Adversarial Search

Chastine Fatichah
Departemen Teknik Informatika
April 2023

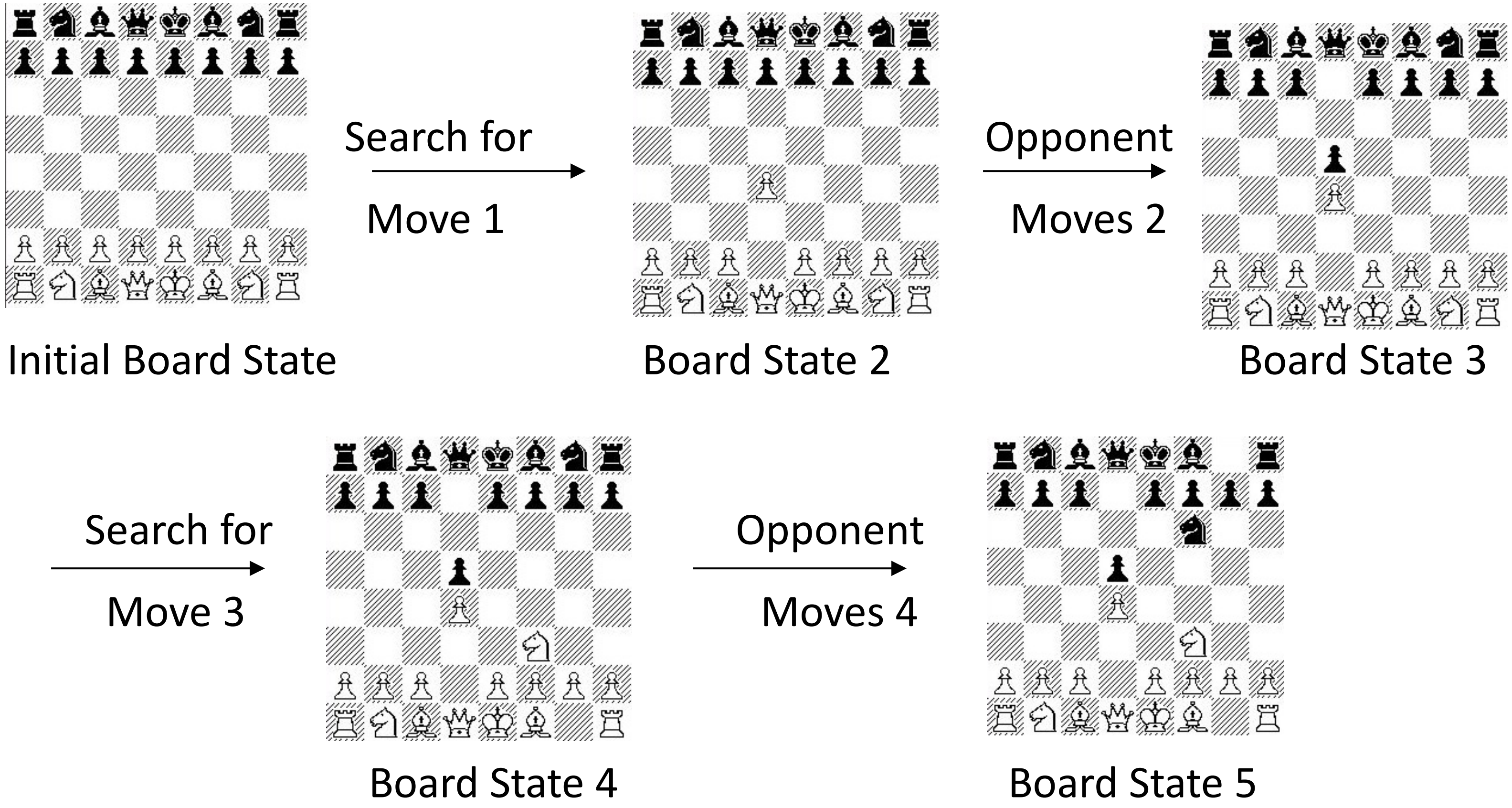
Capaian Pembelajaran Matakuliah

Mahasiswa mampu menjelaskan, mengidentifikasi, merancang, dan menerapkan *intelligent agent* untuk *problem* yang sesuai dengan memanfaatkan algoritma pencarian yang meliputi *uninformed search*, *informed search*, *heuristic search*, *adversarial search*, serta algoritma *search* untuk *Constraint Satisfaction Problem*

Pencarian dengan Lihat Status Lawan (*Adversarial Search*)

- Teknik pencarian sebelumnya *uninformed, informed, local search*
  - Tidak memperhitungkan *state* dari pihak lawan
- *Adversarial Search* peduli dengan *state* lawan:
  - Contoh problem umumnya pada permainan *game,* misal: Catur (orang lawan komputer)
  - Solusi dari algoritma *Adversarial search* menjadi langkah komputer
  - Algoritma *Adversarial Search: Minimax* dan *Alpha-beta*

Game Playing: Pencarian *best move* setiap saat



Initial Board State

Search for Move 1

Board State 2

Opponent Moves 2

Board State 3

Search for Move 3

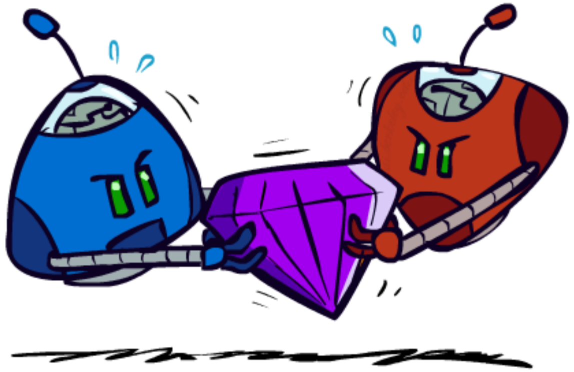
Board State 4

Opponent Moves 4

Board State 5

Jenis *Game*

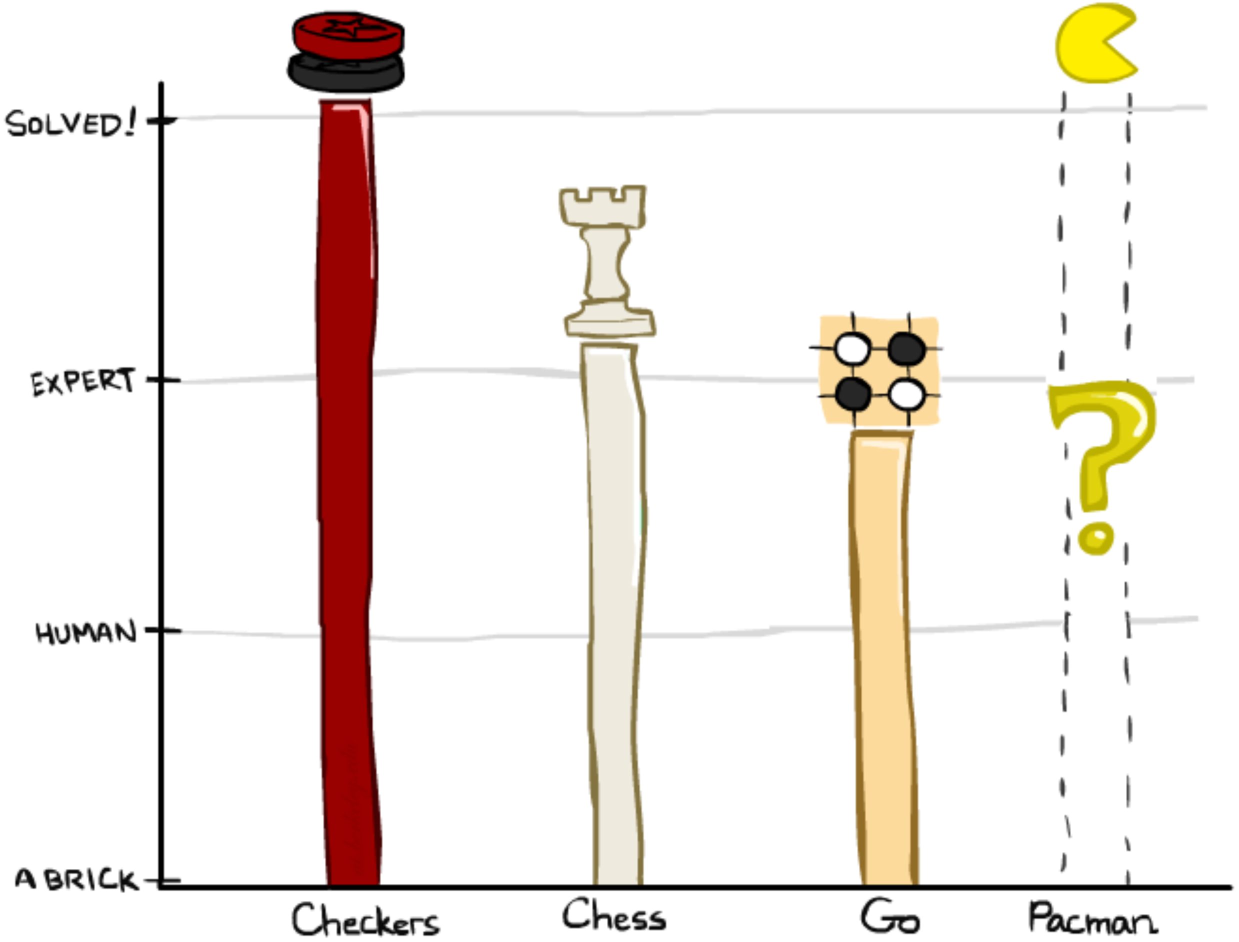
- *General Games*
  - Agen mempunyai *independent utilities* (luarannya berupa nilai)
  - *Cooperation, indifference, competition*
    - Setiap AI Agent perlu menyelesaikan game
- Zero-Sum Games
  - Agen mempunyai *opposite utilities* (luarannya berupa nilai)
  - Setiap nilai ada yang memaksimalkan dan yang lainnya meminimalkan
  - *Adversarial* (kompetisi)



Sumber: Adversarial Search by Dan Klein and Pieter Abbeel for CS188 Intro to AI at UC Berkeley

Zero-Sum Games

- **Checkers:** 1950: First computer player. 1994: First computer champion: Chinook ended 40-year-reign of human champion Marion Tinsley using complete 8-piece endgame. 2007: Checkers solved!
- **Chess:** 1997: Deep Blue defeats human champion Gary Kasparov in a six-game match. Deep Blue examined 200M positions per second, used very sophisticated evaluation and undisclosed methods for extending some lines of search up to 40 ply. Current programs are even better, if less historic.
- **Go :2016: Alpha GO defeats human champion. Uses Monte Carlo Tree Search, learned evaluation function.**
- **Pacman**



Sumber: Adversarial Search by Dan Klein and Pieter Abbeel for CS188 Intro to AI at UC Berkeley

Konsep *Game*

Game secara formal didefinisikan dengan elemen berikut:

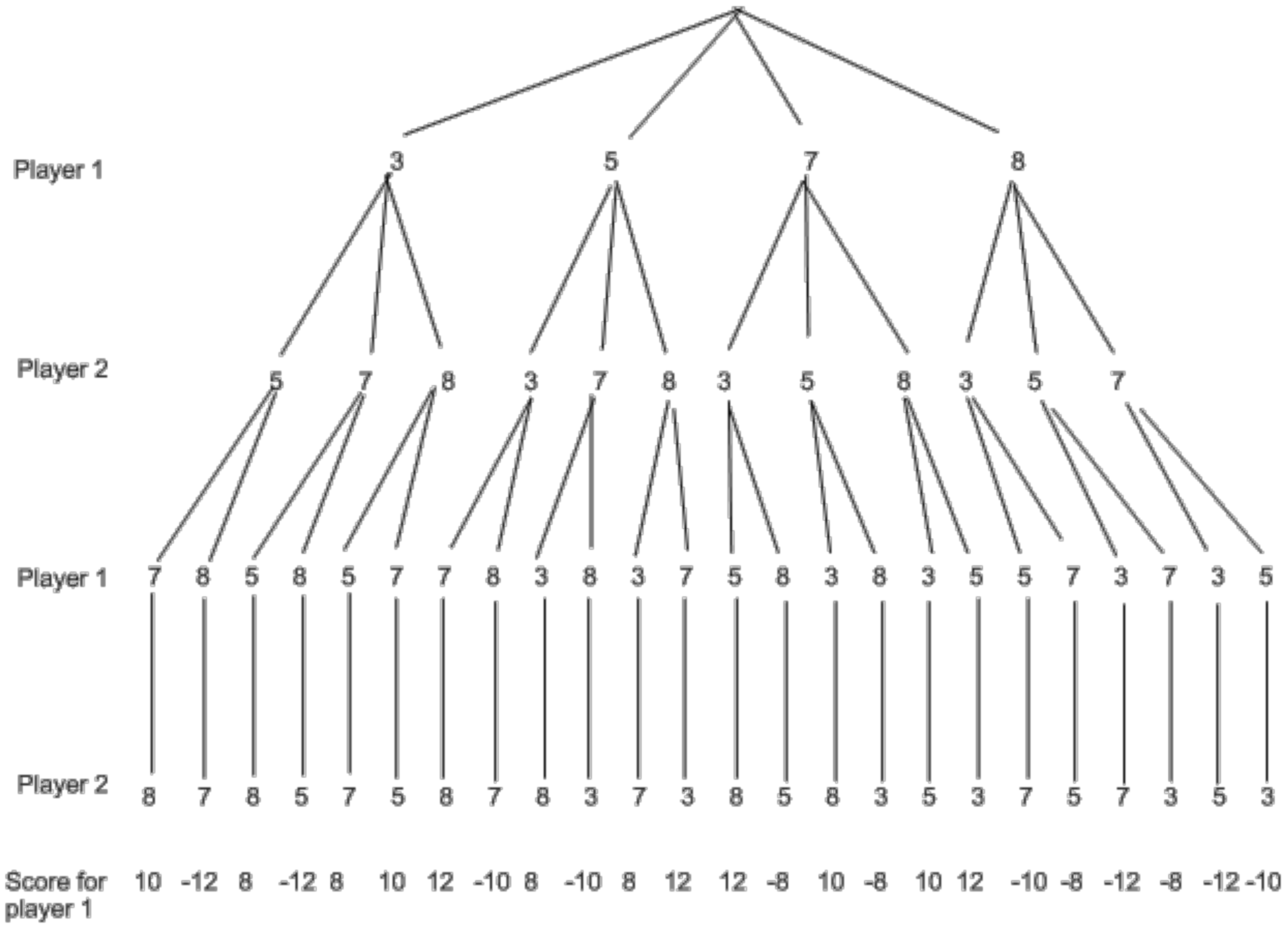
- S_0: *State* awal
- To-MOVE(*s*) : *Player* yang mendapat giliran bermain pada state *s*
- ACTIONS(*s*) : Aksi-aksi yang bisa dilakukan pada *state s*
- RESULTS(*s, a*) : *Transition model,* mendefinisikan *state* yang dihasilkan dari sebuah aksi *a* pada *state s*
- Is-TERMINAL(*s*) :*Terminal test,* penentuan syarat *game* berakhir yaitu bernilai **True** jika *game* selesai, dan **False** jika sebaliknya. *State* dimana *game* selesai disebut *terminal state*.
- UTILITY(*s, p*): Sebuah fungsi *utility* (fungsi obyektif atau fungsi *payoff*), yang mendefinisikan nilai numerik untuk *player p* ketika *game* selesai di *terminal state s.* Contoh luaran *game* catur adalah menang, kalah, dan *draw* dengan nilai 1, 0, dan ½.

Deterministic Game dengan *Terminal Utilities*

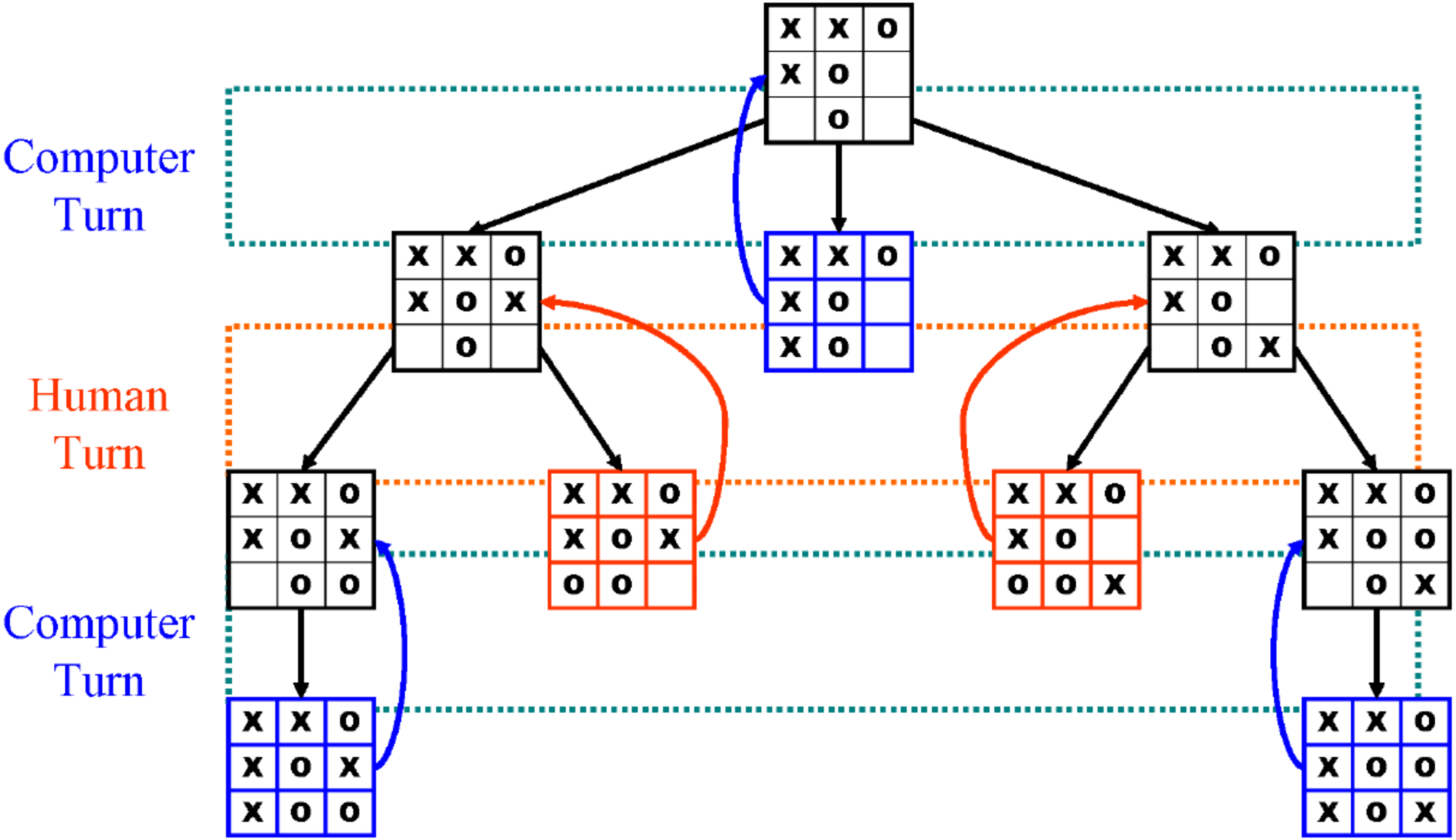
- Formulasi:
  - *States*: *S* (*State* awal s_0)
  - *Players*: *P = {1...N}*
  - *Actions*: *A* (tergantung *player/state*)
  - *Transition Function*: $S \times A \rightarrow S$
  - *Terminal Test*: $S \rightarrow \{true, false\}$
  - *Terminal Utilities*: $S \times P \rightarrow R$
- Solusi untuk setiap *player* adalah sebuah *policy*: $S \rightarrow A$

Search Space: Game Tree

- *Game tree*: semua kemungkinan langkah pemain berdasarkan langkah sebelumnya



Contoh *Game* Tic Tac Toe



Contoh *Game* Tic Tac Toe

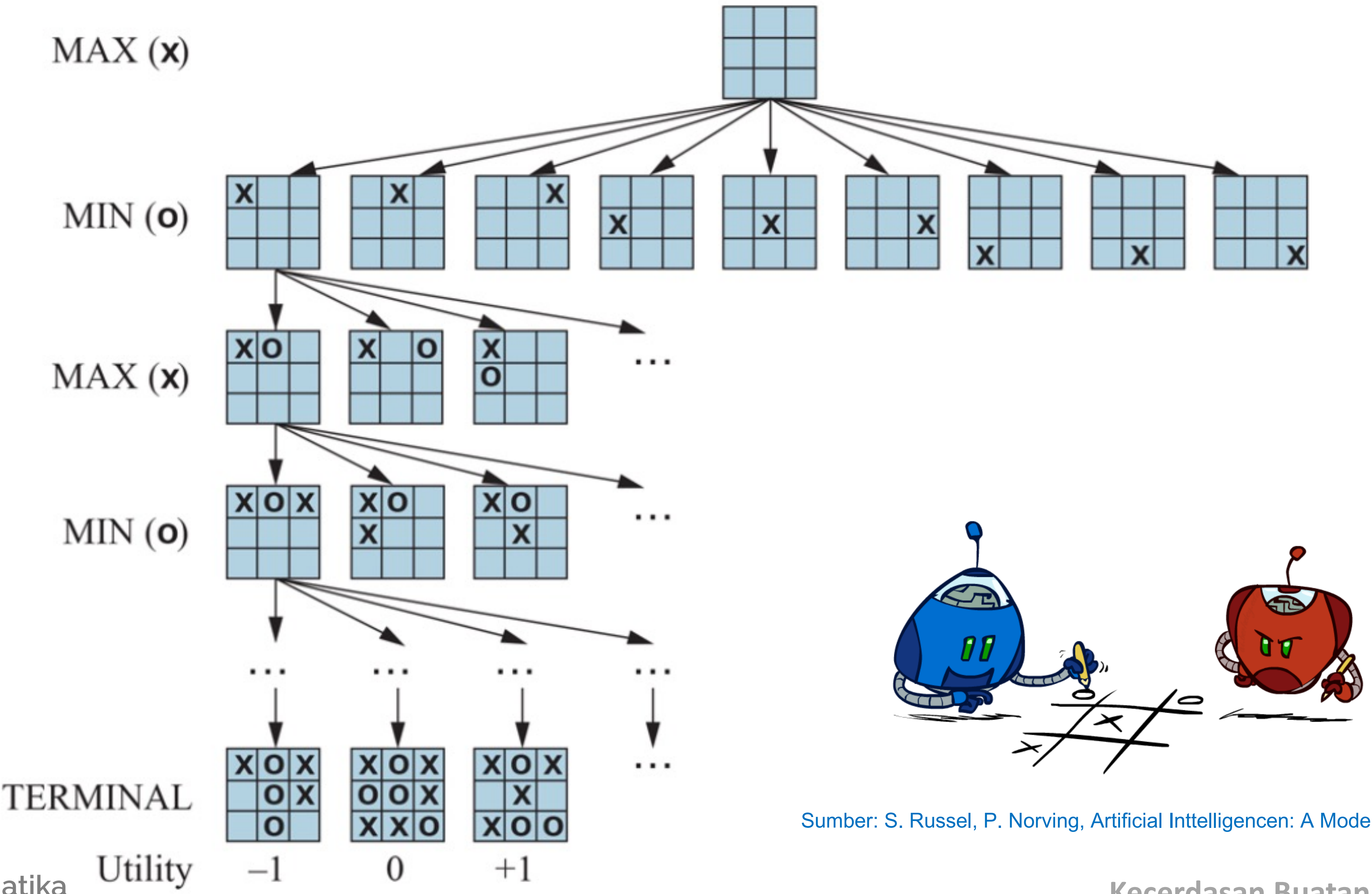
computer's turn

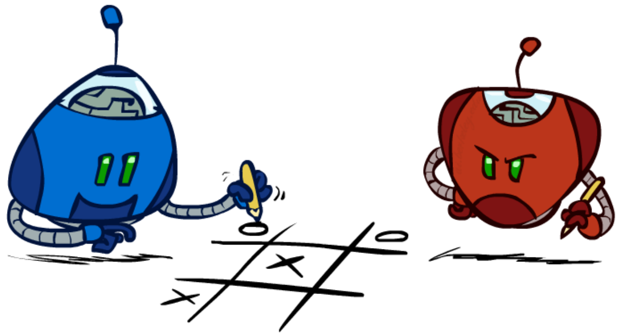
opponent's turn

computer's turn

opponent's turn

leaf nodes are evaluated





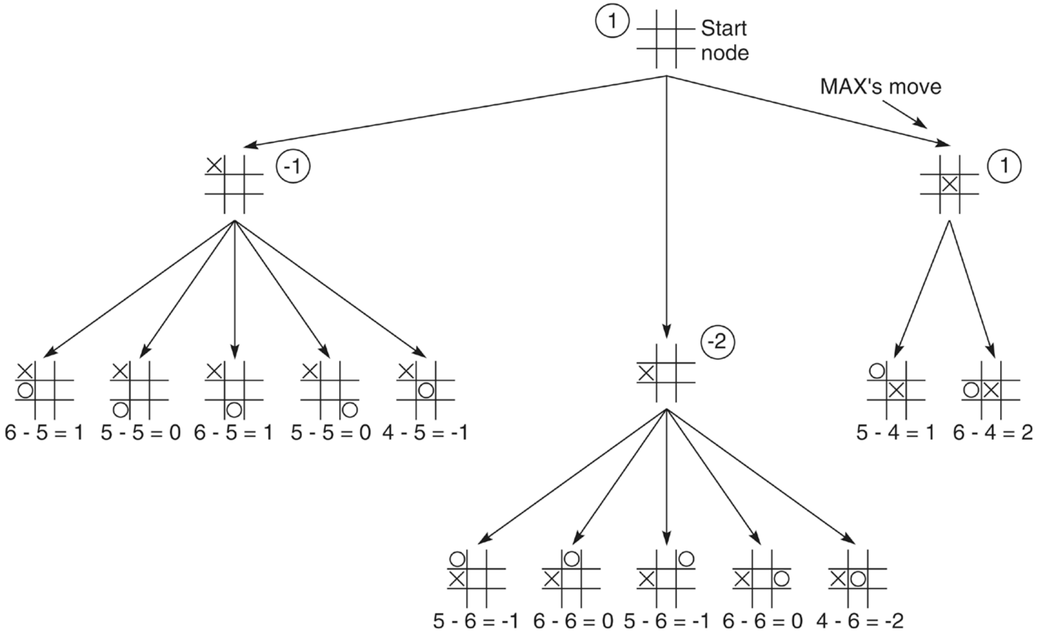
Sumber: S. Russel, P. Norving, Artificial Intteligencen: A Modern Approach

