TÌM KIẾM CÓ ĐỐI THỦ
ADVERSARIAL SEARCH

TRÒ CHƠI (GAMES)

- Games are *competitive environments*, in which agents' goals are in conflict, giving rise to ***adversarial search*** problems.
- Typical AI assumptions
  1. Two agents whose actions alternate
  2. Utility values for each agent are the opposite of the other
  3. Fully observable environments
  4. Zero-sum games (result is draw, or a win and a lost)
- Examples: chess, checkers, tic tac toe, …

GAMES VS. SEARCH PROBLEM

- Search – no adversary
  - Solution: Method for finding game
  - Can find optimal solution
  - Evaluation function: estimate of cost from start to goal through given node
- Games - adversary
  - Solution: strategy (specifies move for every possible opponent reply)
  - Optimality depends on opponent
  - Time limits force an *approximate* solution
  - Evaluation function: evaluate “goodness” of game position

BÀI TOÁN

- Giả sử 2 người chơi: một người cầm quân Trắng, một người cầm quân Đen
- Mục tiêu: nghiên cứu chiến lược chọn nước đi cho Trắng (máy tính cầm quân Trắng)
- Vấn đề: Trắng cần tìm *một dãy các nước đi xen kẽ với các nước đi của Đen* tạo thành đường đi từ trạng thái ban đầu tới trạng thái kết thúc là thắng cho Trắng

KHÔNG GIAN TRẠNG THÁI

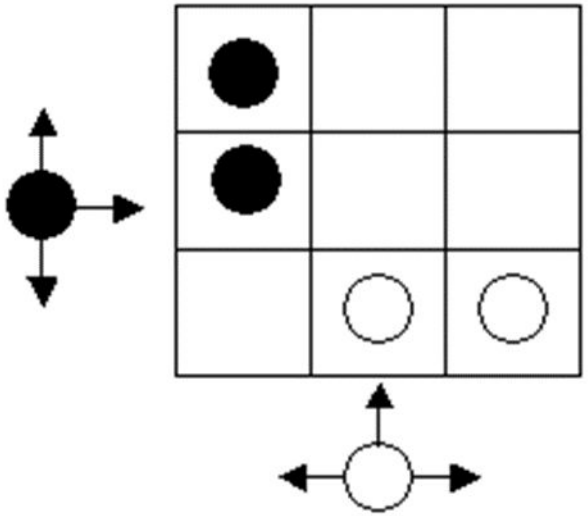
- Trạng thái: sự bố trí các quân của hai bên trên bàn cờ
- Trạng thái ban đầu: sự sắp xếp các quân của hai bên lúc bắt đầu chơi
- Các trạng thái kết thúc: các tình thế mà cuộc chơi dừng (xác định bởi điều kiện dừng)
- Toán tử: các nước đi hợp lệ
- Hàm kết cuộc (utility function): ứng mỗi trạng thái kết thúc với một giá trị nào đó
  - Ví dụ: =1 – thắng, = -1 – thua, = 0 - hòa

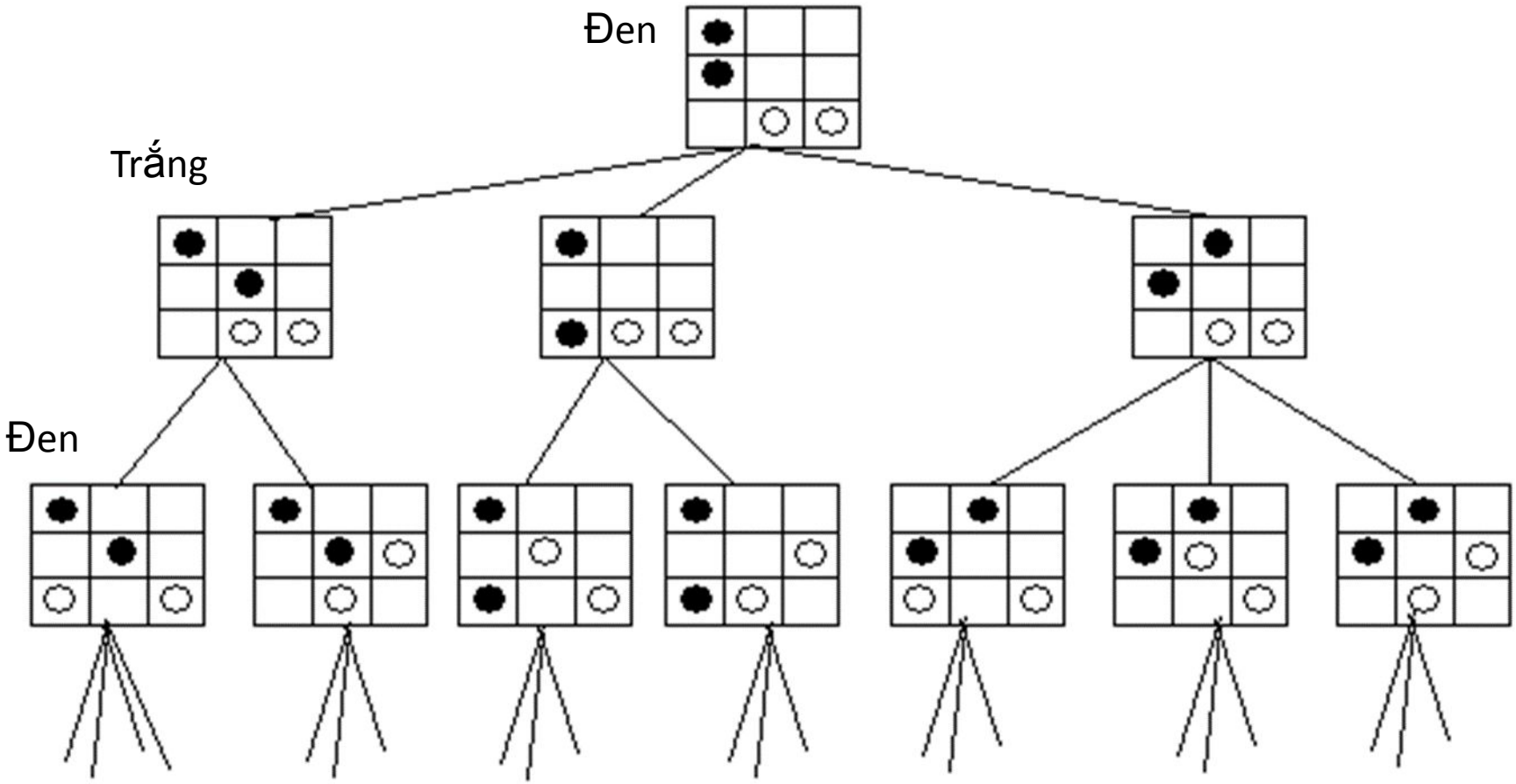
CÂY TRÒ CHƠI (GAME TREE)

- Để thuận lợi, không gian trạng thái được biểu diễn dưới dạng cây trò chơi
  - Gốc: ứng với trạng thái ban đầu
  - Đỉnh ứng với trạng thái mà Trắng sẽ đưa ra nước đi gọi là đỉnh Trắng
  - Đỉnh ứng với trạng thái mà Đen sẽ đưa ra nước đi gọi là đỉnh Đen

CÂY TRÒ CHƠI DODGEM

- Quân đen có thể đi tới ô trống bên phải, ở trên hoặc ở dưới
- Quân trắng có thể đi tới ô trống bên trái, bên phải, ở trên
- Quân đen nếu ở cột ngoài cùng bên phải có thể đi ra ngoài bàn cờ
- Quân trắng nếu ở hàng trên cùng có thể đi ra khỏi bàn cờ
- Ai đưa cả hai quân của mình ra khỏi bàn cờ trước sẽ thắng, hoặc tạo ra tình huống mà đối phương không đi được cũng sẽ thắng



