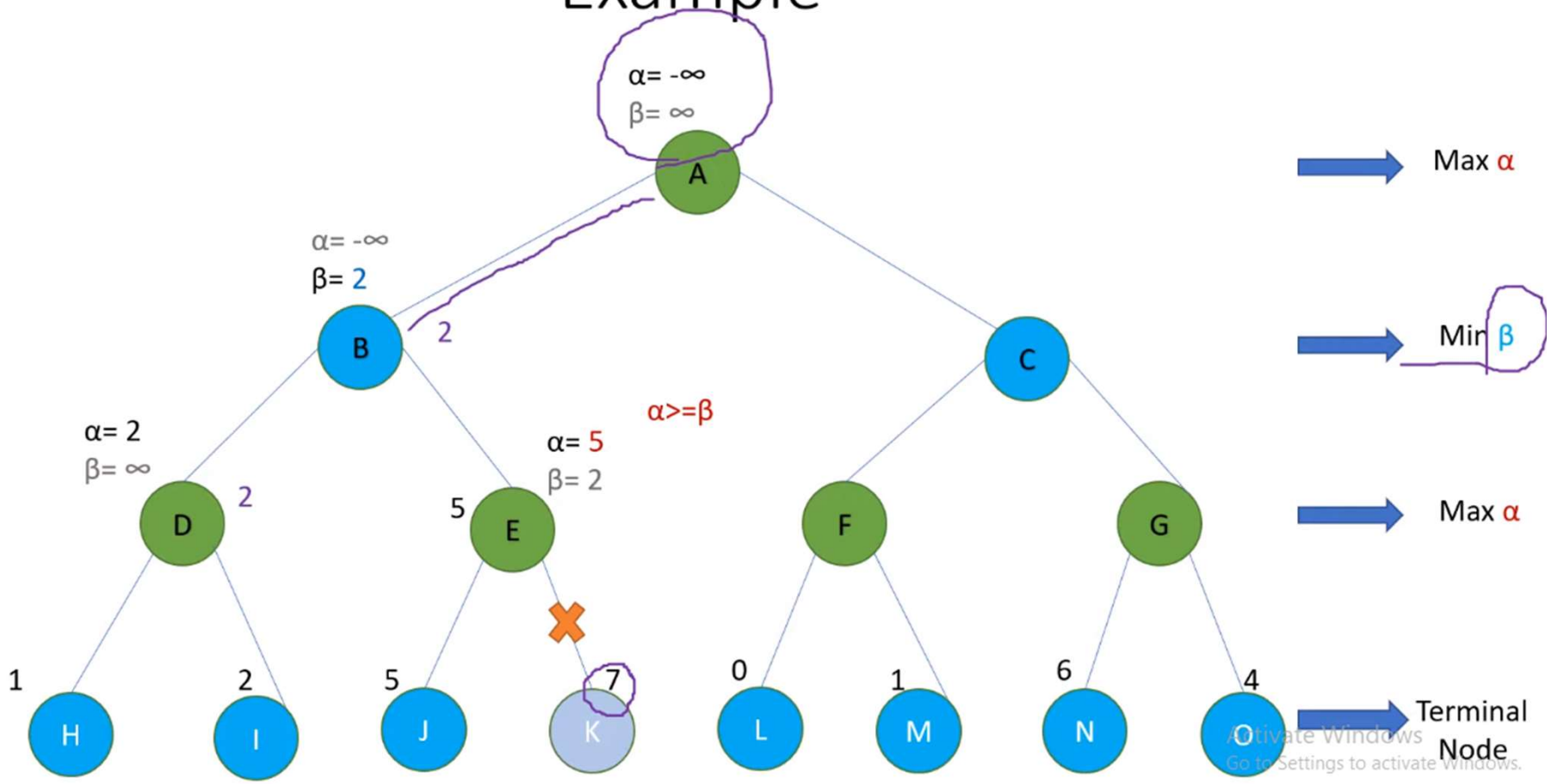



Update node value by Min (2, 5) = 2  i.e. No change

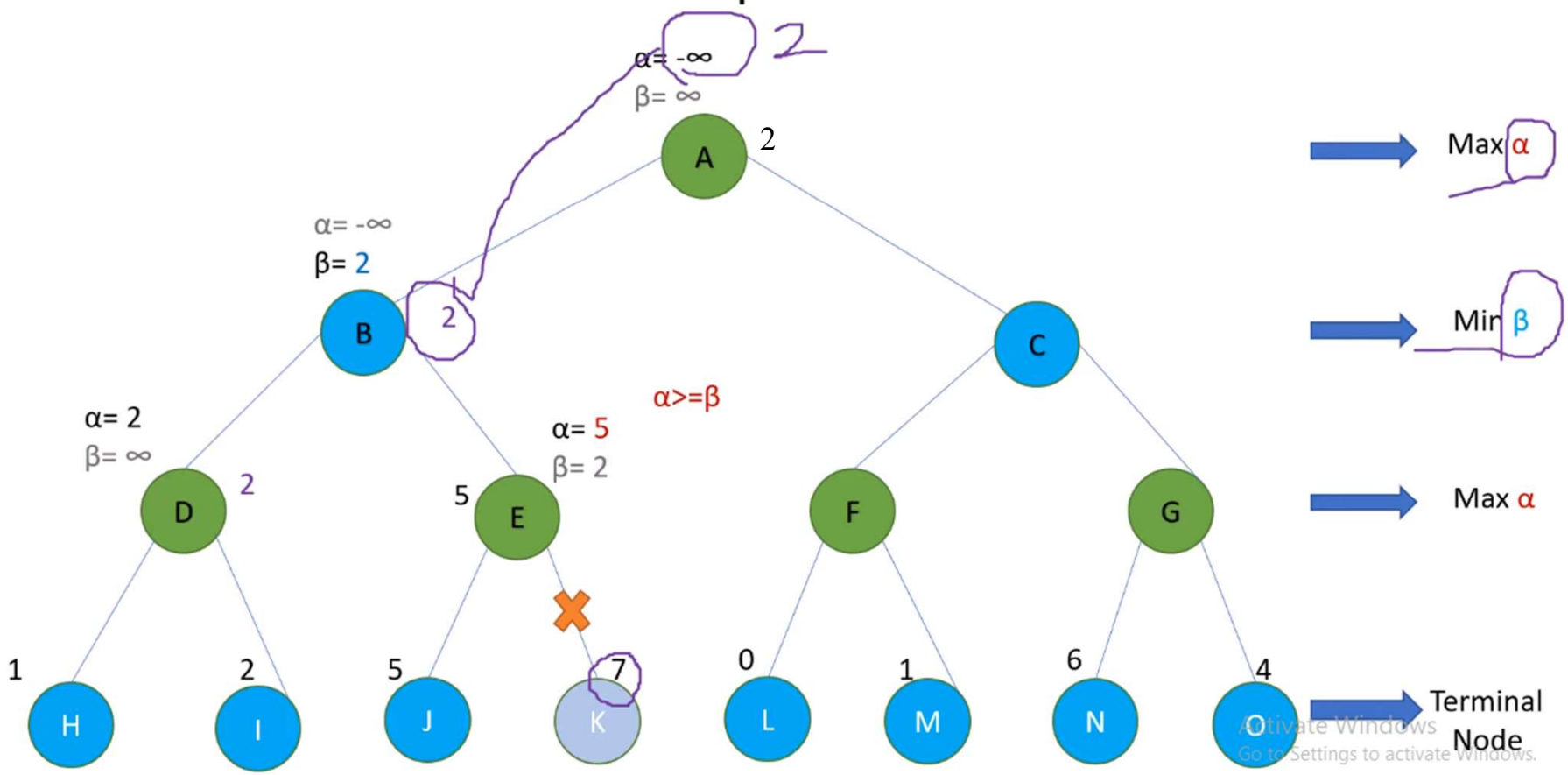
[ Comparing the values of D and E ]