Evaluation Function

- Fungsi evaluasi static:
  - High positive = computer is winning
  - Zero = even game
  - High negative = opponent is winning
  - Contoh *game chess*: Memberikan bobot setiap kategori (queen=9, rook=5, knight/bishop=3, pawn=1)
- Fungsi evaluasi *score non-terminal*:
  - Fungsi heuristik
  - E(n) = M(n) – O(n) where
    - M(n) is the total of Computer (MAX) possible winning lines
    - O(n) is the total of Opponent’s (MIN) possible winning lines
    - E(n) is the total evaluation for state n
  - Weighted linear sum of features:

$$Eval(s) = w_1 f_1(s) + w_2 f_2(s) + \ldots + w_n f_n(s)$$

    - Contoh: $f_1(s)$ = (num white queens – num black queens)



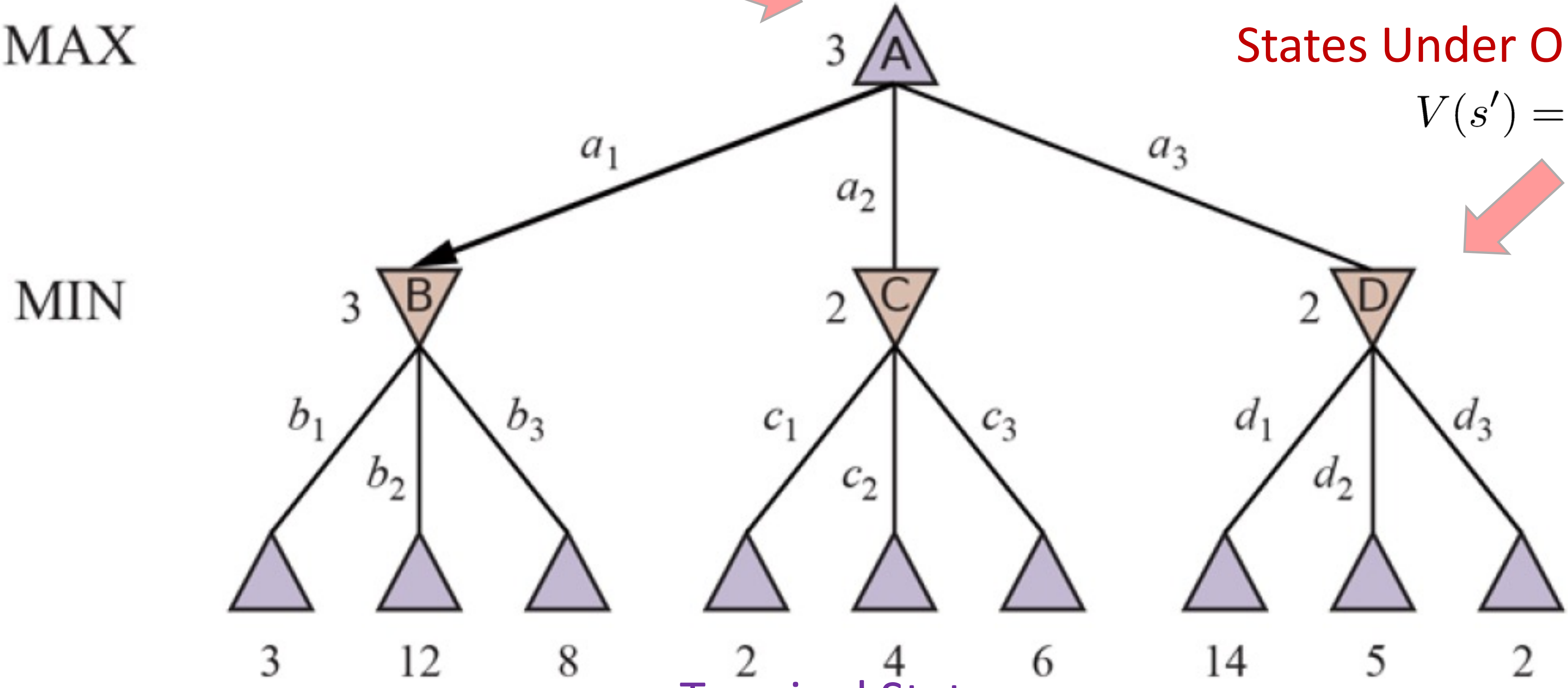
Minimax Values

States Under Agent's Control:

$$V(s) = \max_{s' \in \text{successors}(s)} V(s')$$

States Under Opponent's Control:

$$V(s') = \min_{s \in \text{successors}(s')} V(s)$$

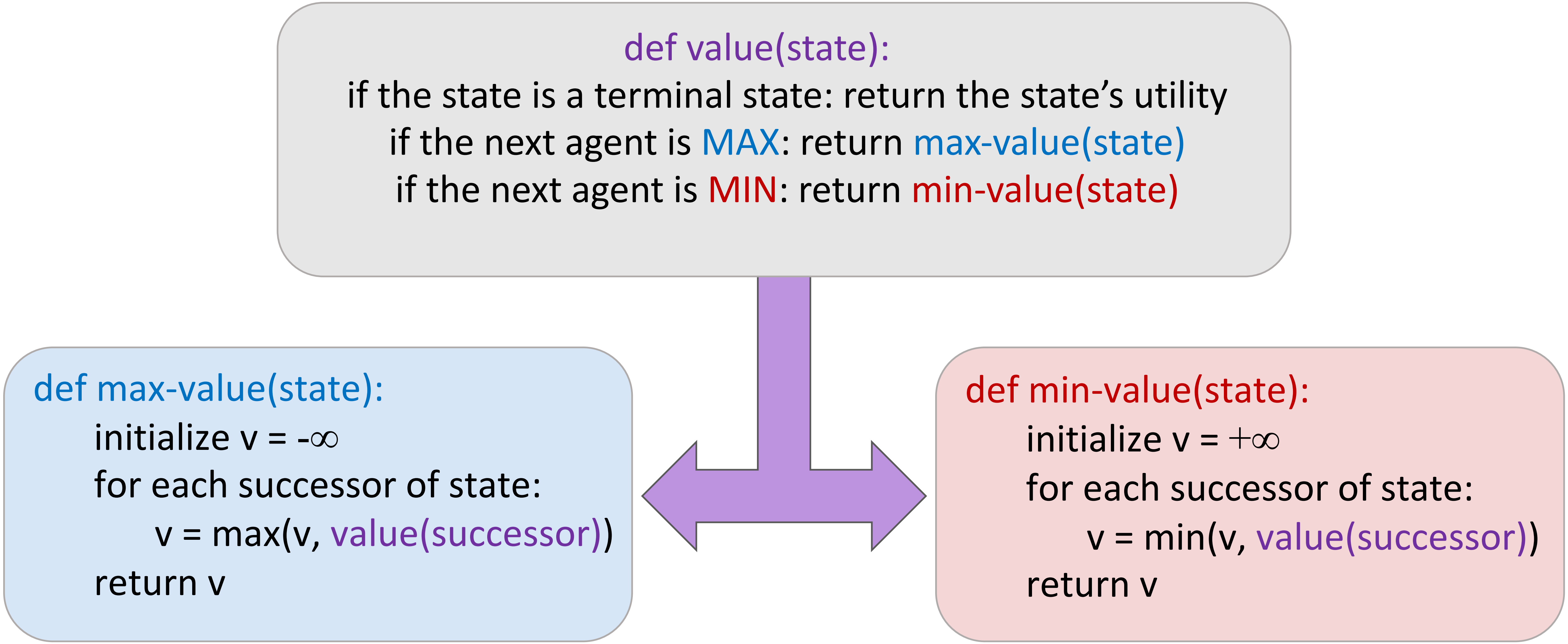


Terminal States:

$$V(s) = \text{known}$$

Sumber: S. Russel, P. Norving, Artificial Intteligencen: A Modern Approach

Algoritma Minimax



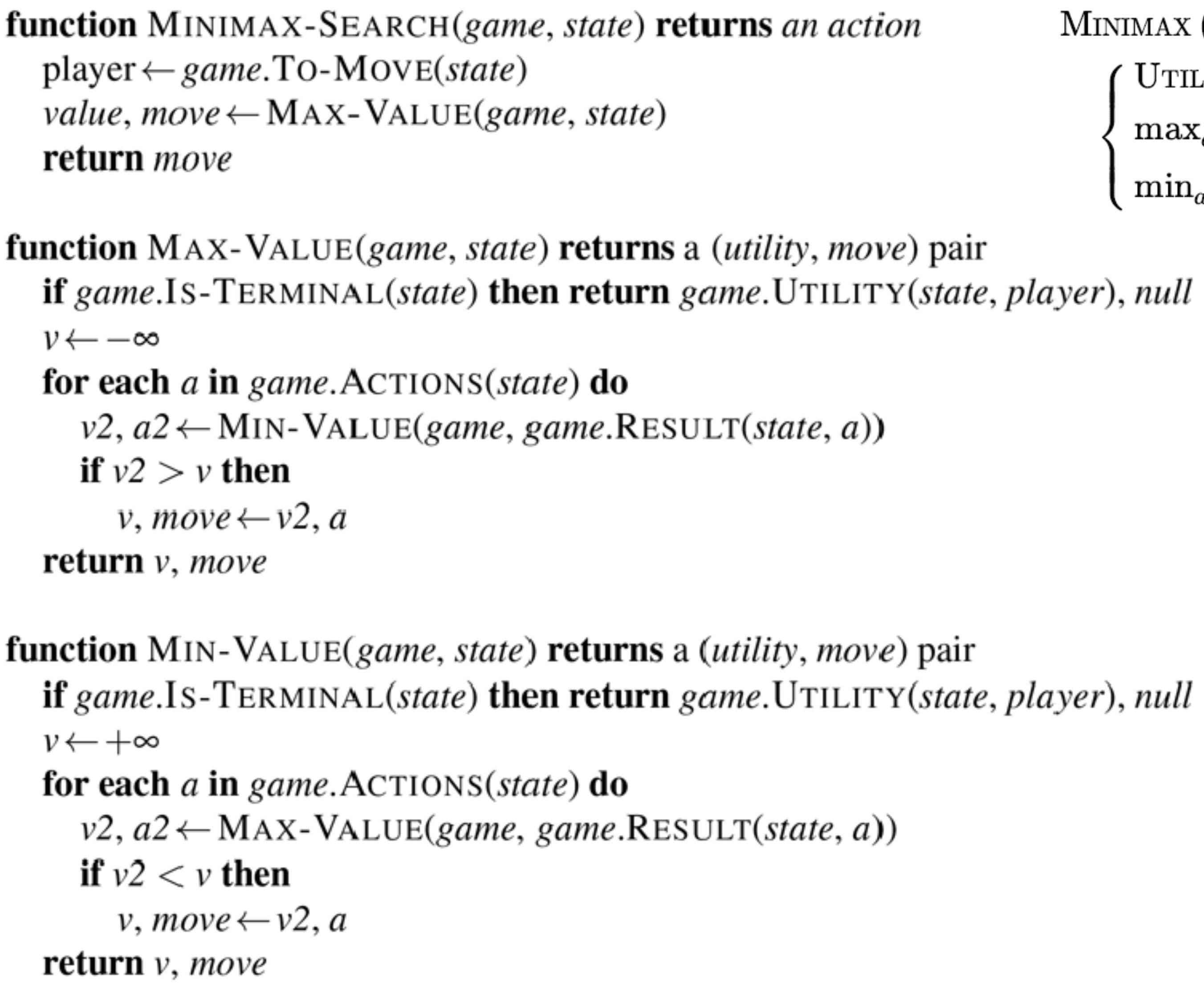
```
def value(state):
    if the state is a terminal state: return the state's utility
    if the next agent is MAX: return max-value(state)
    if the next agent is MIN: return min-value(state)
```

```
def max-value(state):
    initialize v = -∞
    for each successor of state:
        v = max(v, value(successor))
    return v
```

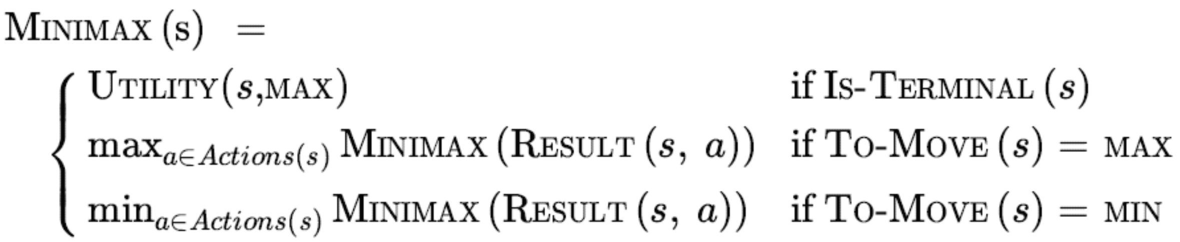
```
def min-value(state):
    initialize v = +∞
    for each successor of state:
        v = min(v, value(successor))
    return v
```

Sumber: Adversarial Search by Dan Klein and Pieter Abbeel for CS188 Intro to AI at UC Berkeley

Algoritma Minimax



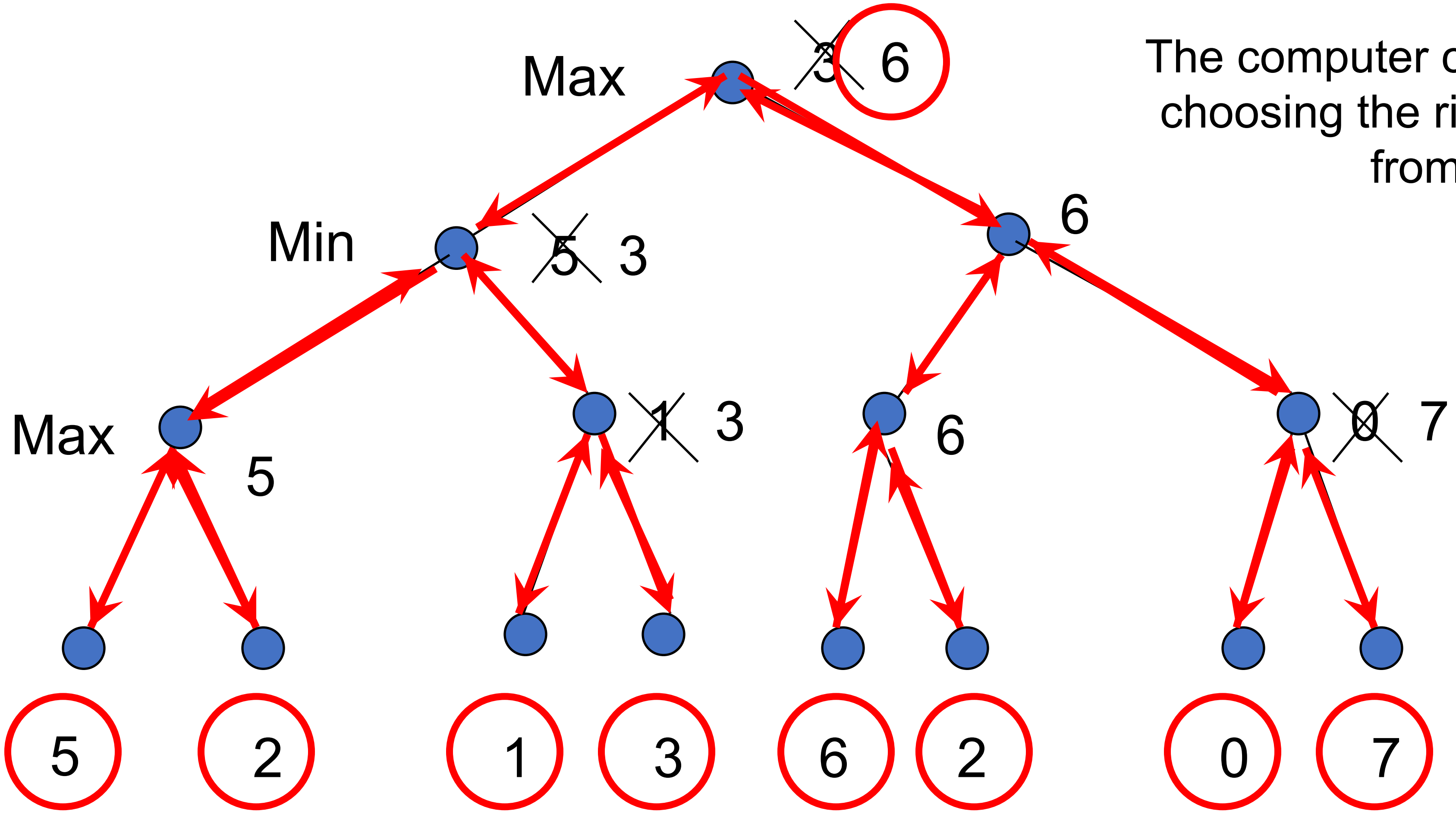
```
function MINIMAX-SEARCH(game, state) returns an action
    player ← game.TO-MOVE(state)
    value, move ← MAX-VALUE(game, state)
    return move

function MAX-VALUE(game, state) returns a (utility, move) pair
    if game.IS-TERMINAL(state) then return game.UTILITY(state, player), null
    $v \leftarrow -\infty$
    for each a in game.ACTIONS(state) do
        v2, a2 ← MIN-VALUE(game, game.RESULT(state, a))
        if $v2 > v$ then
            v, move ← v2, a
    return v, move

function MIN-VALUE(game, state) returns a (utility, move) pair
    if game.IS-TERMINAL(state) then return game.UTILITY(state, player), null
    $v \leftarrow +\infty$
    for each a in game.ACTIONS(state) do
        v2, a2 ← MAX-VALUE(game, game.RESULT(state, a))
        if $v2 < v$ then
            v, move ← v2, a
    return v, move
```



$$\textsc{Minimax}\,(\mathrm{s}) \;=\; \begin{cases} \textsc{Utility}(s,\textsc{max}) & \text{if } \textsc{Is-Terminal}\,(s) \\ \max_{a\in Actions(s)} \textsc{Minimax}\,(\textsc{Result}\,(s,\ a)) & \text{if } \textsc{To-Move}\,(s) \;=\; \textsc{max} \\ \min_{a\in Actions(s)} \textsc{Minimax}\,(\textsc{Result}\,(s,\ a)) & \text{if } \textsc{To-Move}\,(s) \;=\; \textsc{min} \end{cases}$$

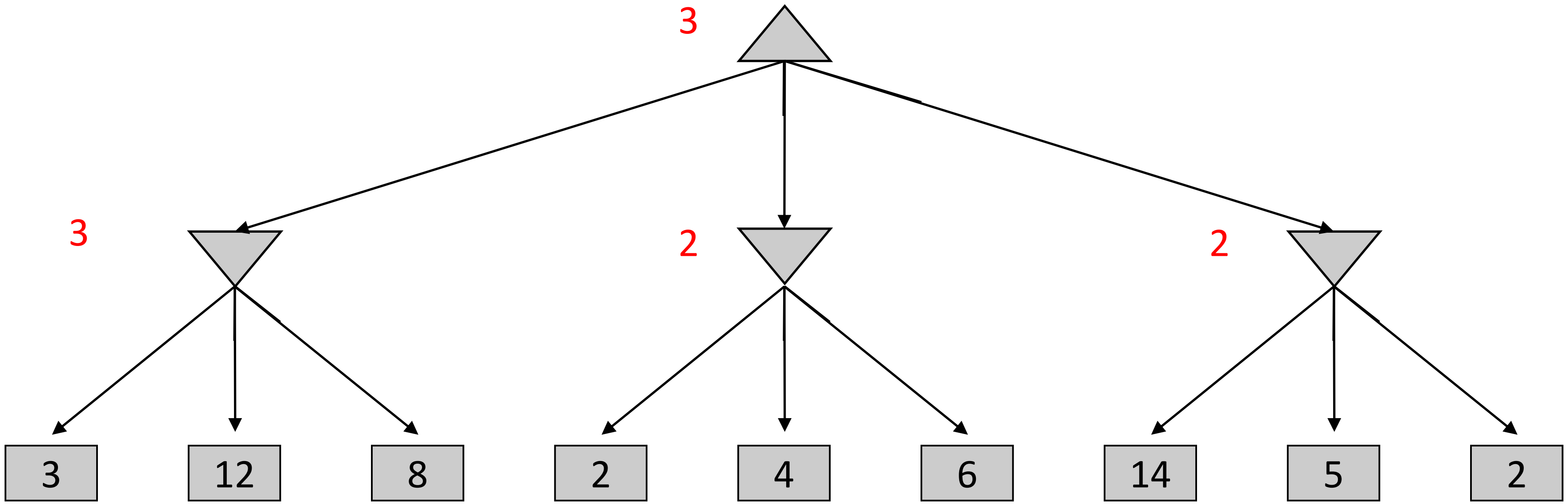
Sumber: S. Russel, P. Norving, Artificial Inttelligencen: A Modern Approach

Contoh: Minimax



The computer can obtain 6 by choosing the right hand edge from the first node.

Contoh: Minimax

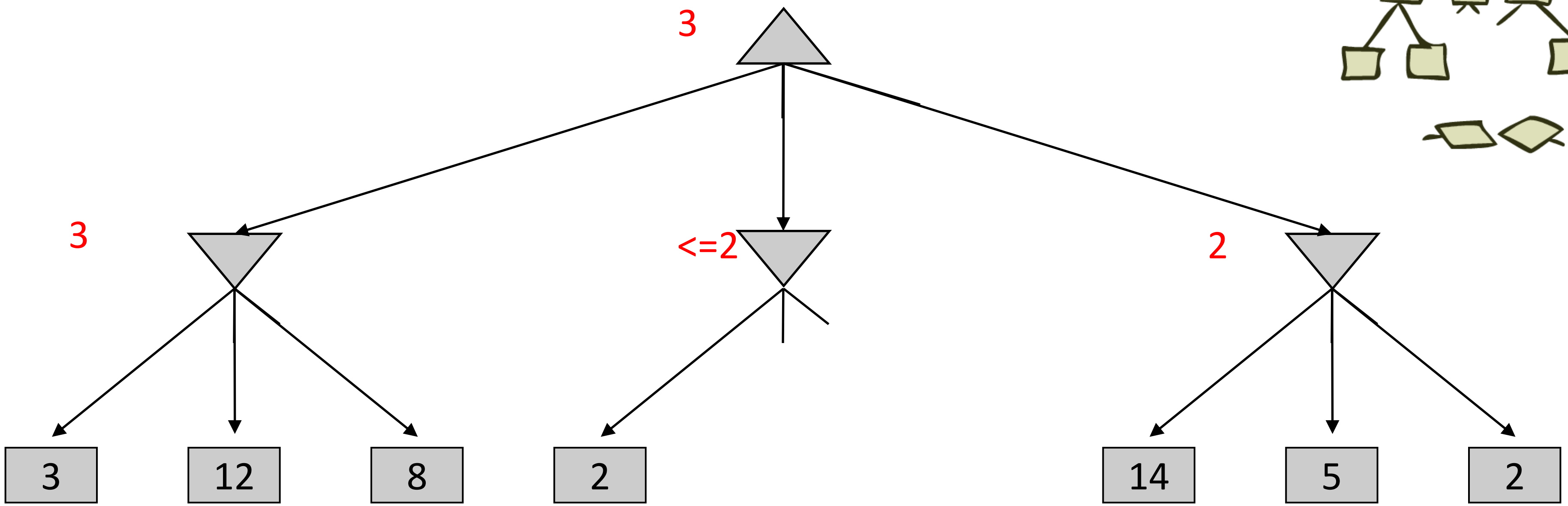


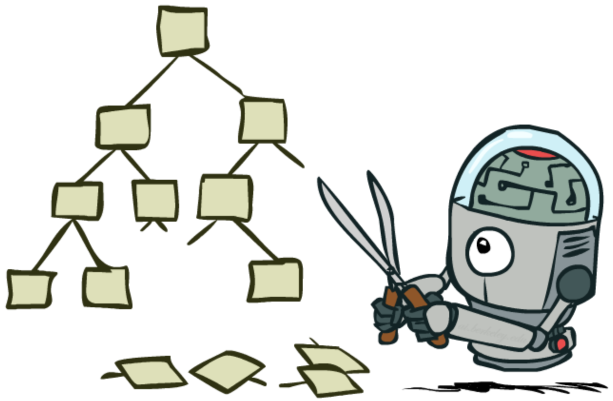
Sumber: Adversarial Search by Dan Klein and Pieter Abbeel for CS188 Intro to AI at UC Berkeley

Game Tree Pruning

Pruning = pemotongan cabang pada game tree

Mengurangi waktu pembuatan dan penelusuran tree

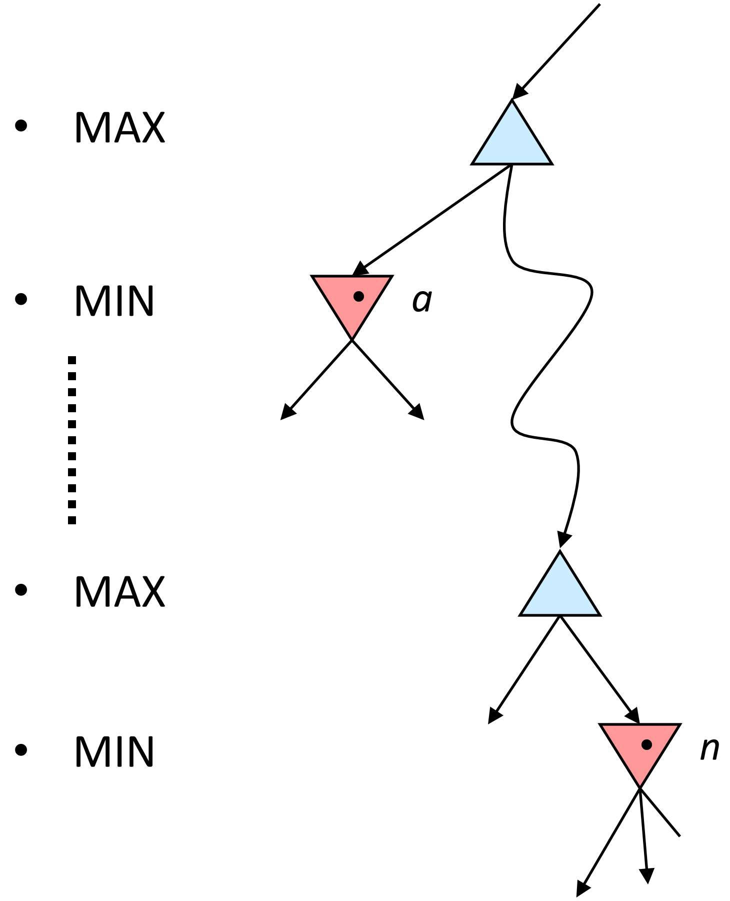




Sumber: Adversarial Search by Dan Klein and Pieter Abbeel for CS188 Intro to AI at UC Berkeley