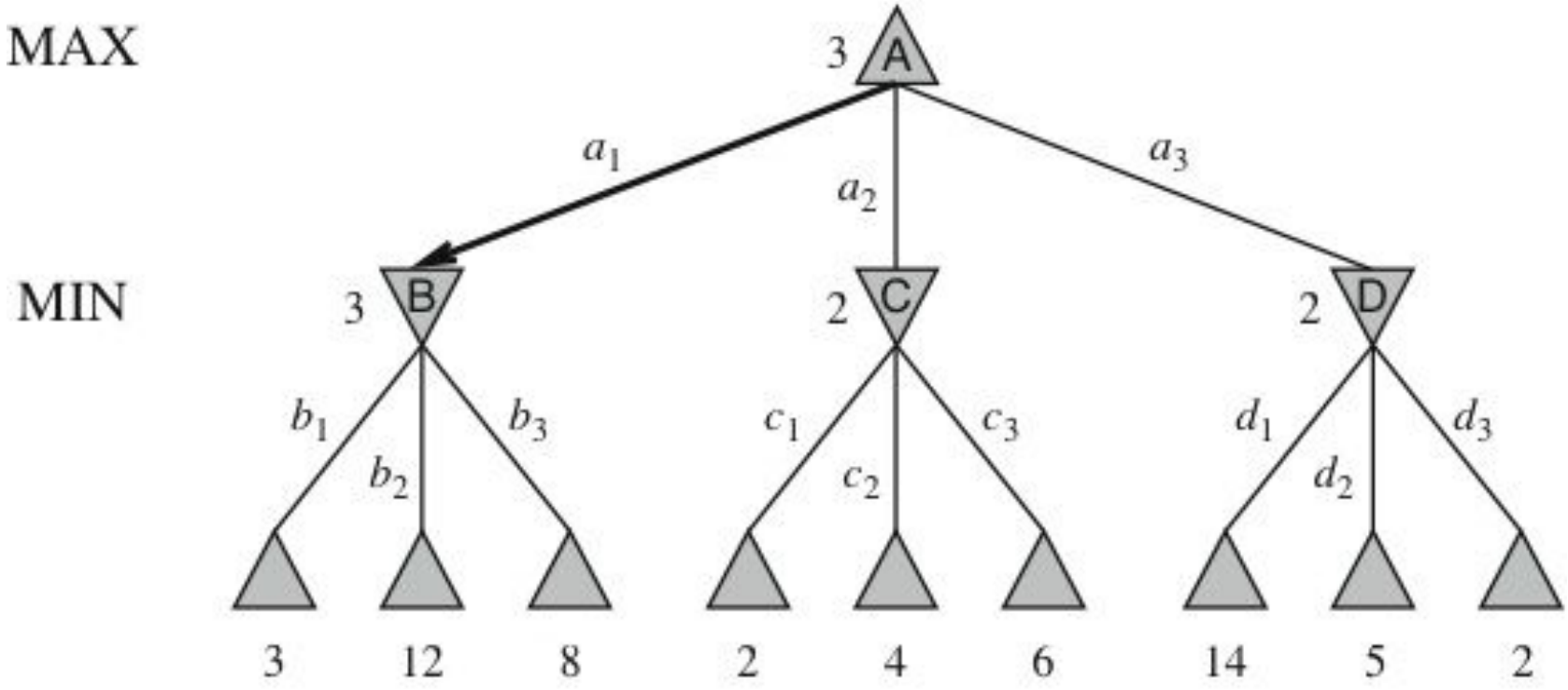


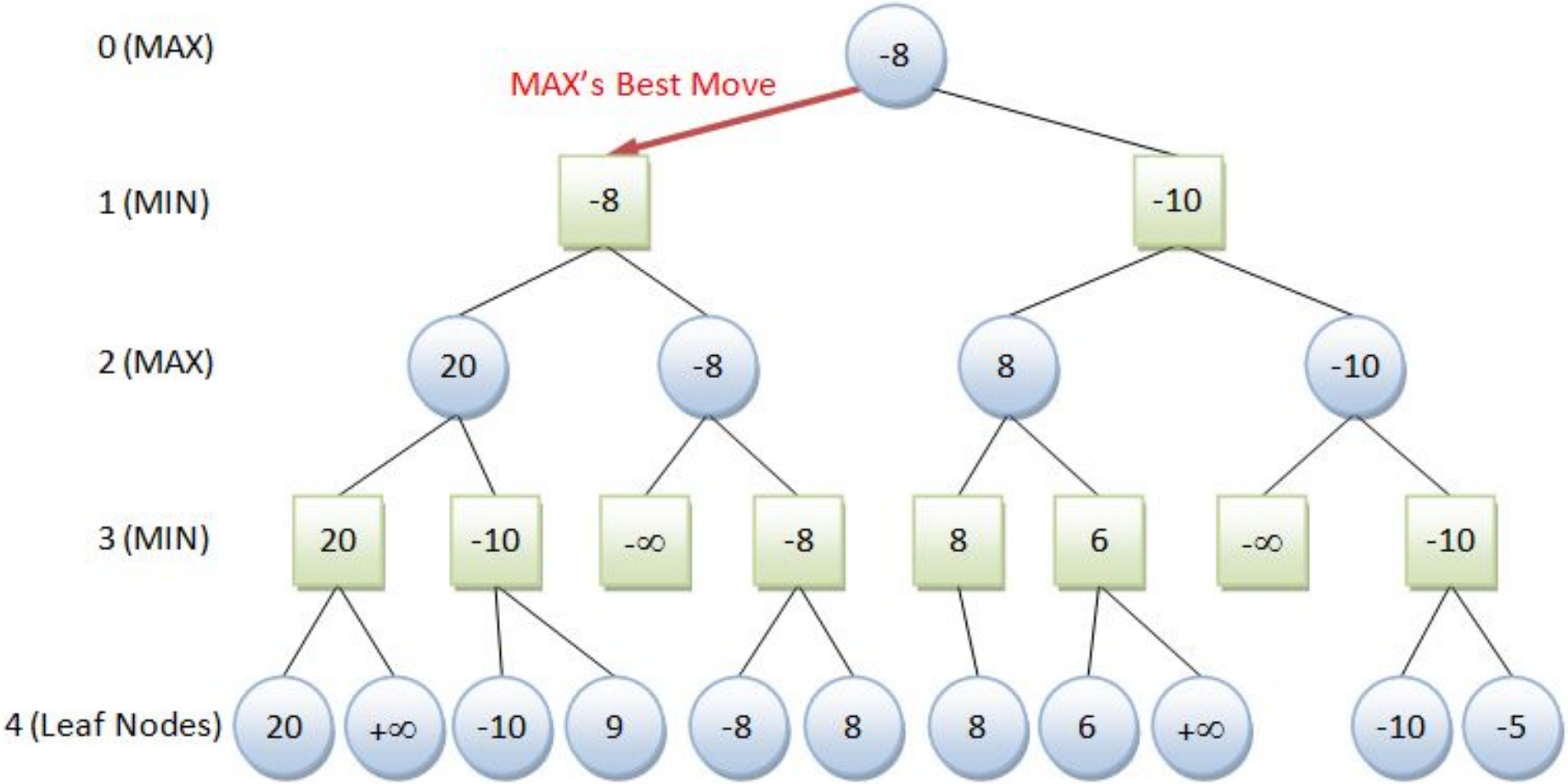
Cây trò chơi Dodgem với Đen đi trước

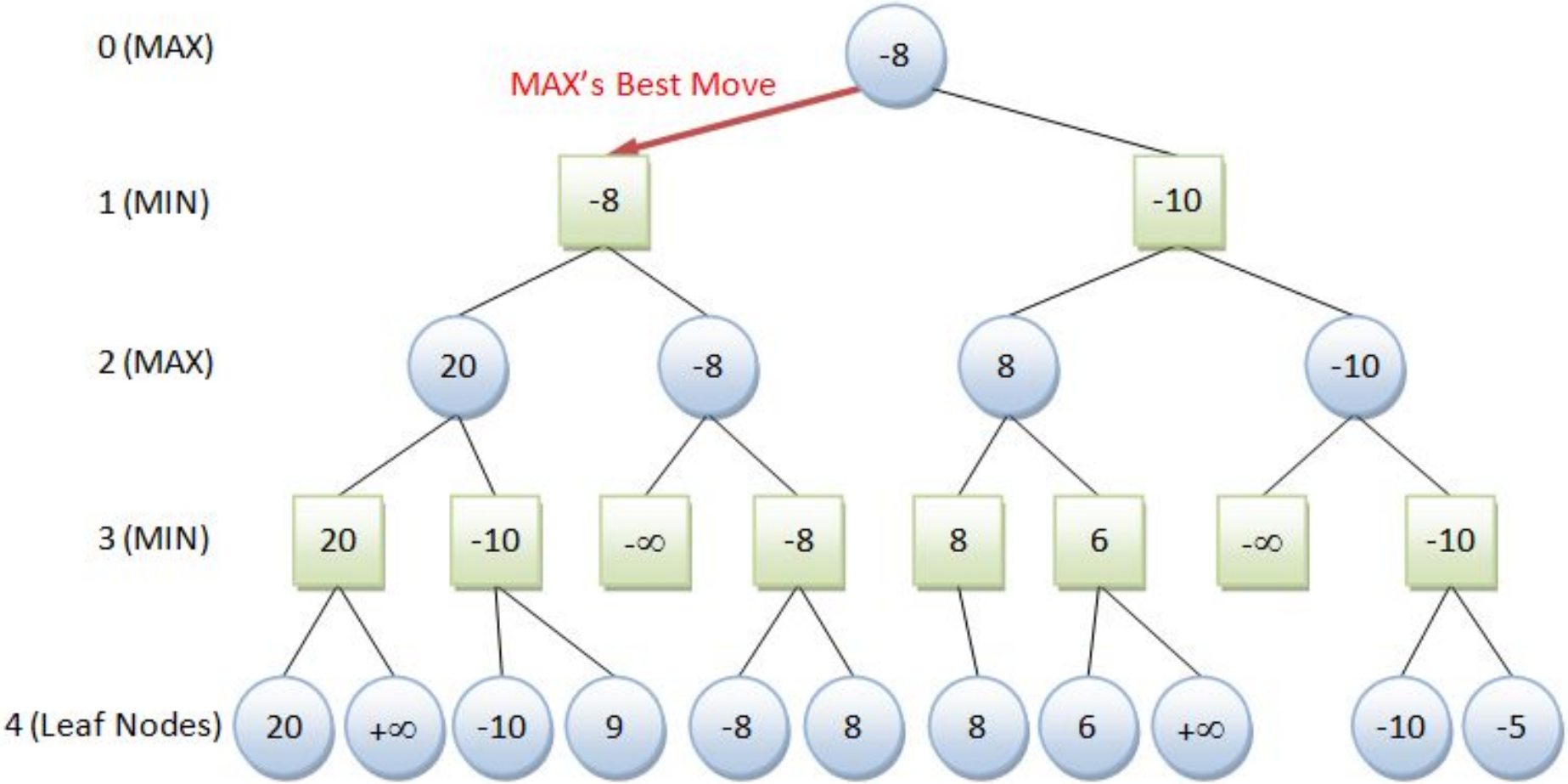
CHIẾN LƯỢC MINIMAX

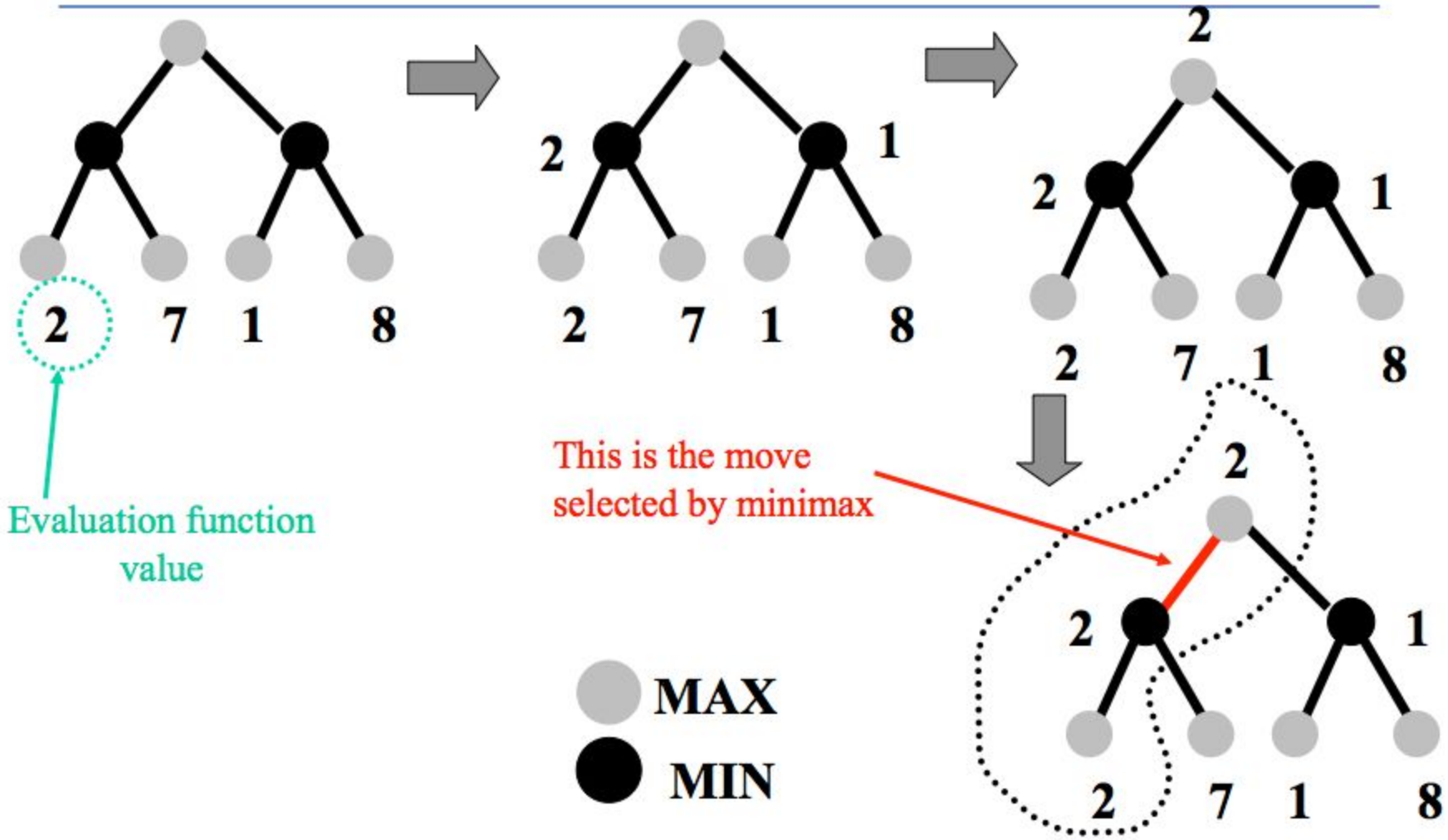
- Hai người chơi:
  - Trắng = MAX
  - Đen = MIN
- Đi ngược từ trạng thái kết thúc
- Gán giá trị cho các trạng thái kết thúc là giá trị của hàm kết cuộc
- Đi ngược từ dưới lên
  - Nếu là đỉnh Trắng (MAX) thì gán giá trị là GTLN của giá trị những nút con của nó
  - Nếu là đỉnh Đen (MIN) thì gán giá trị là GTNN của giá trị những nút con của nó
- Trắng (MAX): chọn nước đi là nút con có giá trị lớn nhất