Next Step : Backtrack to A from B

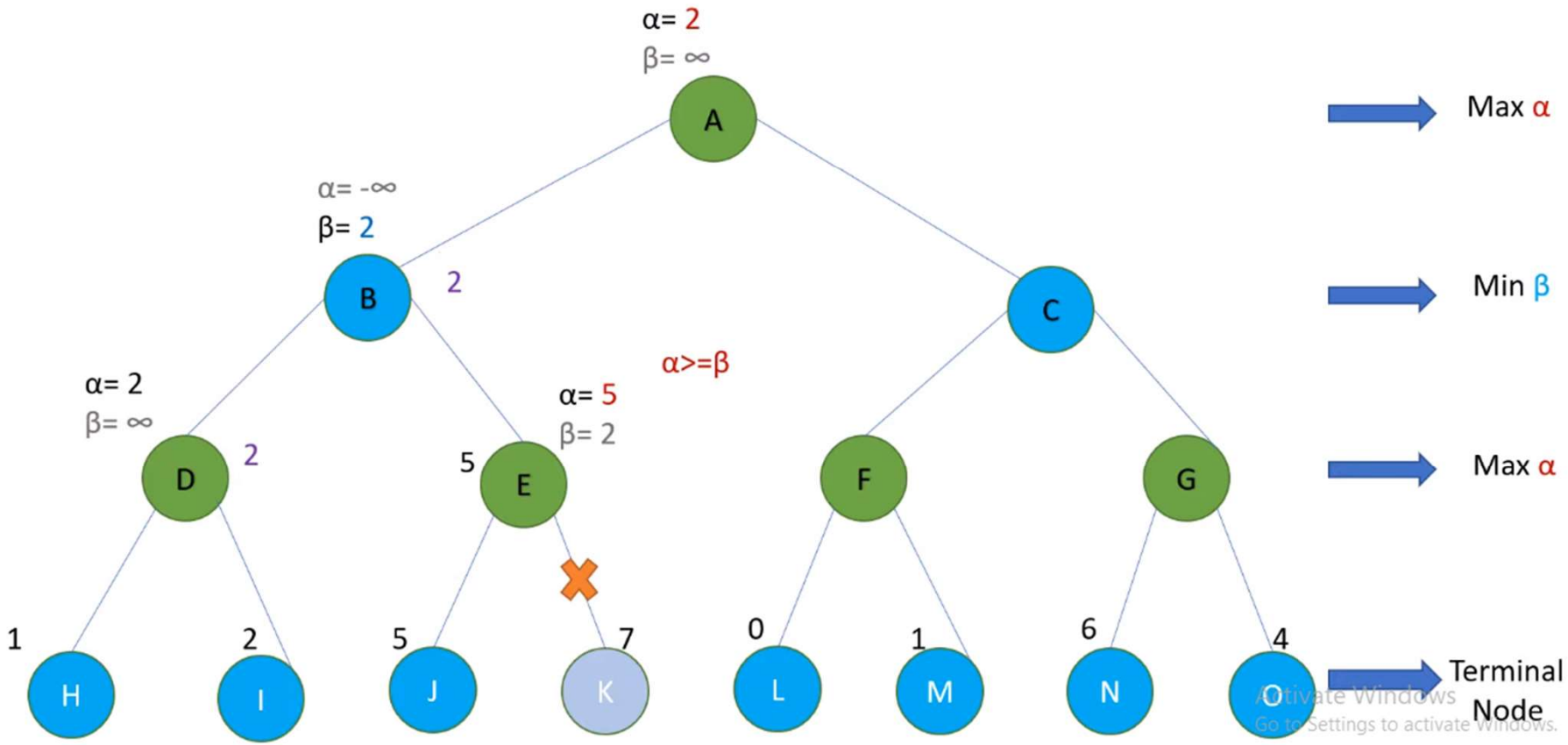
# Example

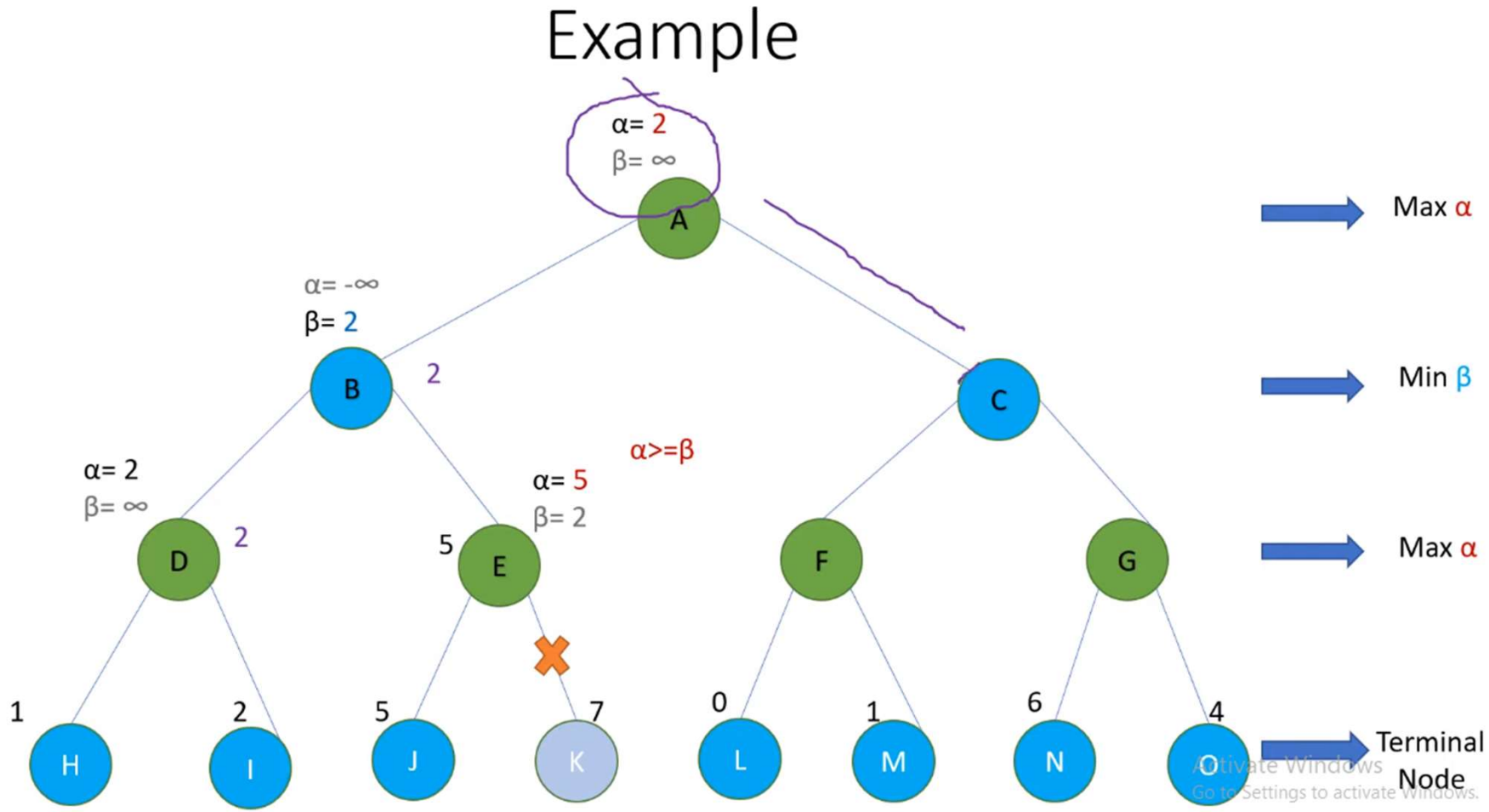


# Example



# Example

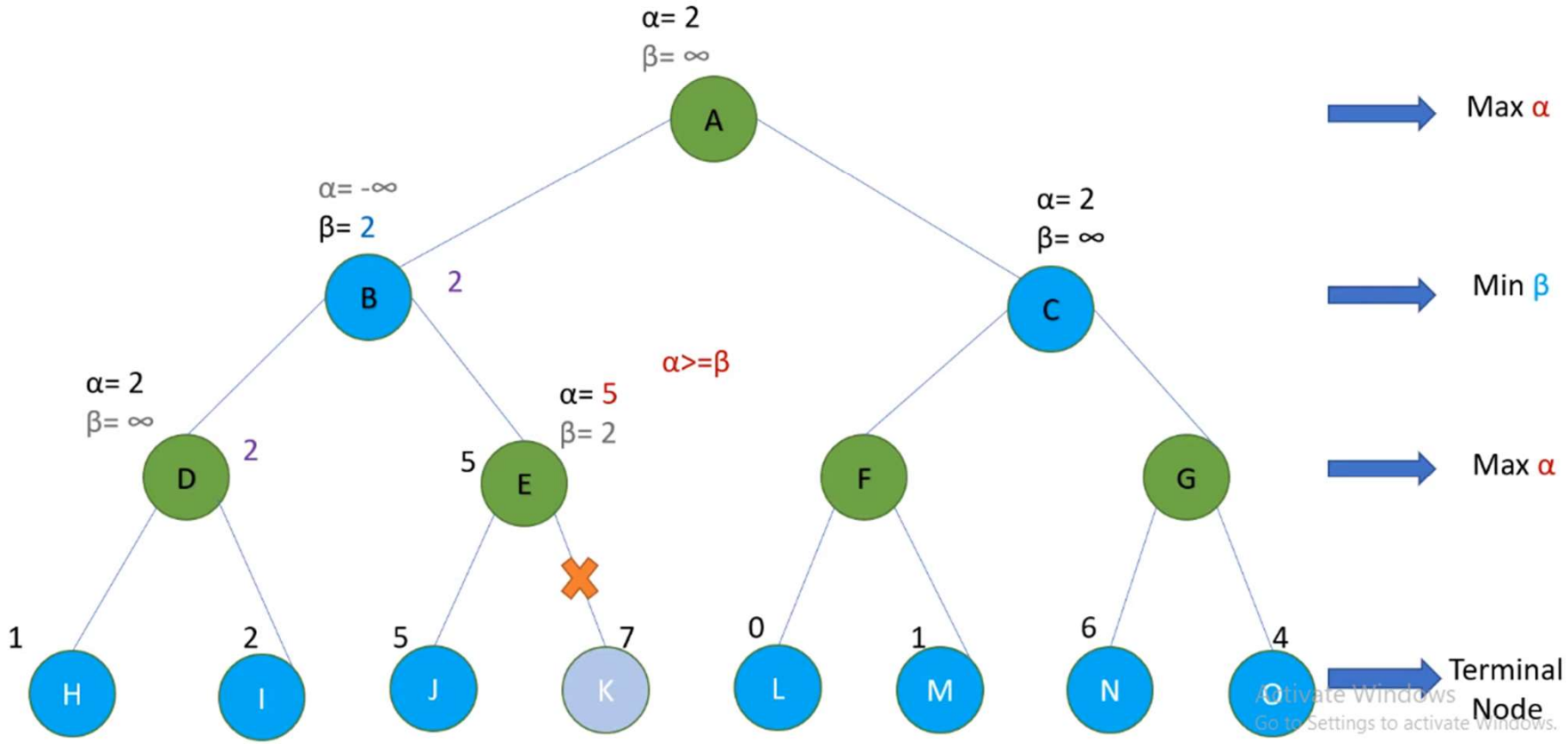

α= 2
β= ∞
A
Max α
α= -∞
β= 2
B
2
C
Min β
α>=β
α= 2
β= ∞
D
2
α= 5
β= 2
5
E
F
G
Max α
1
H
2
I
5
J
7
K
0
L
1
M
6
N
4
O
Terminal Node
Activate Windows
Go to Settings to activate Windows.