Alpha Beta Pruning

- General configuration (MIN version)
  - We’re computing the MIN-VALUE at some node *n*
  - We’re looping over *n*’s children
  - *n*’s estimate of the childrens’ min is dropping
  - Who cares about *n*’s value? MAX
  - Let *a* be the best value that MAX can get at any choice point along the current path from the root
  - If *n* becomes worse than *a*, MAX will avoid it, so we can stop considering *n*’s other children (it’s already bad enough that it won’t be played)
- MAX version is symmetric



- MAX
- MIN
- MAX
- MIN

Sumber: Adversarial Search by Dan Klein and Pieter Abbeel for CS188 Intro to AI at UC Berkeley

Alpha Beta Pruning

α: MAX’s best option on path to root

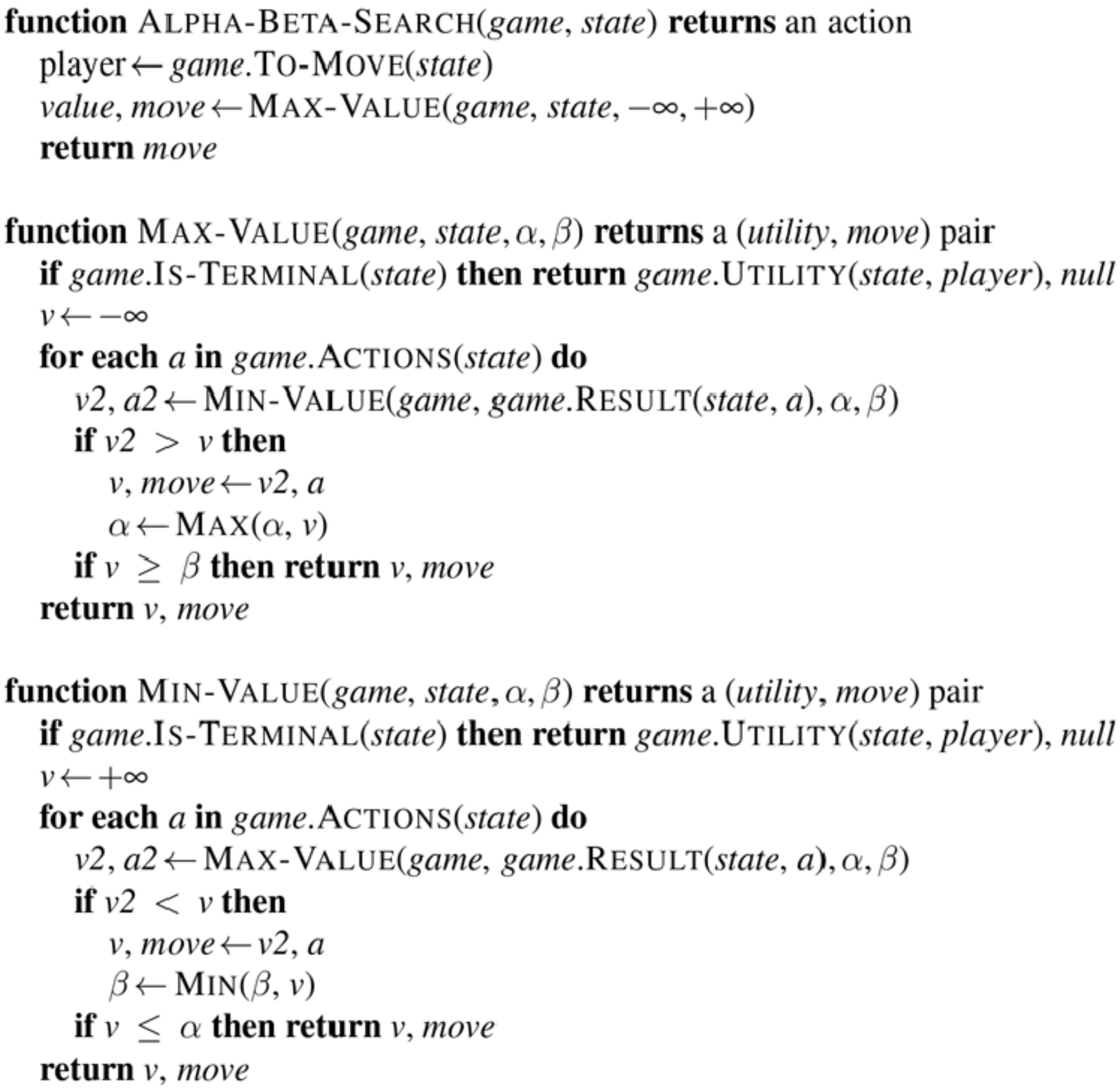
β: MIN’s best option on path to root

```
def max-value(state, $\alpha$, $\beta$):
    initialize v = $-\infty$
    for each successor of state:
        v = max(v, value(successor, $\alpha$, $\beta$))
        if $v \geq \beta$ return v
        $\alpha$ = max($\alpha$, v)
    return v
```

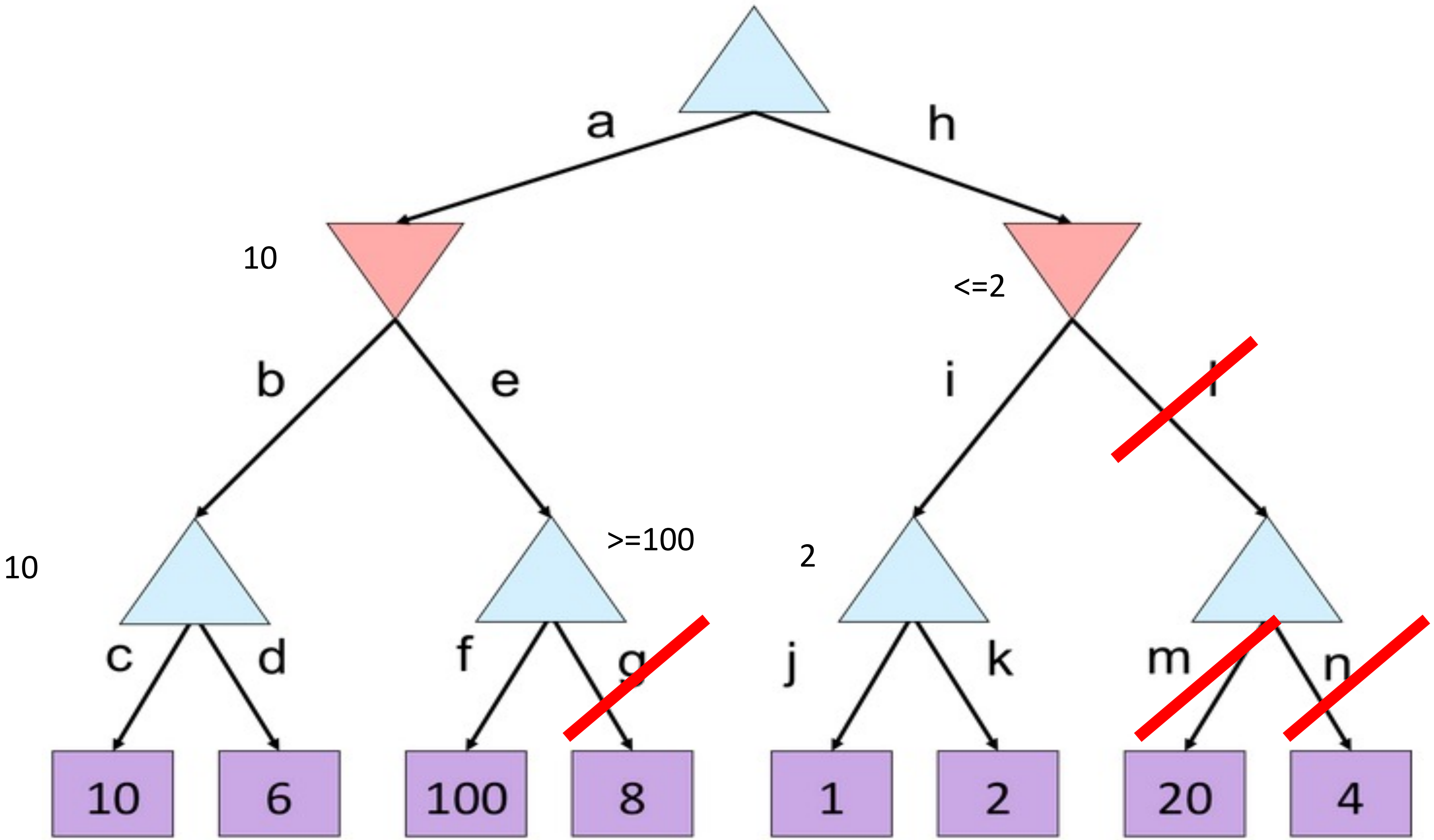
```
def min-value(state , $\alpha$, $\beta$):
    initialize v = $+\infty$
    for each successor of state:
        v = min(v, value(successor, $\alpha$, $\beta$))
        if $v \leq \alpha$ return v
        $\beta$ = min($\beta$, v)
    return v
```

Sumber: Adversarial Search by Dan Klein and Pieter Abbeel for CS188 Intro to AI at UC Berkeley

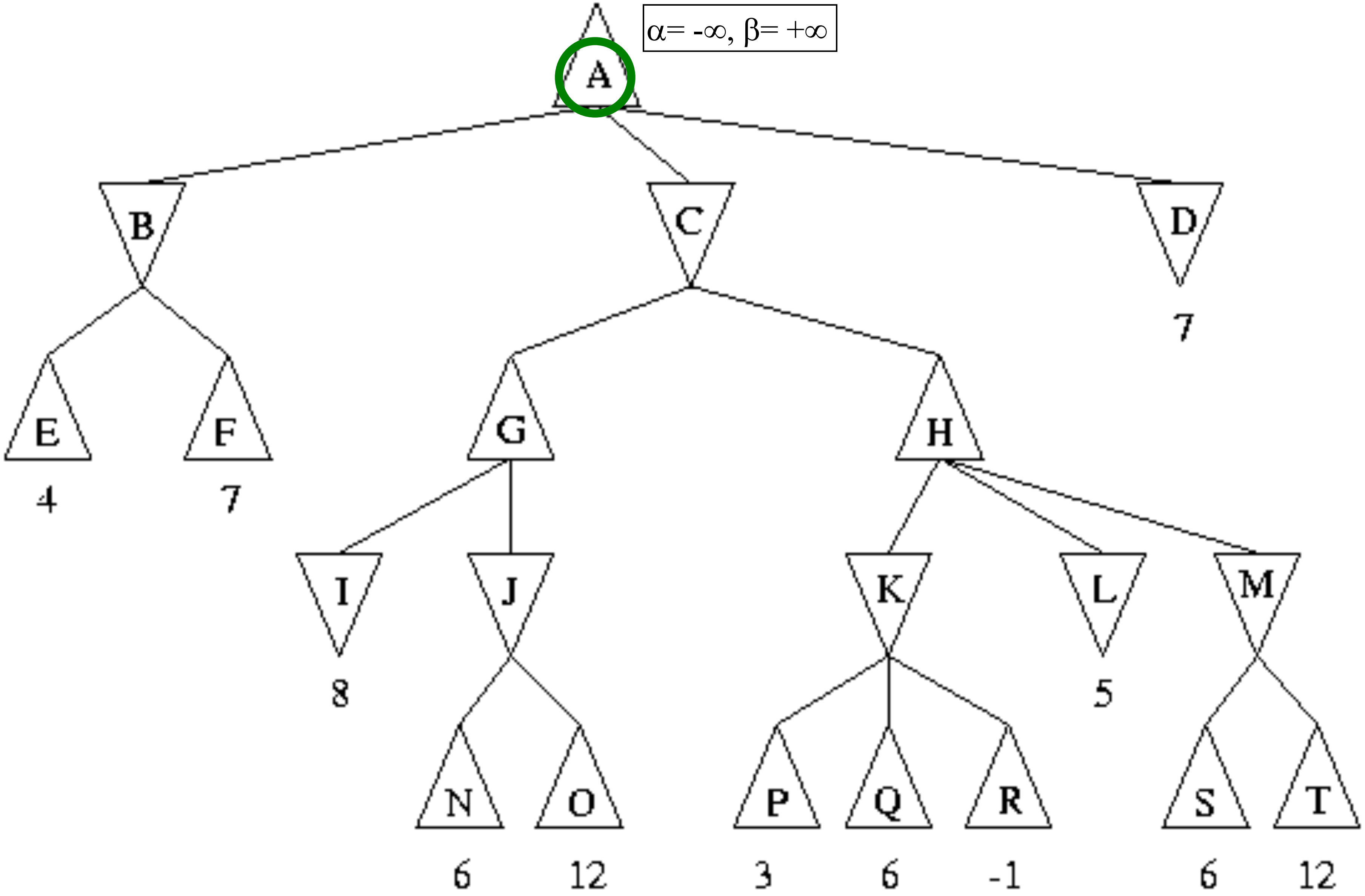
Algoritma Alpha Beta



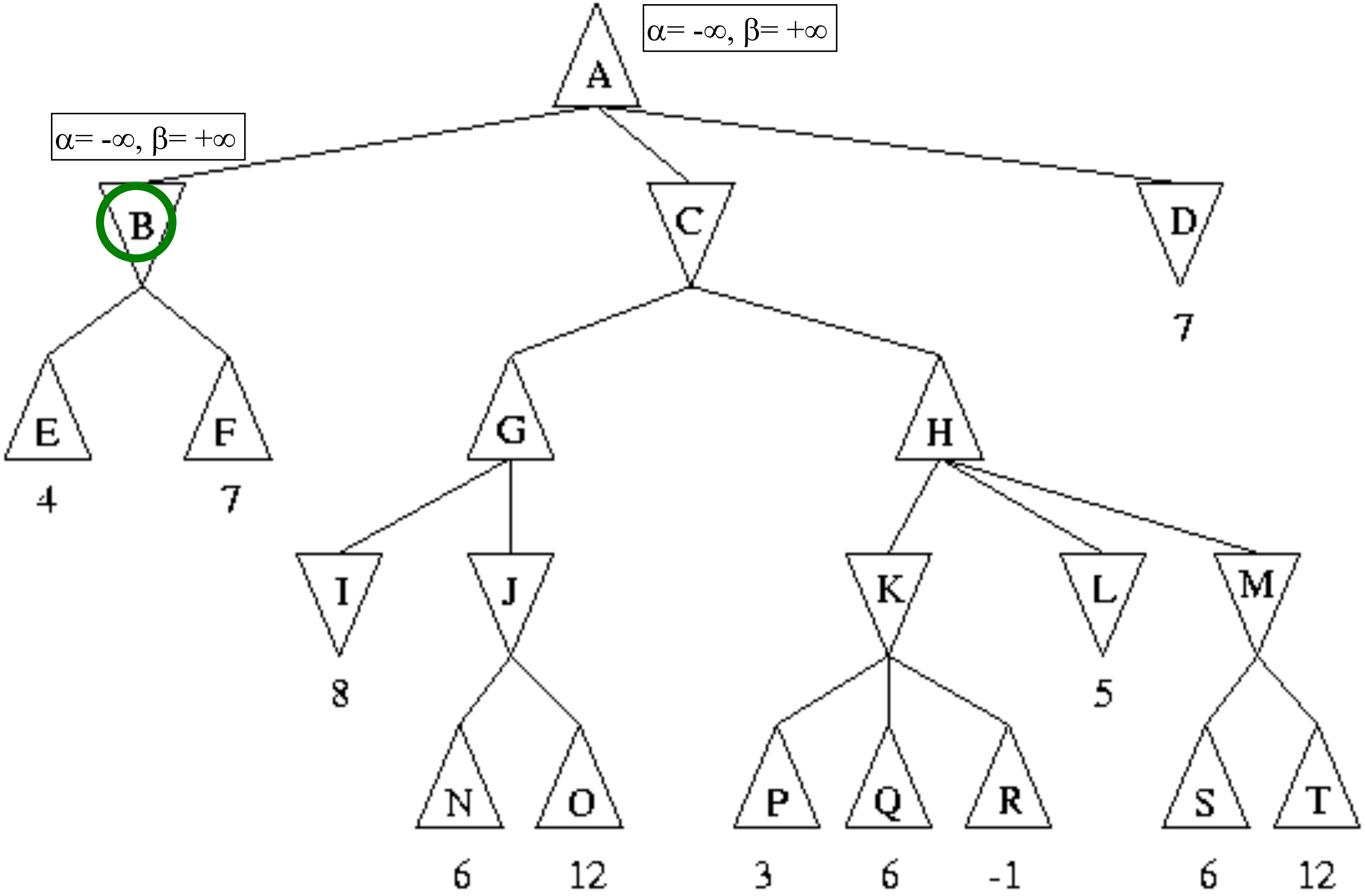
```
function ALPHA-BETA-SEARCH(game, state) returns an action
  player ← game.TO-MOVE(state)
  value, move ← MAX-VALUE(game, state, $-\infty$, $+\infty$)
  return move

function MAX-VALUE(game, state, $\alpha$, $\beta$) returns a (utility, move) pair
  if game.IS-TERMINAL(state) then return game.UTILITY(state, player), null
  $v \leftarrow -\infty$
  for each a in game.ACTIONS(state) do
    v2, a2 ← MIN-VALUE(game, game.RESULT(state, a), $\alpha$, $\beta$)
    if $v2 > v$ then
      v, move ← v2, a
      $\alpha \leftarrow$ MAX($\alpha$, v)
    if $v \geq \beta$ then return v, move
  return v, move

function MIN-VALUE(game, state, $\alpha$, $\beta$) returns a (utility, move) pair
  if game.IS-TERMINAL(state) then return game.UTILITY(state, player), null
  $v \leftarrow +\infty$
  for each a in game.ACTIONS(state) do
    v2, a2 ← MAX-VALUE(game, game.RESULT(state, a), $\alpha$, $\beta$)
    if $v2 < v$ then
      v, move ← v2, a
      $\beta \leftarrow$ MIN($\beta$, v)
    if $v \leq \alpha$ then return v, move
  return v, move
```

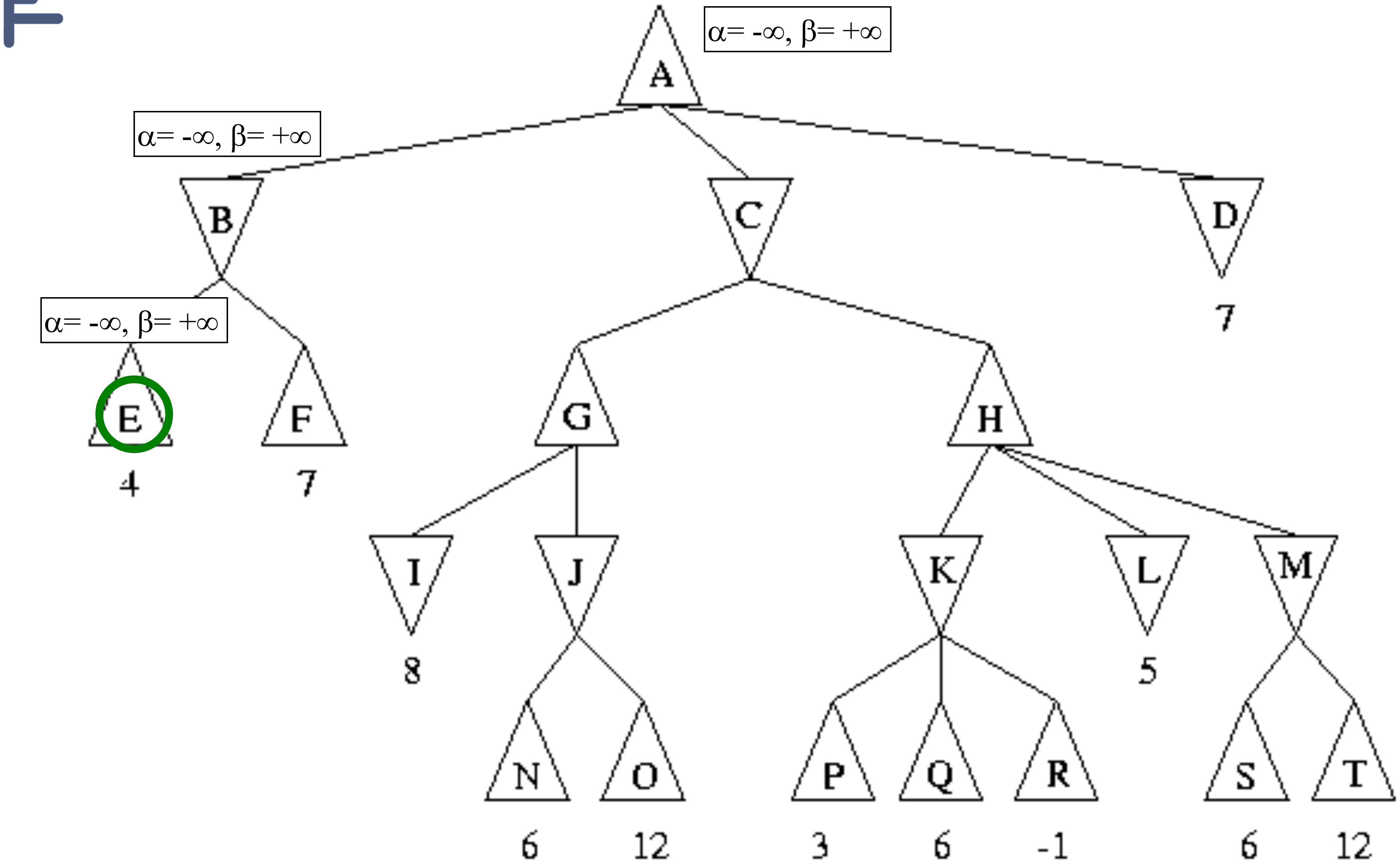


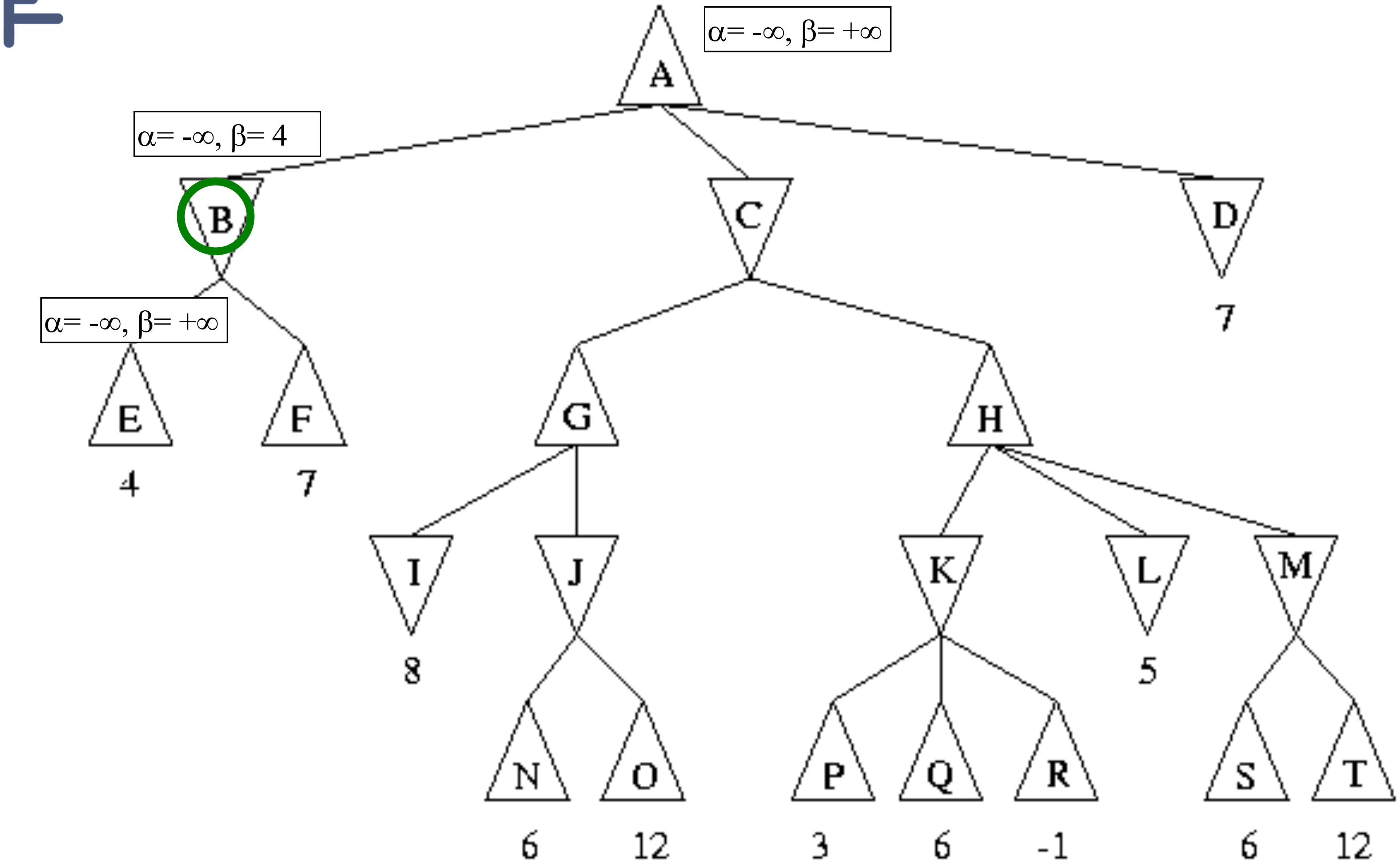
Sumber: Adversarial Search by Dan Klein and Pieter Abbeel for CS188 Intro to AI at UC Berkeley