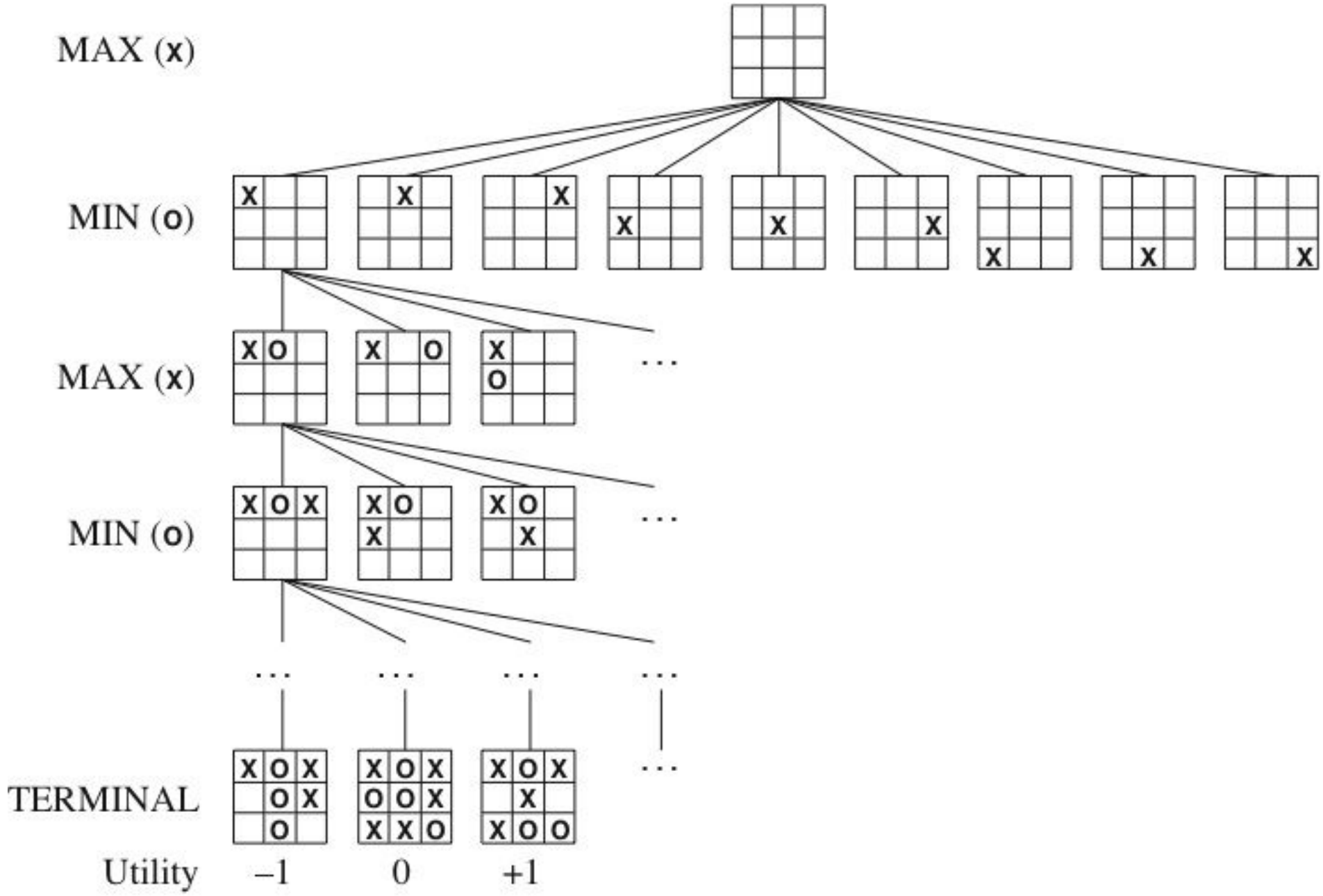
CHIẾN LƯỢC MINIMAX

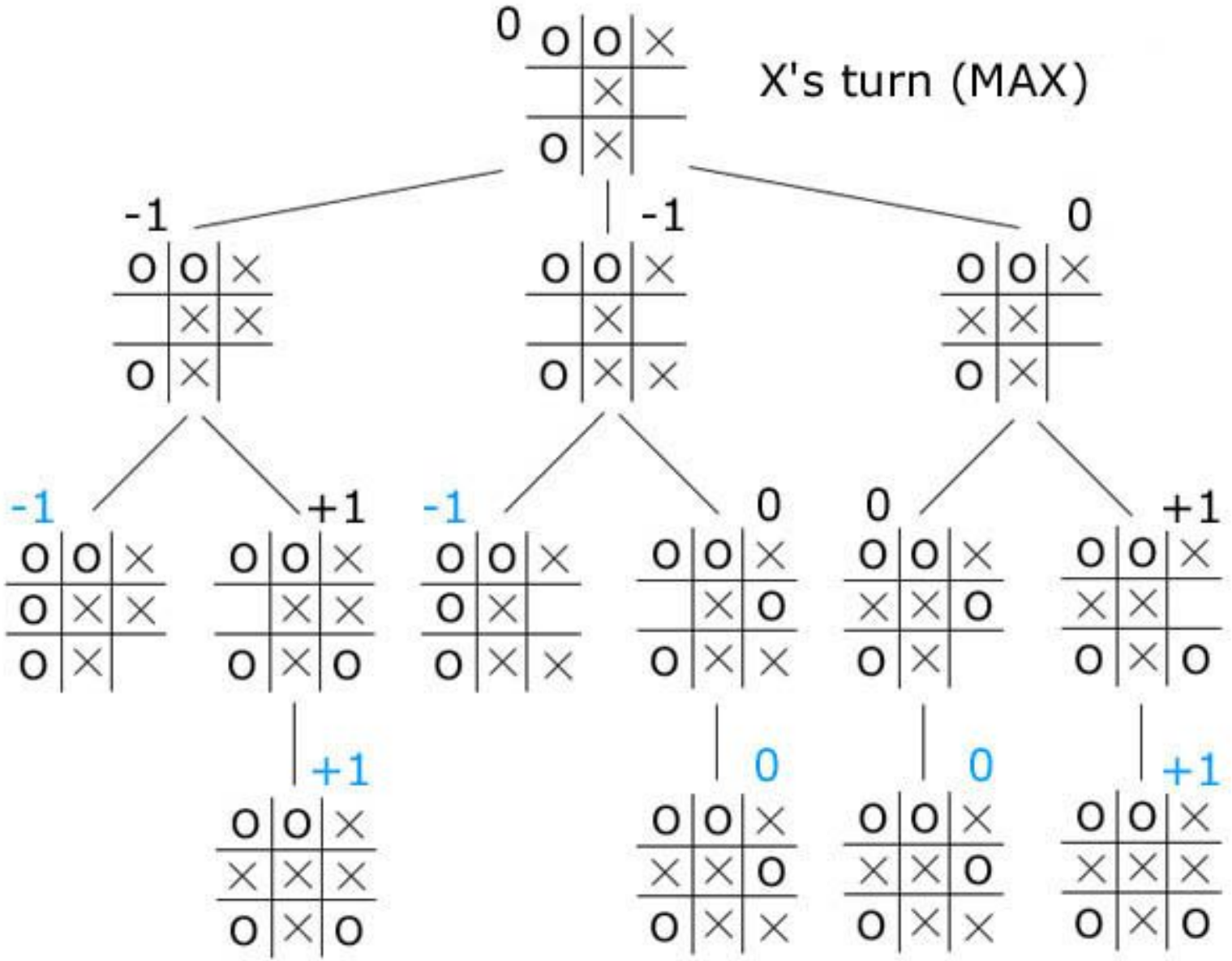


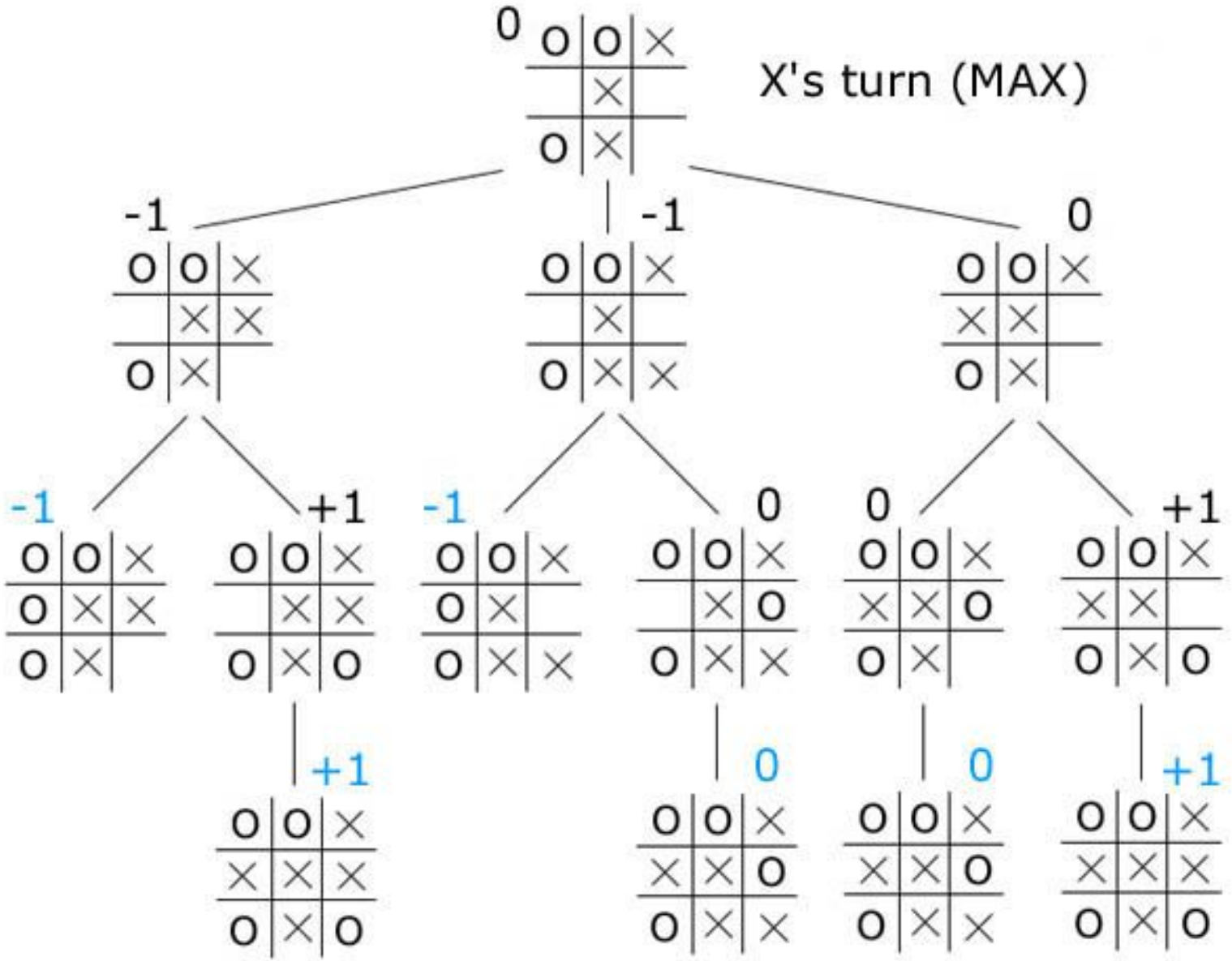

0 (MAX)
1 (MIN)
2 (MAX)
3 (MIN)
4 (Leaf Nodes)
MAX's Best Move
-8
-8
-10
20
-8
8
-10
20
-10
-∞
-8
8
6
-∞
-10
20
+∞
-10
9
-8
8
8
6
+∞
-10
-5


0 (MAX)
1 (MIN)
2 (MAX)
3 (MIN)
4 (Leaf Nodes)
MAX's Best Move
-8
-8
-10
20
-8
8
-10
20
-10
-∞
-8
8
6
-∞
-10
20
+∞
-10
9
-8
8
8
6
+∞
-10
-5


2
7
1
8
Evaluation function value
This is the move selected by minimax
MAX
MIN


MAX (x)
MIN (o)
MAX (x)
MIN (o)
TERMINAL
Utility
–1
0
+1


0
X's turn (MAX)
-1
-1
0
-1
+1
-1
0
0
+1
+1
0
0
+1
MIN
MAX
MIN


0
X's turn (MAX)
-1
-1
0
-1
+1
-1
0
0
+1
+1
0
0
+1
MIN
MAX
MIN

HÀM XÁC ĐỊNH GIÁ TRỊ CHO CÁC ĐỈNH

Function MaxVal(u);