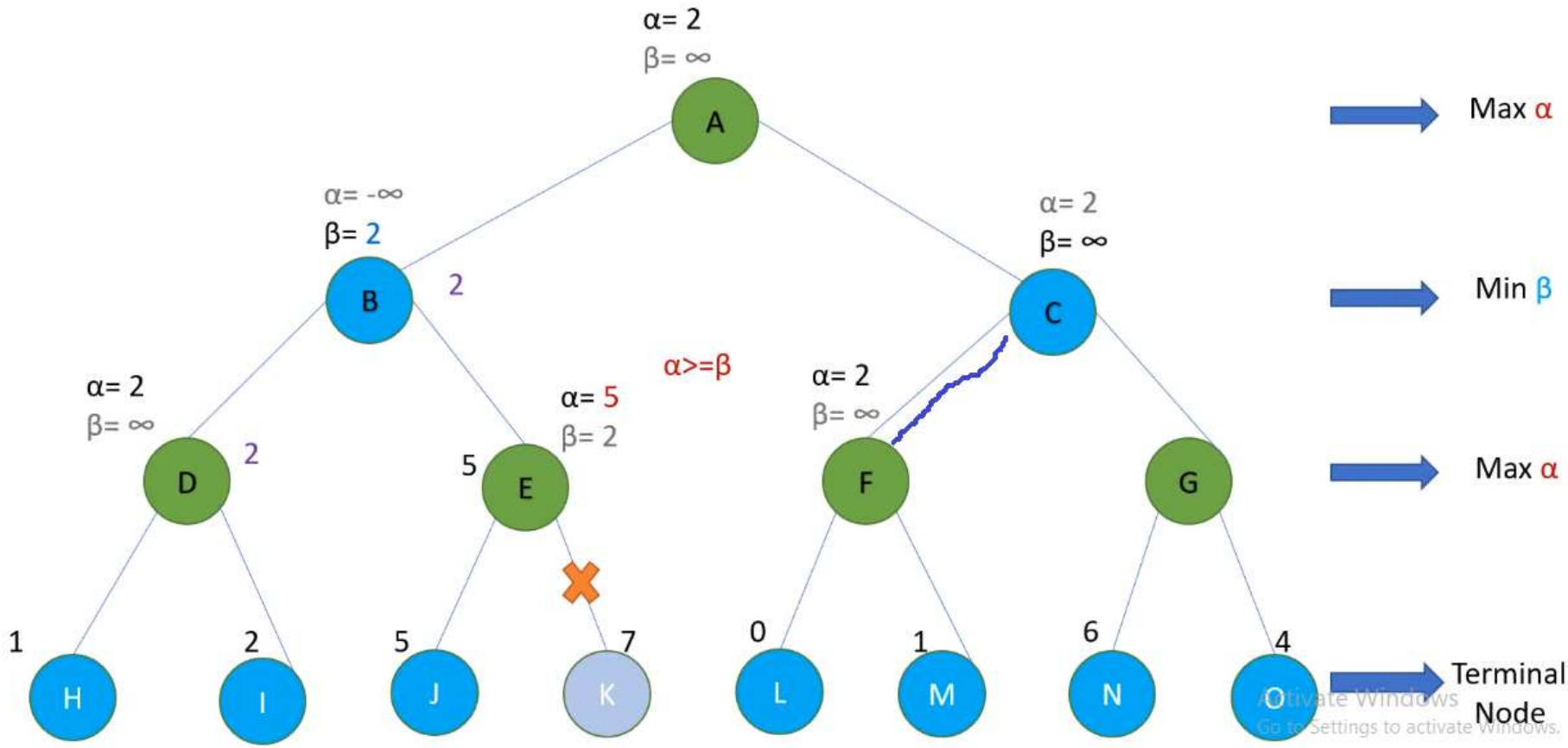

Next Step : A To C

- Bring down (α and β ) of node A to node C

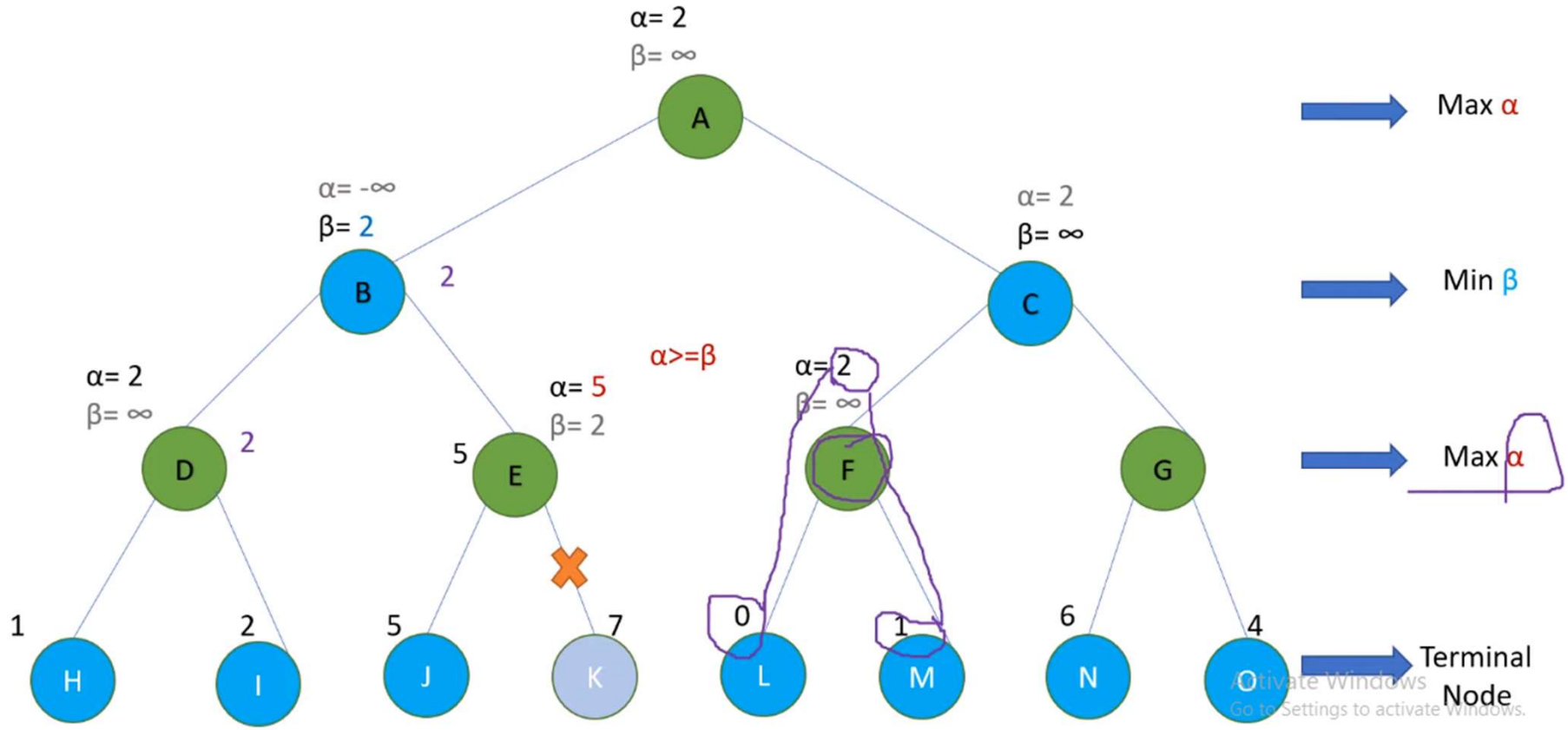
# Example


α= 2
β= ∞
A
α= -∞
β= 2
B
2
C
α= 2
β= ∞
α>=β
α= 2
β= ∞
D
2
α= 5
β= 2
5
E
F
G
1
H
2
I
5
J
7
K
0
L
1
M
6
N
4
O
Max α
Min β
Max α
Terminal Node
Activate Windows
Go to Settings to activate Windows.

# Example


α= 2
β= ∞
A
α= -∞
β= 2
B
2
C
α= 2
β= ∞
α>=β
α= 2
β= ∞
D
2
α= 5
β= 2
5
E
α= 2
β= ∞
F
G
1
H
2
I
5
J
7
K
0
L
1
M
6
N
4
O
Max α
Min β
Max α
Terminal Node
Activate Windows
Go to Settings to activate Windows.