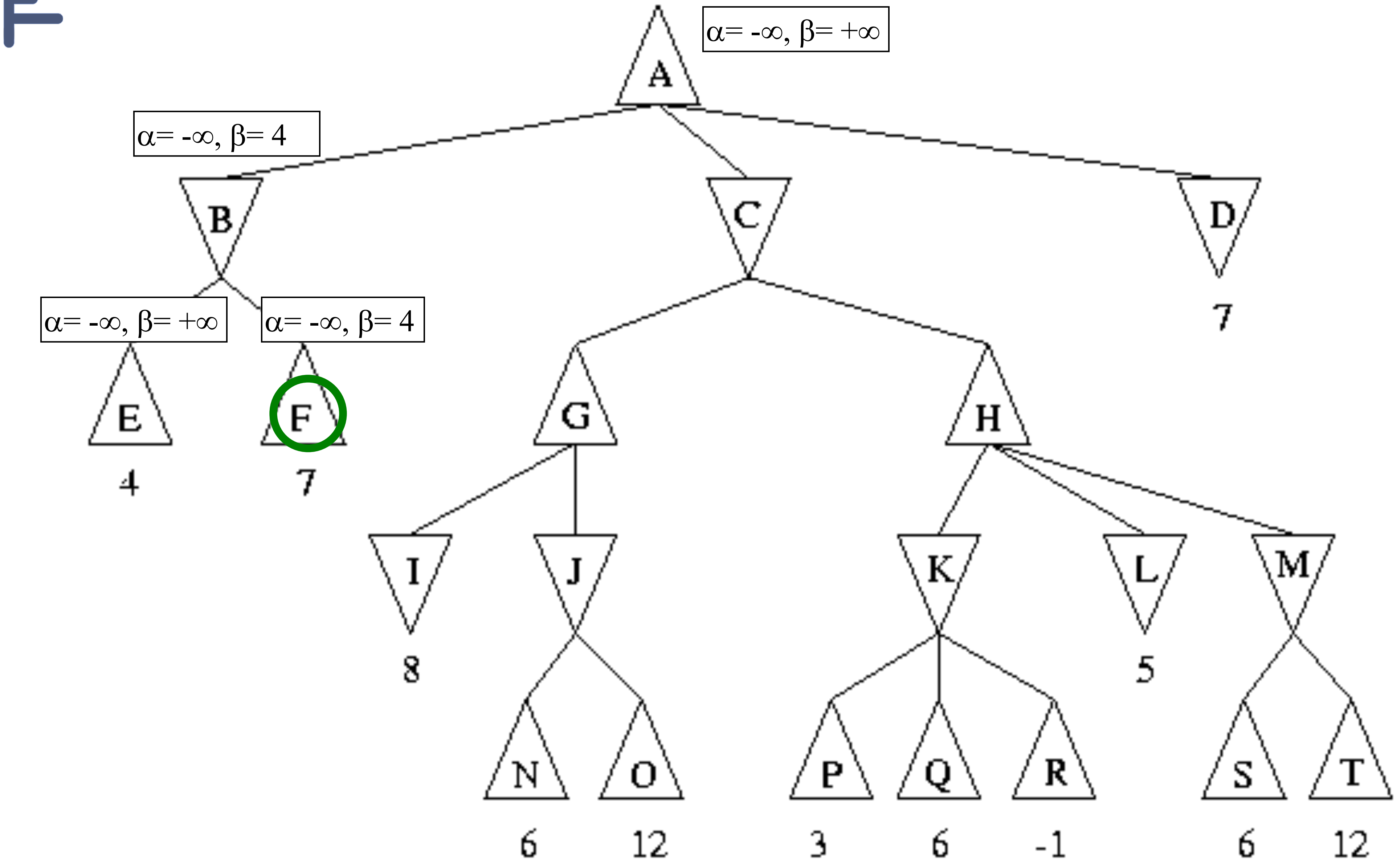


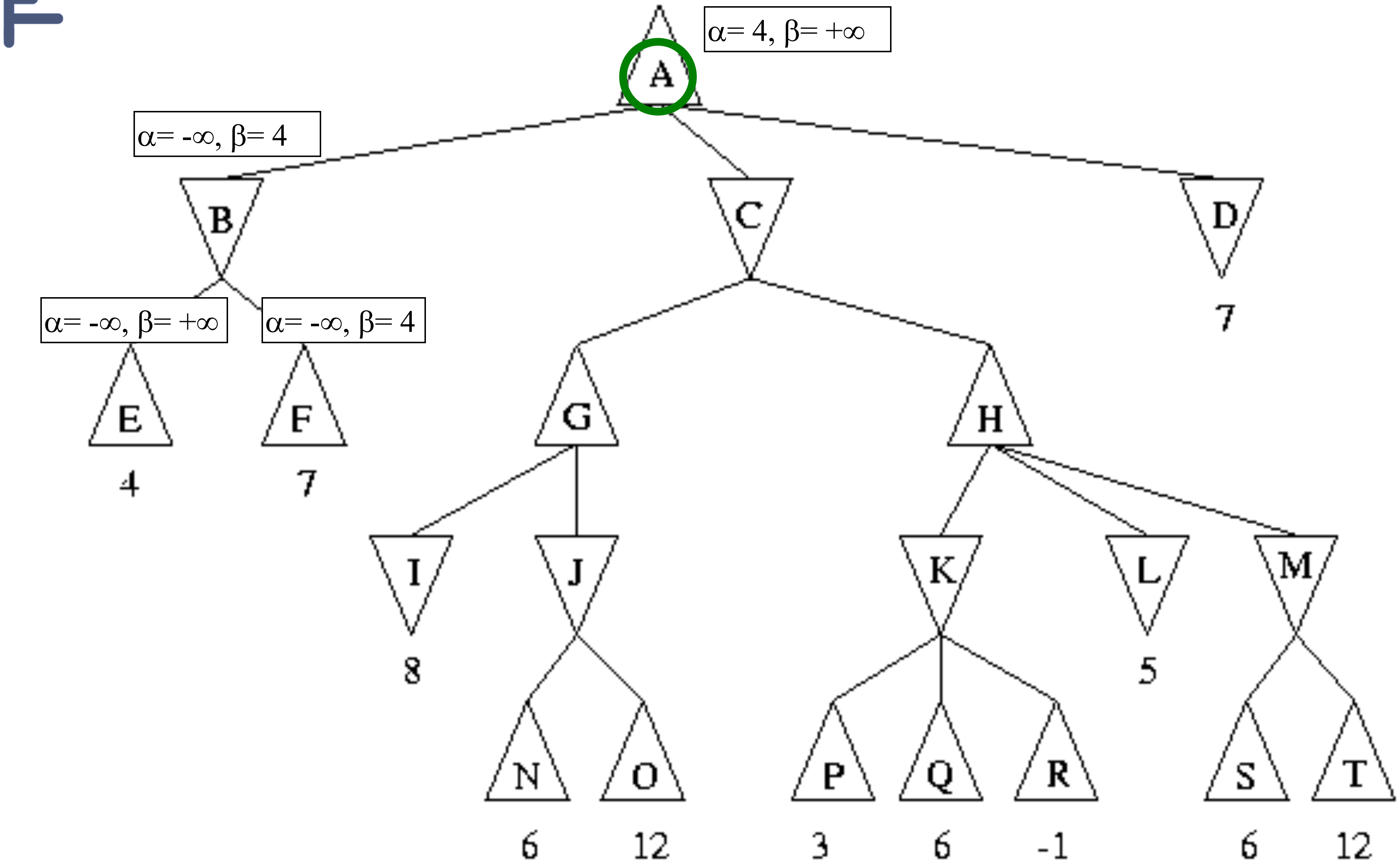
α= -∞, β= +∞
A
B
C
D
7
E
F
4
7
G
H
I
J
K
L
M
8
5
N
O
P
Q
R
S
T
6
12
3
6
-1
6
12

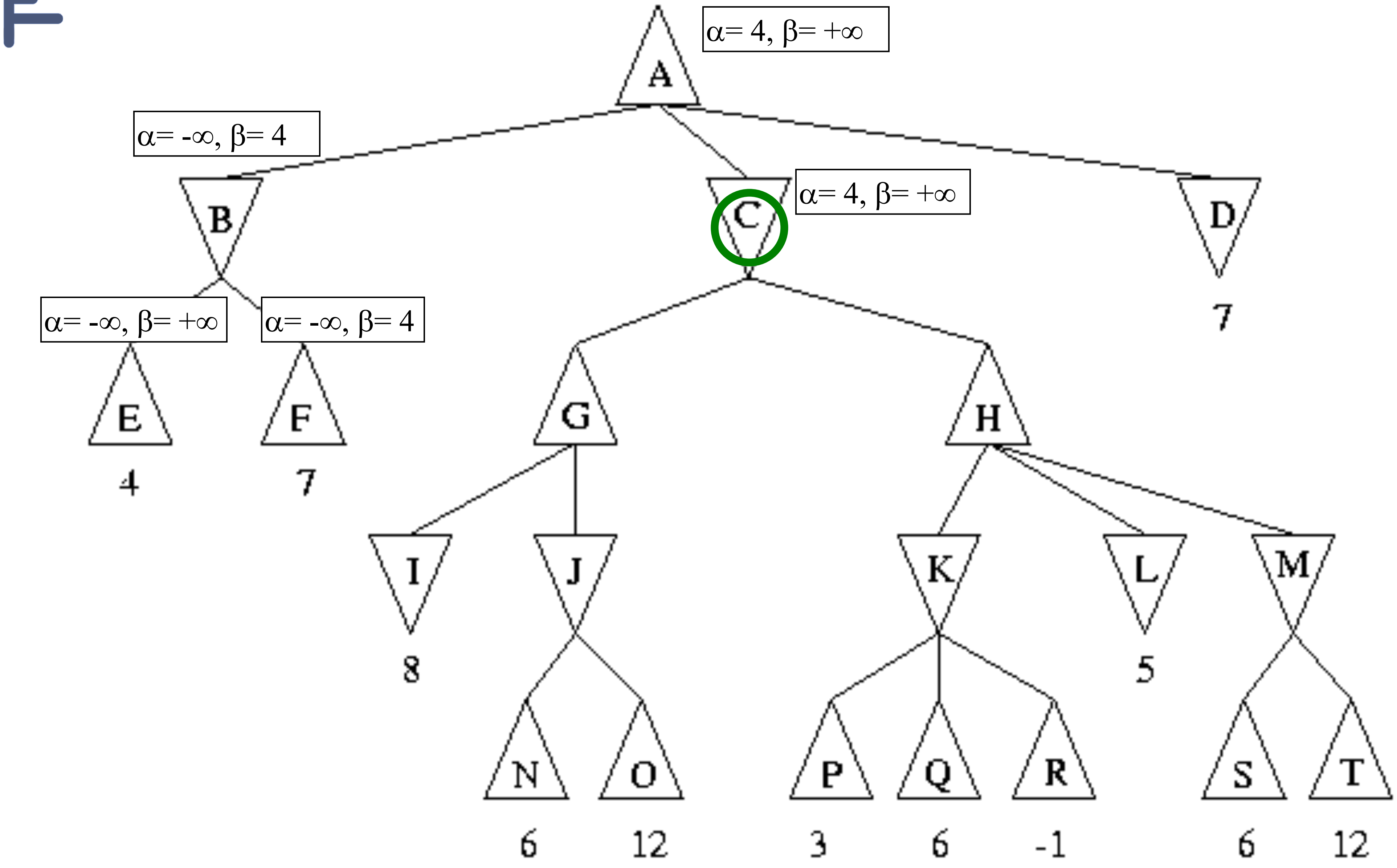

α= -∞, β= +∞
A
α= -∞, β= +∞
B
C
D
7
E
F
4
7
G
H
I
J
K
L
M
8
5
N
O
P
Q
R
S
T
6
12
3
6
-1
6
12

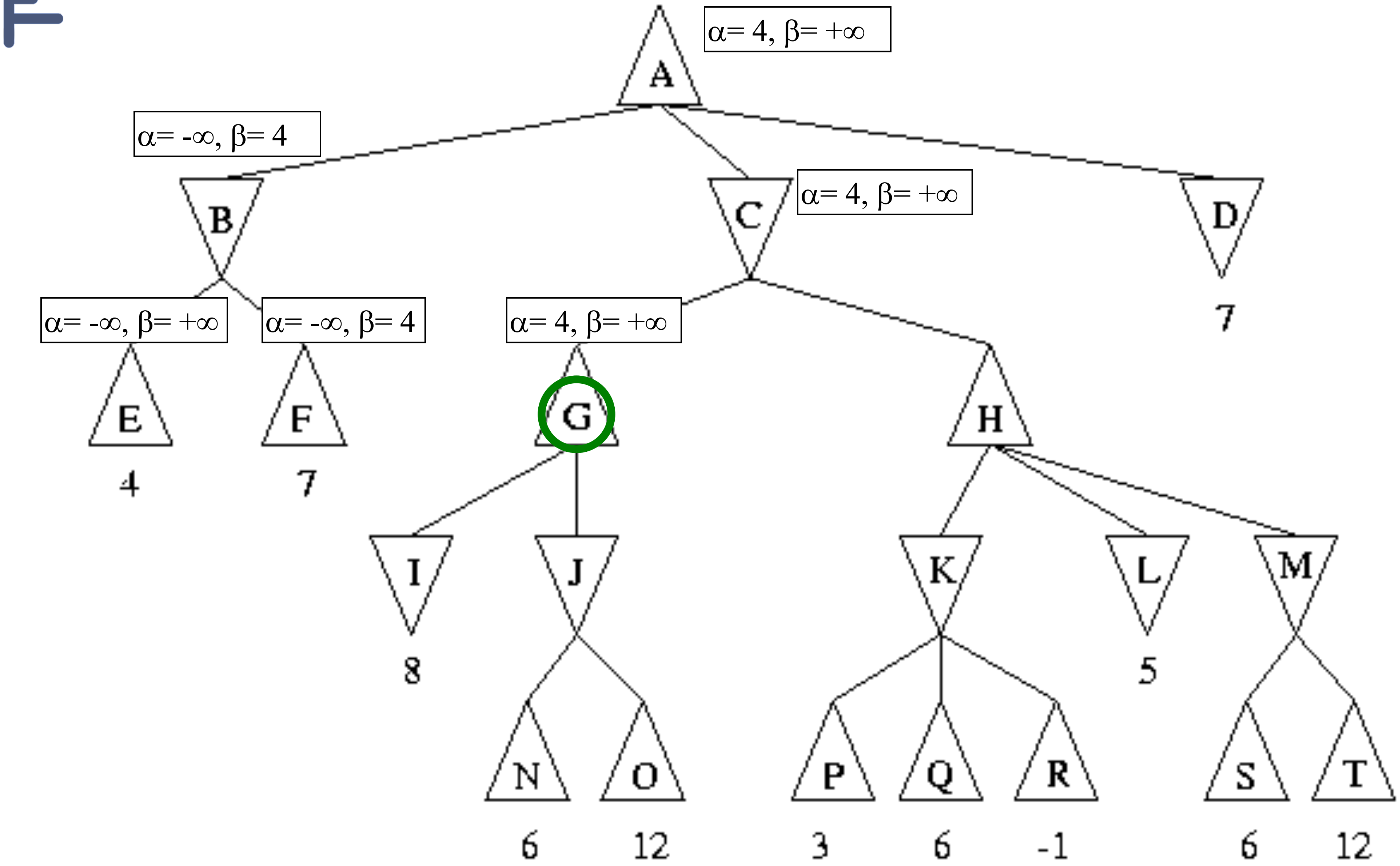

α= -∞, β= +∞
A
α= -∞, β= +∞
B
C
D
7
α= -∞, β= +∞
E
4
F
7
G
H
I
8
J
K
L
5
M
N
6
O
12
P
3
Q
6
R
-1
S
6
T
12

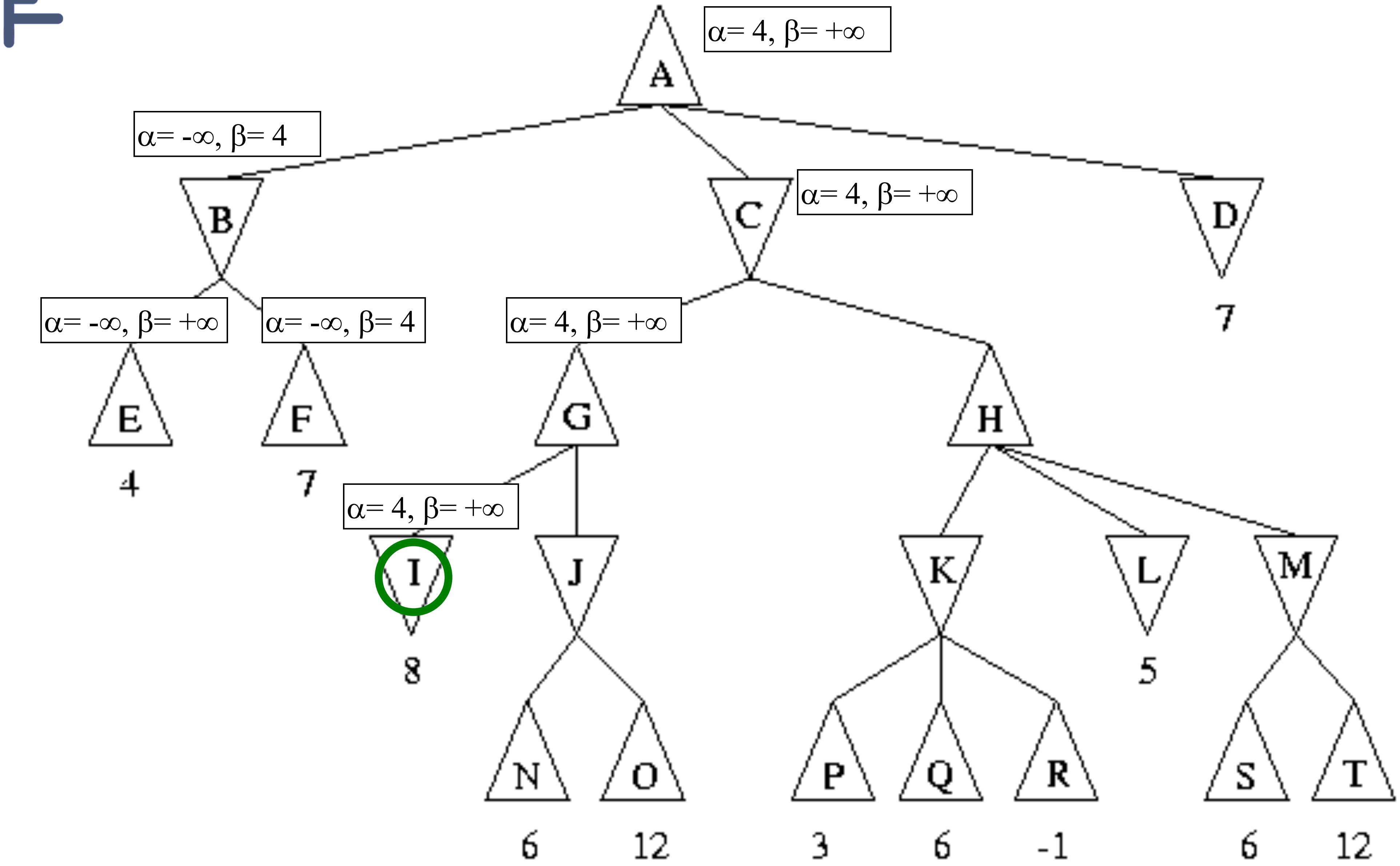

α= -∞, β= +∞
A
α= -∞, β= 4
B
C
D
7
α= -∞, β= +∞
E
F
4
7
G
H
I
J
K
L
M
8
5
N
O
P
Q
R
S
T
6
12
3
6
-1
6
12

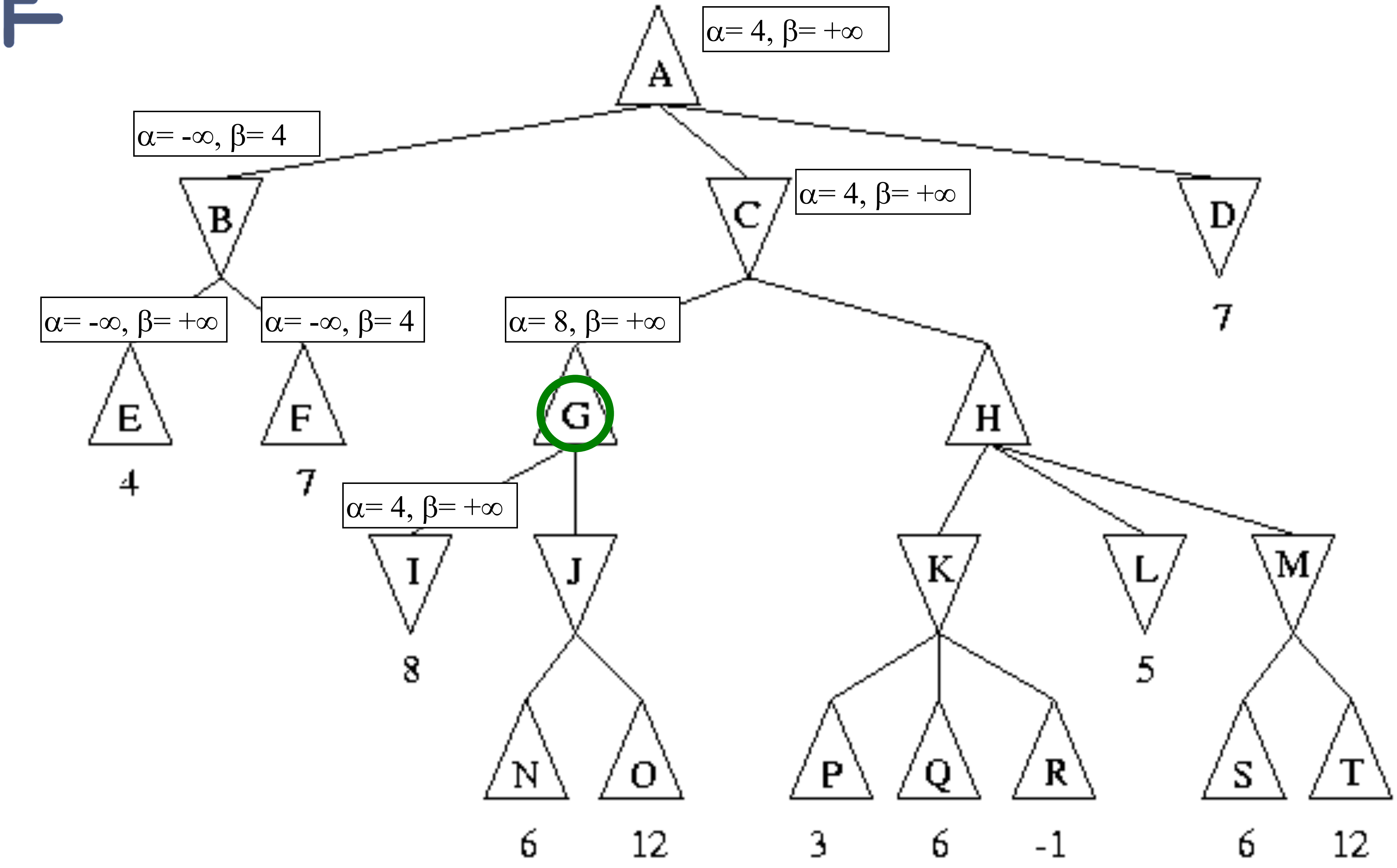

α= -∞, β= +∞
A
α= -∞, β= 4
B
C
D
7
α= -∞, β= +∞
α= -∞, β= 4
E
F
4
7
G
H
I
J
K
L
M
8
5
N
O
P
Q
R
S
T
6
12
3
6
-1
6
12

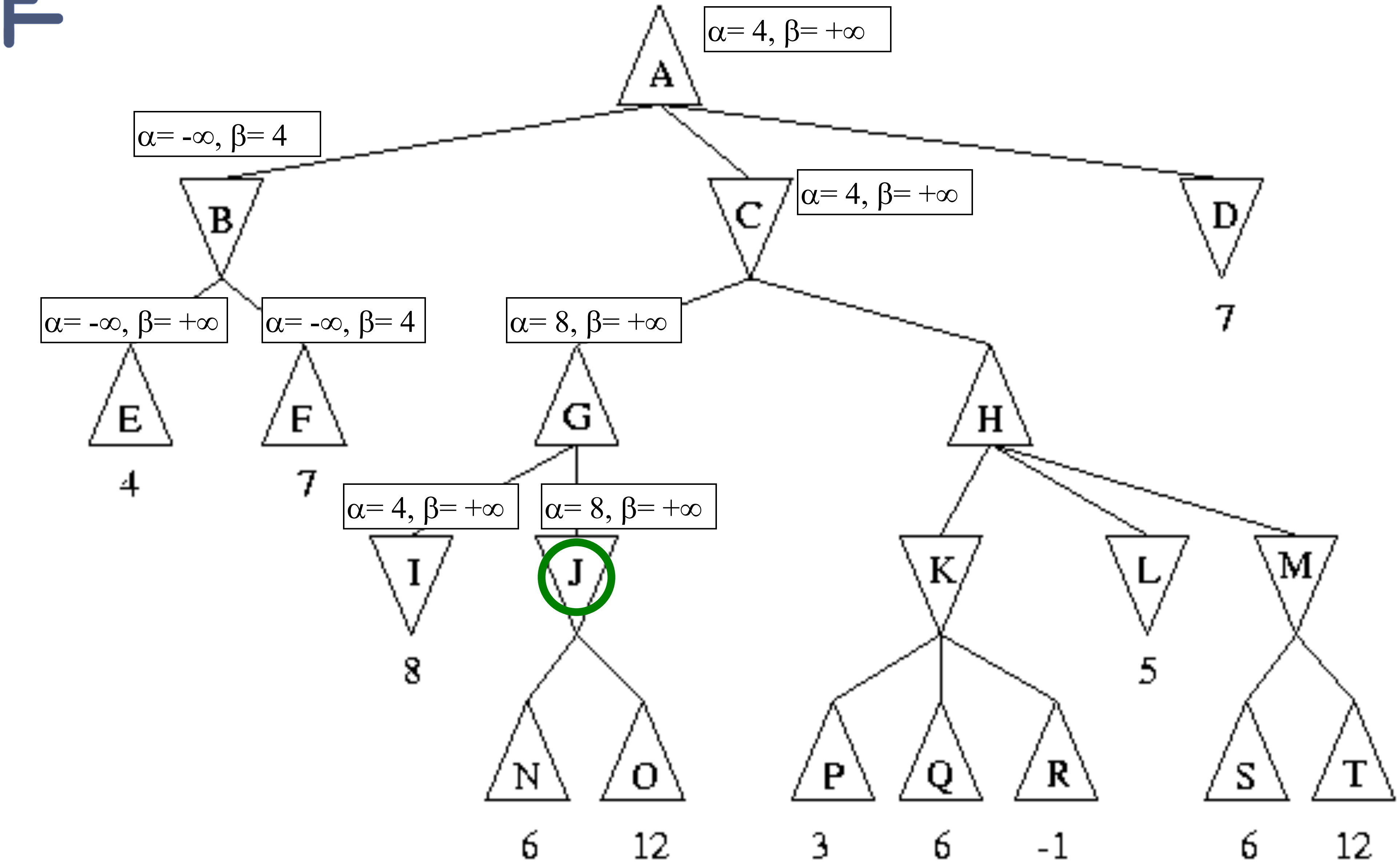

α= 4, β= +∞
A
α= -∞, β= 4
B
C
D
7
α= -∞, β= +∞
α= -∞, β= 4
E
F
4
7
G
H
I
J
K
L
M
8
5
N
O
P
Q
R
S
T
6
12
3
6
-1
6
12

