Elective Course
Course Code: CS4103
Autumn 2025-26

Lecture #14

Artificial Intelligence for Data Science

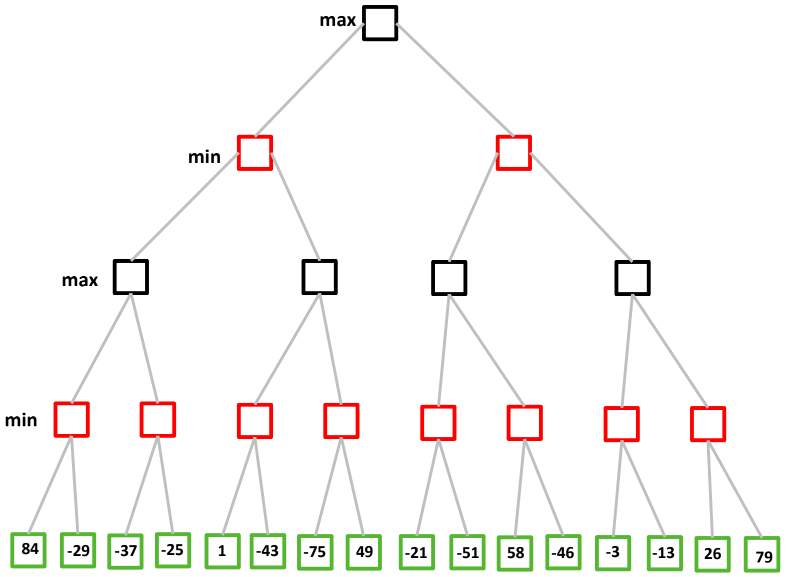
Week-4: PROBLEM SOLVING BY SEARCH

Adversarial Search Problem--- Games [Part-II]

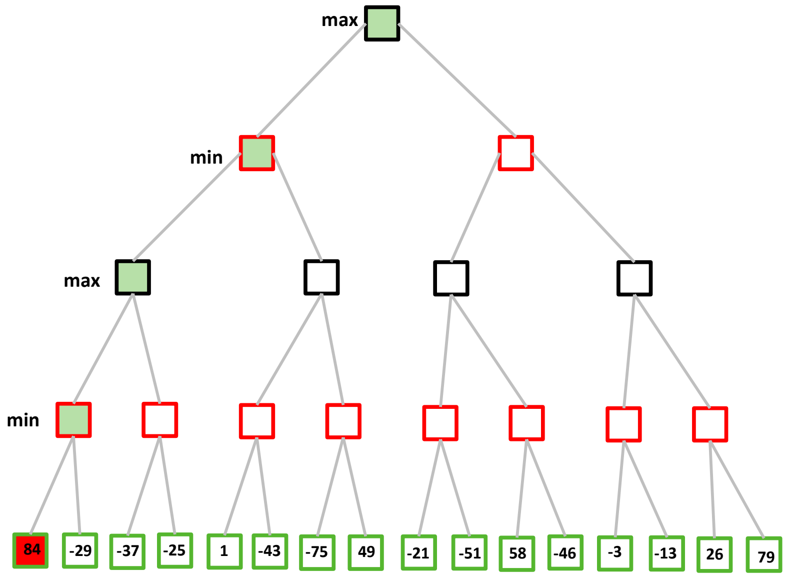
Course Instructor:

Dr. Monidipa Das
Assistant Professor
Department of Computational and Data Sciences
Indian Institute of Science Education and Research Kolkata, India 741246

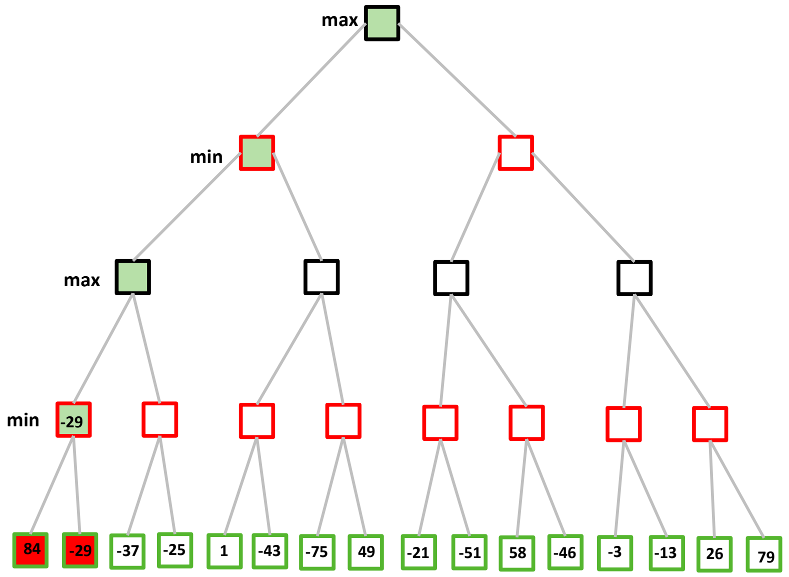
Minimax Example (revisited)



Minimax Example (revisited)

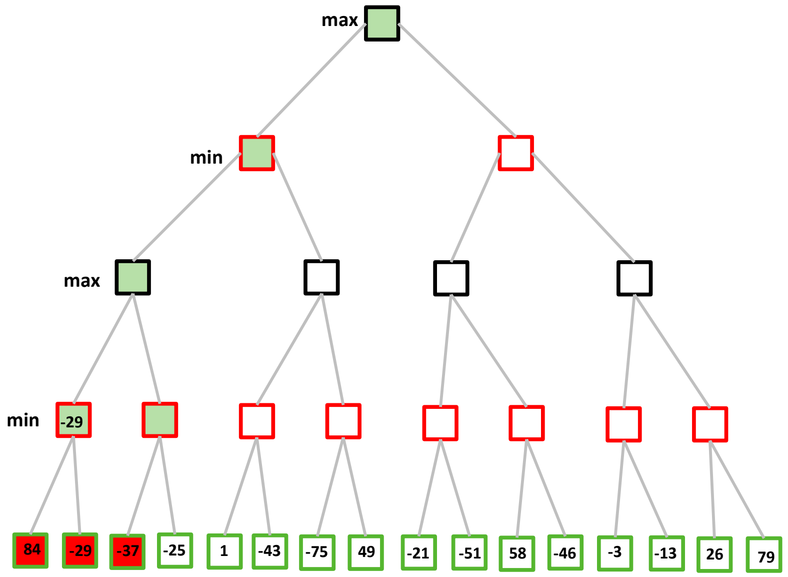


Minimax Example (revisited)



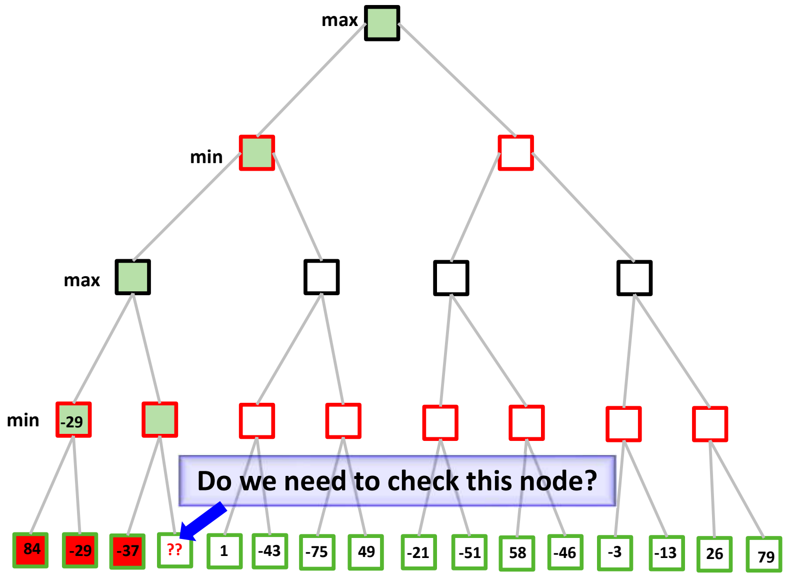
Minimax Example (revisited)

5



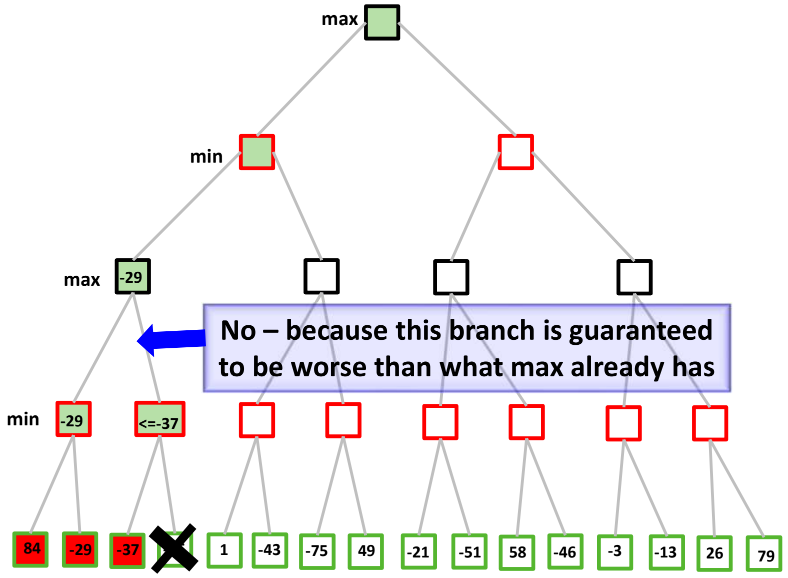
Minimax Example (revisited)

6



Do we need to check this node?