Begin

if u là đỉnh kết thúc then MaxVal = f(u)

else MaxVal = max {MinVal(v) | v là đỉnh con u}

End;

Function MinVal(u);

Begin

if u là đỉnh kết thúc then MinVal = f(u)

else MinVal = min {MaxVal(v) | v là đỉnh con u}

End;

THỦ TỤC CHỌN NƯỚC ĐI CHO TRẮNG (MAX)

```
Procedure Minimax(u,v);
Begin
    Val = -∞ ;
    for mỗi w là đỉnh con của u do
        if val <= MinVal(w) then
            {val = MinVal(w); v = w}
End;
```

CHIẾN LƯỢC MINIMAX

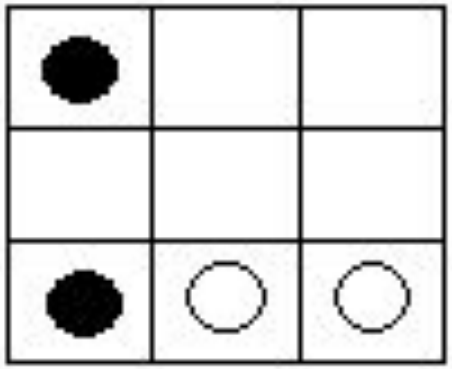
- Là thuật toán tìm kiếm theo độ sâu
- Cho phép chọn được nước đi tối ưu
- Hạn chế: Độ phức tạp lớn
- Giải pháp:
  - Hạn chế không gian tìm kiếm: xem xét cây trò chơi gốc u tới độ cao h nào đó
  - Là của cây trò chơi hạn chế này có thể không phải là trạng thái kết thúc không sử dụng được hàm kết cuộc Sử dụng ***hàm đánh giá***

HÀM ĐÁNH GIÁ

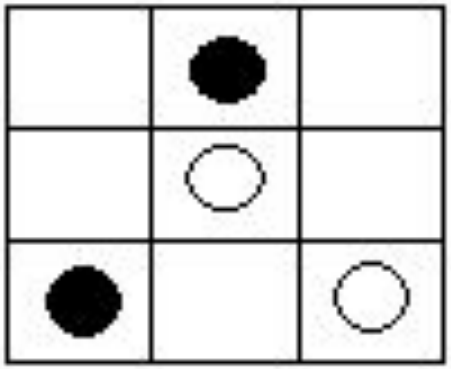
- Hàm đánh giá eval ứng mỗi trạng thái u của trò chơi với một giá trị eval(u)
- Giá trị này đánh giá "độ lợi thế" của trạng thái u
  - eval(u) = 0 – không có lợi thế cho ai cả
  - eval(u) = số dương càng lớn – càng lợi thế cho Trắng (MAX)
  - eval(u) = số âm càng nhỏ - càng lợi thế cho Đen (MIN)
  - eval(u) = +∞ - Trắng thắng
  - eval(u) = -∞ - Đen thắng
- Chất lượng chương trình phụ thuộc vào hàm đánh giá
- Độ tốt của hàm đánh giá thường mâu thuẫn với thời gian để tính nó

VÍ DỤ VỀ HÀM ĐÁNH GIÁ

- Trò chơi Dodgem
  - Xác định 2 trạng thái sau có lợi cho Trắng hay Đen



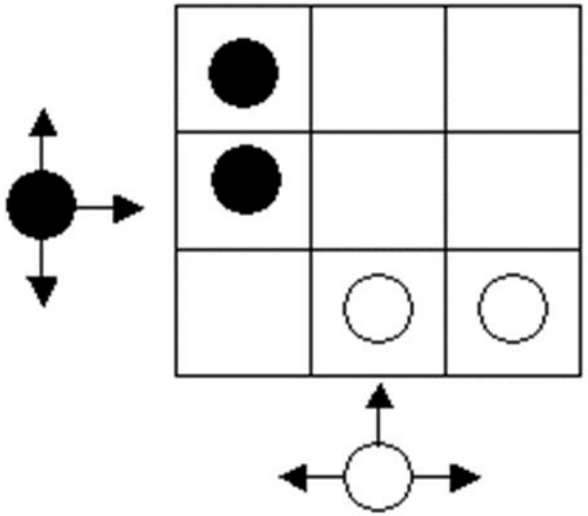
Trạng thái u

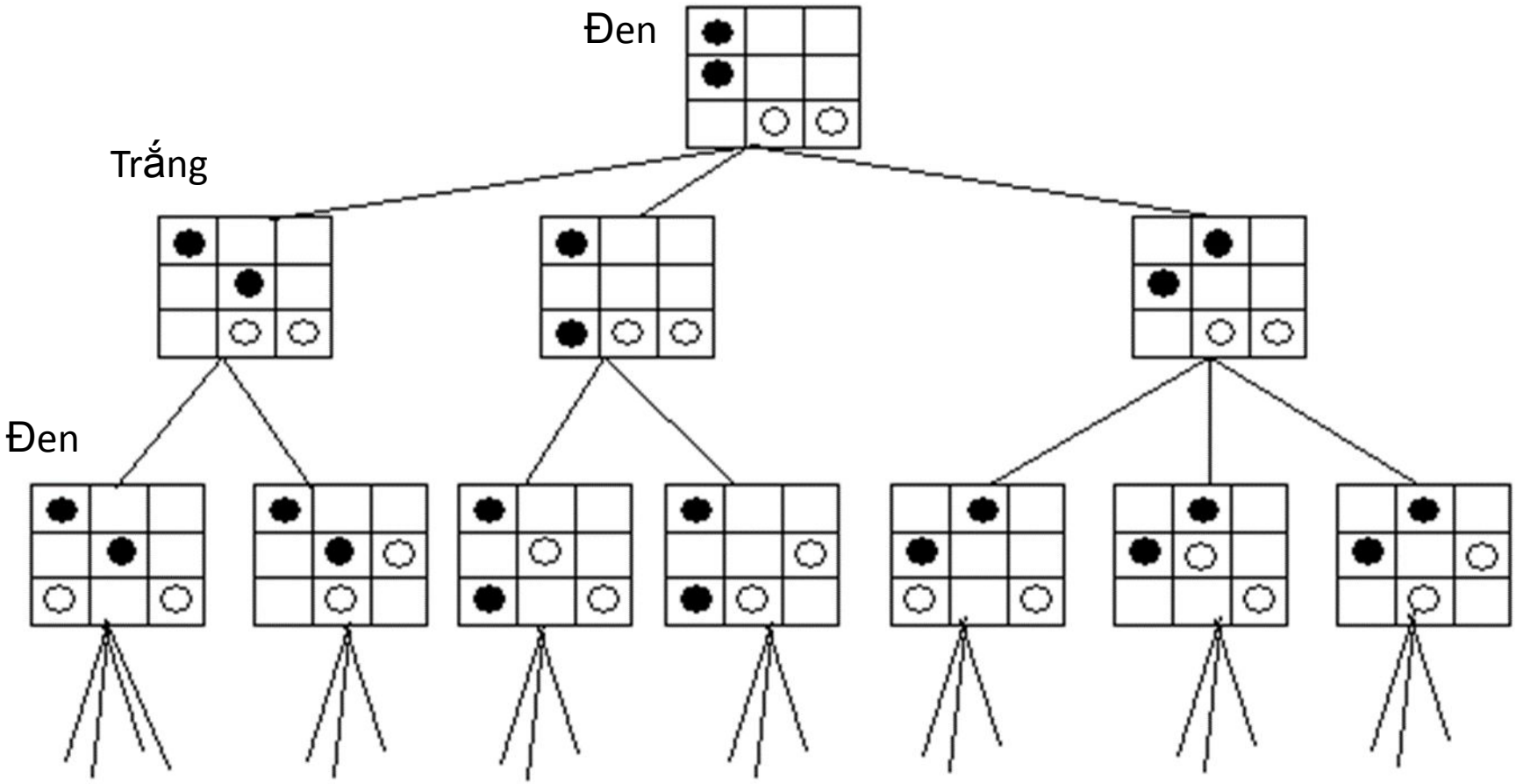


Trạng thái v

CÂY TRÒ CHƠI DODGEM

- Quân đen có thể đi tới ô trống bên phải, ở trên hoặc ở dưới
- Quân trắng có thể đi tới ô trống bên trái, bên phải, ở trên
- Quân đen nếu ở cột ngoài cùng bên phải có thể đi ra ngoài bàn cờ
- Quân trắng nếu ở hàng trên cùng có thể đi ra khỏi bàn cờ
- Ai đưa cả hai quân của mình ra khỏi bàn cờ trước sẽ thắng, hoặc tạo ra tình huống mà đối phương không đi được cũng sẽ thắng