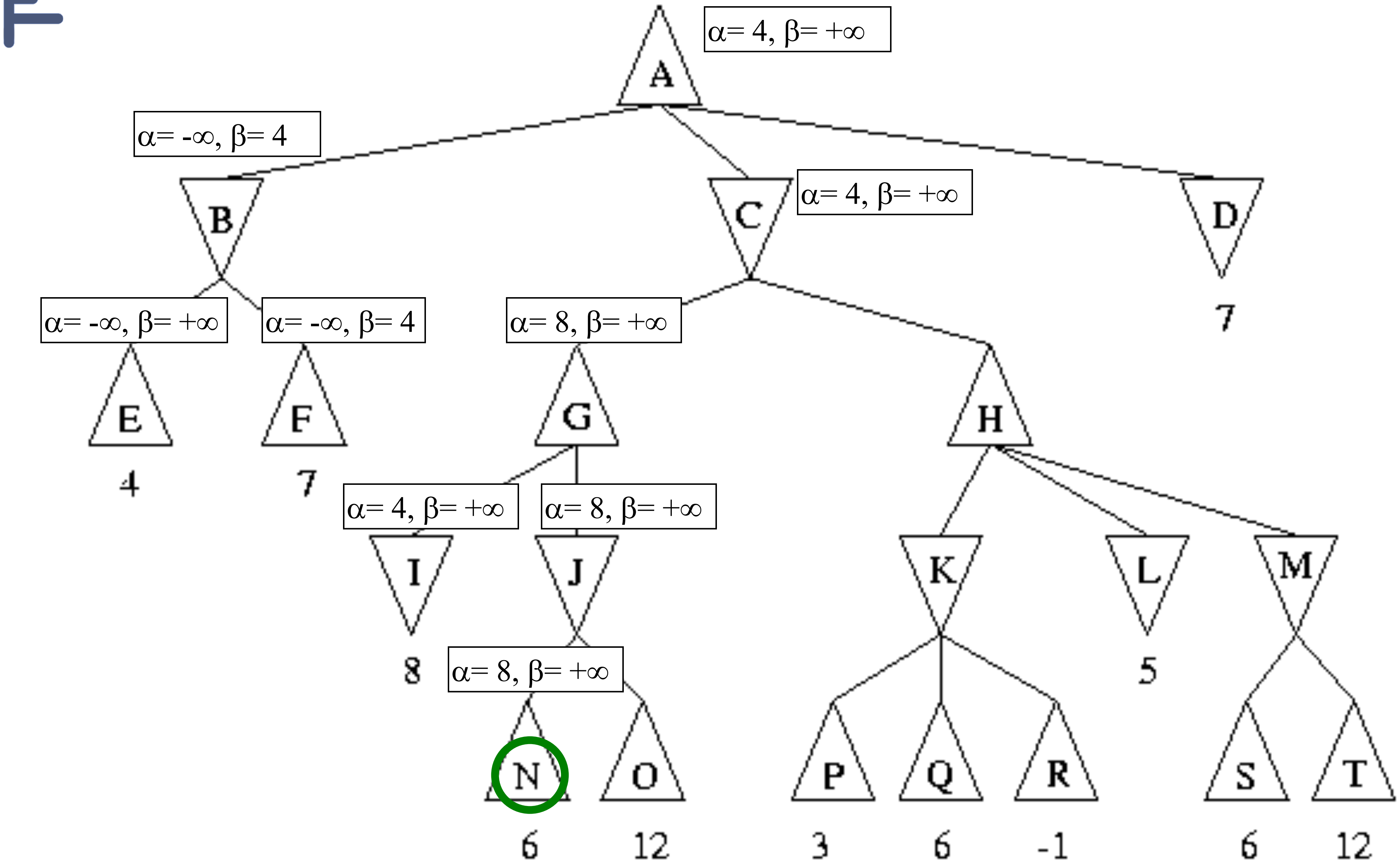

α= 4, β= +∞
A
α= -∞, β= 4
B
C
α= 4, β= +∞
D
7
α= -∞, β= +∞
α= -∞, β= 4
E
F
4
7
G
H
I
J
K
L
M
8
5
N
O
P
Q
R
S
T
6
12
3
6
-1
6
12

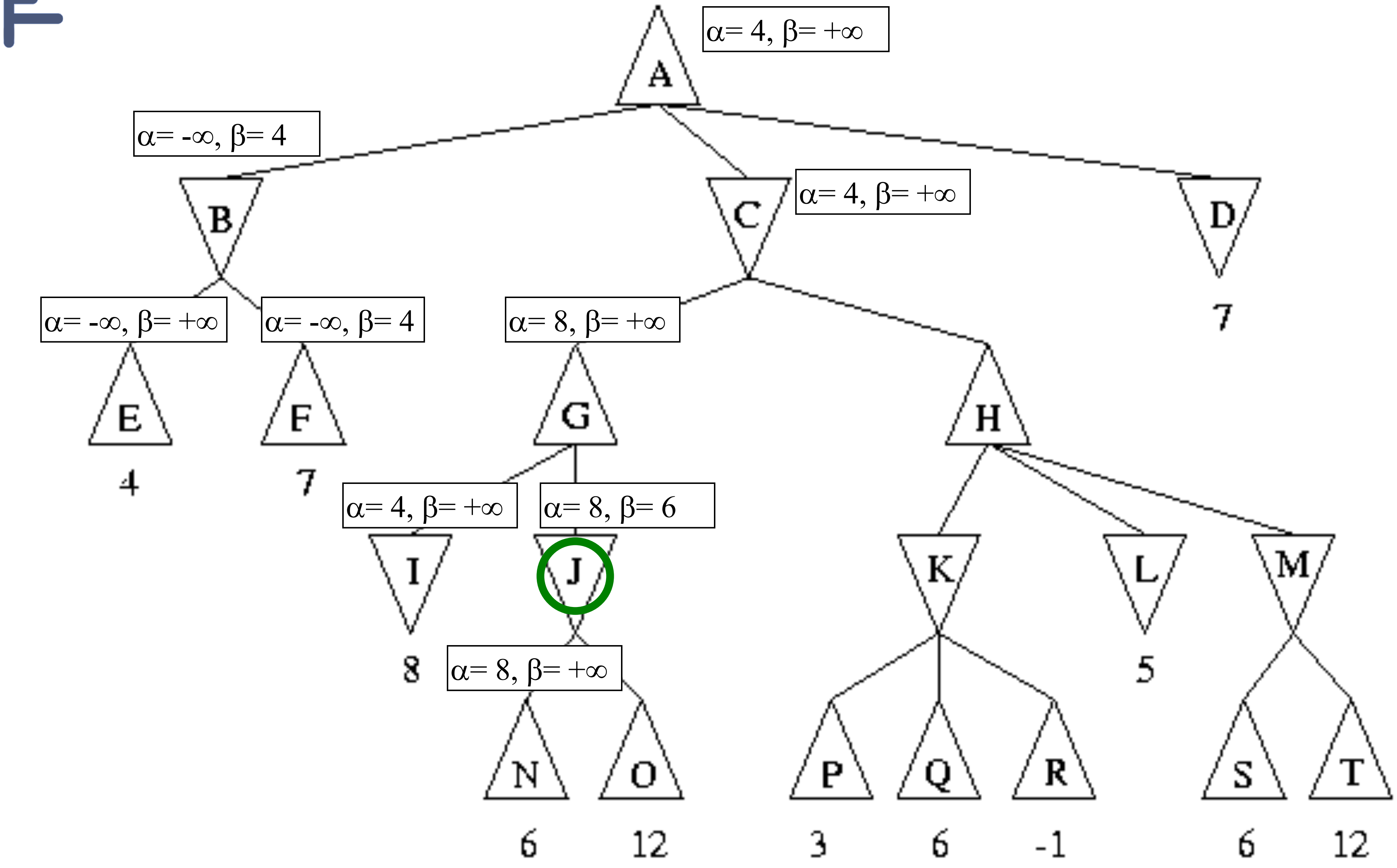

α= 4, β= +∞
A
α= -∞, β= 4
B
C
α= 4, β= +∞
D
7
α= -∞, β= +∞
α= -∞, β= 4
α= 4, β= +∞
E
F
G
H
4
7
I
J
K
L
M
8
5
N
O
P
Q
R
S
T
6
12
3
6
-1
6
12

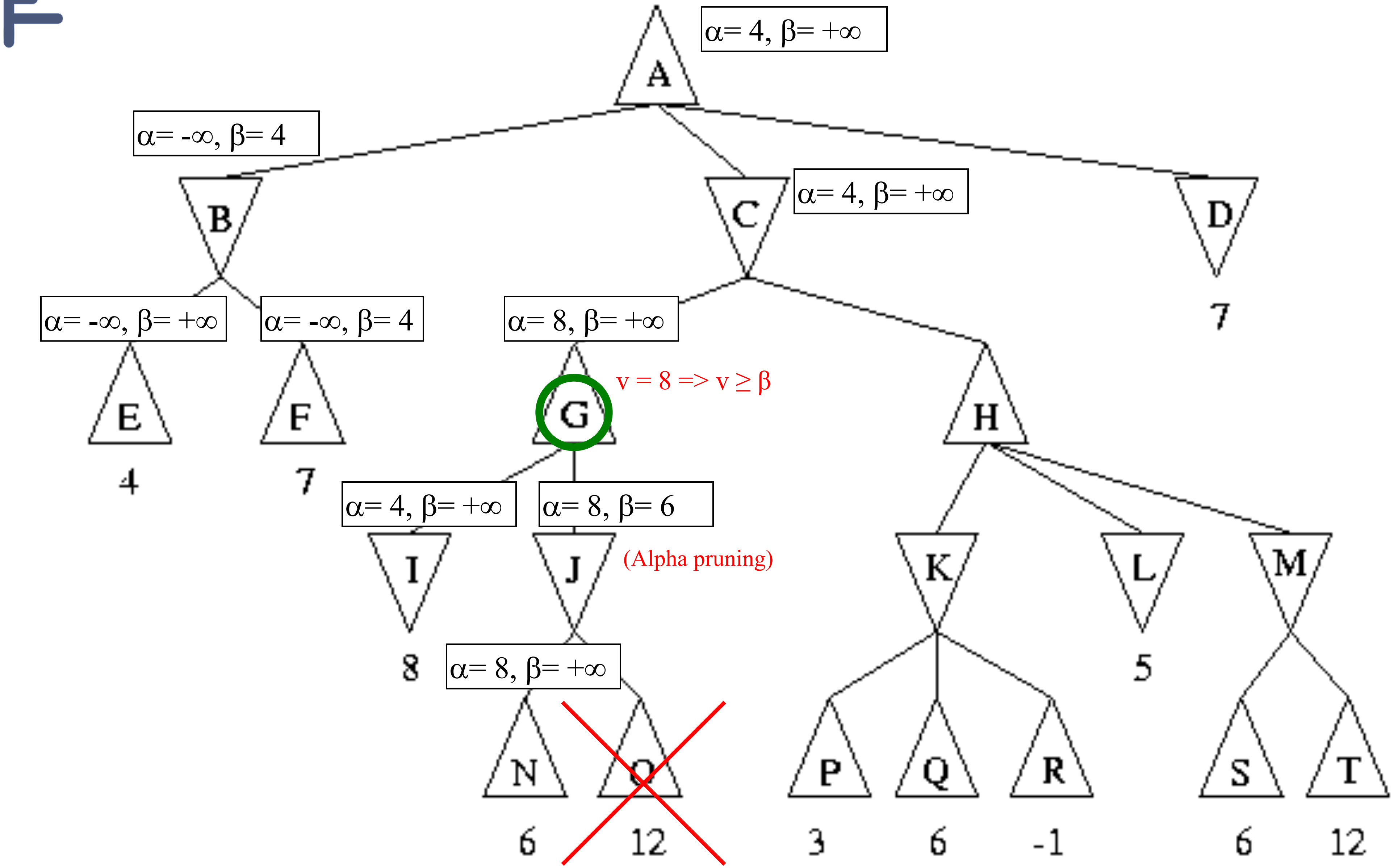

α= 4, β= +∞
A
α= -∞, β= 4
B
C
α= 4, β= +∞
D
7
α= -∞, β= +∞
α= -∞, β= 4
α= 4, β= +∞
E
F
G
H
4
7
α= 4, β= +∞
I
J
K
L
M
8
5
N
O
P
Q
R
S
T
6
12
3
6
-1
6
12

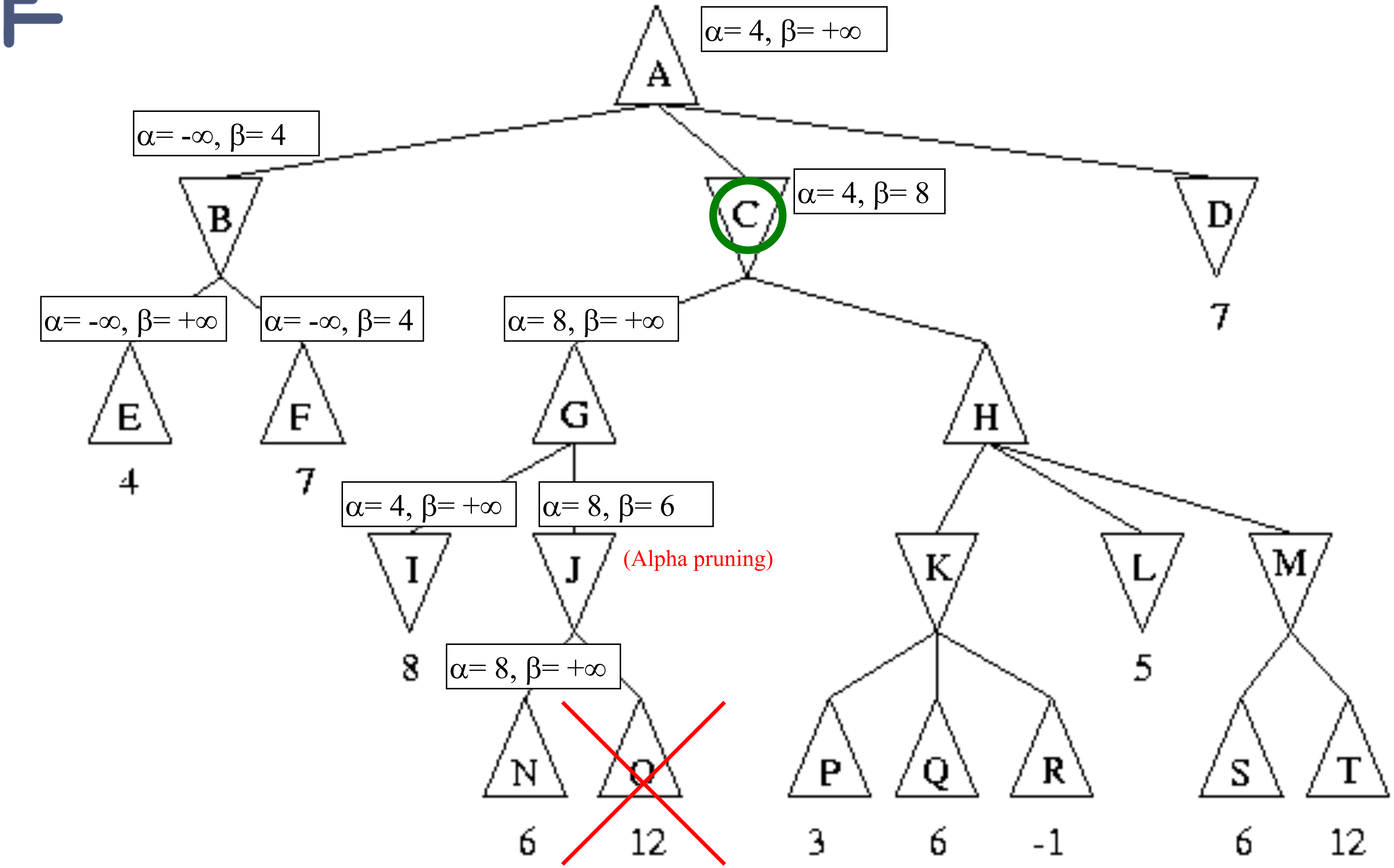

α= 4, β= +∞
A
α= -∞, β= 4
B
C
α= 4, β= +∞
D
7
α= -∞, β= +∞
α= -∞, β= 4
α= 8, β= +∞
E
F
G
H
4
7
α= 4, β= +∞
I
J
K
L
M
8
5
N
O
P
Q
R
S
T
6
12
3
6
-1
6
12

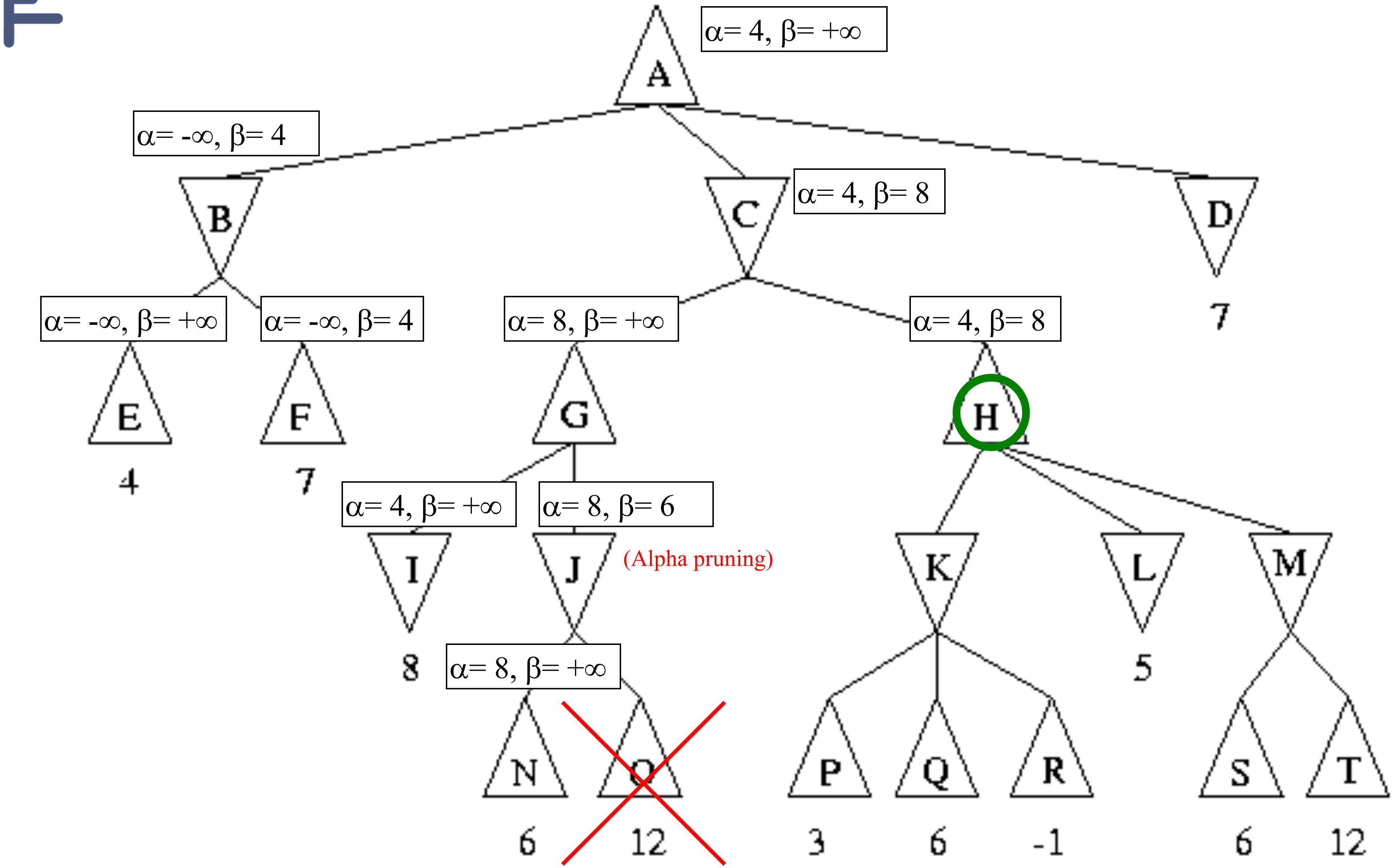

α= 4, β= +∞
A
α= -∞, β= 4
B
C
α= 4, β= +∞
D
7
α= -∞, β= +∞
α= -∞, β= 4
α= 8, β= +∞
E
F
G
H
4
7
α= 4, β= +∞
α= 8, β= +∞
I
J
K
L
M
8
5
N
O
P
Q
R
S
T
6
12
3
6
-1
6
12

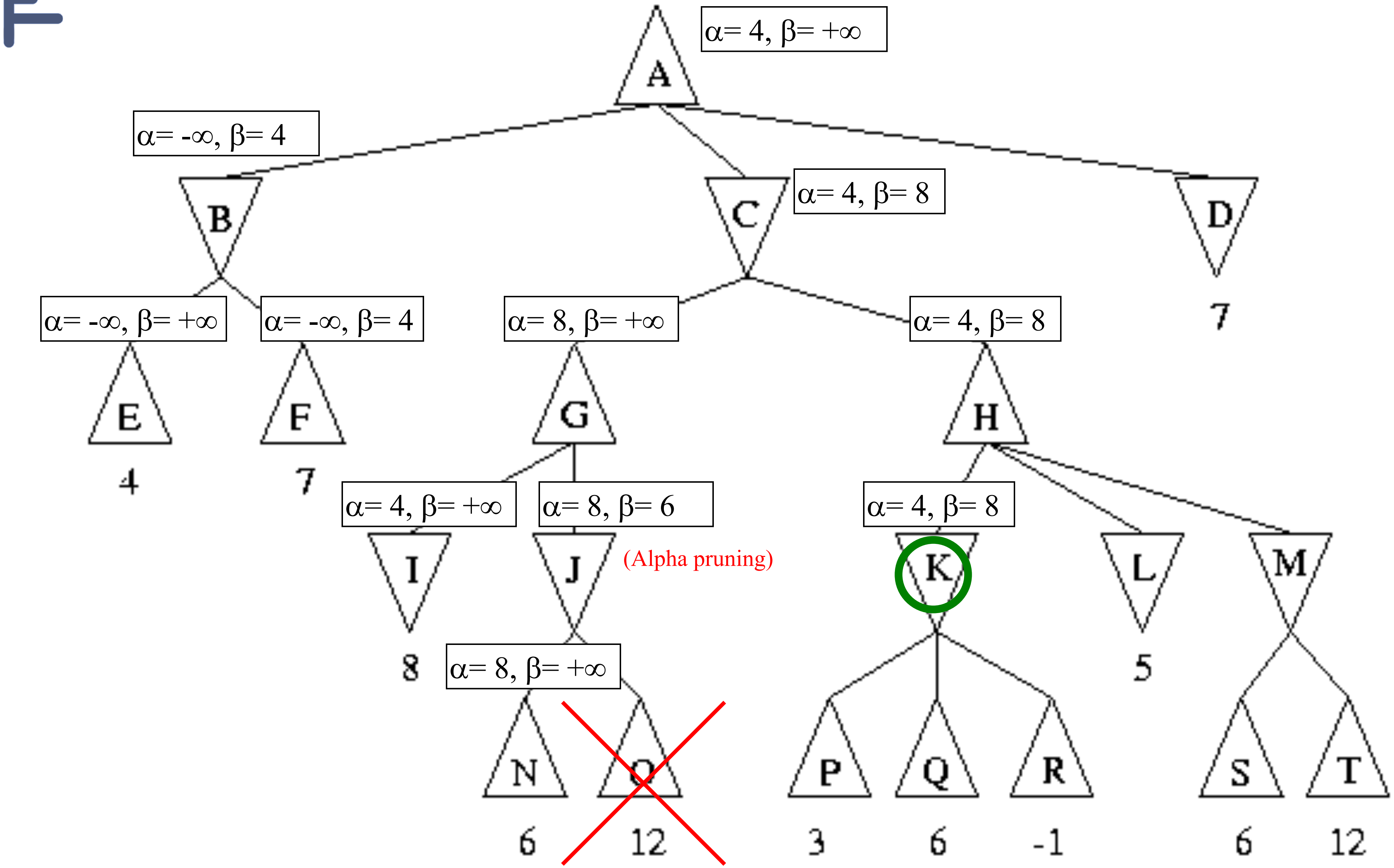

α= 4, β= +∞
A
α= -∞, β= 4
B
α= 4, β= +∞
C
D
7
α= -∞, β= +∞
α= -∞, β= 4
α= 8, β= +∞
E
F
G
H
4
7
α= 4, β= +∞
α= 8, β= +∞
I
J
K
L
M
8
α= 8, β= +∞
5
N
O
P
Q
R
S
T
6
12
3
6
-1
6
12

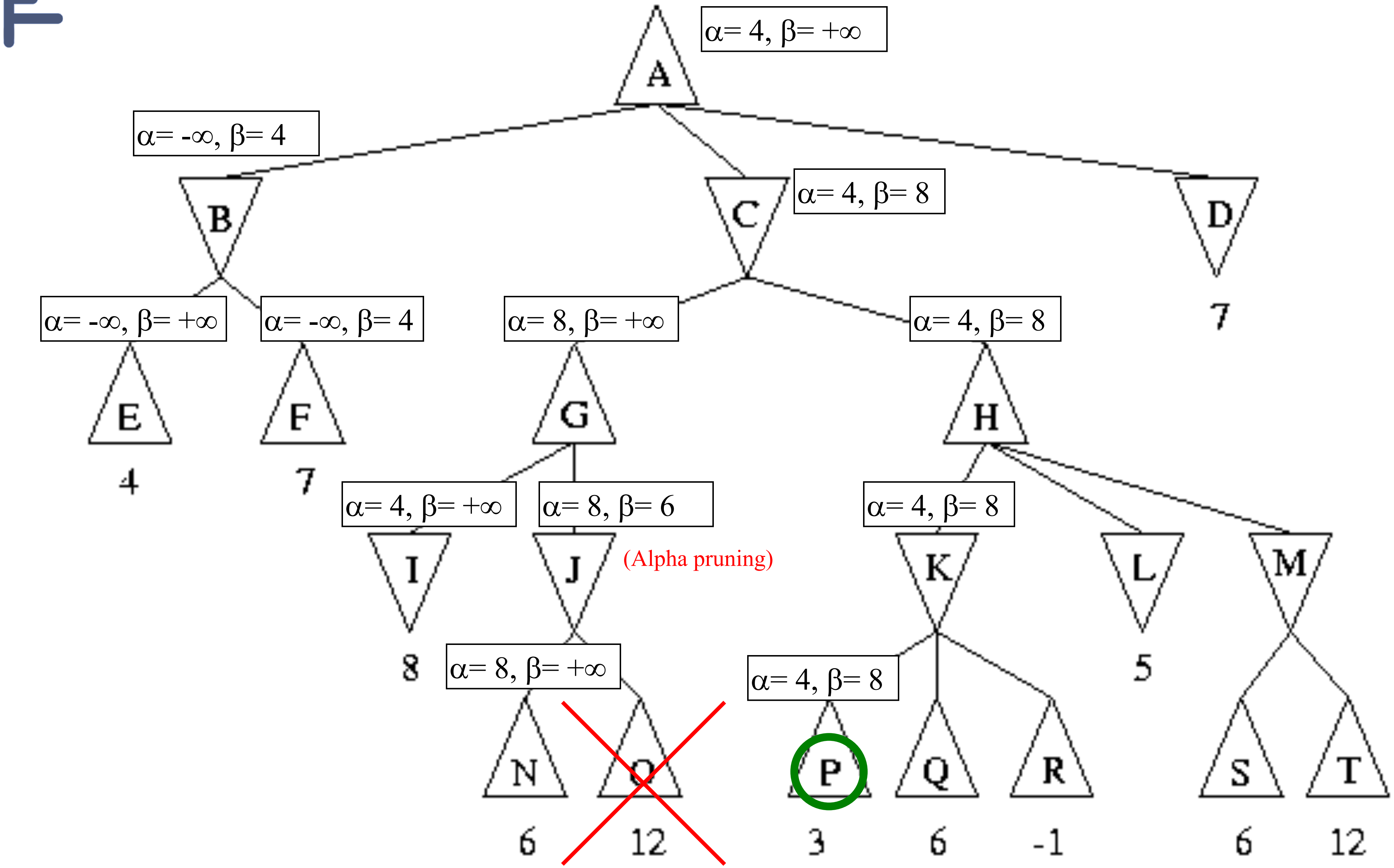

A
α= 4, β= +∞
B
α= -∞, β= 4
C
α= 4, β= +∞
D
7
α= -∞, β= +∞
E
4
α= -∞, β= 4
F
7
α= 8, β= +∞
G
H
α= 4, β= +∞
I
8
α= 8, β= 6
J
α= 8, β= +∞
N
6
O
12
K
L
5
M
P
3
Q
6
R
-1
S
6
T
12