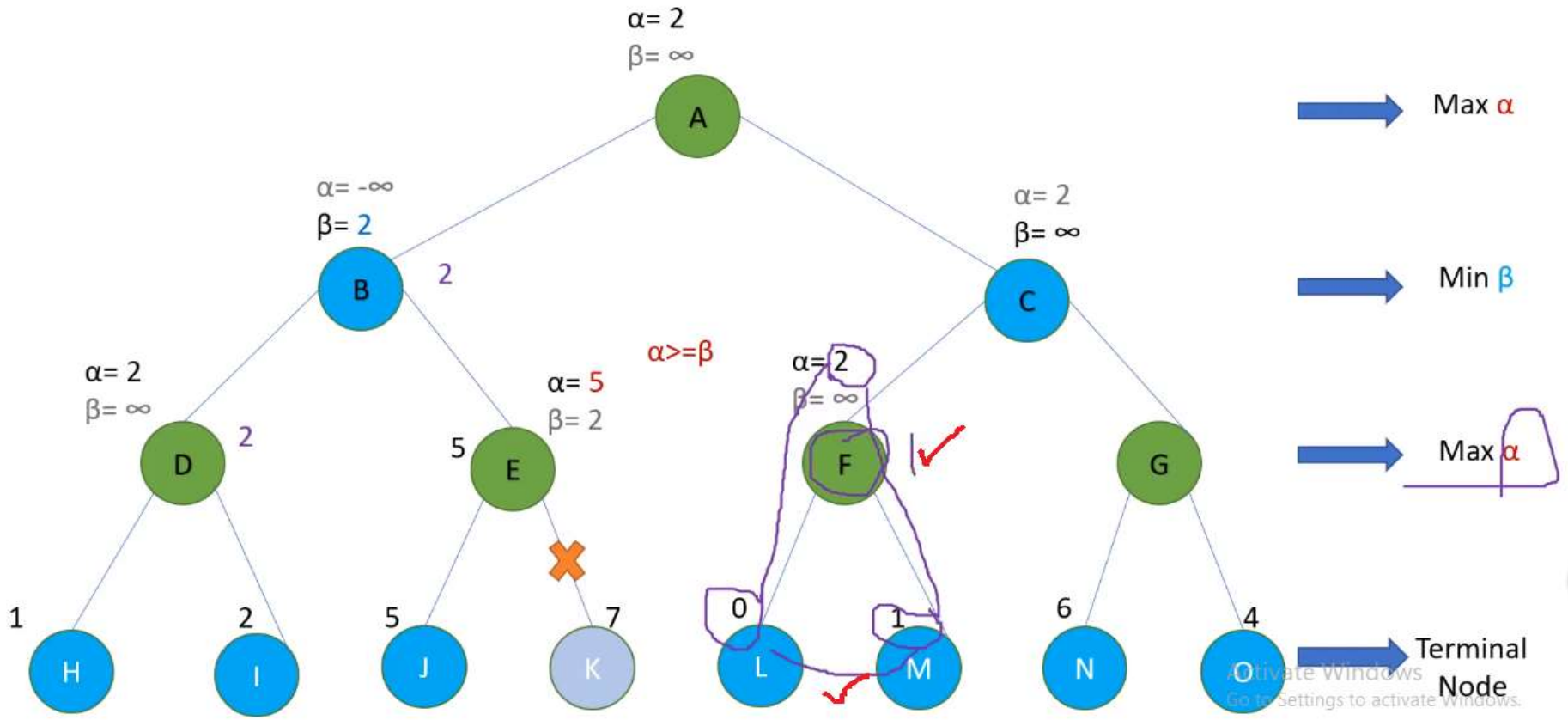
# Example




Max(2,0) = 2, So no replacement

Max(2,1) = 2, So no replacement

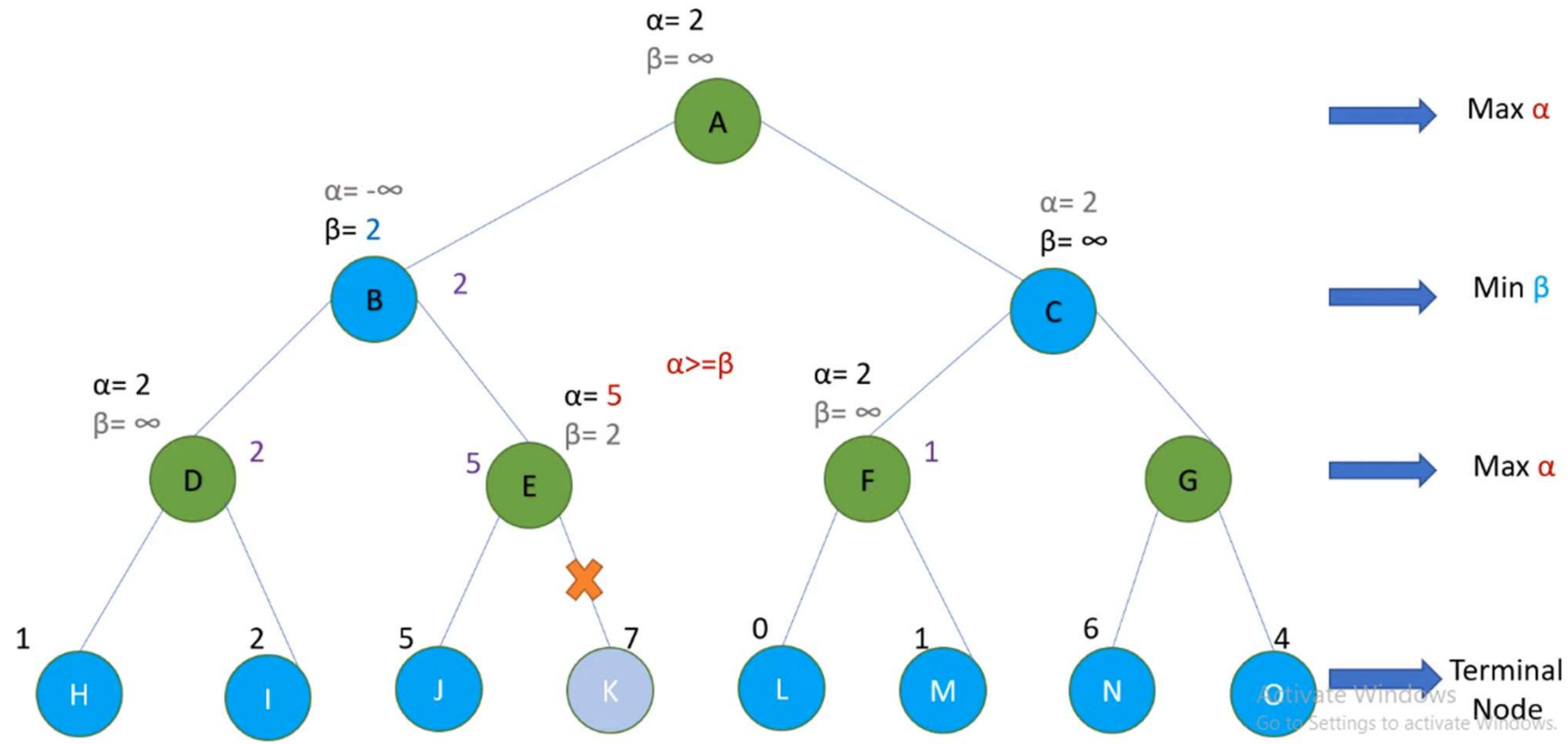
# Example




Node Value : Compare L and M values : Max (0, 1) = 1

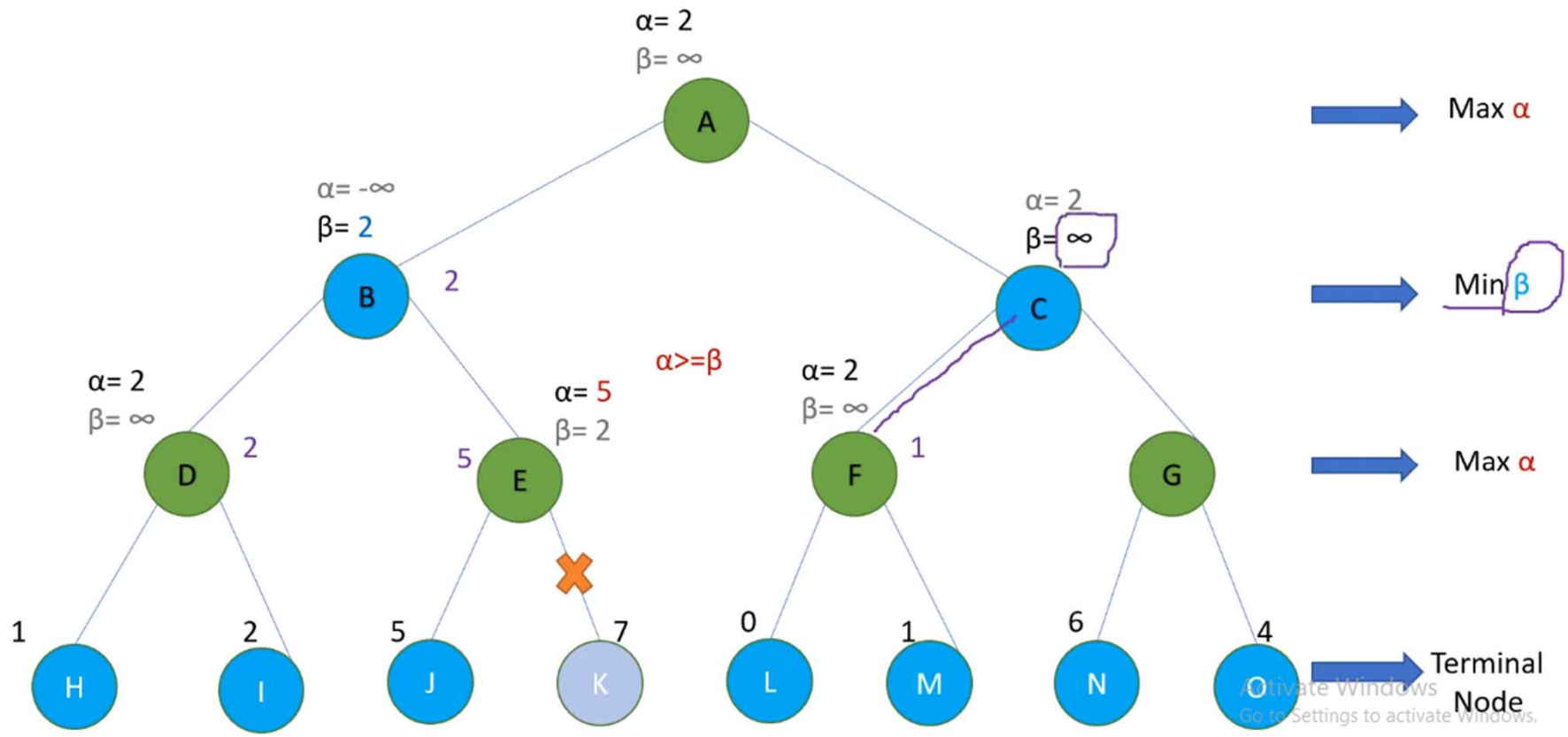
Hence Node value = 1

# Example



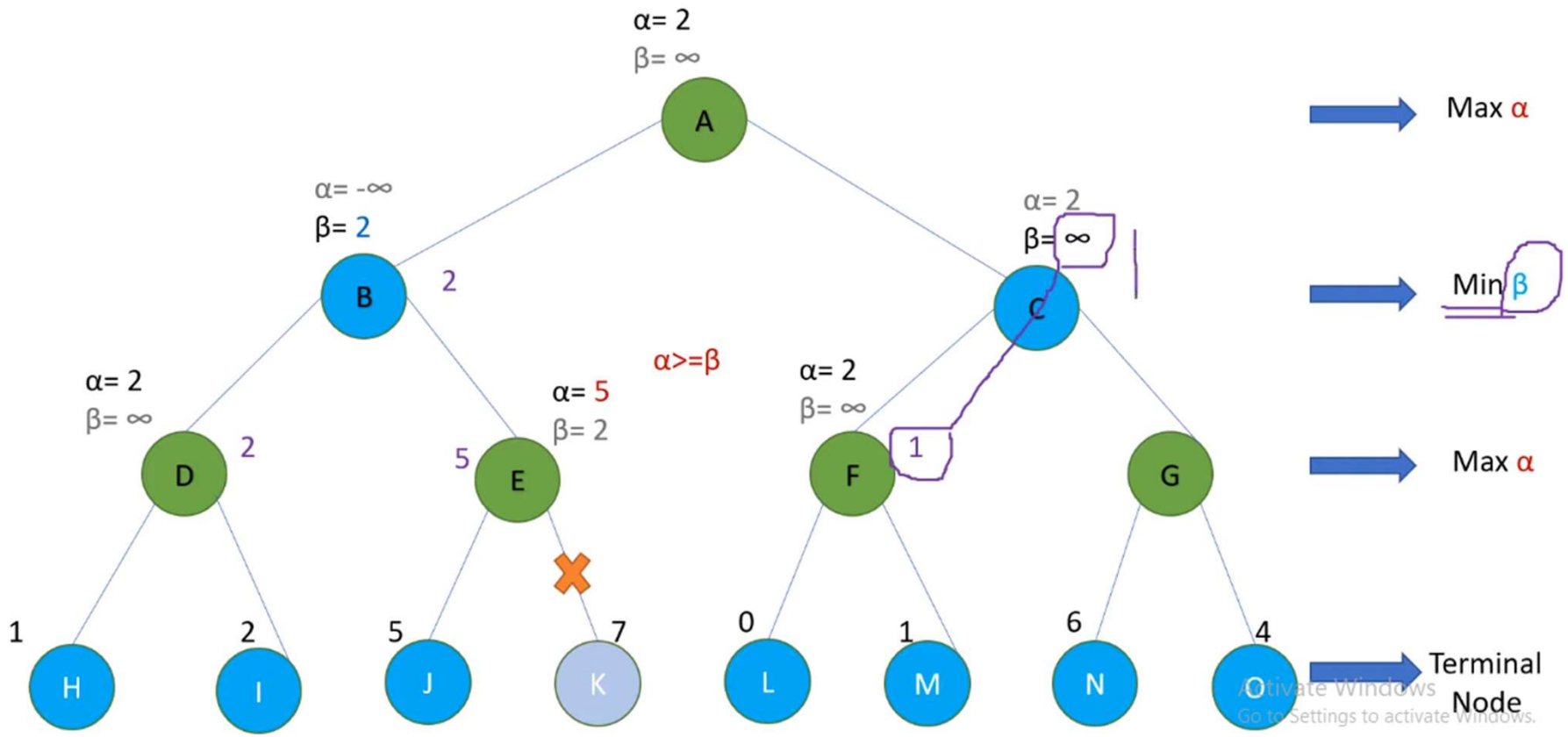