Minimax Example (revisited)



No – because this branch is guaranteed to be worse than what max already has

Alpha-Beta Procedure

- The alpha-beta procedure can speed up a depth-first minimax search.
- **Alpha:** a lower bound on the value that a max node may ultimately be assigned

$$v \geq \alpha$$

- **Beta:** an upper bound on the value that a min node may ultimately be assigned

$$v \leq \beta$$

Alpha-Beta Pruning

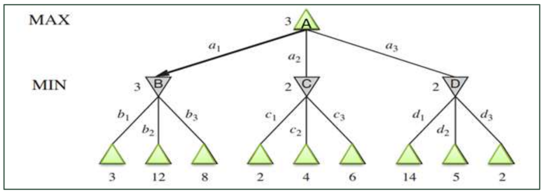
9

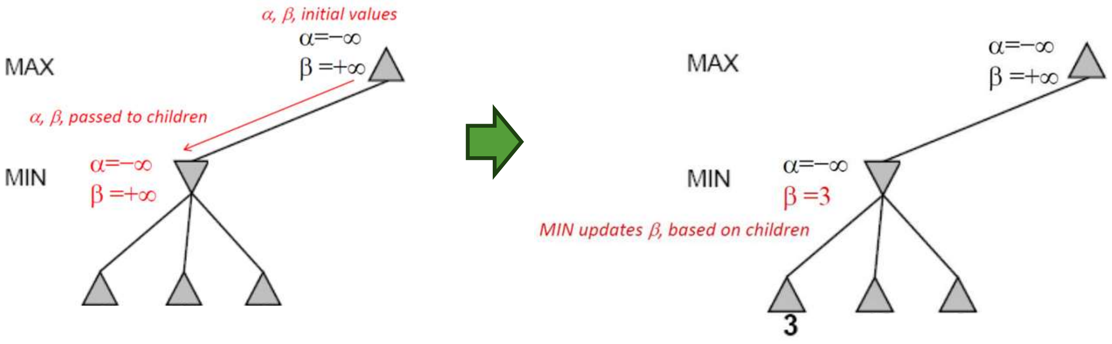
- Traverse the search tree in depth-first order
- At each MAX node n, $alpha(n)$ = maximum value found so far
- At each MIN node n, $beta(n)$ = minimum value found so far
  - **Note:** The alpha values start at -infinity and only increase, while beta values start at +infinity and only decrease.
- **Beta cutoff:** Given a MAX node n, cut off the search below n (i.e., don't generate or examine any more of n's children) if $alpha(n) >= beta(i)$ for some MIN node ancestor i of n.
- **Alpha cutoff:** stop searching below MIN node n if $beta(n) <= alpha(i)$ for some MAX node ancestor i of n.

Alpha Beta Example

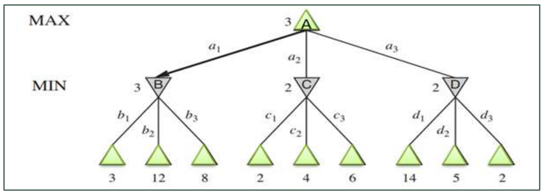
10

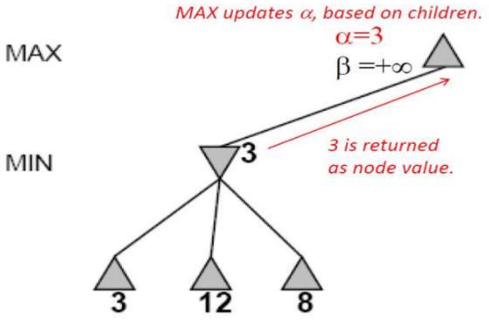
- Do depth-first search until first leaf





Alpha Beta Example



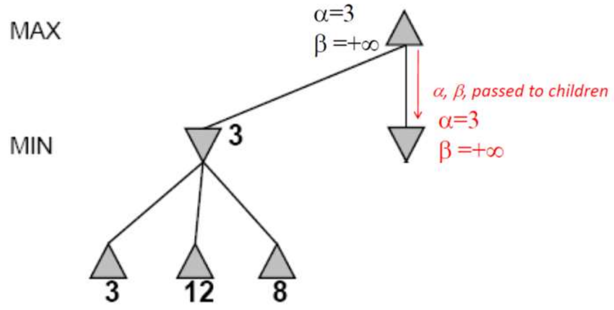


MAX updates α, based on children.

α=3

β =+∞

3 is returned as node value.



α=3

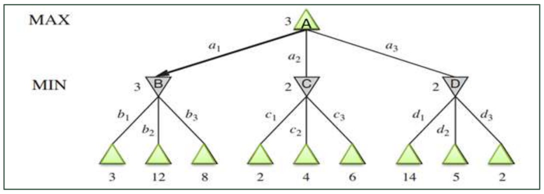
β =+∞

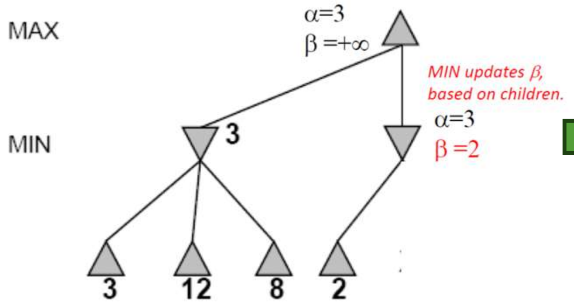
α, β, passed to children

α=3

β =+∞

Alpha Beta Example





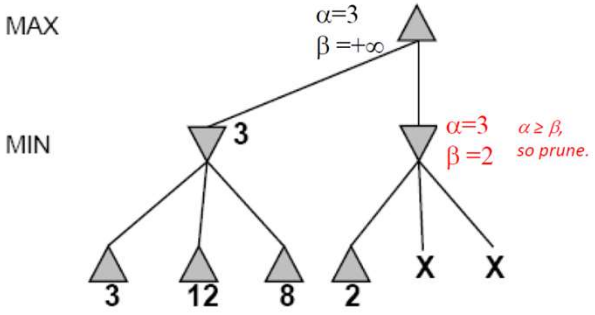
α=3

β =+∞

MIN updates β, based on children.

α=3

β =2



α=3

β =+∞

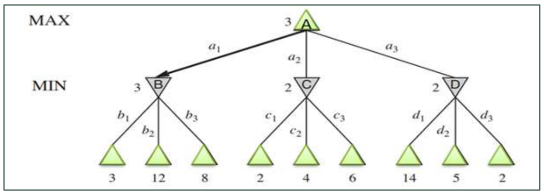
α=3

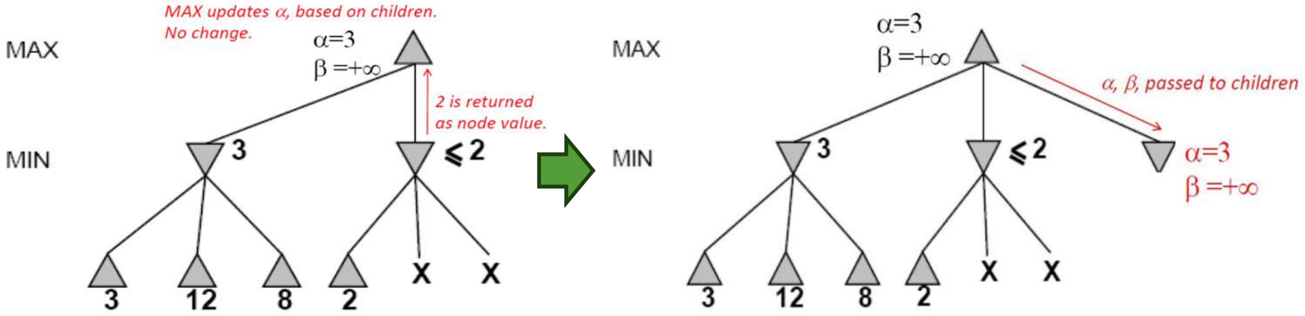
β =2

$\alpha \geq \beta$, so prune.

Alpha Beta Example

13





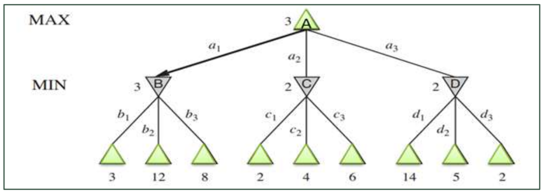
MAX updates α, based on children. No change.