Cây trò chơi Dodgem với Đen đi trước

1. Cho điểm các vị trí

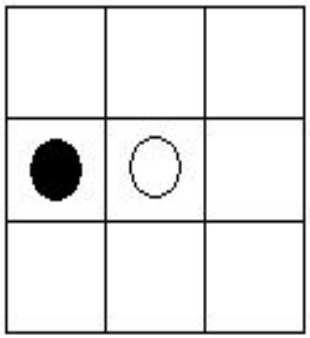
30	35	40
15	20	25
0	5	10

Giá trị quân trắng

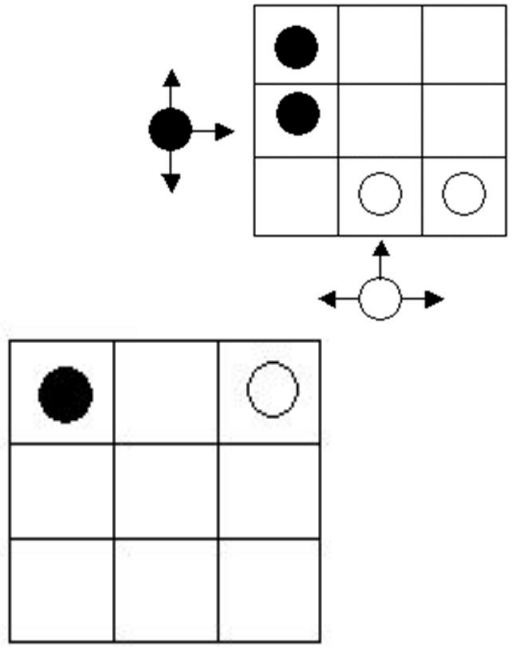
-10	-25	-40
-5	-20	-35
0	-15	-30

Giá trị quân đen

2. Cho điểm cản
   - Cản trực tiếp: 40 điểm
   - Cản gián tiếp: 30 điểm



Trắng cản trực tiếo đen được 40 điểm



Trắng cản gián tiếp đen được 30 điểm

```
Function MaxVal(u,h);
Begin
    if h=0 hoặc u là đỉnh kết thúc then MaxVal = eval(u)
    else MaxVal = max {MinVal(v,h-1) |  v là đỉnh con u}
End;
Function MinVal(u,h);
Begin
    if h=0 hoặc u là đỉnh kết thúc then MinVal = eval(u)
    else MinVal = min {MaxVal(v,h-1) |  v là đỉnh con u}
End;
```

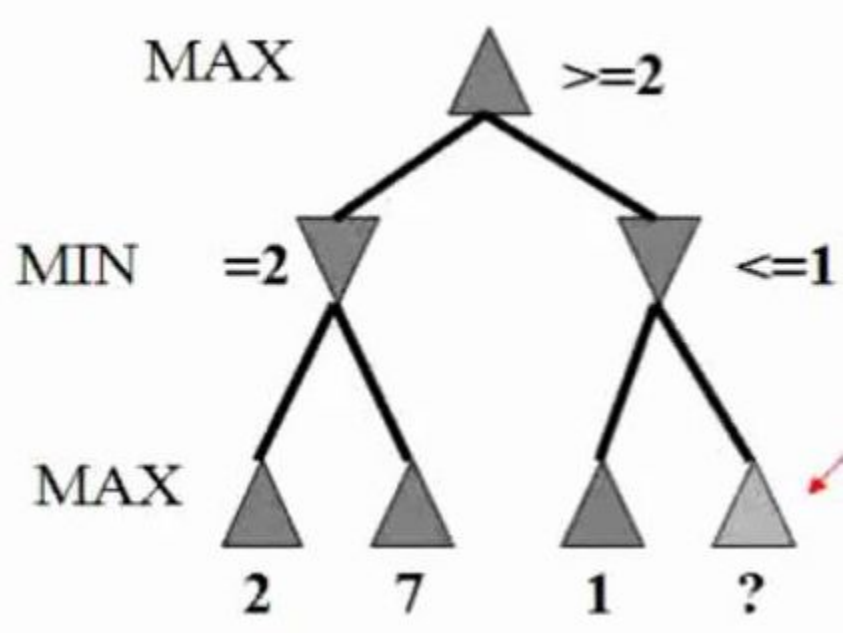
```
Procedure Minimax(u,v,h);
Begin
    val = -∞ ;
    for mỗi w là đỉnh con của u do
        if val <= MinVal(w,h-1) then
            {val = MinVal(w,h-1); v = w}
End;
```

TIC TAC TOE EVALUATION FUNCTION

- **Calculating A Score**
- How do we actually give a score to the state of of the board? This is the only part of the algorithm that ends up being heuristic. You have to choose a heuristic board evaluation function to calculate this value. I chose to calculate a score for each line (the three rows, the three columns, and two diagonals) on the board, and then sum these values. I scored each line as follows:
- +100 for each three-in-a-row for the AI
- +10 for each two-in-a-row (and empty cell) for the AI
- +1 for each one-in-a-row (and two empty cells) for the AI
- -1 for each one-in-a-row (and two empty cells) for the other player
- -10 for each two-in-a-row (and empty cell) for the other player
- -100 for each three-in-a-row for the other player
- 0 for all other states

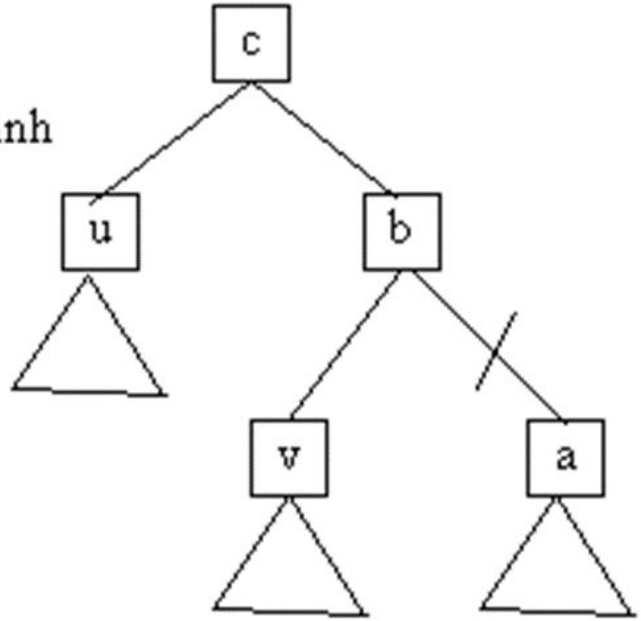
PHƯƠNG PHÁP CẮT CỤT ALPHA-BETA

- Phương pháp cho phép giảm bớt số đỉnh cần đánh giá của cây trò chơi gốc u (độ sâu h) mà không ảnh hưởng tới sự đánh giá u



TƯ TƯỞNG CỦA PHƯƠNG PHÁP

Nếu eval(u)>eval(v)
thì không cần đi xuống để đánh giá đỉnh a nữa



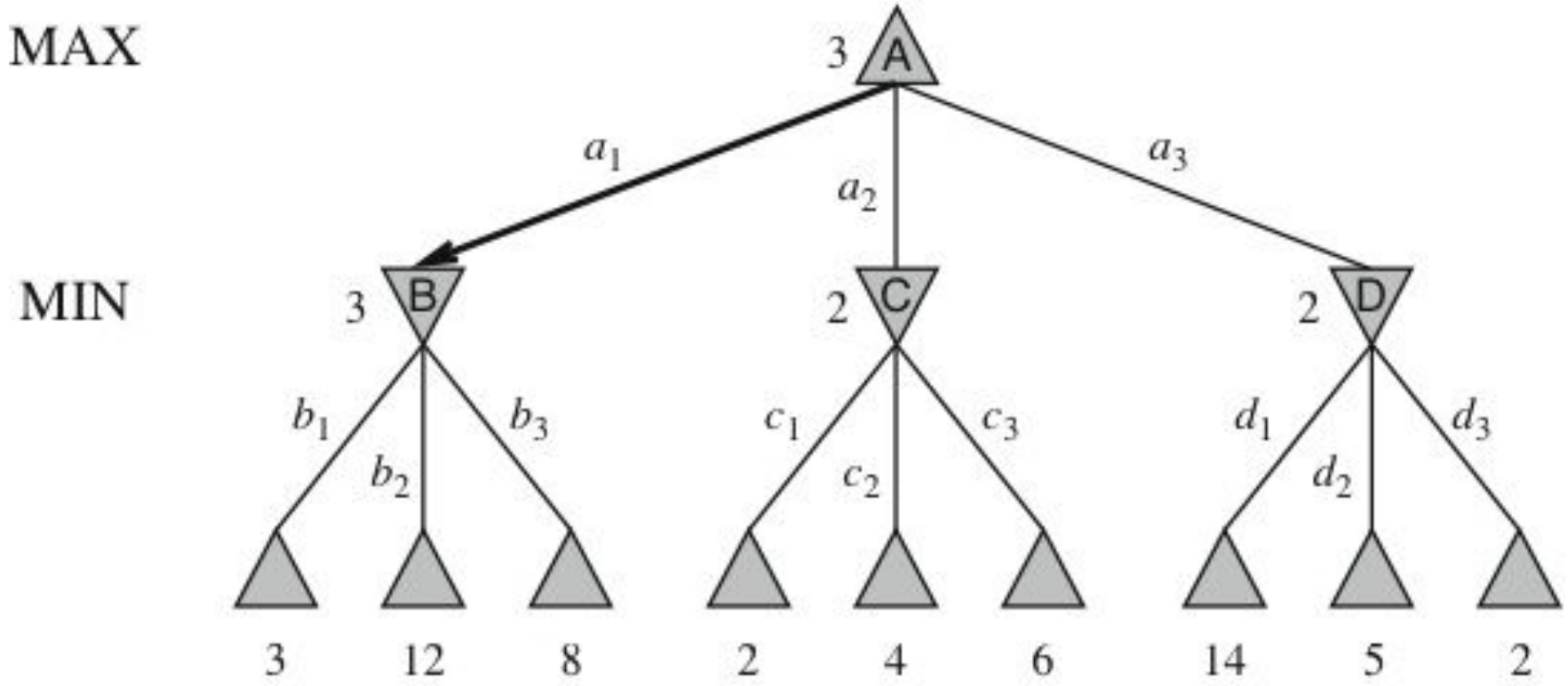
max – ít nhất bằng eval(u)

min - nhiều nhất bằng eval(v)