Updated

# Example



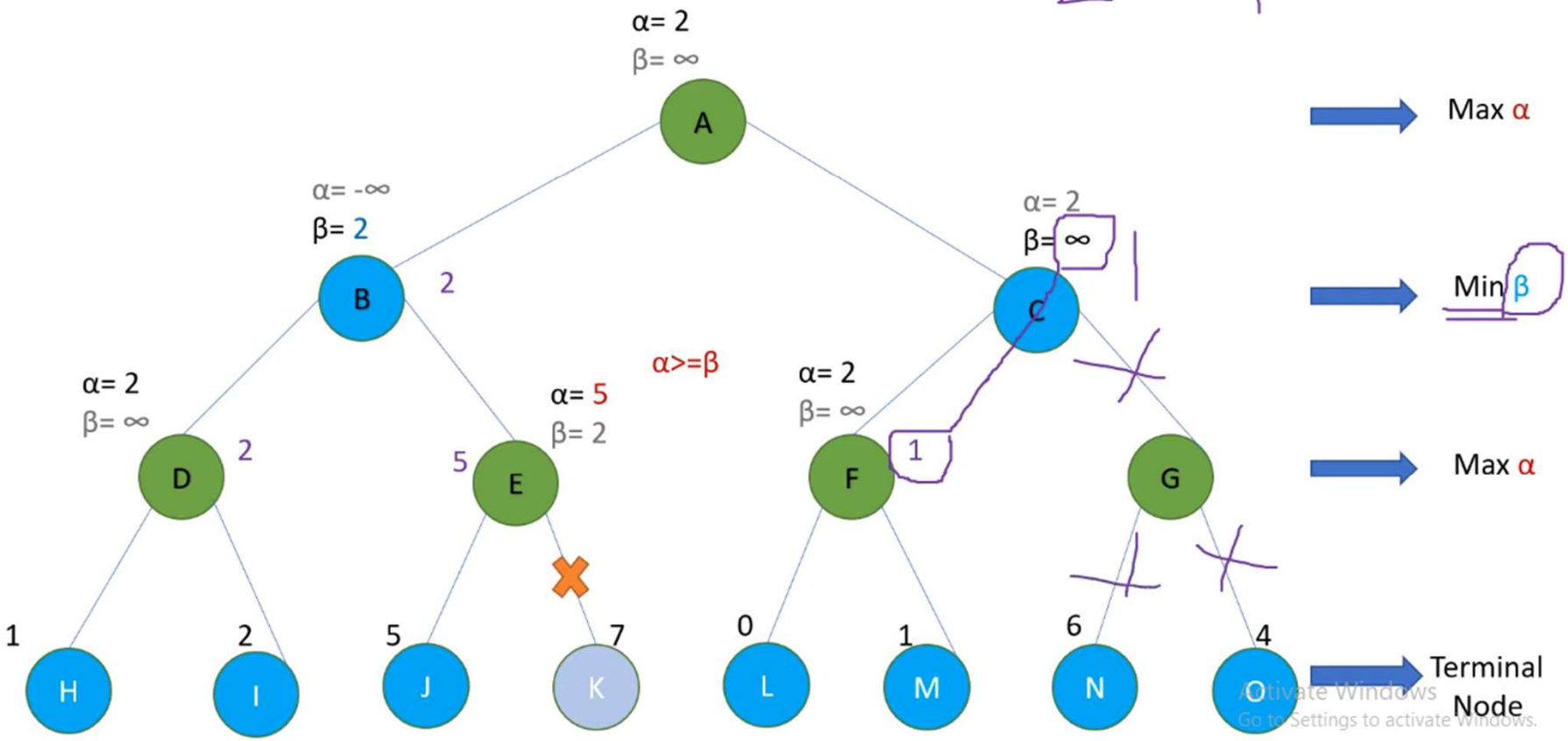

- Backtrack to C from F
- Update β

# Example

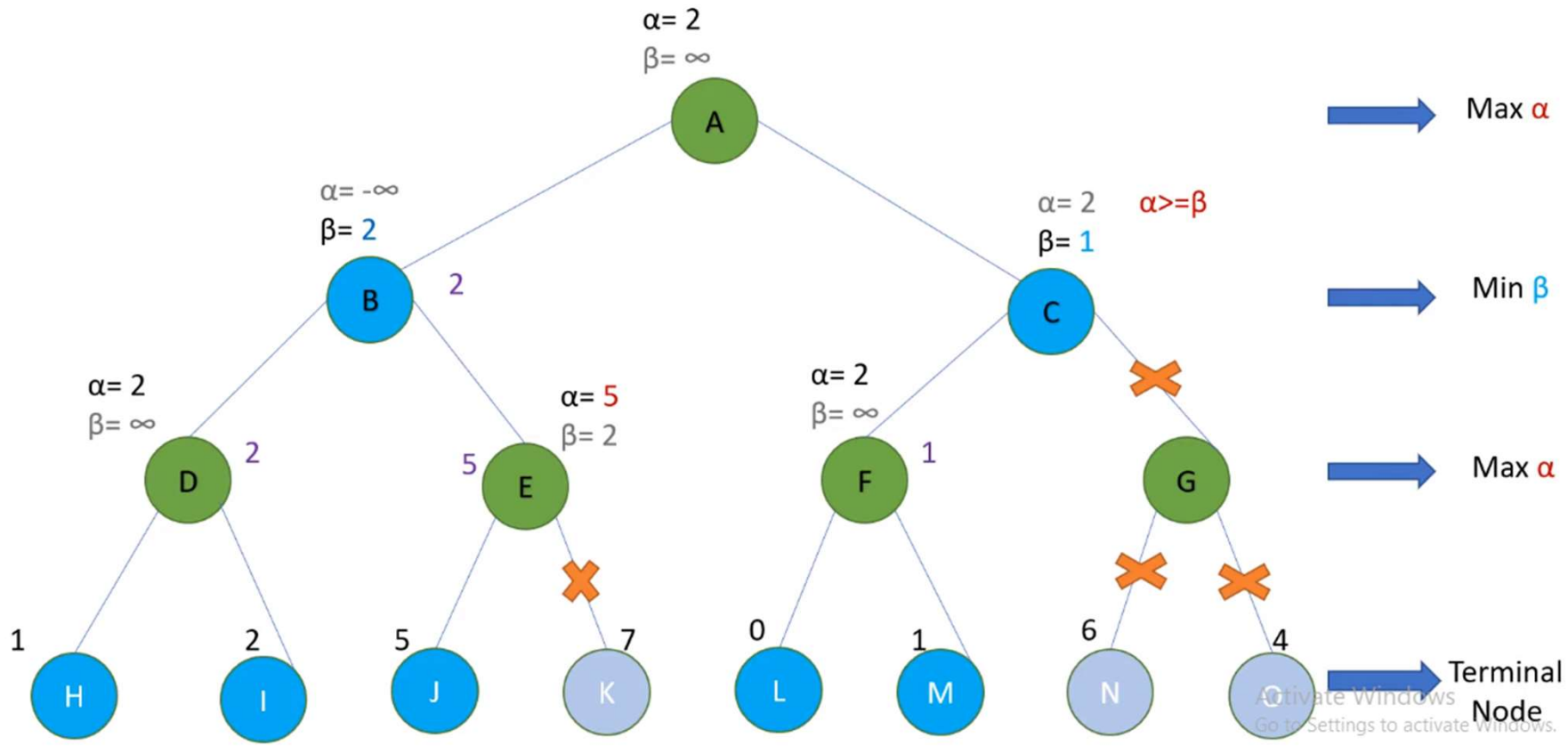



- Min(β=∞, 1) = 1 So replace β = 1
- Check Constraint α >= β

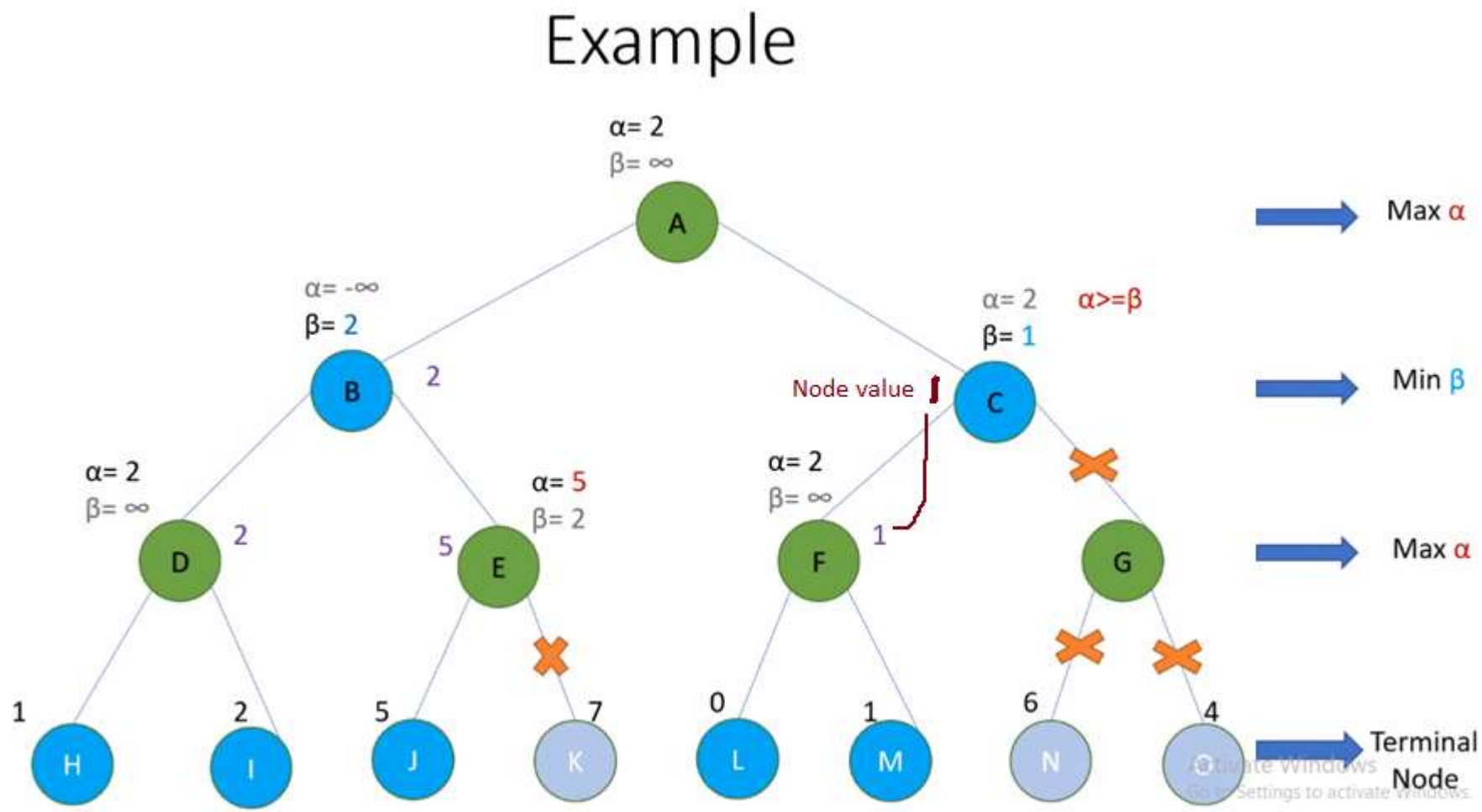
# Example



# Example

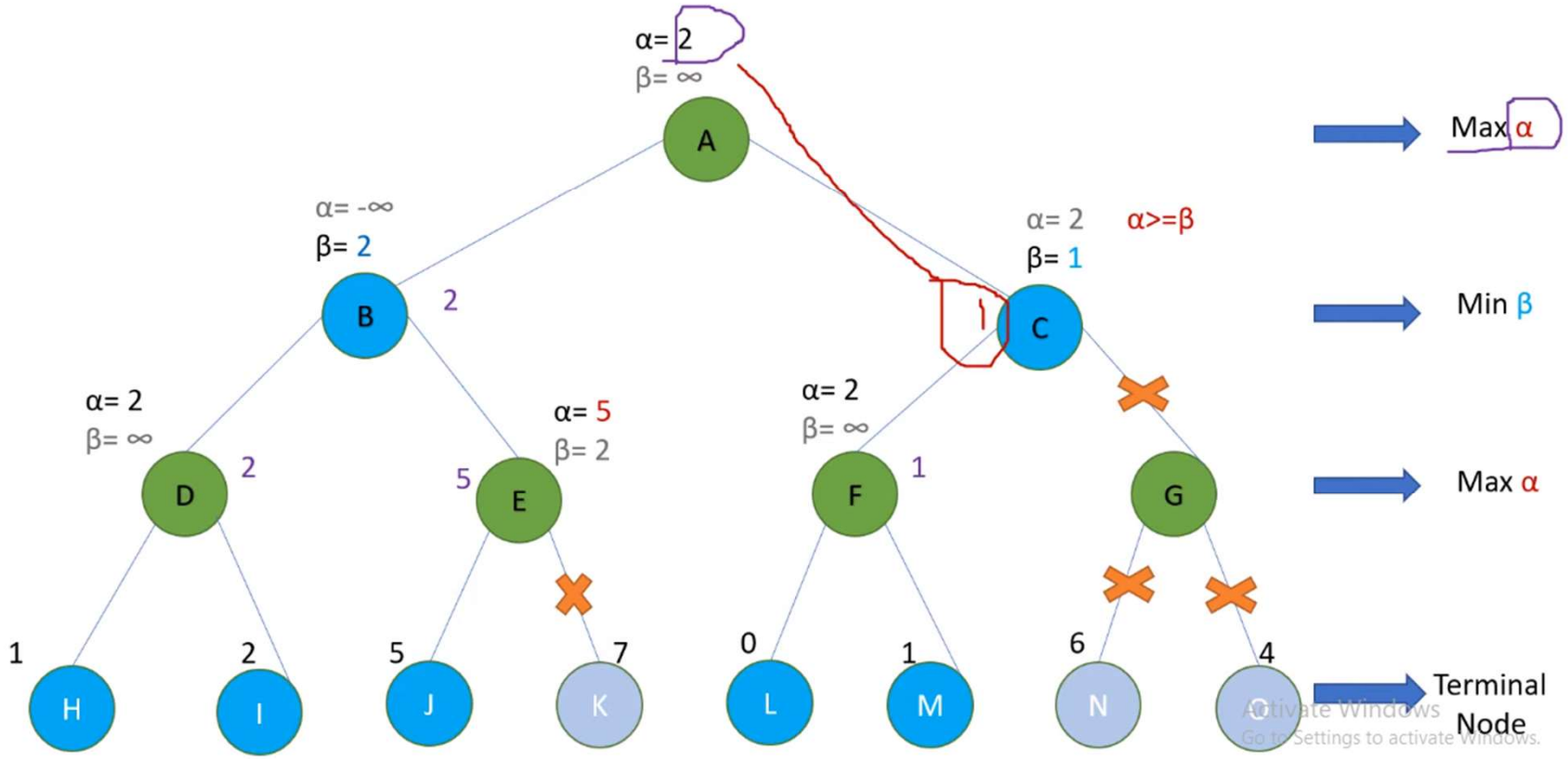



- So pruning