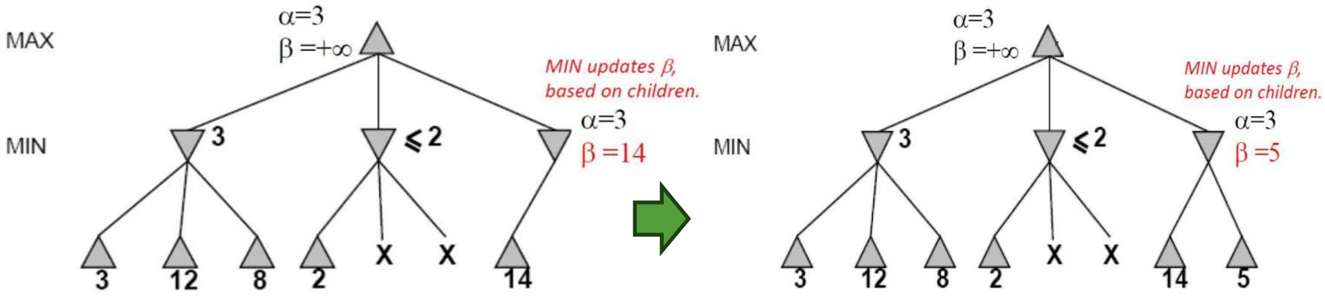
2 is returned as node value.

α, β, passed to children

Alpha Beta Example

14



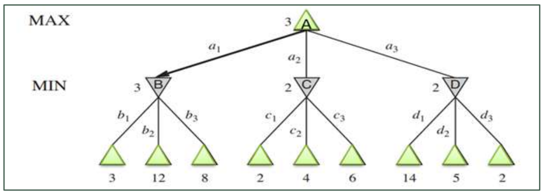


MIN updates β, based on children.

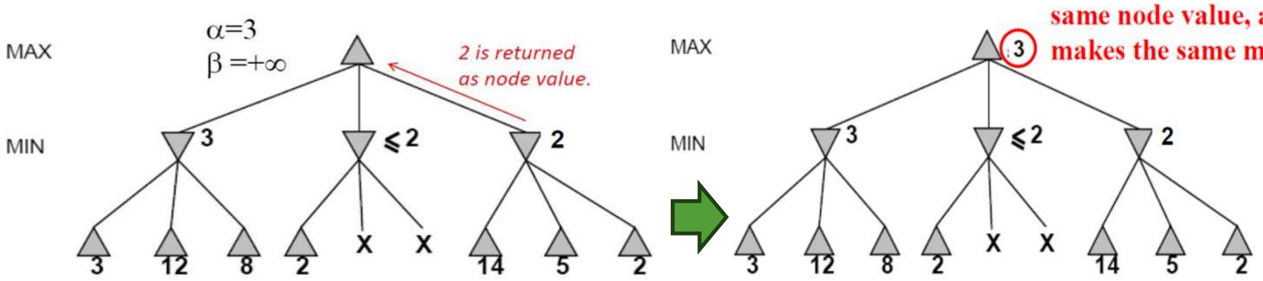
MIN updates β, based on children.

Alpha Beta Example

15



Max calculates the same node value, and makes the same move!



Alpha–Beta Search Algorithm

16

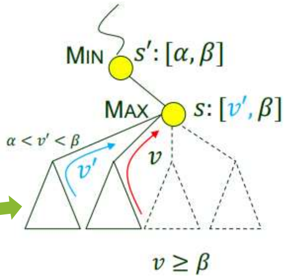
```
function ALPHA-BETA-SEARCH(state) returns an action
   v ← MAX-VALUE(state, −∞, +∞)
   return the action in ACTIONS(state) with value v

function MAX-VALUE(state, α, β) returns a utility value
   if TERMINAL-TEST(state) then return UTILITY(state)
   v ← −∞
   for each a in ACTIONS(state) do
      v ← MAX(v, MIN-VALUE(RESULT(s,a), α, β))
      if v ≥ β then return v
      α ← MAX(α, v)
   return v

function MIN-VALUE(state, α, β) returns a utility value
   if TERMINAL-TEST(state) then return UTILITY(state)
   v ← +∞
   for each a in ACTIONS(state) do
      v ← MIN(v, MAX-VALUE(RESULT(s,a) , α, β))
      if v ≤ α then return v
      β ← MIN(β, v)
   return v
```

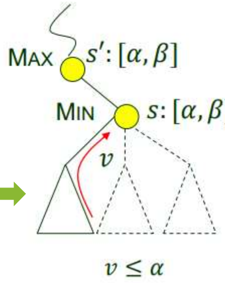
// no change of β value within MAX-VALUE()

$\alpha \leftarrow MAX(\alpha, v)$
$if\, \alpha \geq \beta\ then\ return\ v$

// no change of α value within MIN-VALUE()