α= 4, β= +∞
A
α= -∞, β= 4
B
C
α= 4, β= +∞
D
7
α= -∞, β= +∞
α= -∞, β= 4
α= 8, β= +∞
E
F
G
H
4
7
v = 8 => v ≥ β
α= 4, β= +∞
α= 8, β= 6
I
J
(Alpha pruning)
K
L
M
8
α= 8, β= +∞
5
N
O
P
Q
R
S
T
6
12
3
6
-1
6
12

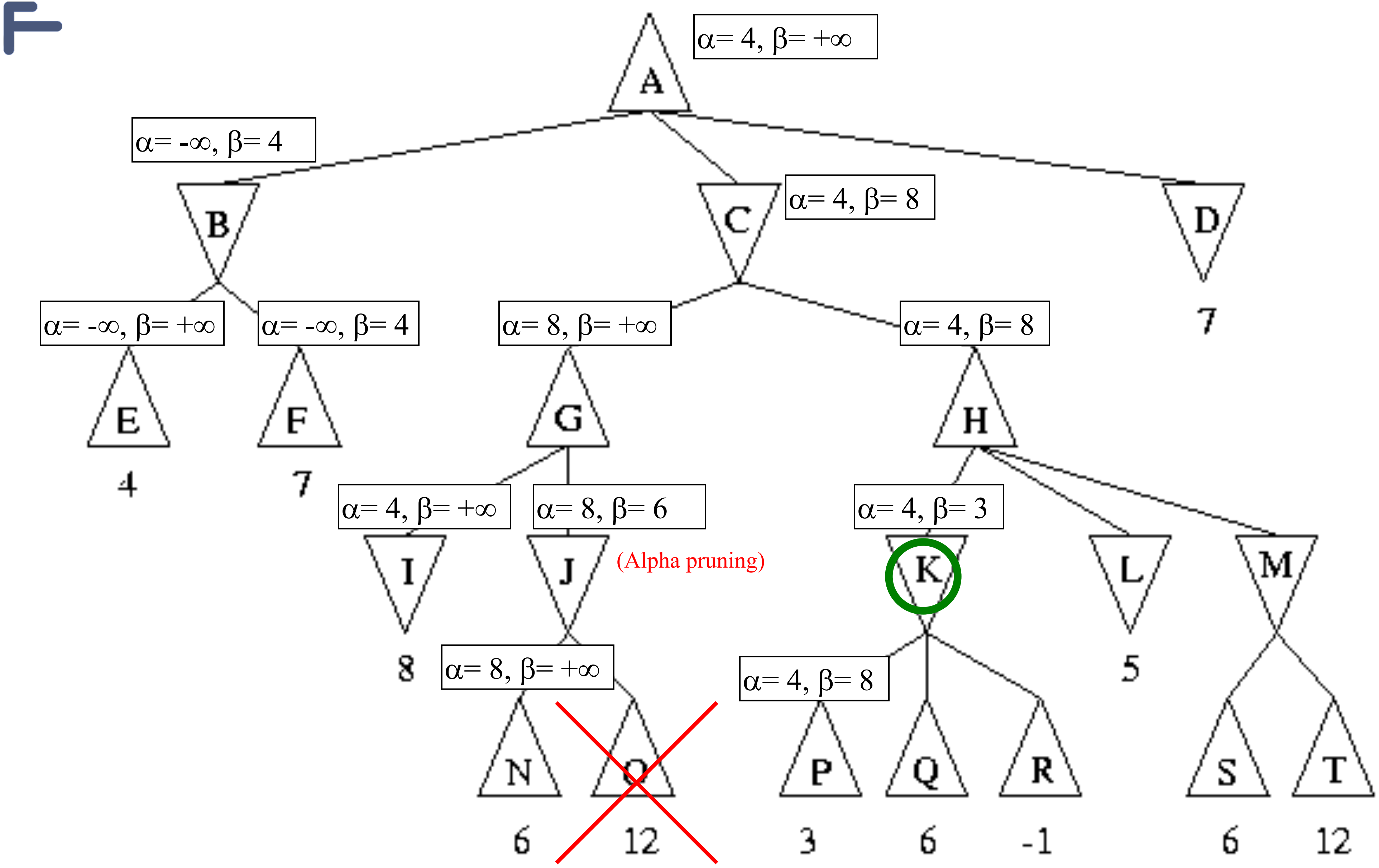

α= 4, β= +∞
A
α= -∞, β= 4
B
C
α= 4, β= 8
D
7
α= -∞, β= +∞
α= -∞, β= 4
α= 8, β= +∞
E
F
G
H
4
7
α= 4, β= +∞
α= 8, β= 6
I
J
(Alpha pruning)
K
L
M
8
α= 8, β= +∞
5
N
O
P
Q
R
S
T
6
12
3
6
-1
6
12

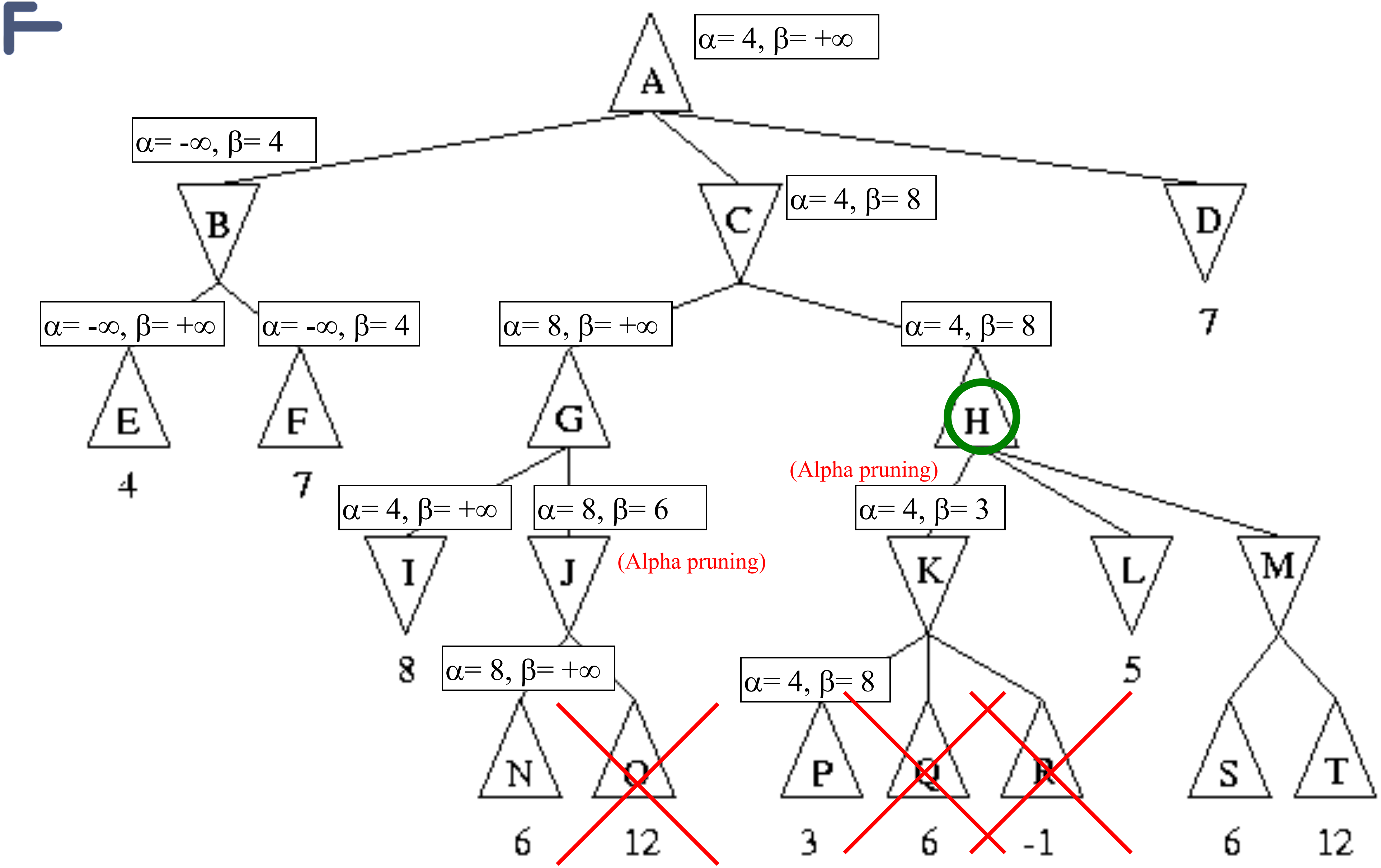

α= 4, β= +∞
A
α= -∞, β= 4
B
C
α= 4, β= 8
D
7
α= -∞, β= +∞
α= -∞, β= 4
α= 8, β= +∞
α= 4, β= 8
E
F
G
H
4
7
α= 4, β= +∞
α= 8, β= 6
I
J
(Alpha pruning)
K
L
M
8
α= 8, β= +∞
5
N
O
P
Q
R
S
T
6
12
3
6
-1
6
12

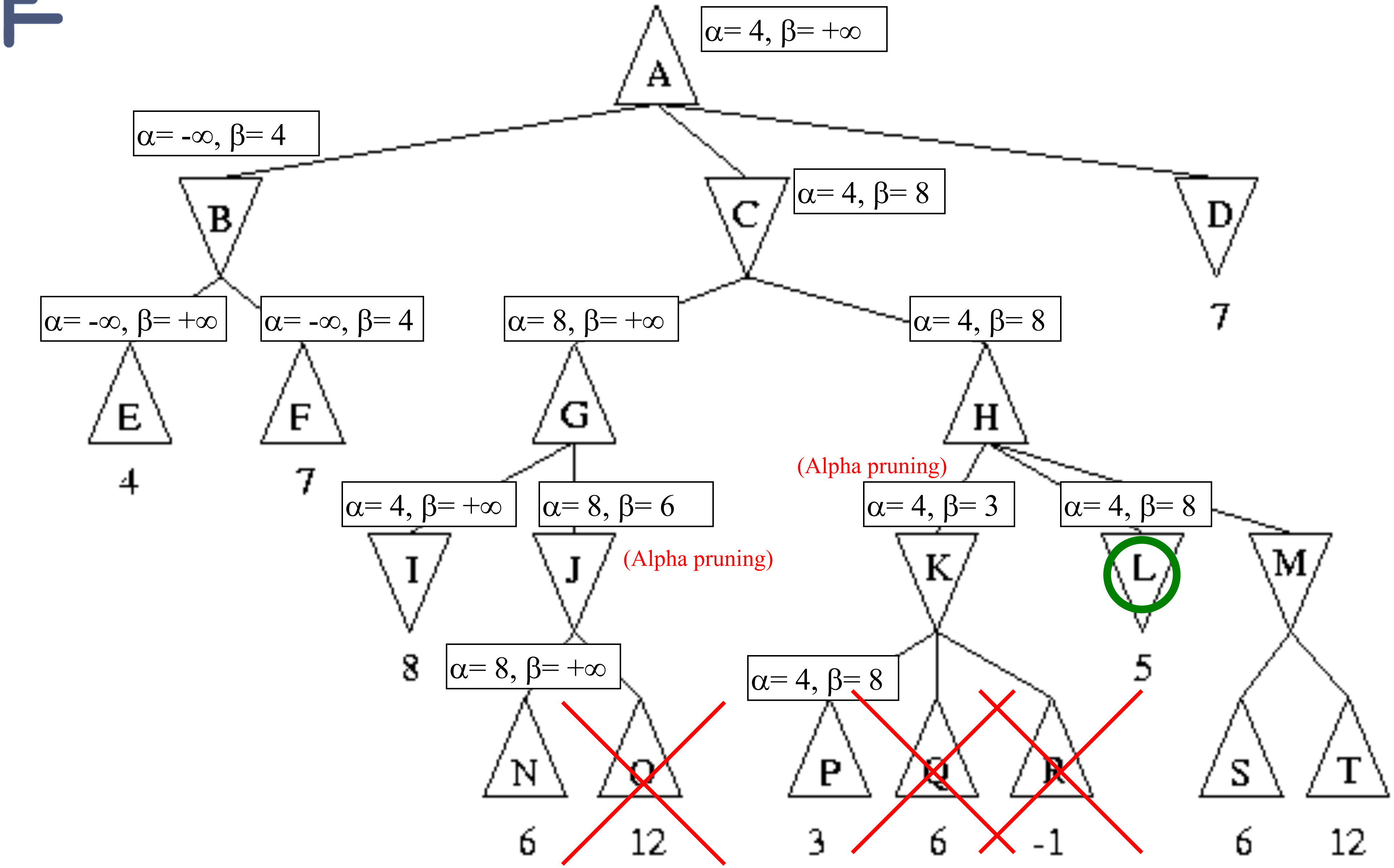

α= 4, β= +∞
A
α= -∞, β= 4
B
C
α= 4, β= 8
D
7
α= -∞, β= +∞
α= -∞, β= 4
α= 8, β= +∞
α= 4, β= 8
E
F
G
H
4
7
α= 4, β= +∞
α= 8, β= 6
α= 4, β= 8
I
J
(Alpha pruning)
K
L
M
8
α= 8, β= +∞
5
N
O
P
Q
R
S
T
6
12
3
6
-1
6
12

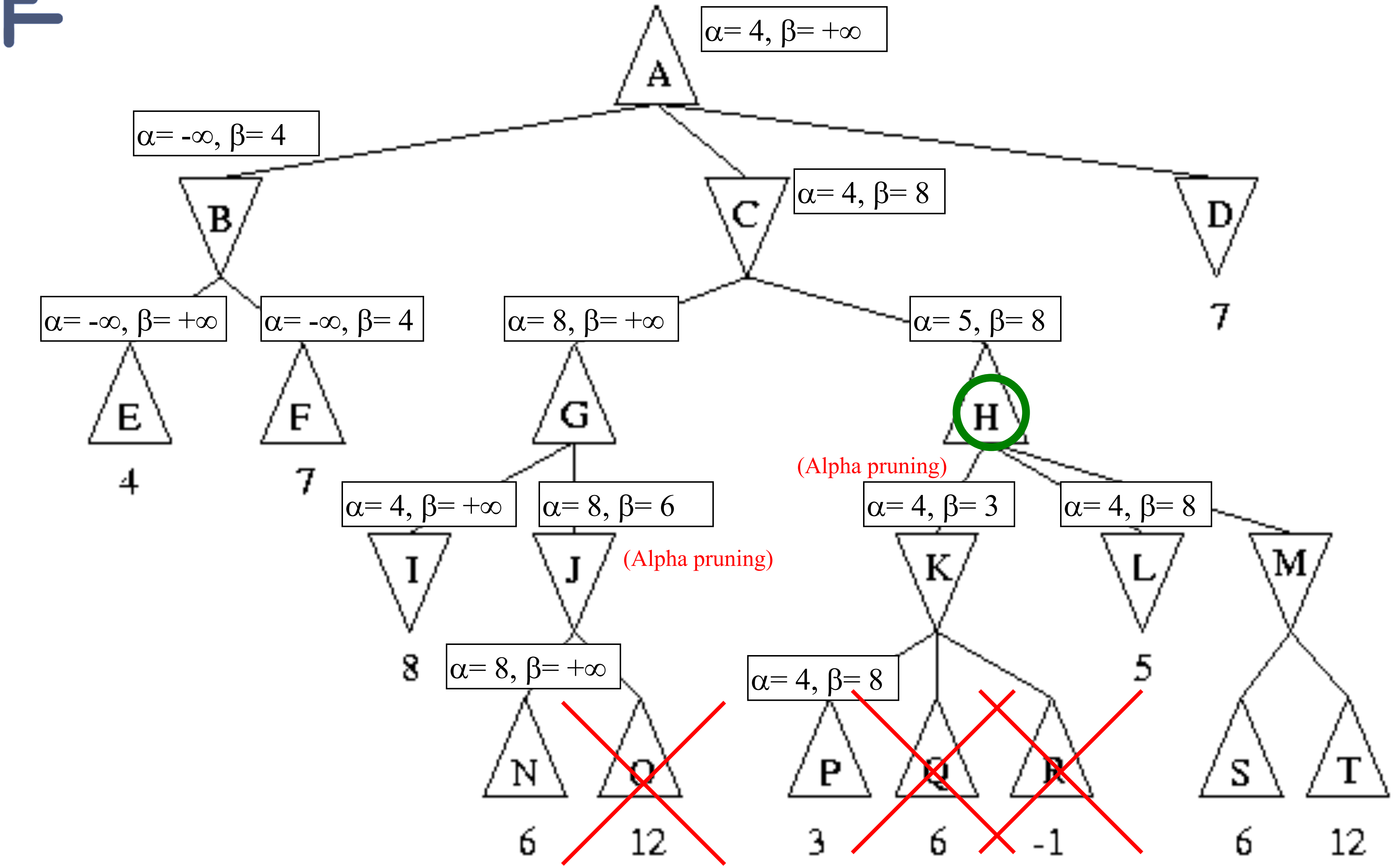


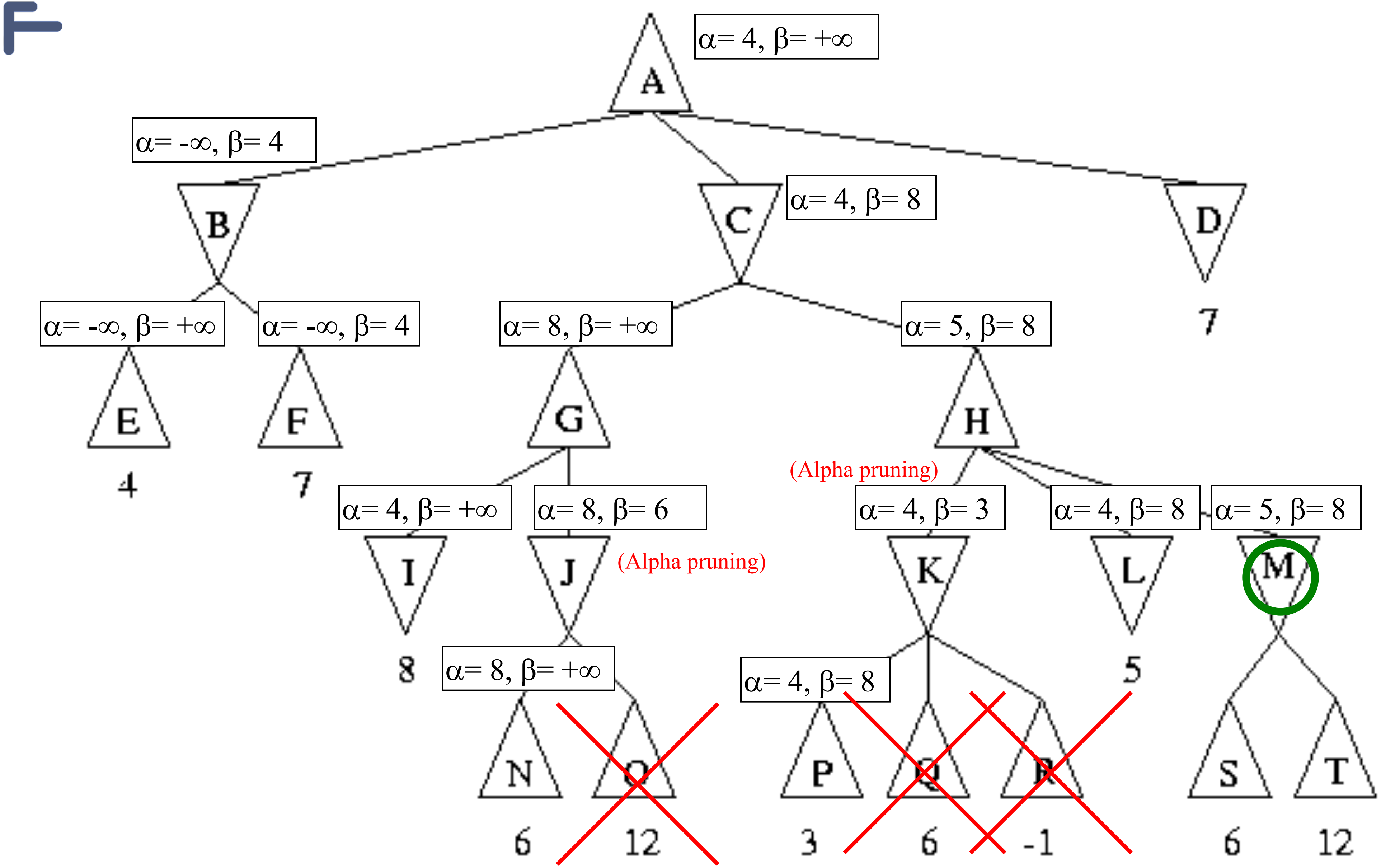
α= 4, β= +∞
A
α= -∞, β= 4
B
α= 4, β= 8
C
D
7
α= -∞, β= +∞
α= -∞, β= 4
α= 8, β= +∞
α= 4, β= 8
E
F
G
H
4
7
α= 4, β= +∞
α= 8, β= 6
α= 4, β= 8
I
J
(Alpha pruning)
K
L
M
8
α= 8, β= +∞
α= 4, β= 8
5
N
O
P
Q
R
S
T
6
12
3
6
-1
6
12

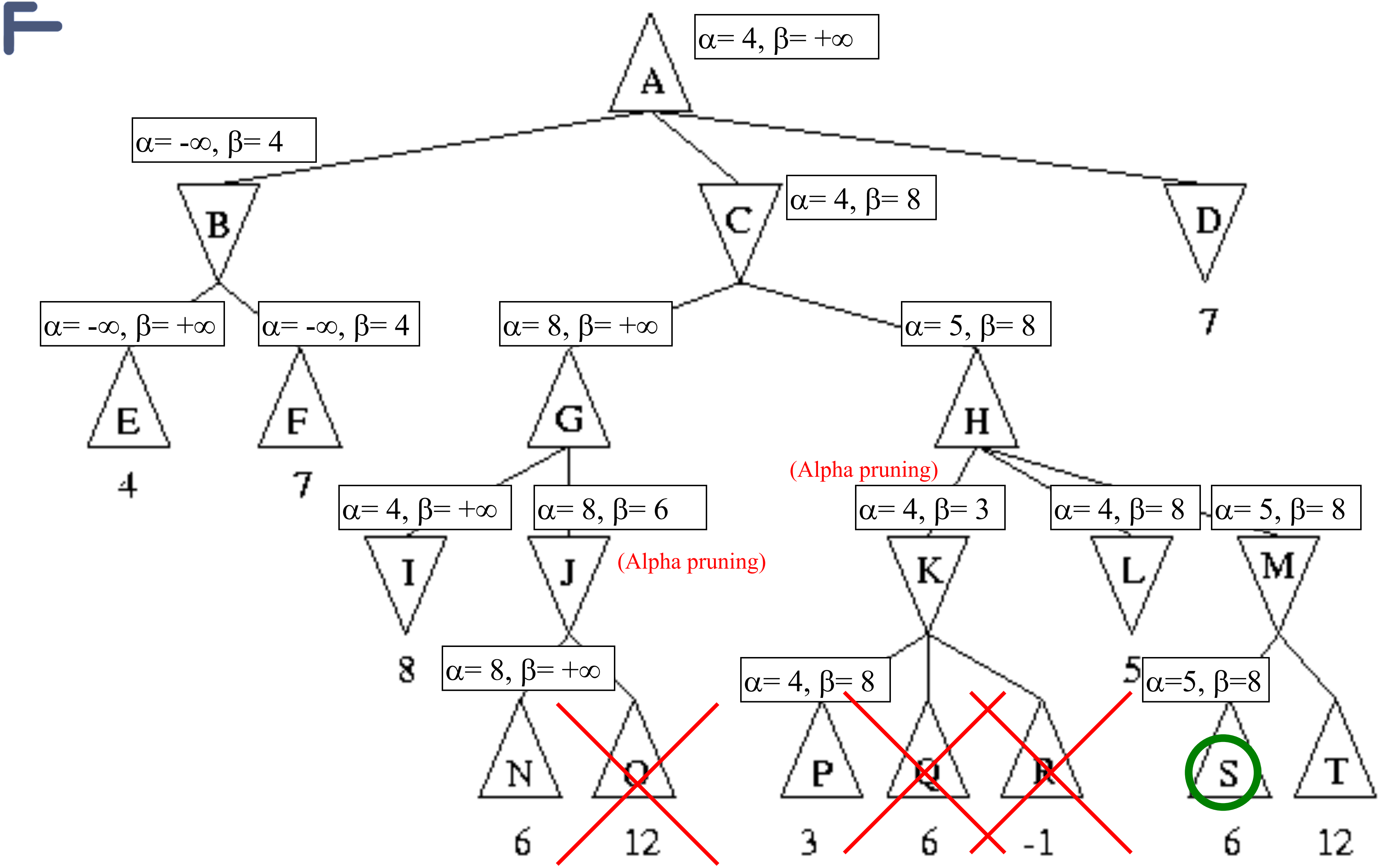

α= 4, β= +∞
A
α= -∞, β= 4
B
C
α= 4, β= 8
D
7
α= -∞, β= +∞
α= -∞, β= 4
α= 8, β= +∞
α= 4, β= 8
E
F
G
H
4
7
α= 4, β= +∞
α= 8, β= 6
α= 4, β= 3
I
J
(Alpha pruning)
K
L
M
8
α= 8, β= +∞
α= 4, β= 8
5
N
O
P
Q
R
S
T
6
12
3
6
-1
6
12