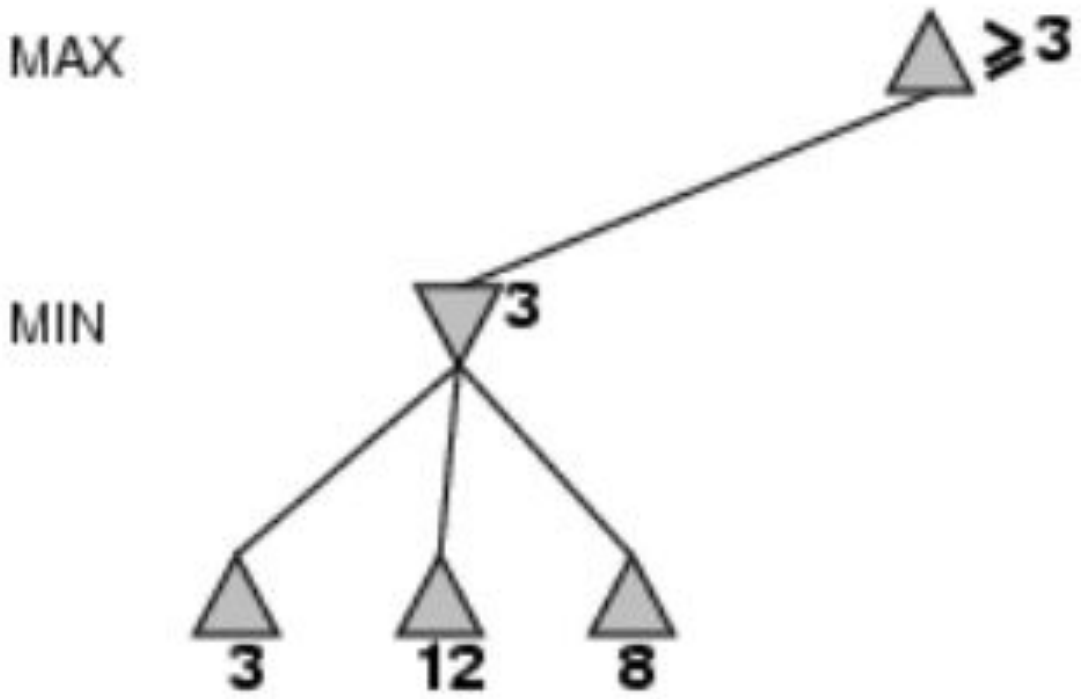
max

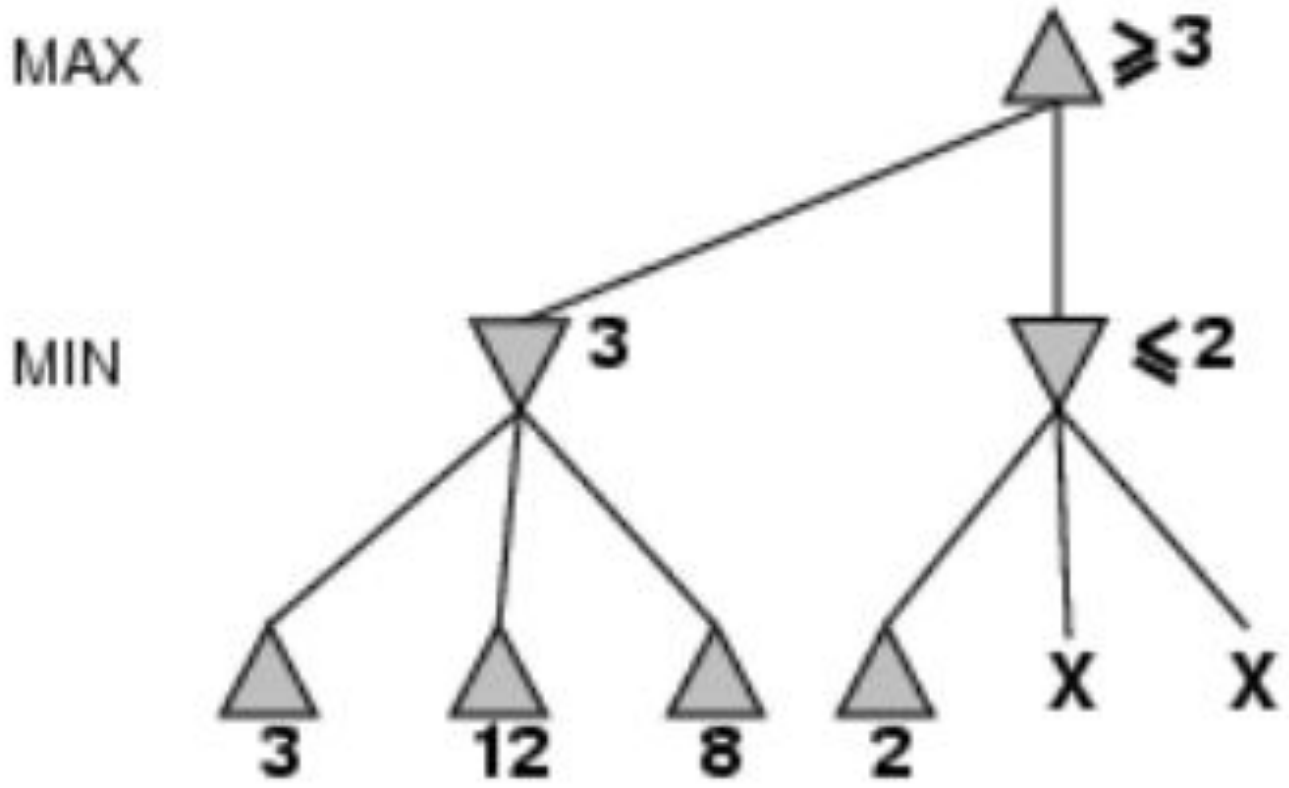
Cắt bỏ cây con gốc a nếu eval(u)>eval(v)