$\beta \leftarrow MIN(\beta, v)$
$if\, \beta \leq \alpha\ then\ return\ v$



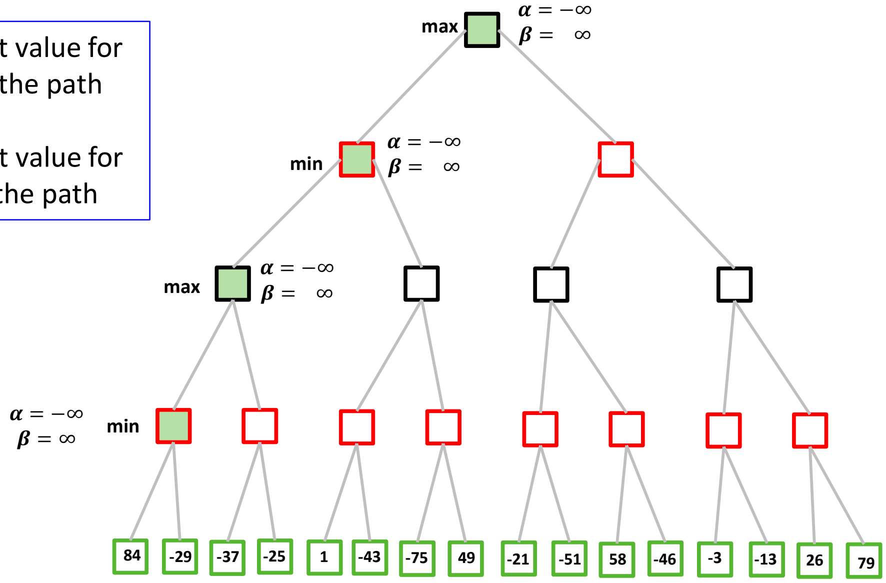


Example

17

α - the best value for max along the path

β - the best value for min along the path

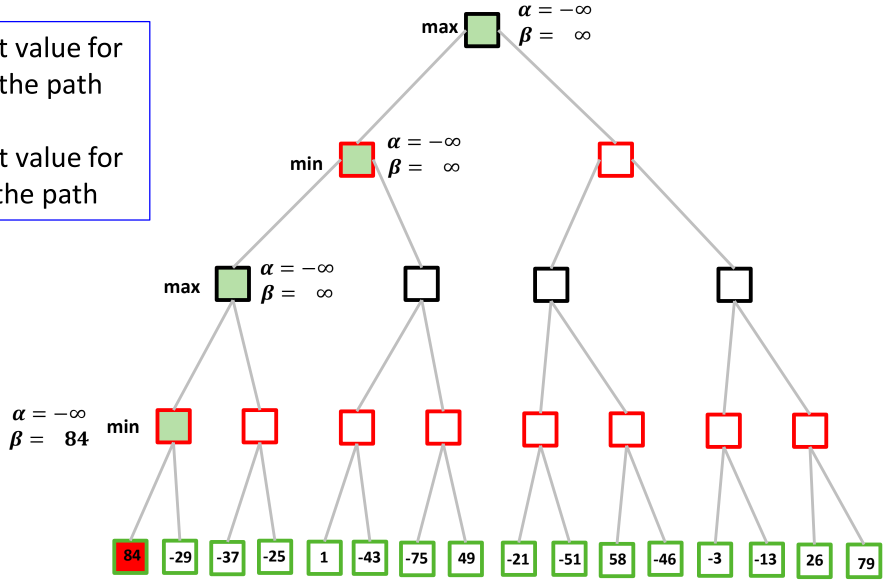


Example

18

α - the best value for max along the path

β - the best value for min along the path

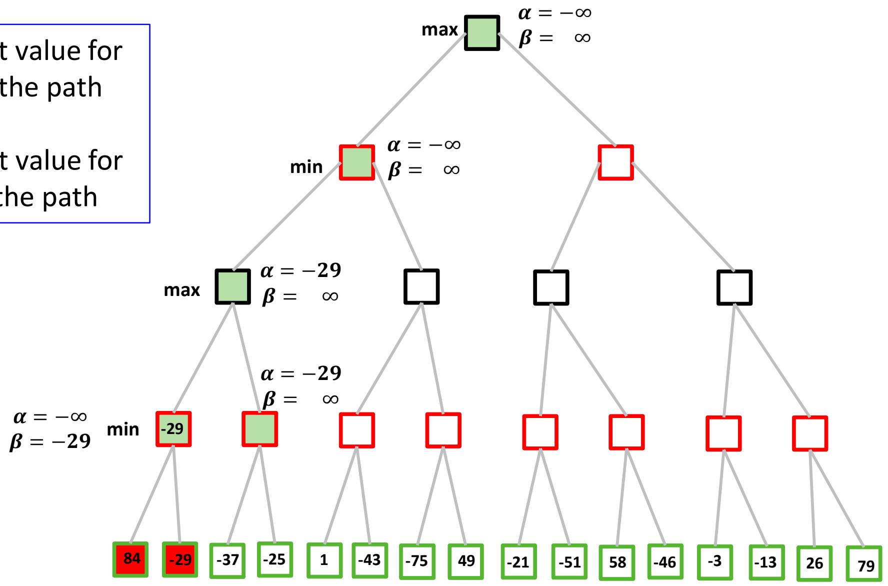


Example

19

α - the best value for max along the path

β - the best value for min along the path

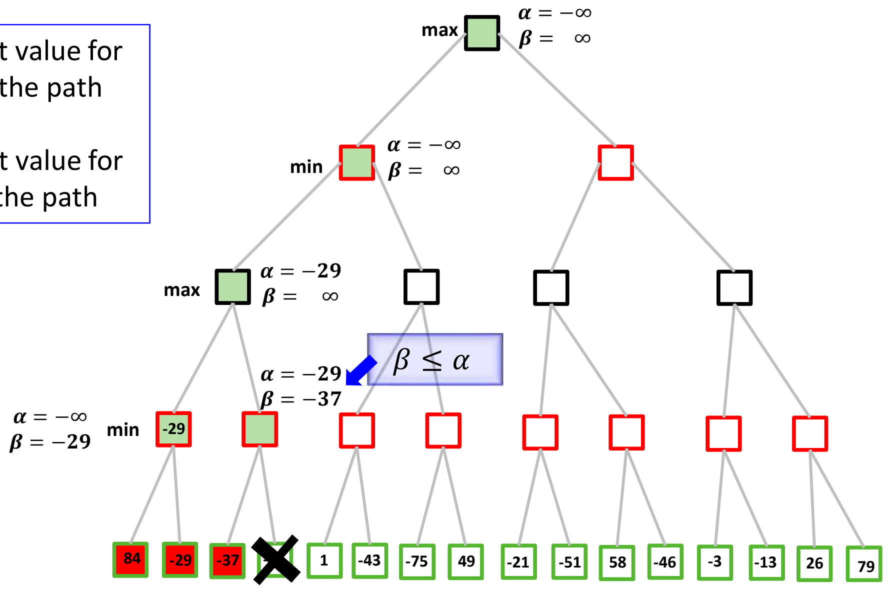


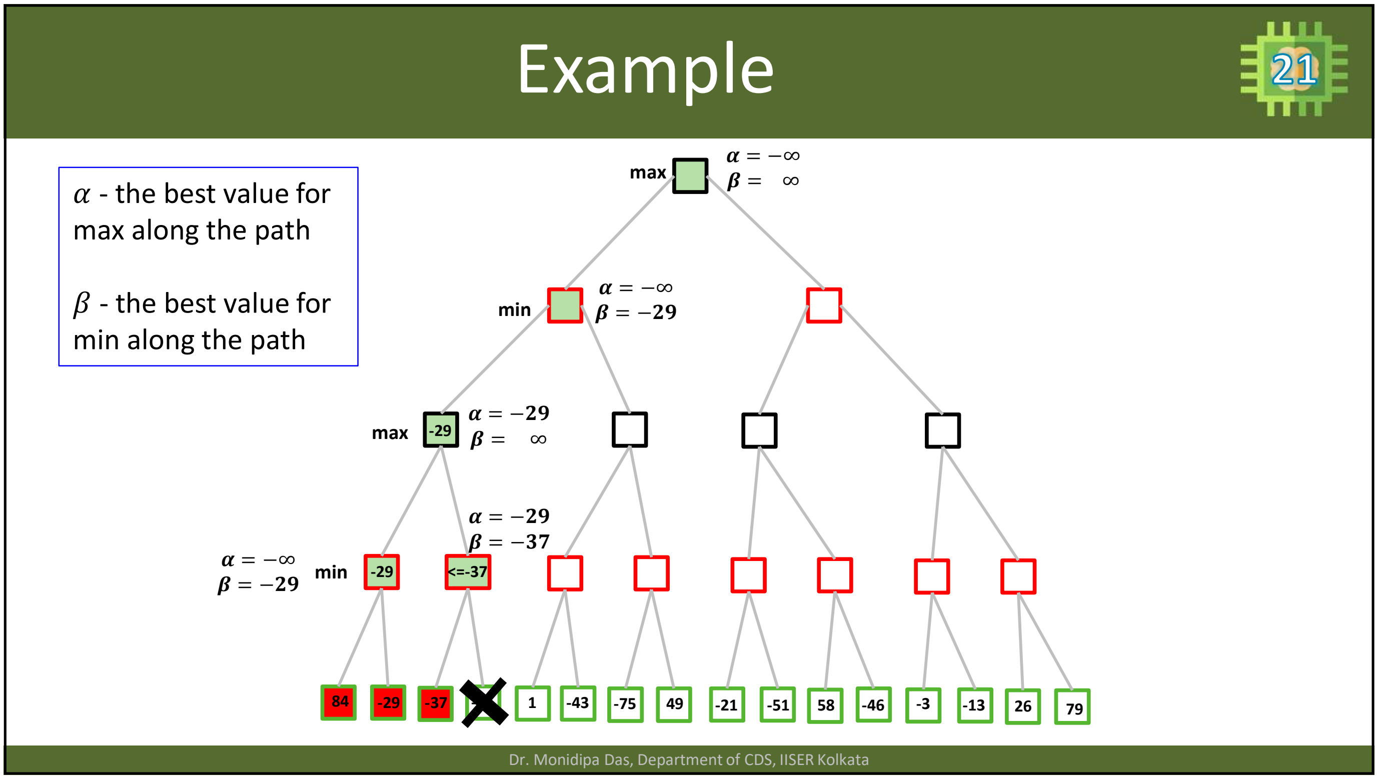
Example

20

α - the best value for max along the path

β - the best value for min along the path