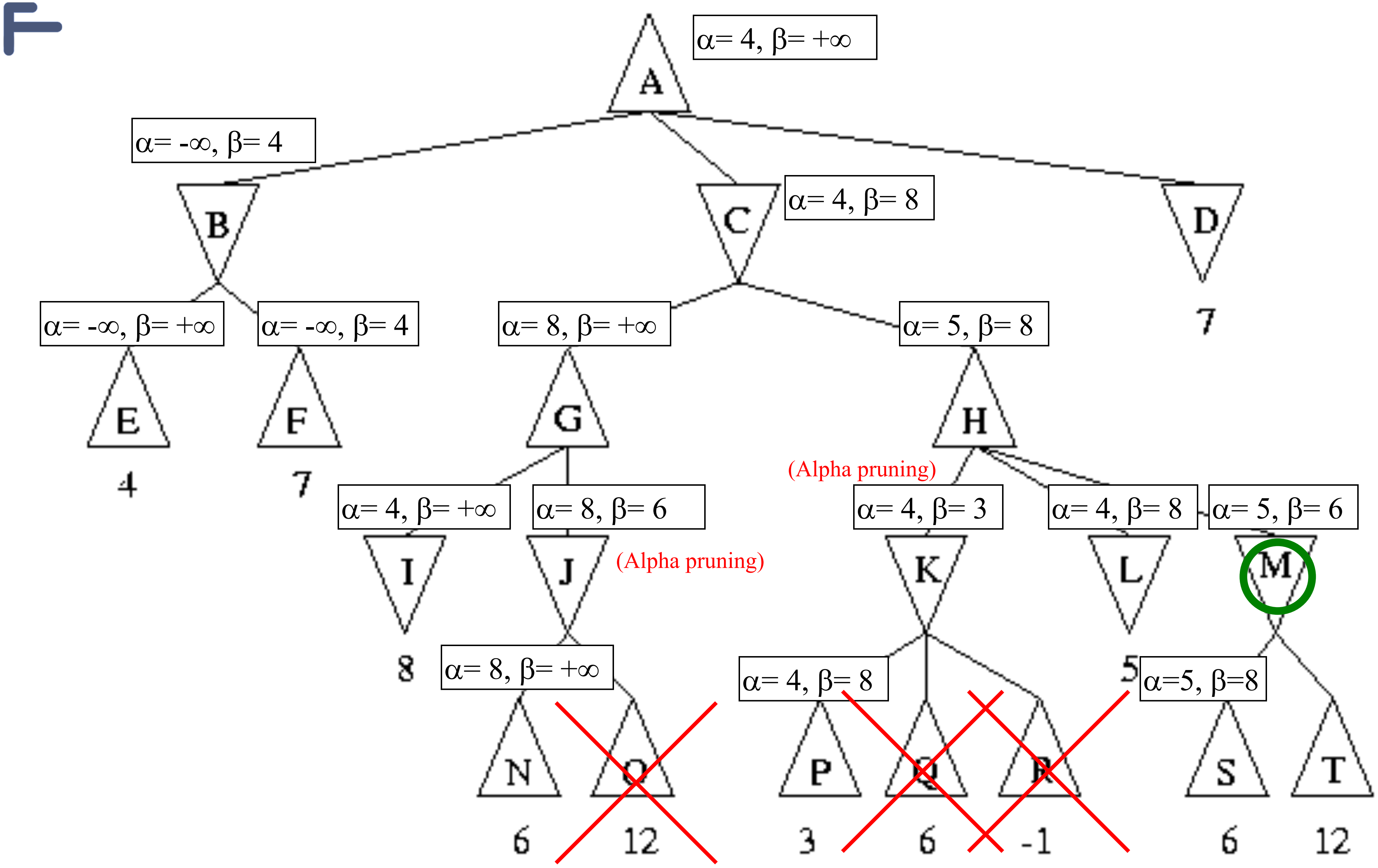

α= 4, β= +∞
A
α= -∞, β= 4
B
C
α= 4, β= 8
D
7
α= -∞, β= +∞
α= -∞, β= 4
α= 8, β= +∞
α= 4, β= 8
E
F
G
H
4
7
(Alpha pruning)
α= 4, β= +∞
α= 8, β= 6
α= 4, β= 3
I
J
(Alpha pruning)
K
L
M
8
α= 8, β= +∞
α= 4, β= 8
5
N
O
P
Q
R
S
T
6
12
3
6
-1
6
12

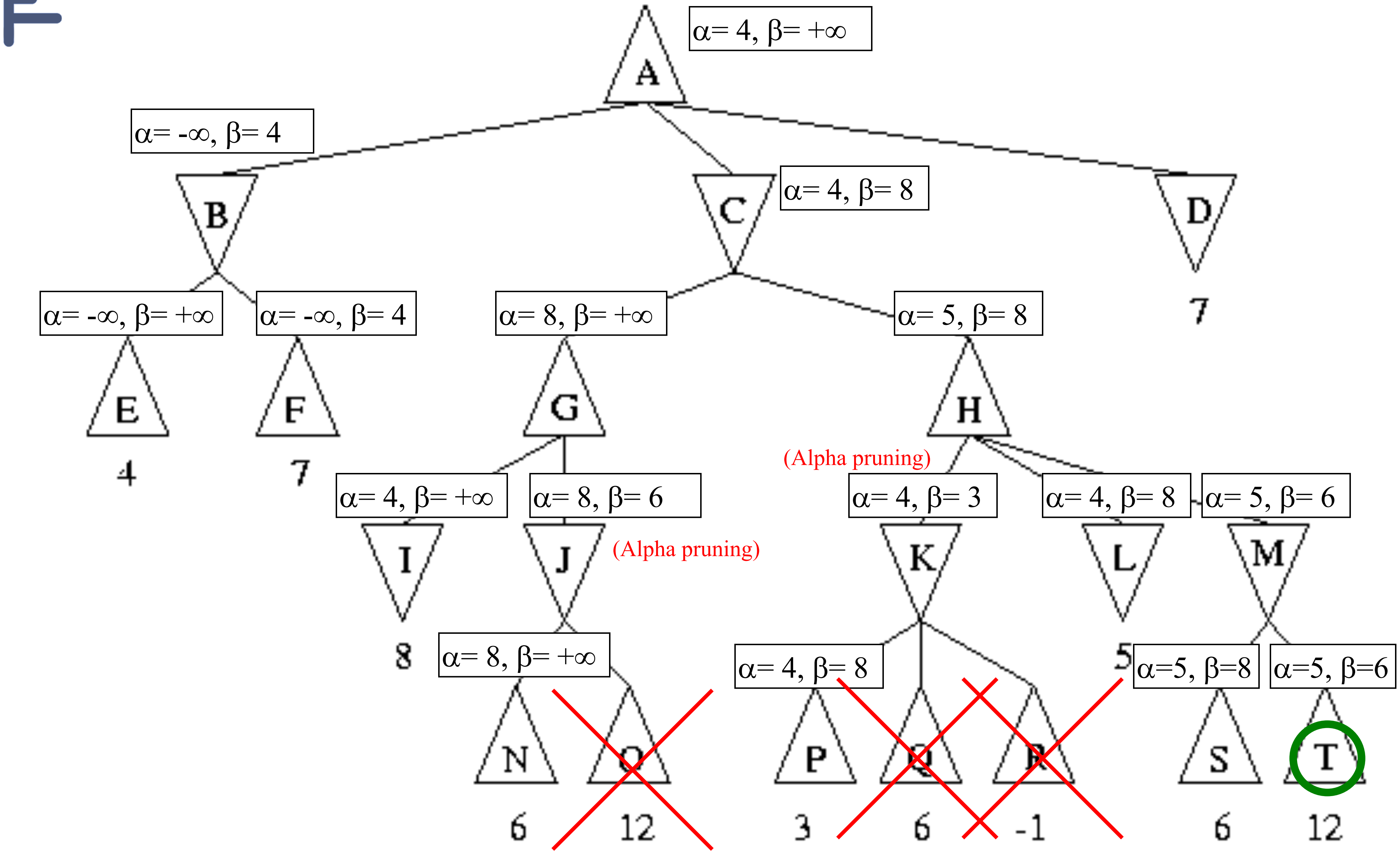

α= 4, β= +∞
A
α= -∞, β= 4
B
C
α= 4, β= 8
D
7
α= -∞, β= +∞
α= -∞, β= 4
α= 8, β= +∞
α= 4, β= 8
E
F
G
H
4
7
(Alpha pruning)
α= 4, β= +∞
α= 8, β= 6
α= 4, β= 3
α= 4, β= 8
I
J
(Alpha pruning)
K
L
M
8
α= 8, β= +∞
α= 4, β= 8
5
N
O
P
Q
R
S
T
6
12
3
6
-1
6
12

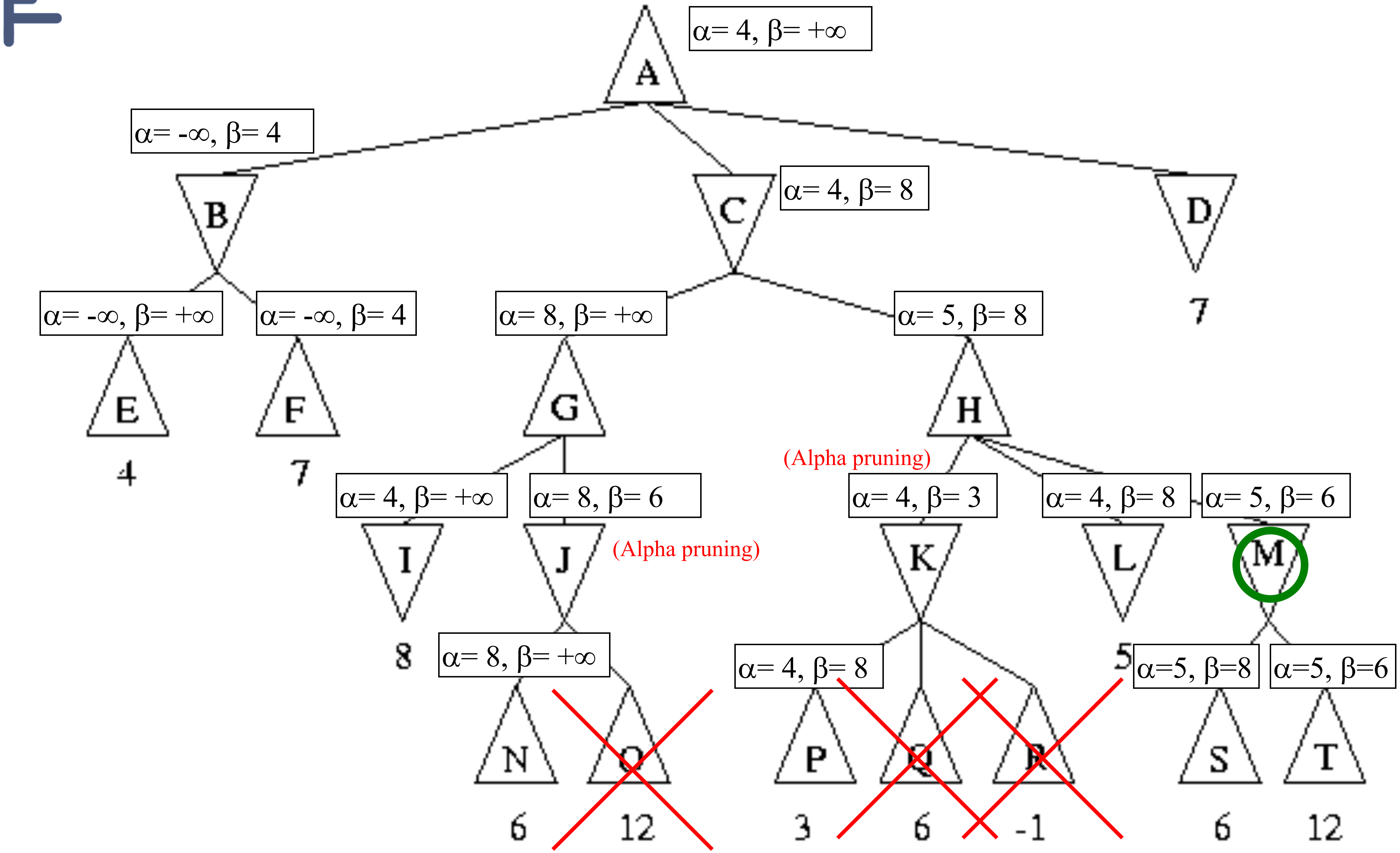

α= 4, β= +∞
A
α= -∞, β= 4
B
C
α= 4, β= 8
D
7
α= -∞, β= +∞
α= -∞, β= 4
α= 8, β= +∞
α= 5, β= 8
E
F
G
H
4
7
(Alpha pruning)
α= 4, β= +∞
α= 8, β= 6
α= 4, β= 3
α= 4, β= 8
I
J
(Alpha pruning)
K
L
M
8
α= 8, β= +∞
α= 4, β= 8
5
N
O
P
Q
R
S
T
6
12
3
6
-1
6
12


α= 4, β= +∞
A
α= -∞, β= 4
B
C
α= 4, β= 8
D
7
α= -∞, β= +∞
α= -∞, β= 4
α= 8, β= +∞
α= 5, β= 8
E
F
G
H
4
7
(Alpha pruning)
α= 4, β= +∞
α= 8, β= 6
α= 4, β= 3
α= 4, β= 8
α= 5, β= 8
I
J
(Alpha pruning)
K
L
M
8
α= 8, β= +∞
α= 4, β= 8
5
N
O
P
Q
R
S
T
6
12
3
6
-1
6
12

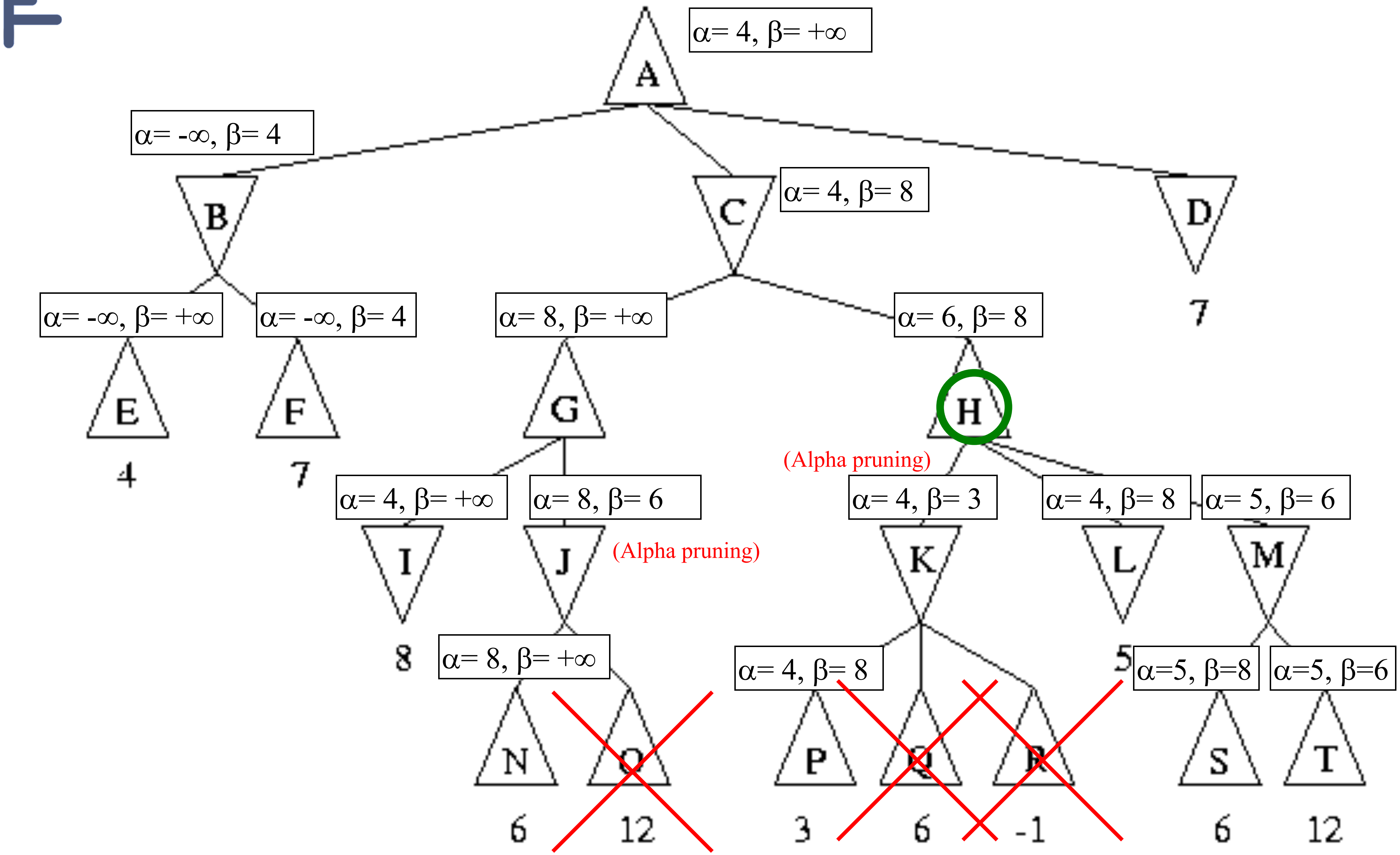

A
α= 4, β= +∞
α= -∞, β= 4
B
C
α= 4, β= 8
D
7
α= -∞, β= +∞
α= -∞, β= 4
α= 8, β= +∞
α= 5, β= 8
E
F
G
H
4
7
(Alpha pruning)
α= 4, β= +∞
α= 8, β= 6
α= 4, β= 3
α= 4, β= 8
α= 5, β= 8
I
J
(Alpha pruning)
K
L
M
8
α= 8, β= +∞
α= 4, β= 8
5
α=5, β=8
N
O
P
Q
R
S
T
6
12
3
6
-1
6
12

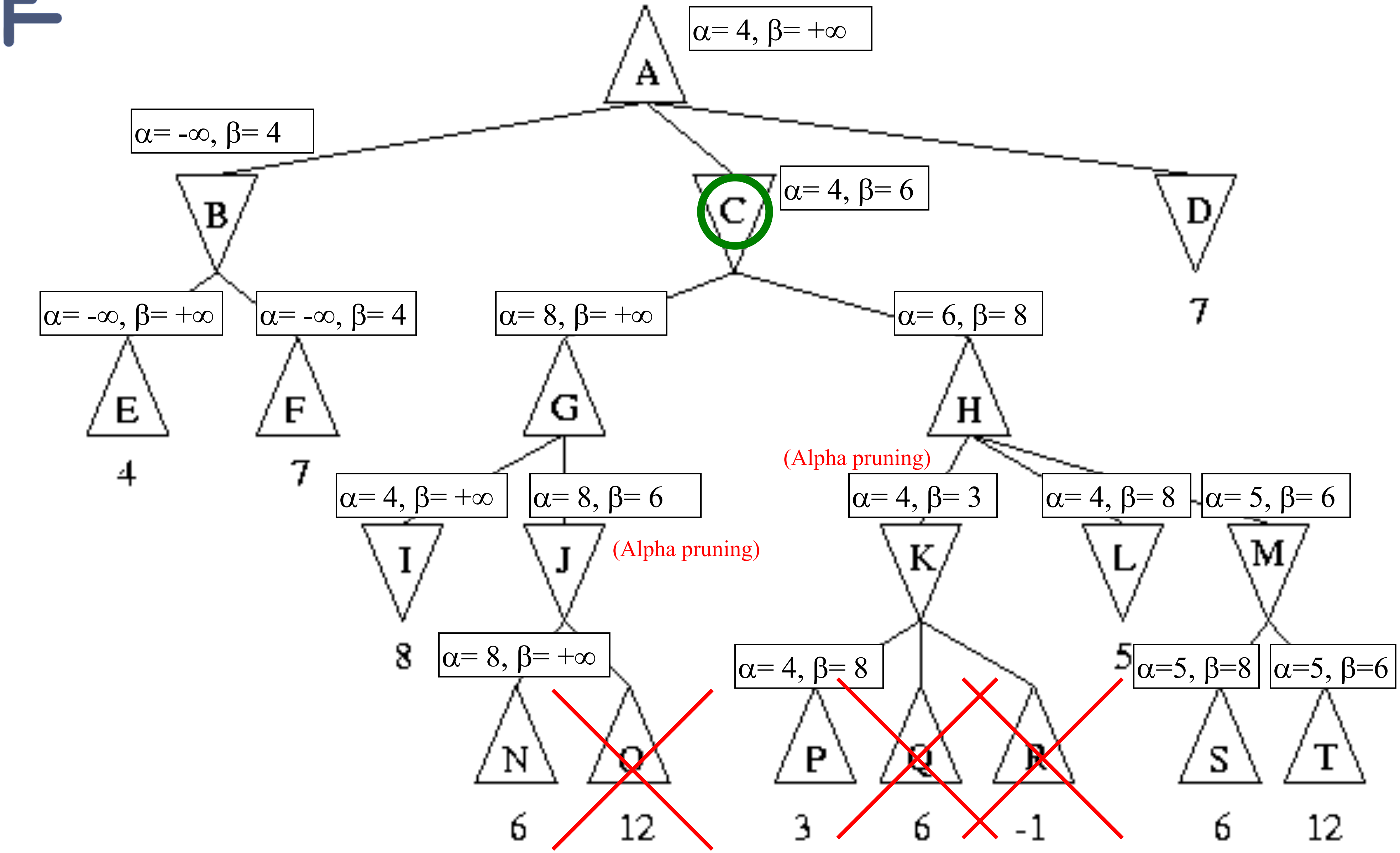

α= 4, β= +∞
A
α= -∞, β= 4
B
C
α= 4, β= 8
D
7
α= -∞, β= +∞
α= -∞, β= 4
α= 8, β= +∞
α= 5, β= 8
E
F
G
H
4
7
(Alpha pruning)
α= 4, β= +∞
α= 8, β= 6
α= 4, β= 3
α= 4, β= 8
α= 5, β= 6
I
J
(Alpha pruning)
K
L
M
8
α= 8, β= +∞
α= 4, β= 8
5
α=5, β=8
N
O
P
Q
R
S
T
6
12
3
6
-1
6
12

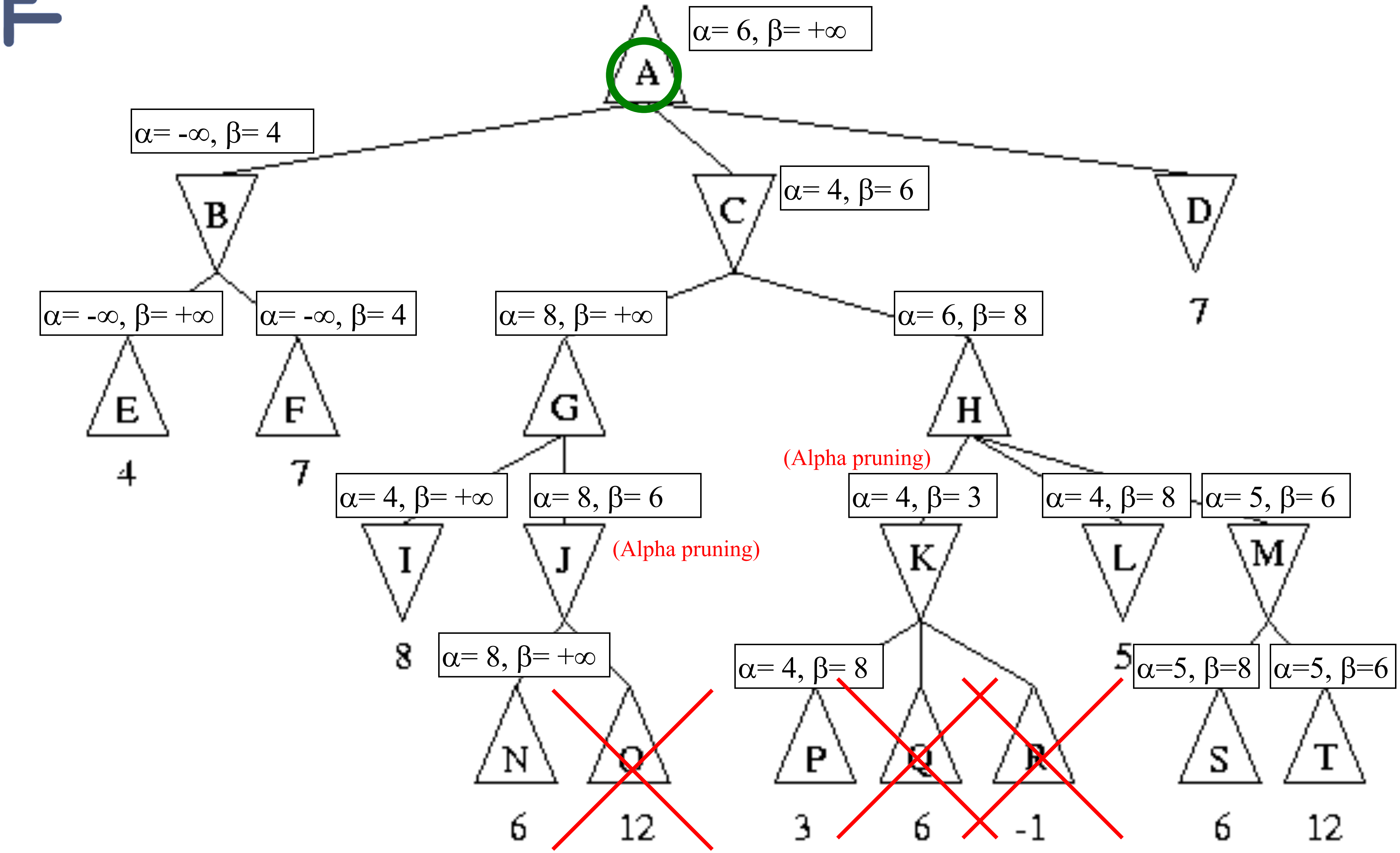

A
α= 4, β= +∞
B
α= -∞, β= 4
C
α= 4, β= 8
D
7
α= -∞, β= +∞
α= -∞, β= 4
α= 8, β= +∞
α= 5, β= 8
E
4
F
7
G
H
(Alpha pruning)
α= 4, β= +∞
α= 8, β= 6
(Alpha pruning)
α= 4, β= 3
α= 4, β= 8
α= 5, β= 6
I
8
J
K
L
5
M
α= 8, β= +∞
α= 4, β= 8
α=5, β=8
α=5, β=6
N
6
O
12
P
3
Q
6
R
-1
S
6
T
12