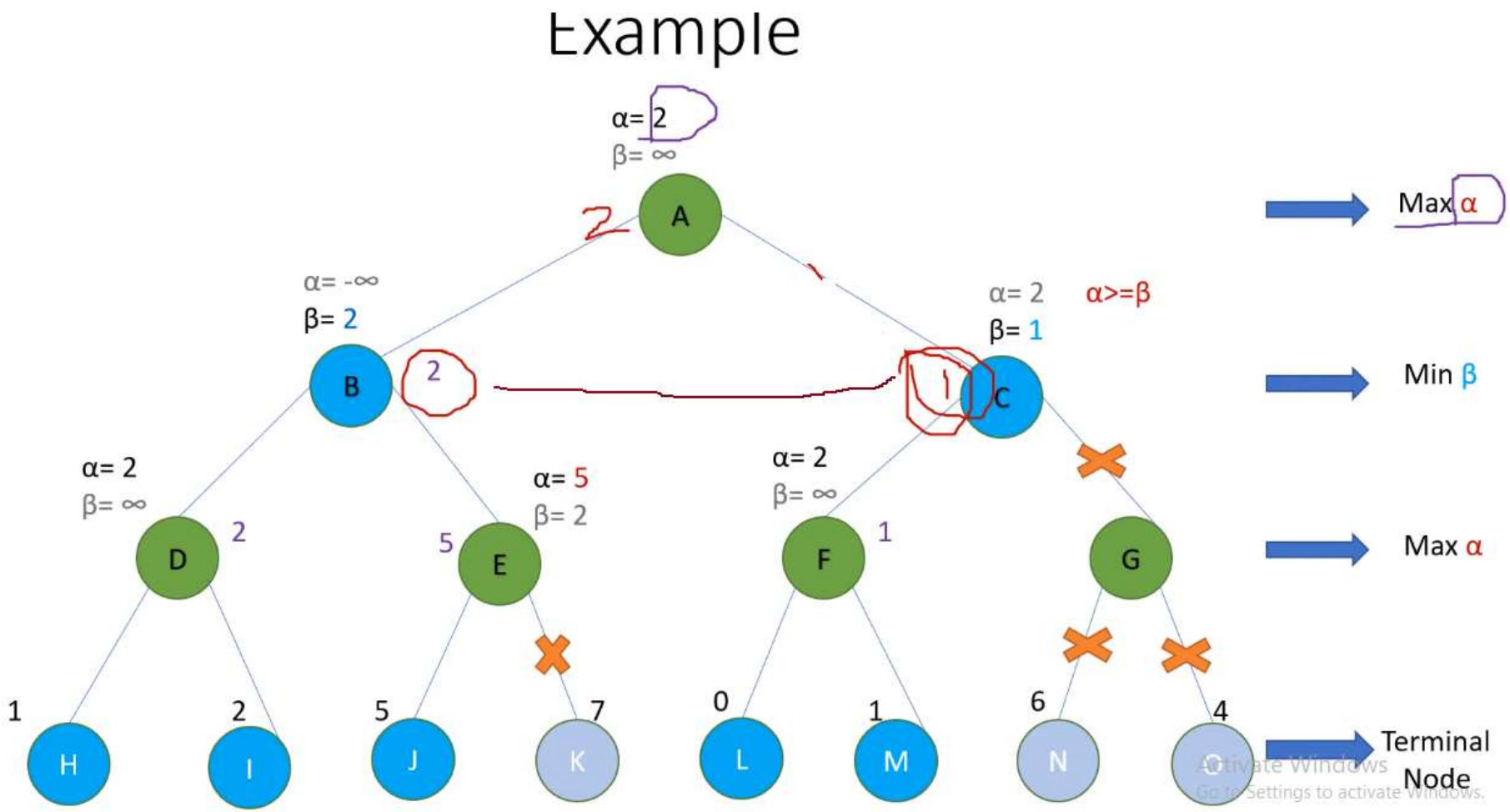

- Node value of C is also 1

# Example




Max(2,1) = 2 . So no replacement

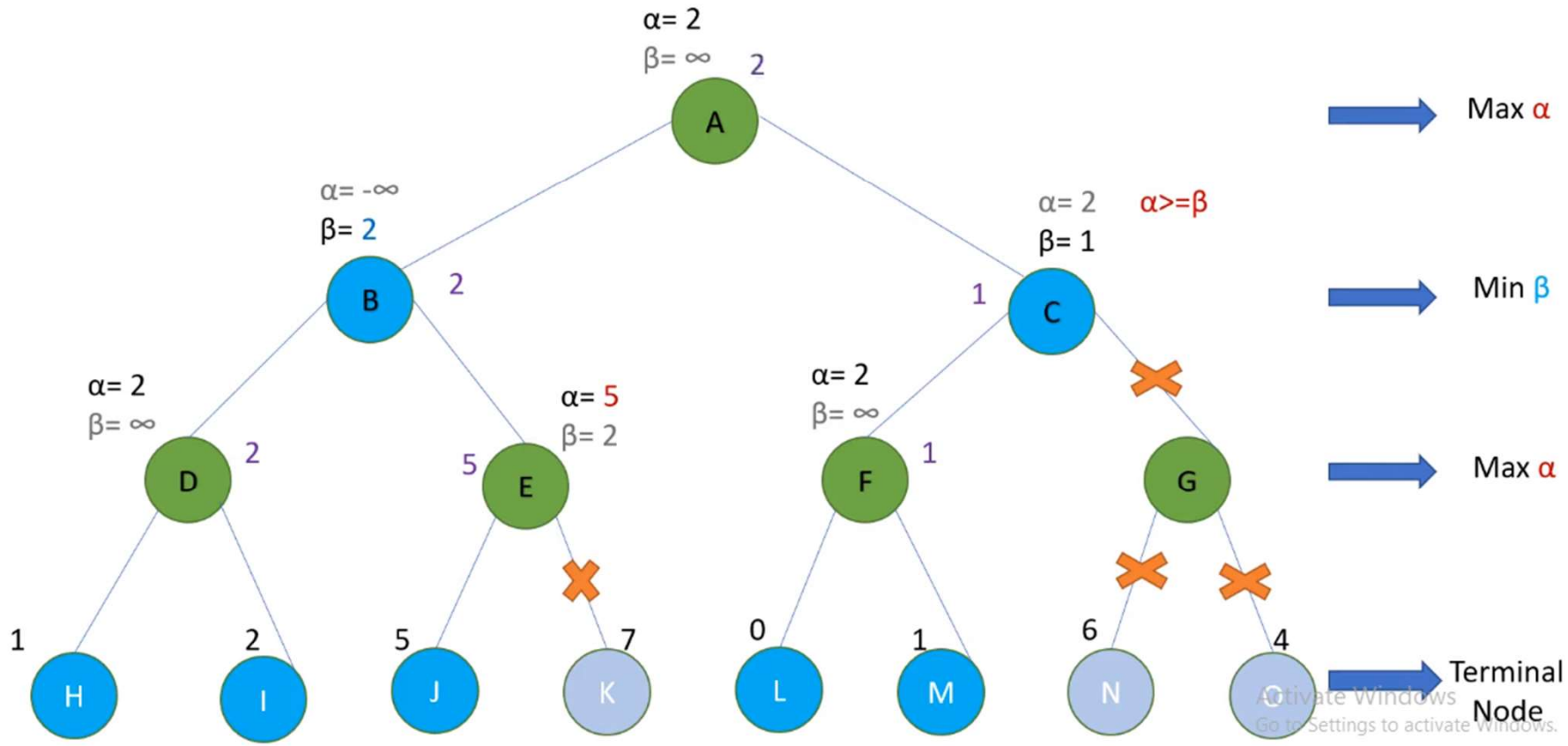
Node value of A = ?




Max(2,1) = 2 . So no replacement

Node value of A = ?

# Example




Max(2,1) = 2 . So no replacement

Node value of A = ?

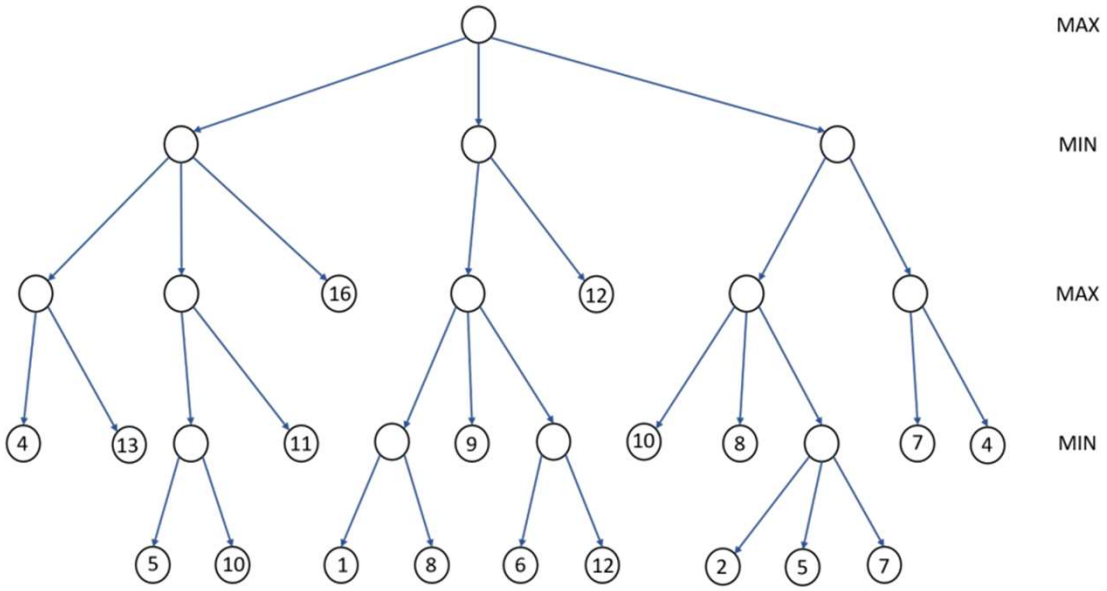
- That’s all. Its  over

# Conclusions

The Alpha-beta pruning to a standard minimax algorithm returns the same move as the standard algorithm does, but **it removes all the nodes which are not really affecting the final decision but making algorithm slow**. Hence by pruning these nodes, it makes the algorithm fast.

# Assessment

- Use the Minimax algorithm to compute the minimax value at each node for the game tree below.




- Use the Alpha-Beta pruning algorithm to prune the game tree above assuming child nodes are visited from left to right. Show all final alpha and beta values computed at the root, each internal node visited, and at the top of pruned branches. Also show the pruned branches.
-