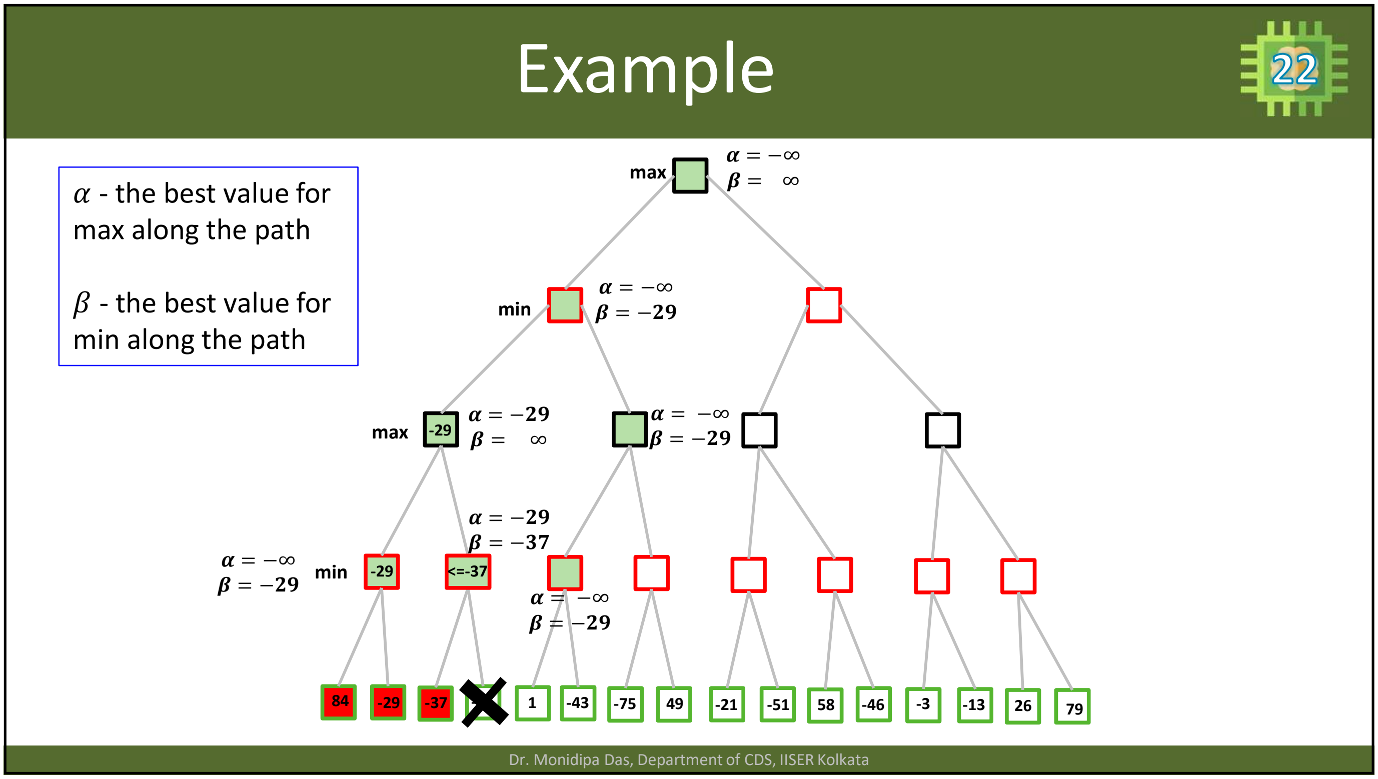




Example

α - the best value for max along the path

β - the best value for min along the path



Example

α - the best value for max along the path

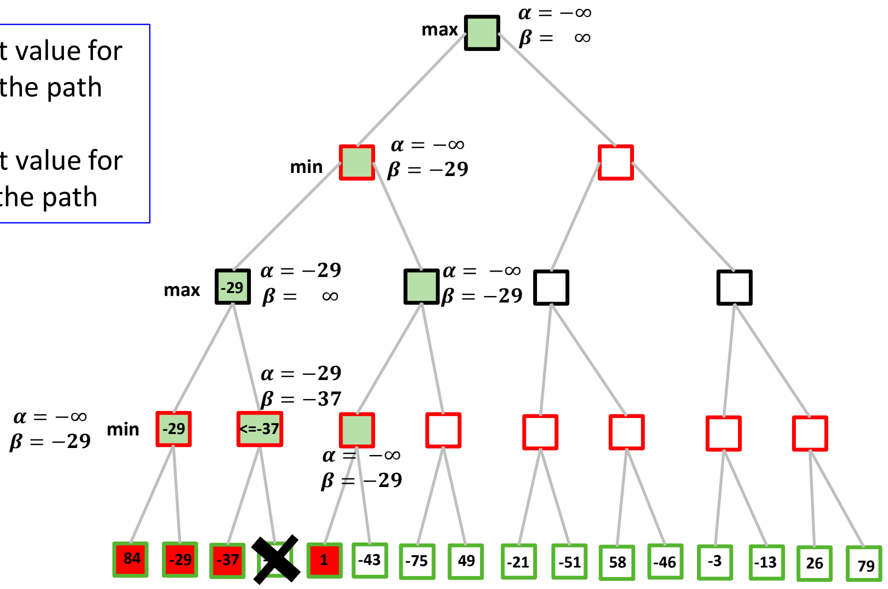
β - the best value for min along the path

Example

23

α - the best value for max along the path

β - the best value for min along the path



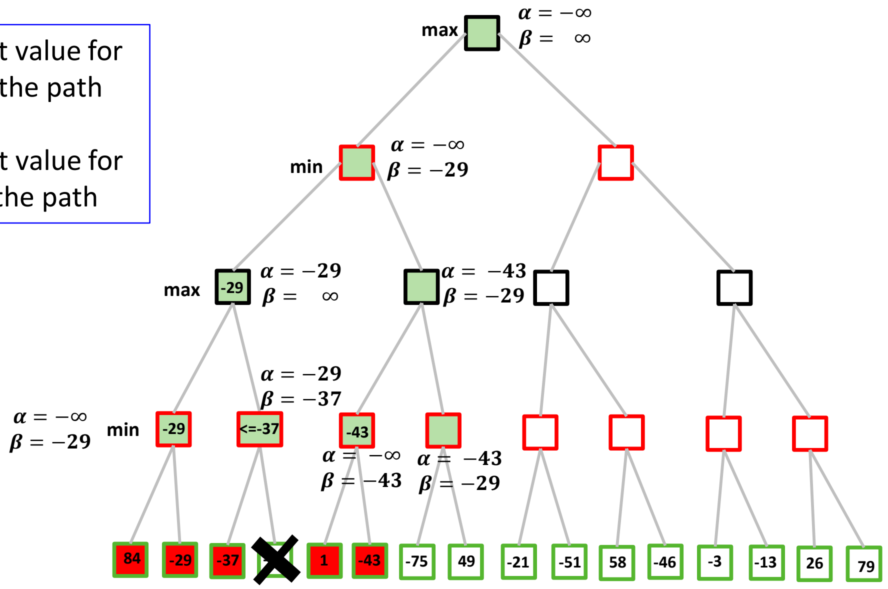
Dr. Monidipa Das, Department of CDS, IISER Kolkata

Example

24

α - the best value for max along the path

β - the best value for min along the path



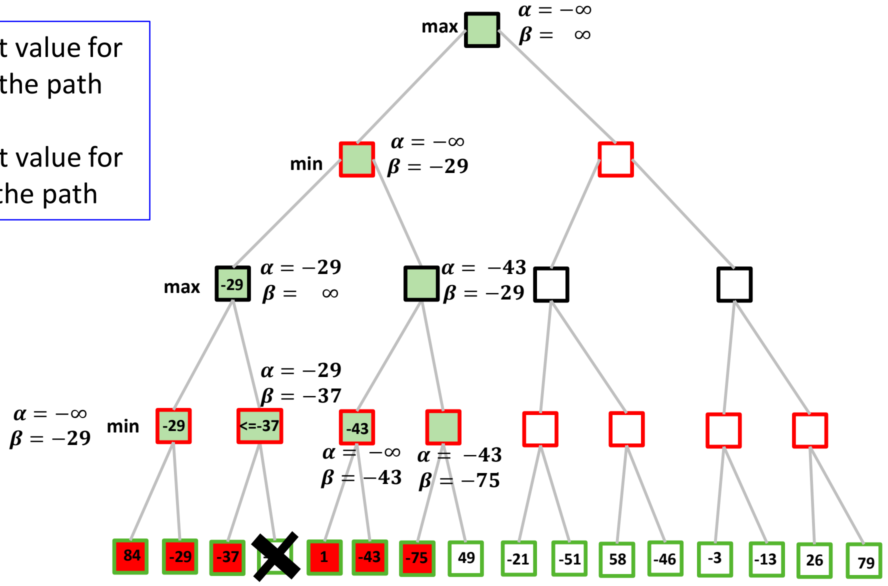
Dr. Monidipa Das, Department of CDS, IISER Kolkata