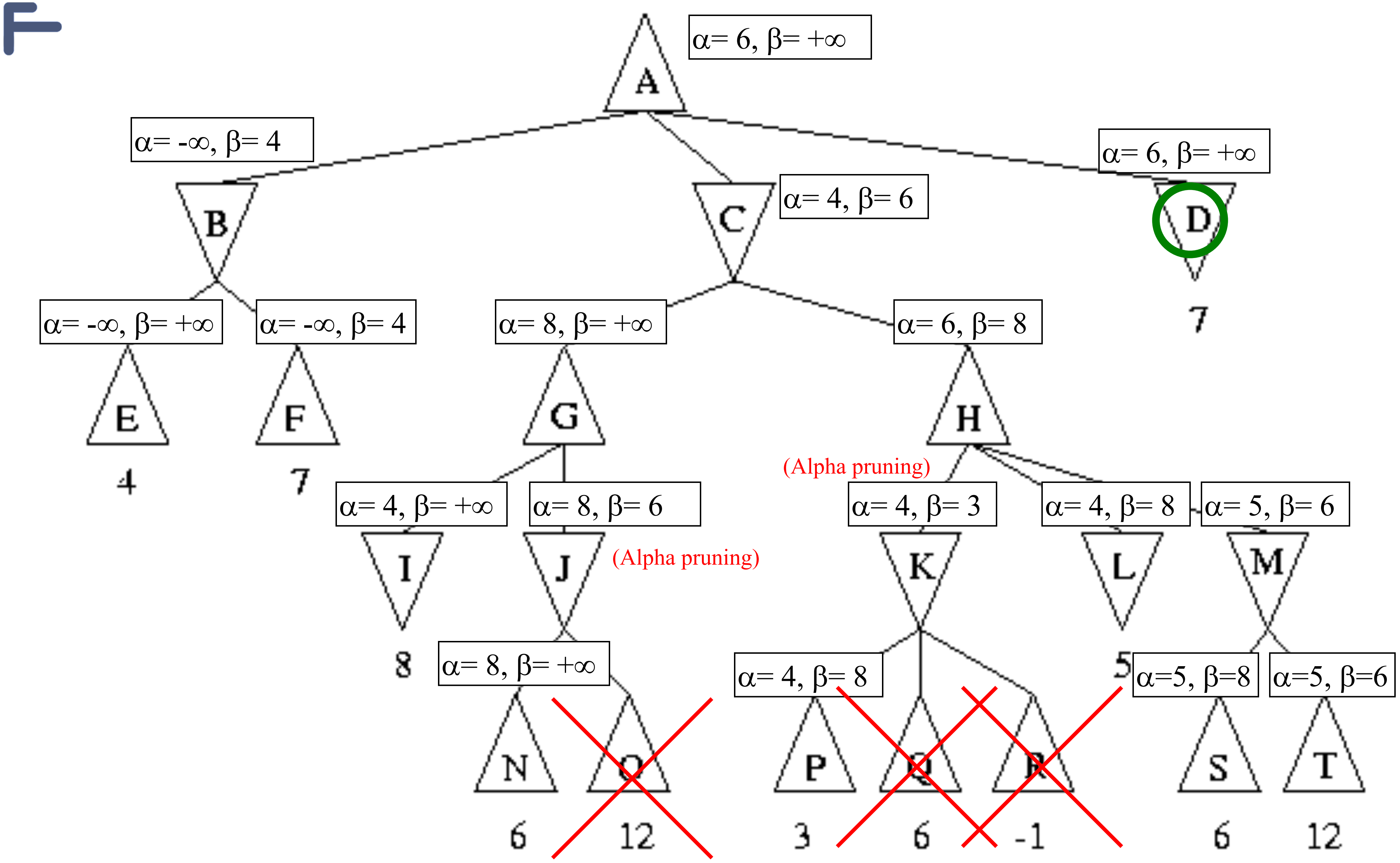


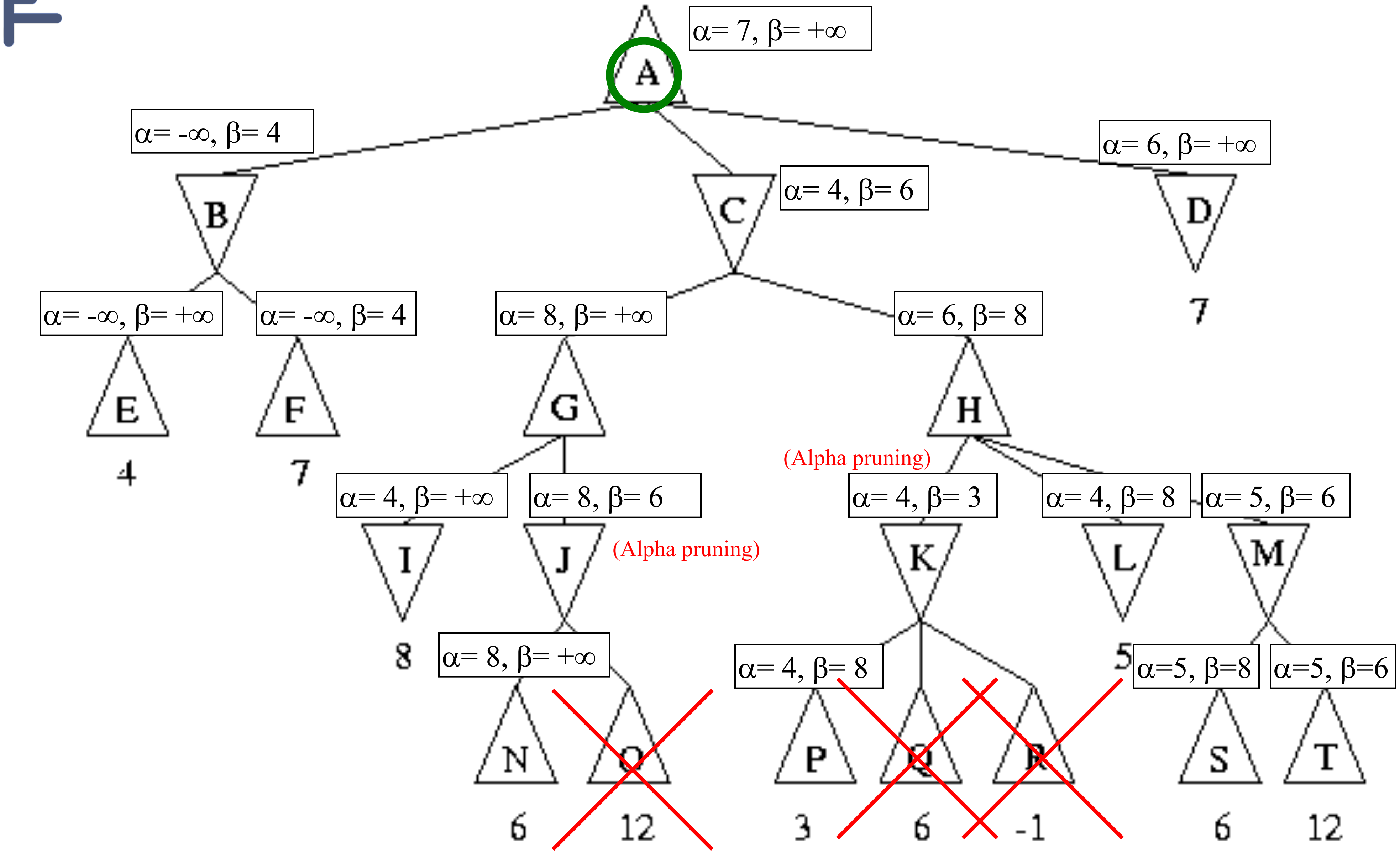
α= 4, β= +∞
A
α= -∞, β= 4
B
C
α= 4, β= 8
D
α= -∞, β= +∞
α= -∞, β= 4
α= 8, β= +∞
α= 5, β= 8
7
E
F
G
H
4
7
(Alpha pruning)
α= 4, β= +∞
α= 8, β= 6
α= 4, β= 3
α= 4, β= 8
α= 5, β= 6
I
J
(Alpha pruning)
K
L
M
8
α= 8, β= +∞
α= 4, β= 8
5
α=5, β=8
α=5, β=6
N
O
P
Q
R
S
T
6
12
3
6
-1
6
12

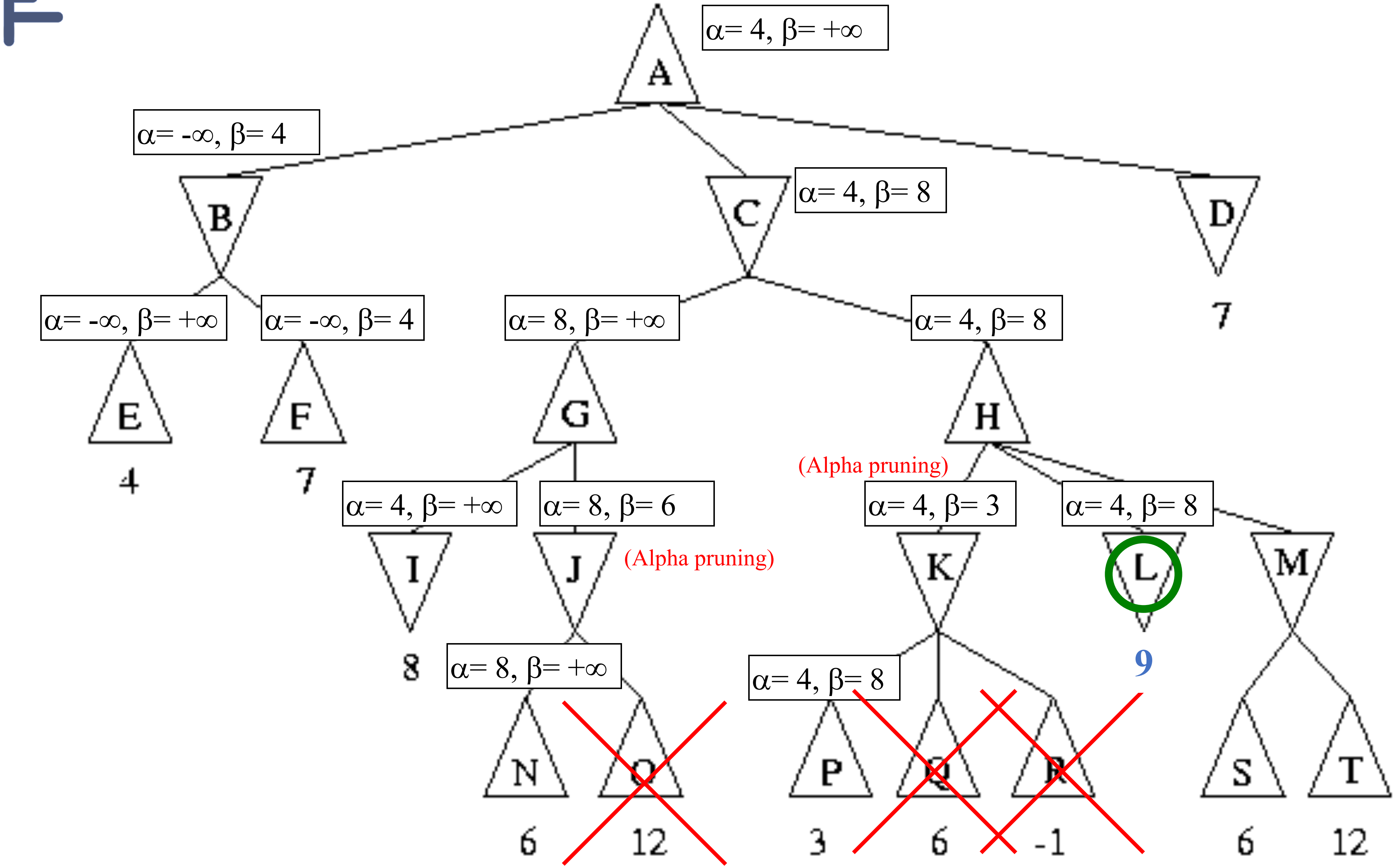

α= 4, β= +∞
A
α= -∞, β= 4
B
C
α= 4, β= 8
D
α= -∞, β= +∞
α= -∞, β= 4
α= 8, β= +∞
α= 6, β= 8
7
E
F
G
H
4
7
(Alpha pruning)
α= 4, β= +∞
α= 8, β= 6
α= 4, β= 3
α= 4, β= 8
α= 5, β= 6
I
J
(Alpha pruning)
K
L
M
8
α= 8, β= +∞
α= 4, β= 8
5
α=5, β=8
α=5, β=6
N
O
P
Q
R
S
T
6
12
3
6
-1
6
12

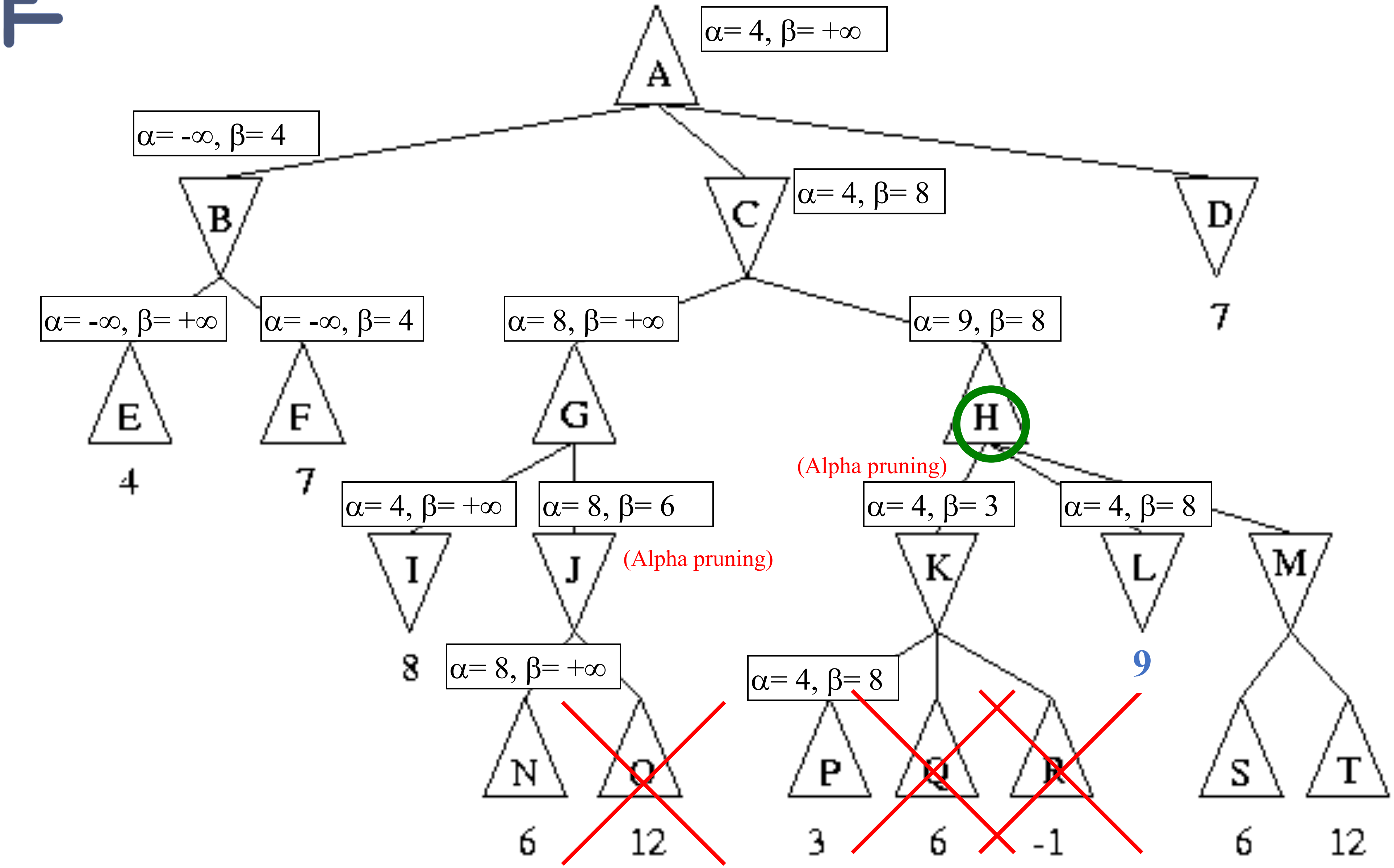

α= 4, β= +∞
A
α= -∞, β= 4
B
C
α= 4, β= 6
D
7
α= -∞, β= +∞
α= -∞, β= 4
α= 8, β= +∞
α= 6, β= 8
E
F
G
H
4
7
(Alpha pruning)
α= 4, β= +∞
α= 8, β= 6
α= 4, β= 3
α= 4, β= 8
α= 5, β= 6
I
J
(Alpha pruning)
K
L
M
8
α= 8, β= +∞
α= 4, β= 8
5
α=5, β=8
α=5, β=6
N
O
P
Q
R
S
T
6
12
3
6
-1
6
12

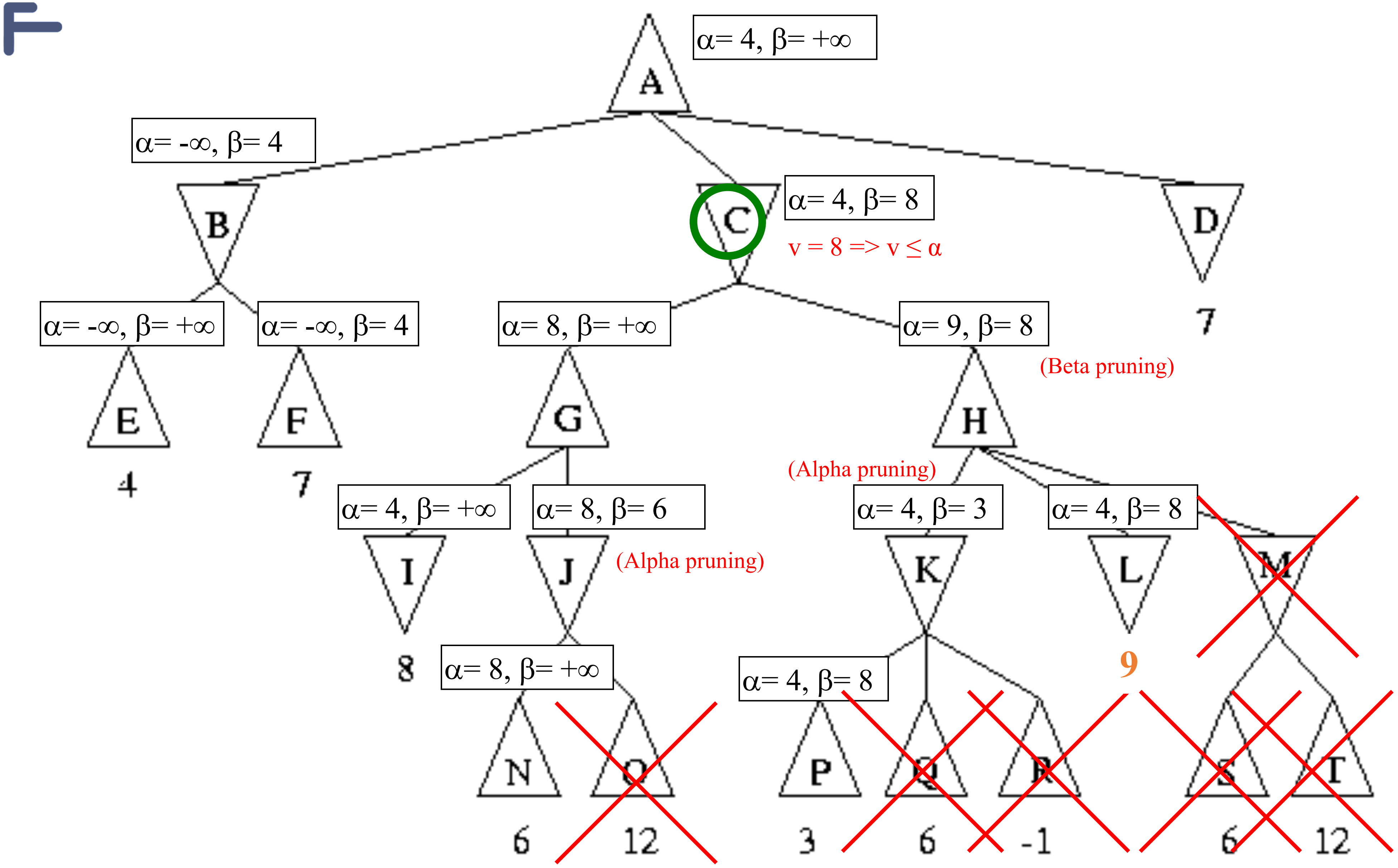

α= 6, β= +∞
A
α= -∞, β= 4
B
α= 4, β= 6
C
D
7
α= -∞, β= +∞
α= -∞, β= 4
α= 8, β= +∞
α= 6, β= 8
E
F
G
H
4
7
(Alpha pruning)
α= 4, β= +∞
α= 8, β= 6
α= 4, β= 3
α= 4, β= 8
α= 5, β= 6
I
J
(Alpha pruning)
K
L
M
8
α= 8, β= +∞
α= 4, β= 8
5
α=5, β=8
α=5, β=6
N
O
P
Q
R
S
T
6
12
3
6
-1
6
12


α= 6, β= +∞
A
α= -∞, β= 4
B
C
α= 4, β= 6
α= 6, β= +∞
D
7
α= -∞, β= +∞
α= -∞, β= 4
α= 8, β= +∞
α= 6, β= 8
E
F
G
H
4
7
(Alpha pruning)
α= 4, β= +∞
α= 8, β= 6
α= 4, β= 3
α= 4, β= 8
α= 5, β= 6
I
J
(Alpha pruning)
K
L
M
8
α= 8, β= +∞
α= 4, β= 8
5
α=5, β=8
α=5, β=6
N
O
P
Q
R
S
T
6
12
3
6
-1
6
12

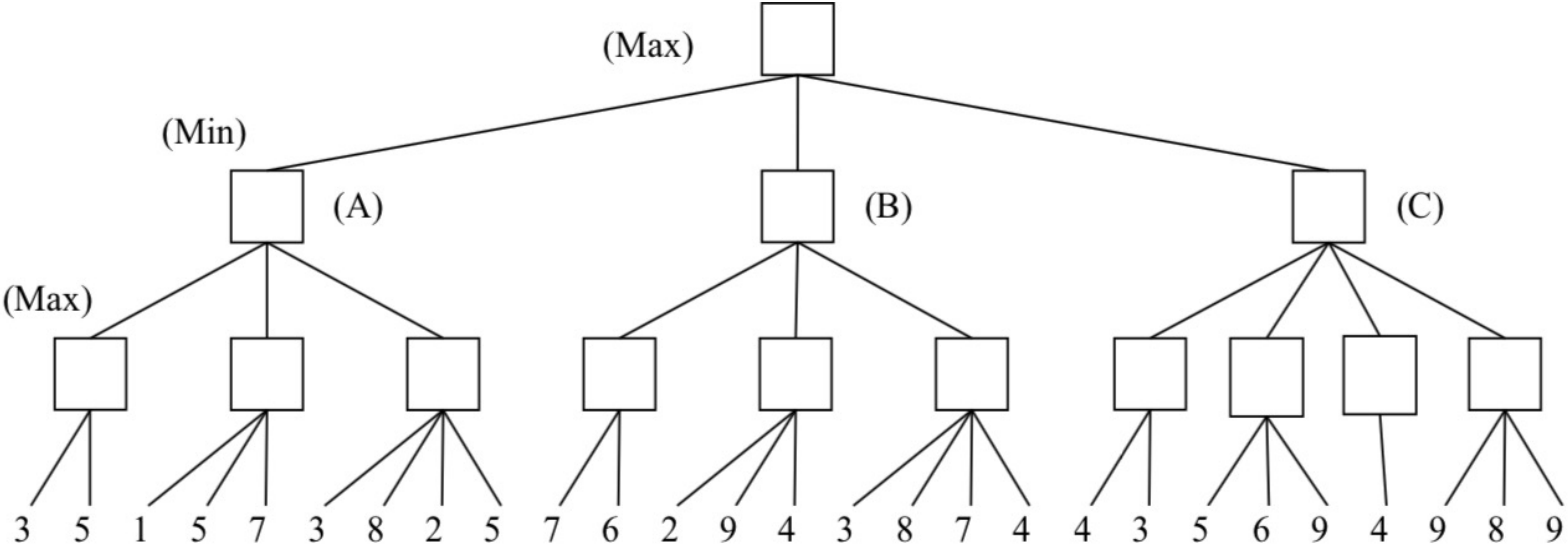

α= 7, β= +∞
A
α= -∞, β= 4
B
C
α= 4, β= 6
α= 6, β= +∞
D
7
α= -∞, β= +∞
α= -∞, β= 4
α= 8, β= +∞
α= 6, β= 8
E
F
G
H
4
7
(Alpha pruning)
α= 4, β= +∞
α= 8, β= 6
α= 4, β= 3
α= 4, β= 8
α= 5, β= 6
I
J
(Alpha pruning)
K
L
M
8
α= 8, β= +∞
α= 4, β= 8
5
α=5, β=8
α=5, β=6
N
O
P
Q
R
S
T
6
12
3
6
-1
6
12


α= 4, β= +∞
A
α= -∞, β= 4
B
α= 4, β= 8
C
D
7
α= -∞, β= +∞
α= -∞, β= 4
α= 8, β= +∞
α= 4, β= 8
E
F
G
H
4
7
(Alpha pruning)
α= 4, β= +∞
α= 8, β= 6
α= 4, β= 3
α= 4, β= 8
I
J
(Alpha pruning)
K
L
M
8
α= 8, β= +∞
9
α= 4, β= 8
N
O
P
Q
R
S
T
6
12
3
6
-1
6
12


α= 4, β= +∞
A
α= -∞, β= 4
B
C
α= 4, β= 8
D
α= -∞, β= +∞
α= -∞, β= 4
α= 8, β= +∞
α= 9, β= 8
7
E
F
G
H
4
7
(Alpha pruning)
α= 4, β= +∞
α= 8, β= 6
α= 4, β= 3
α= 4, β= 8
I
J
(Alpha pruning)
K
L
M
8
α= 8, β= +∞
α= 4, β= 8
9
N
O
P
Q
R
S
T
6
12
3
6
-1
6
12


α= 4, β= +∞
A
α= -∞, β= 4
B
C
α= 4, β= 8
v = 8 => v ≤ α
D
7
α= -∞, β= +∞
α= -∞, β= 4
α= 8, β= +∞
α= 9, β= 8
(Beta pruning)
E
F
G
H
4
7
(Alpha pruning)
α= 4, β= +∞
α= 8, β= 6
α= 4, β= 3
α= 4, β= 8
I
J
(Alpha pruning)
K
L
M
8
α= 8, β= +∞
α= 4, β= 8
9
N
O
P
Q
R
S
T
6
12
3
6
-1
6
12

Latihan Soal



- TERIMA KASIH -