Example

25

α - the best value for max along the path

β - the best value for min along the path

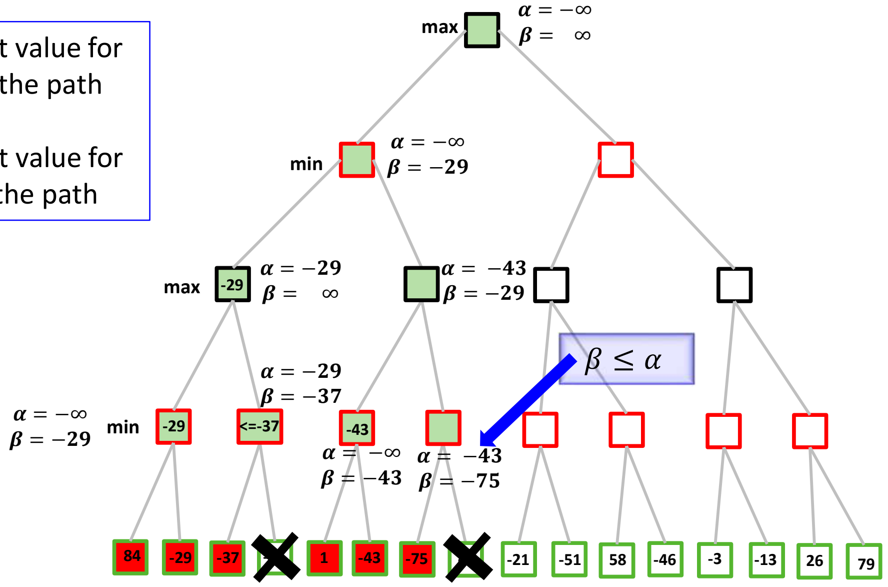


Example

26

α - the best value for max along the path

β - the best value for min along the path

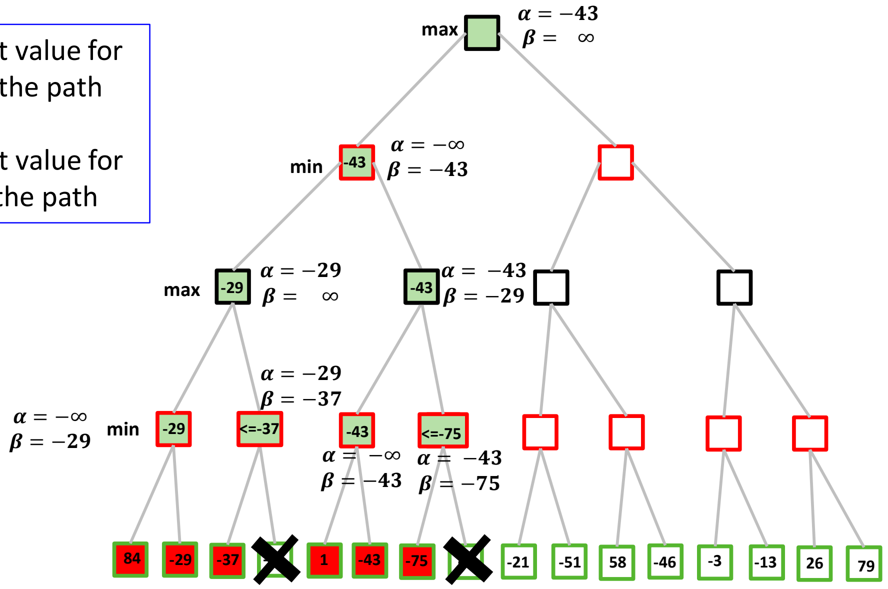


Example

27

α - the best value for max along the path

β - the best value for min along the path

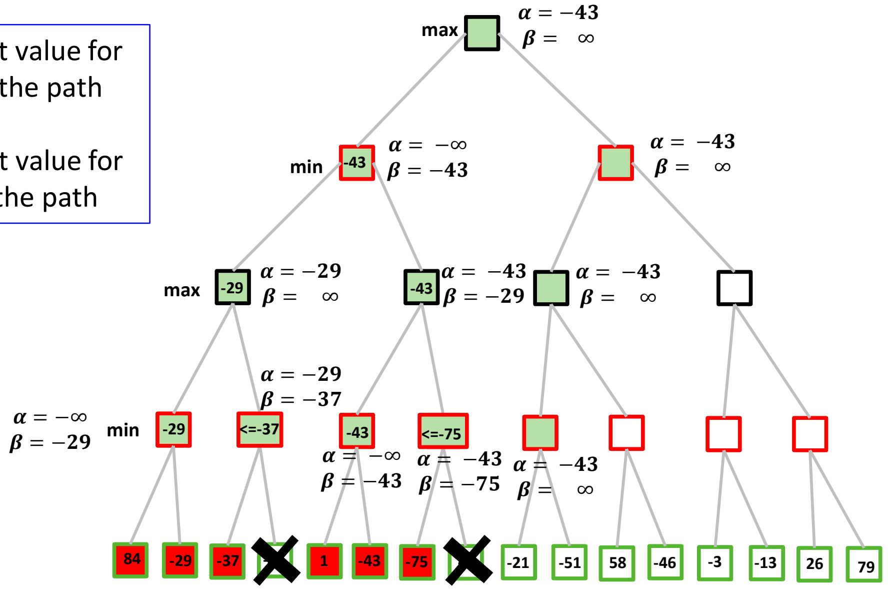


Example

28

α - the best value for max along the path

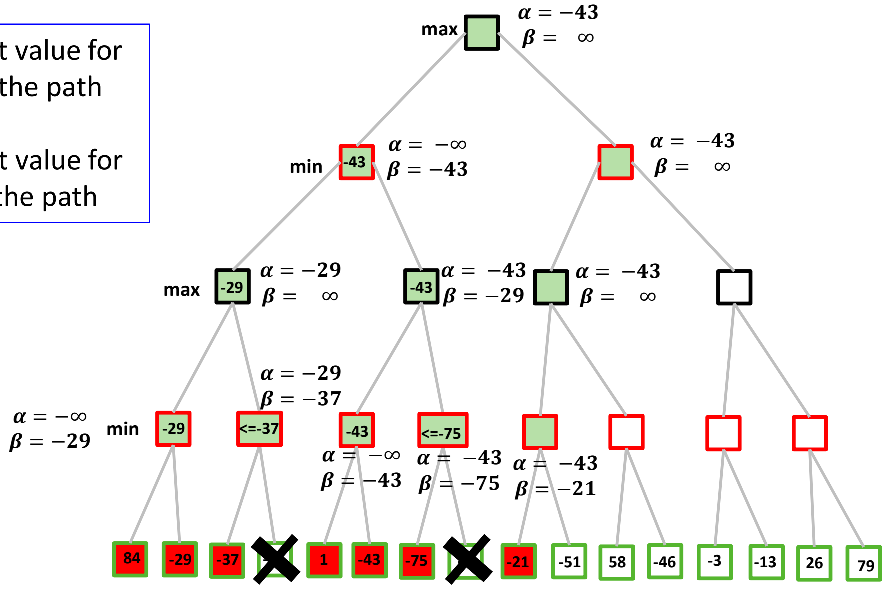
β - the best value for min along the path



Example

α - the best value for max along the path

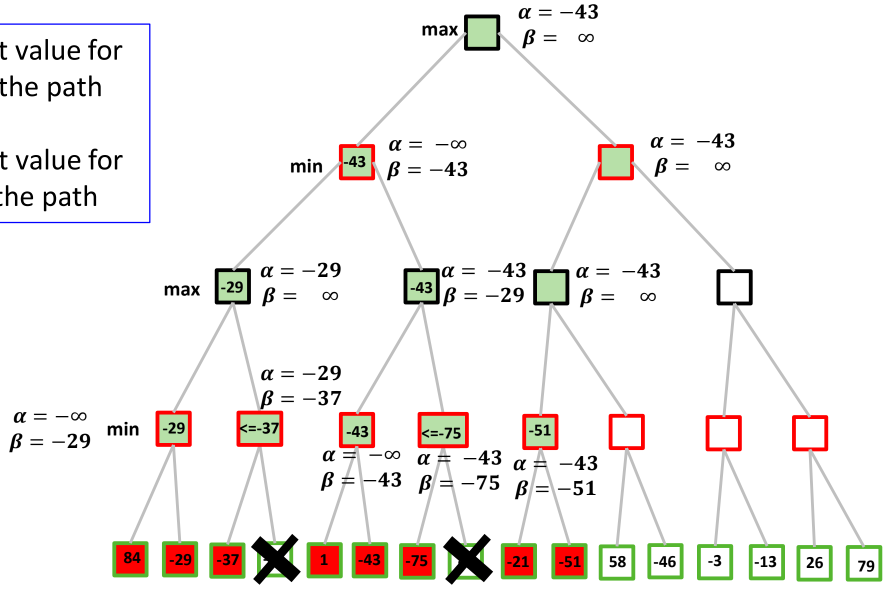
β - the best value for min along the path

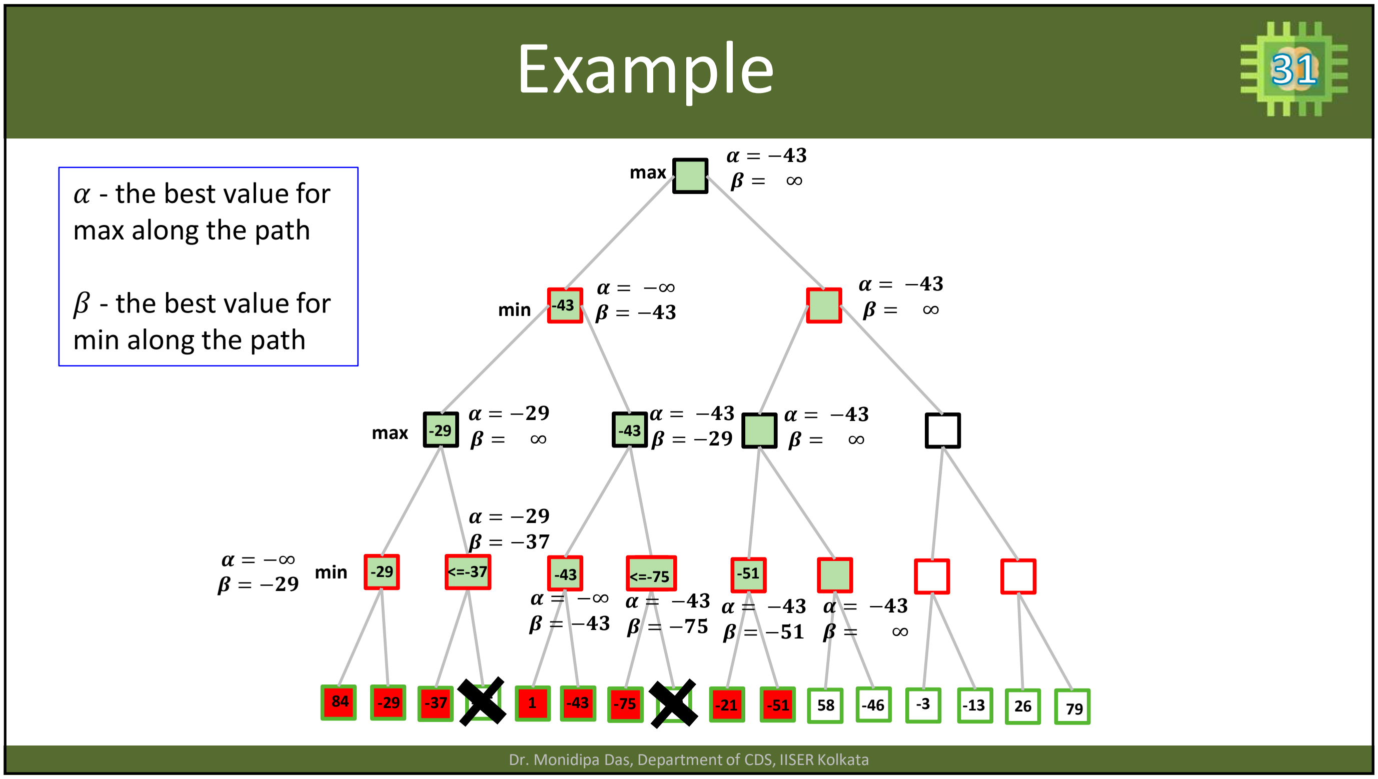


Example

α - the best value for max along the path

β - the best value for min along the path





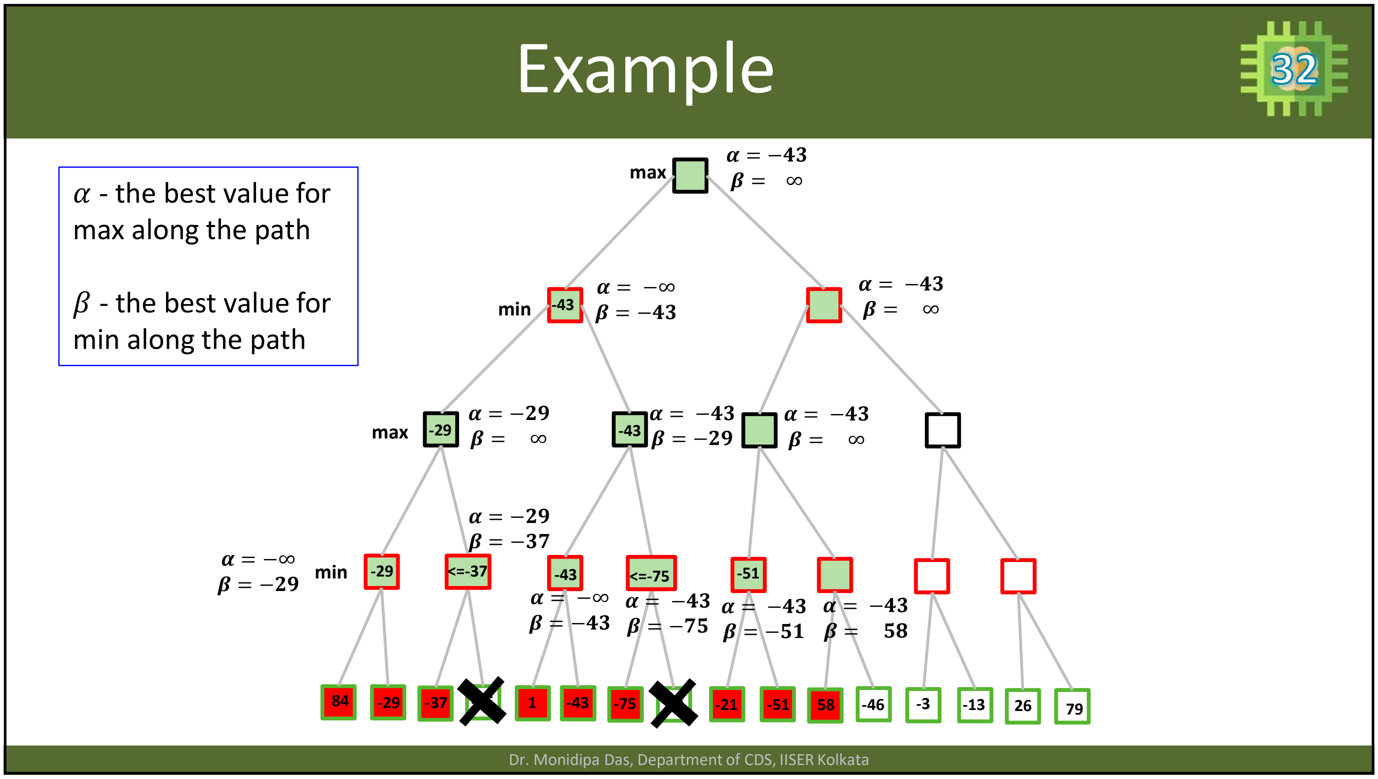
Example

31

α - the best value for max along the path

β - the best value for min along the path

Dr. Monidipa Das, Department of CDS, IISER Kolkata



Example

32

α - the best value for max along the path

β - the best value for min along the path

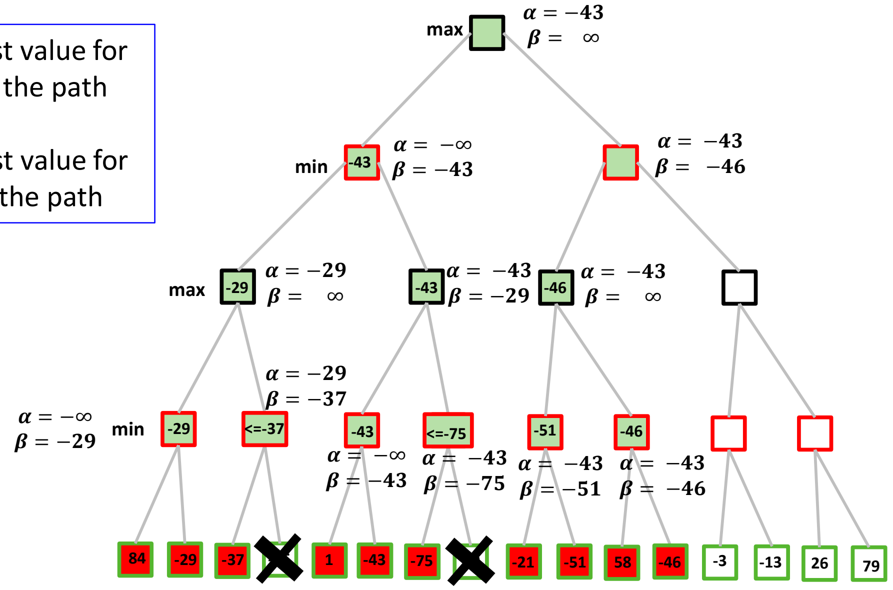
Dr. Monidipa Das, Department of CDS, IISER Kolkata

Example

33

α - the best value for max along the path

β - the best value for min along the path

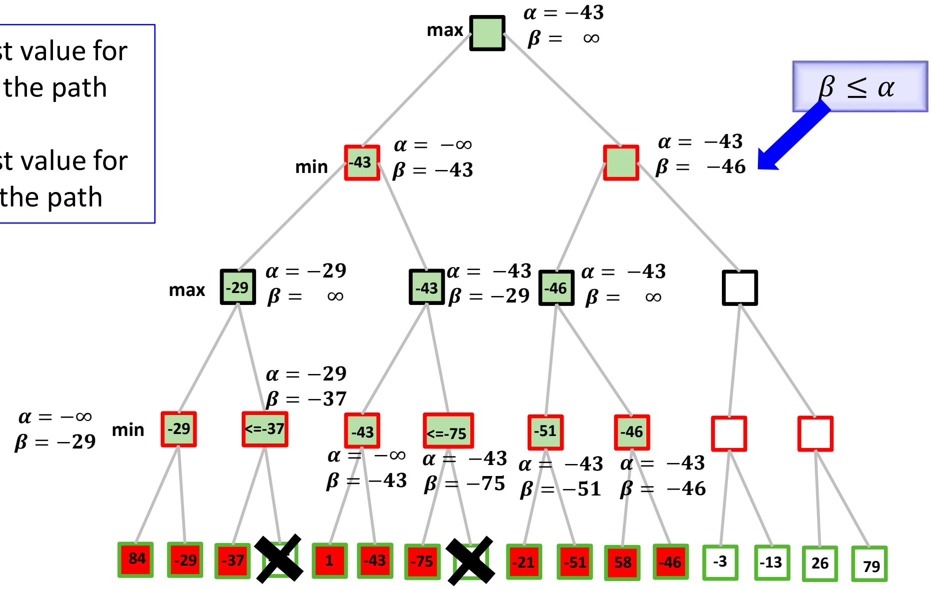


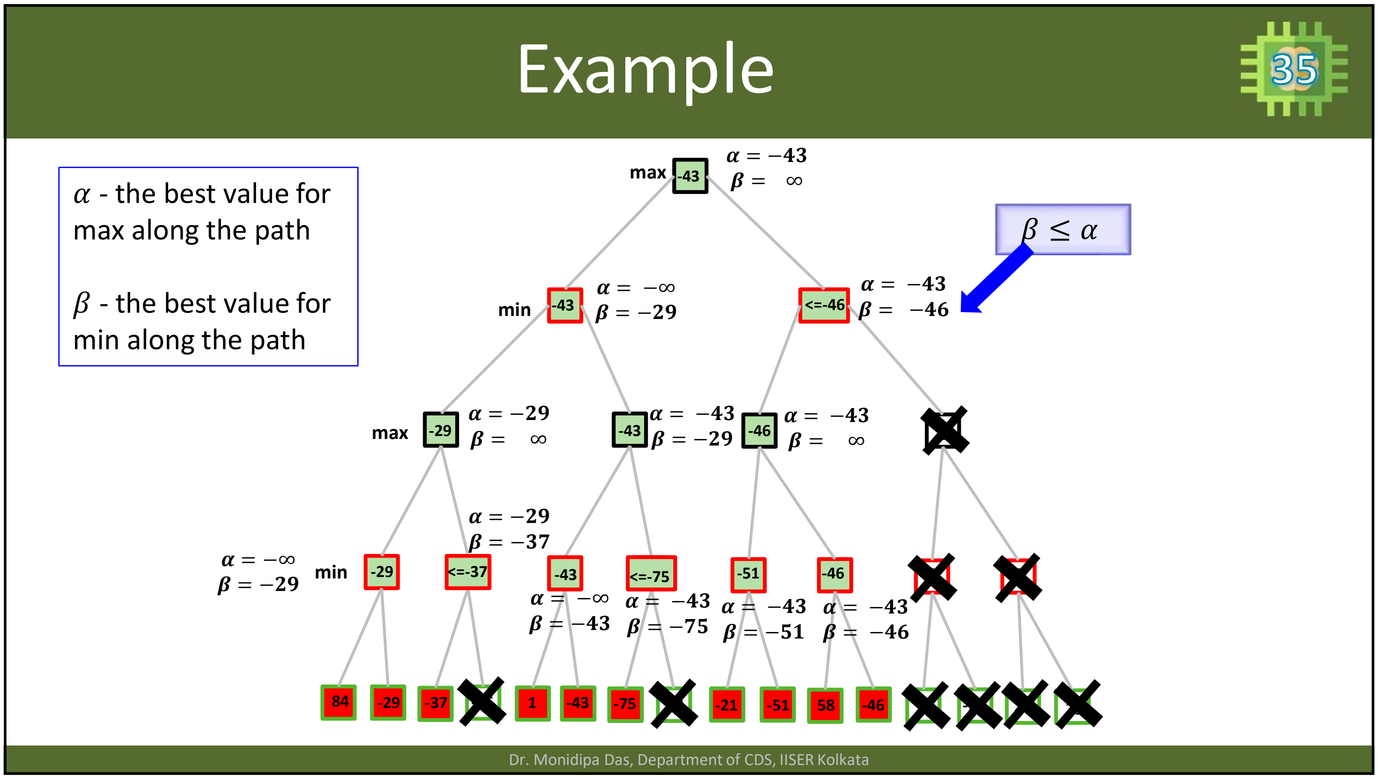
Example

34

α - the best value for max along the path

β - the best value for min along the path


$\beta \leq \alpha$



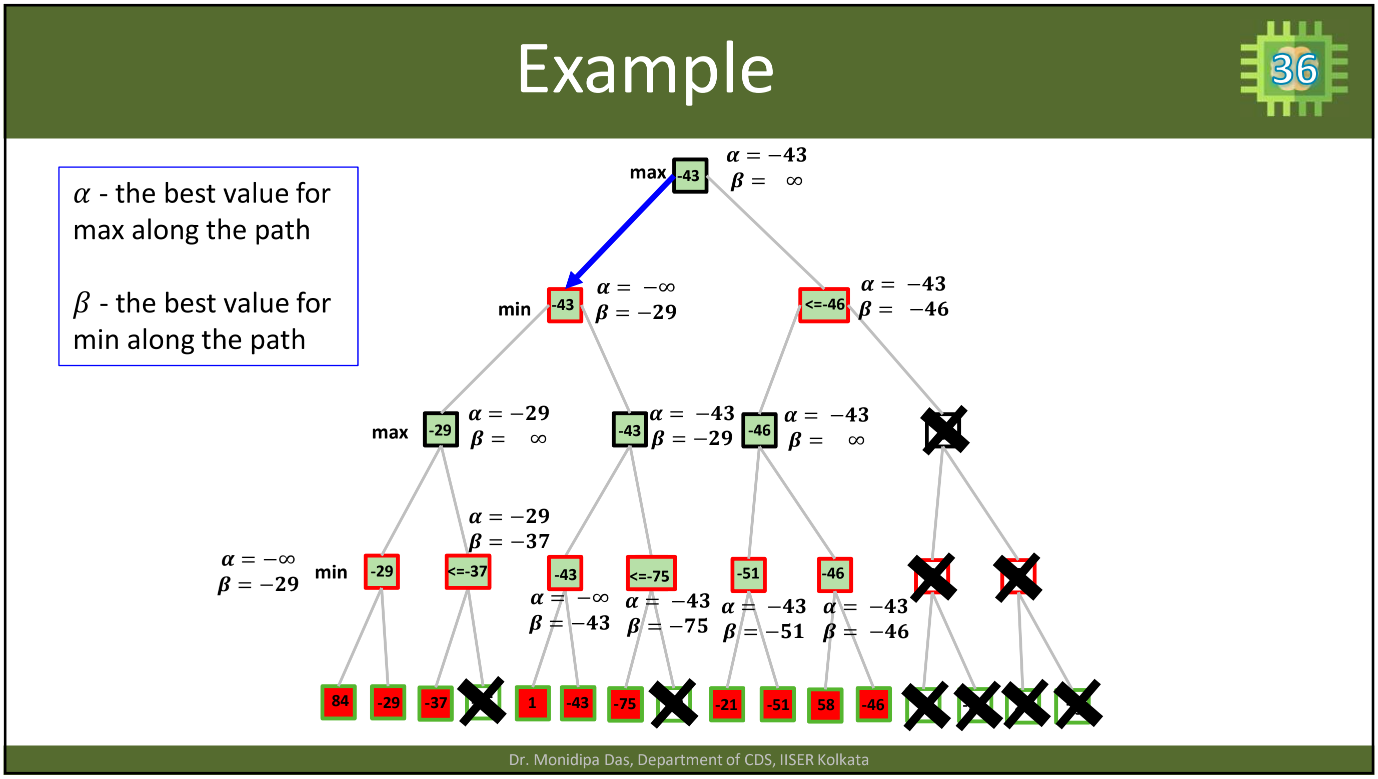
Example

35

α - the best value for max along the path

β - the best value for min along the path

Dr. Monidipa Das, Department of CDS, IISER Kolkata



Example

36

α - the best value for max along the path

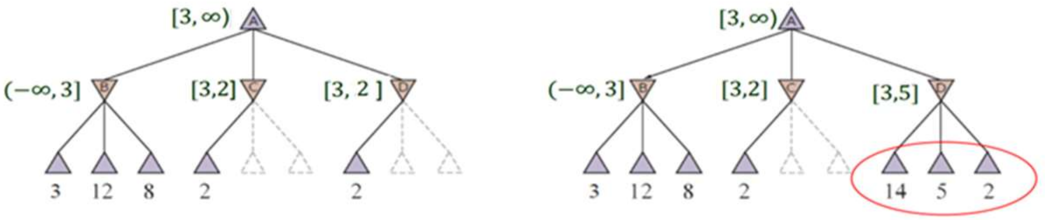
β - the best value for min along the path

Dr. Monidipa Das, Department of CDS, IISER Kolkata

Effectiveness of alpha-beta

- Alpha-beta is guaranteed to compute the same value for the root node as computed by minimax, with less or equal computation
- **Worst case:** no pruning, examining b^m leaf nodes, where each node has b children and a m-ply search is performed
- **Best case:** examines only $O(b^{m/2})$ nodes.
  - Result is you can search twice as deep as minimax!
- **Best case** is when each player's best move is the first alternative generated

Move Ordering



Successors 14 and 5 would've been pruned had 2 been generated first.

- Effectiveness of pruning is highly dependent on the order in which successors are generated.
- "Perfect ordering" has effective branching factor $\sqrt{b}$, which limits examination to only $O(b^{m/2})$ nodes compared to $O(b^m)$ for minimax.
- $O(b^{3m/4}$) nodes for random move ordering.

Good Enough?

39

- Chess (Minimax):
  - branching factor b≈35
  - game length m≈100
  - search space b m ≈ 35^{100}≈10^{154}
  - Exact solution completely infeasible
- Chess (Alpha Beta Pruning):
  - branching factor b≈35
  - game length m≈100
  - search space $b^{m/2} \approx 35^{50} \approx 10^{77}$

40

Questions?