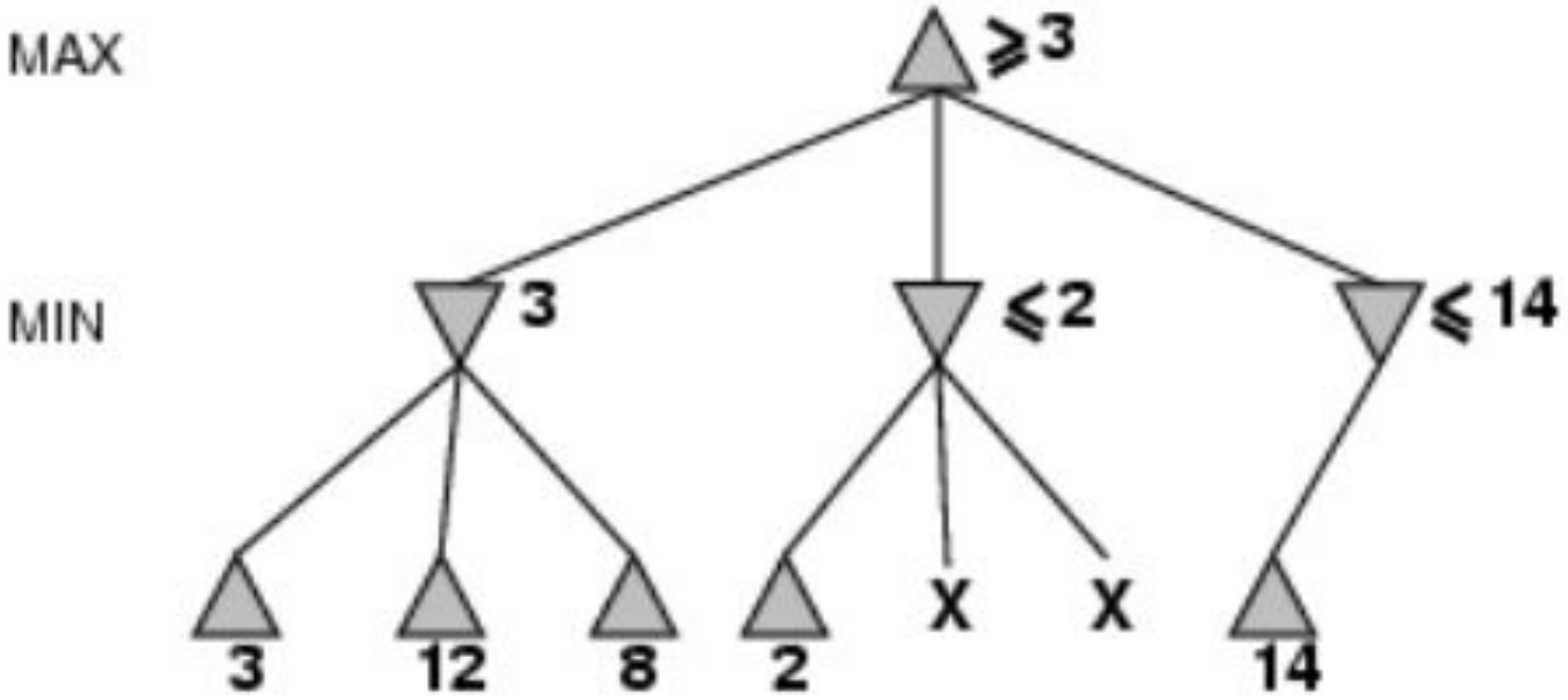
- Ví dụ: Chiến lược Minimax

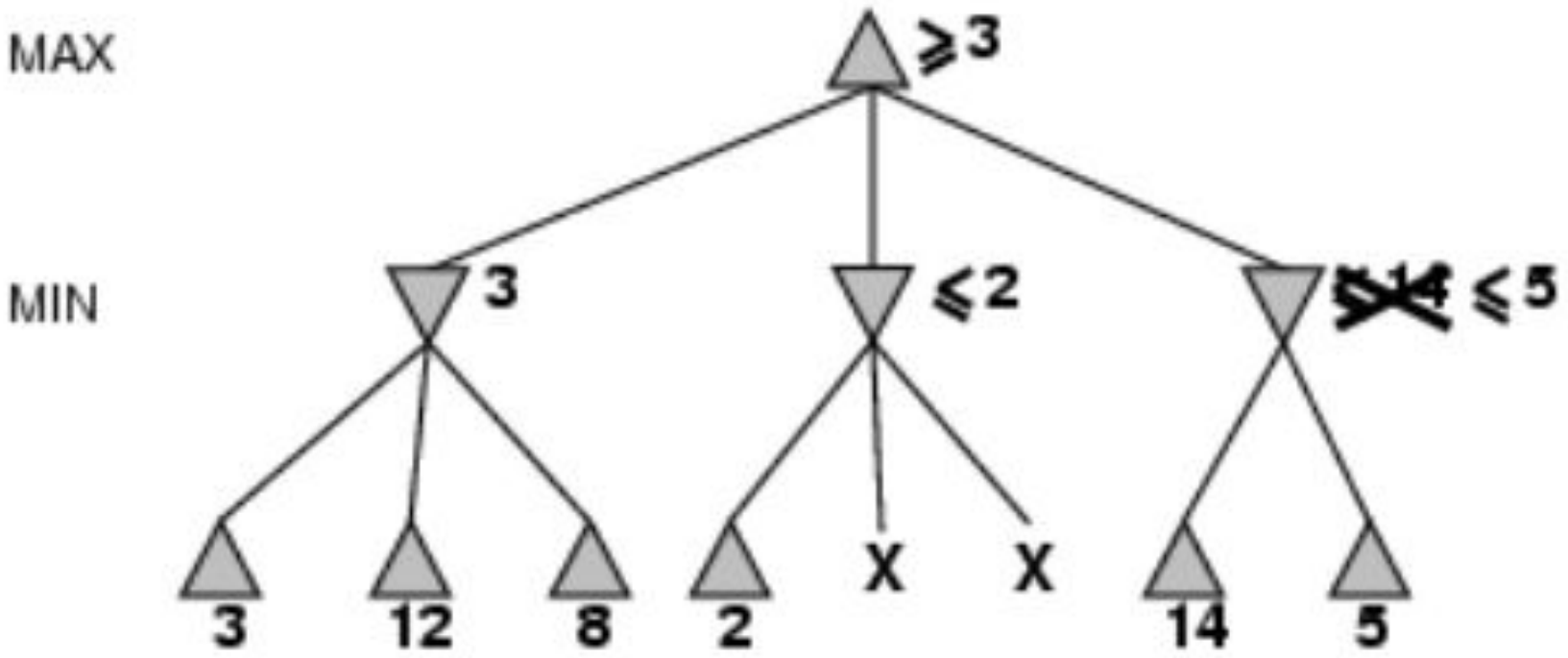


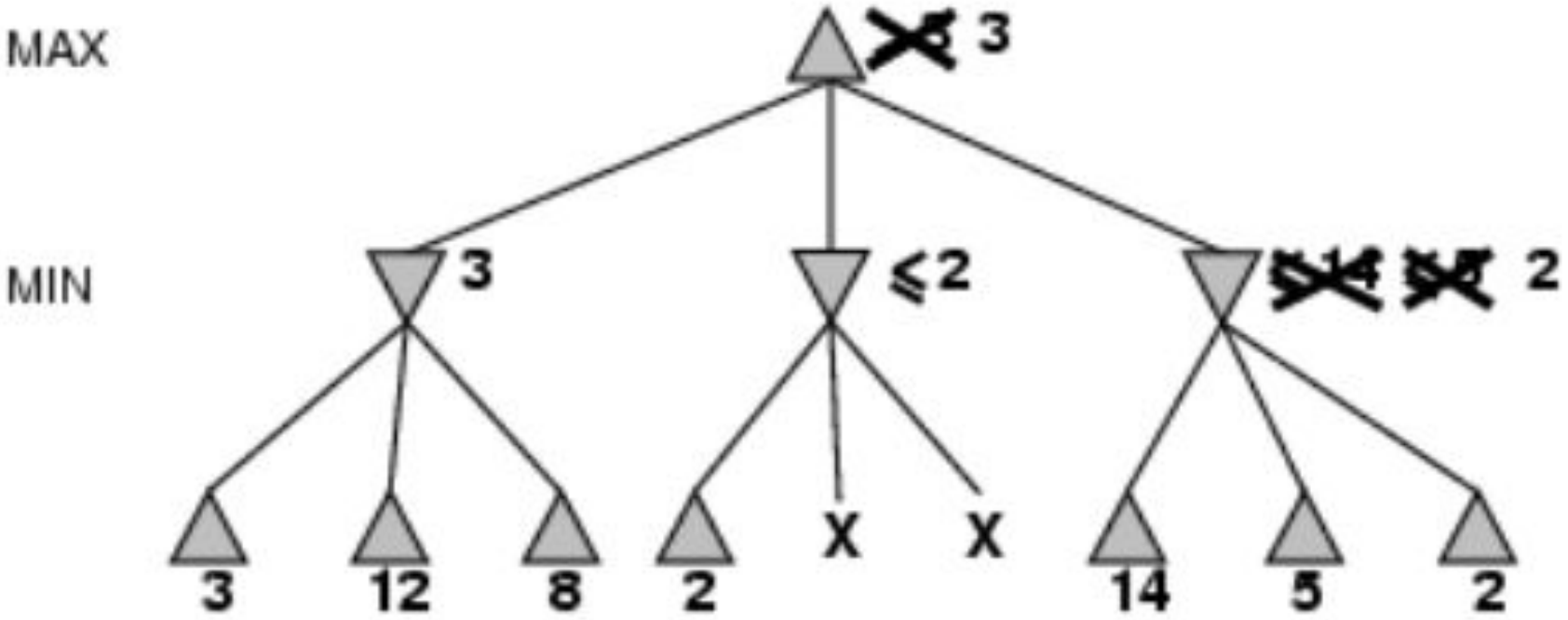
- Phương pháp cắt cụt Alpha-Beta




MAX
MIN
≥3
3
≤2
3
12
8
2
X
X


MAX
MIN
≥3
3
≤2
≤14
3
12
8
2
X
X
14


MAX
MIN
≥3
3
≤2
≤5
3
12
8
2
X
X
14
5


MAX
MIN
3
3
≤2
2
X
X
3
12
8
2
14
5
2

KỸ THUẬT CÀI ĐẶT

- Tìm kiếm theo chiều sâu (DFS)
- Nút MAX có một giá trị α (luôn tăng)
- Nút MIN có một giá trị β (luôn giảm)
- Tìm kiếm kết thúc khi
  - Nút MIN nào có $\beta \leq \alpha$ của bất kỳ nút cha MAX nào (β – cut)
  - Nút MAX nào có $\alpha \geq \beta$ của bất kỳ nút cha MIN nào (α – cut)

```
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
```

```
Function MaxVal(u, α, β);
Begin
    if u là lá của cây hạn chế hoặc u là đỉnh kết thúc
    then MaxVal = eval(u)
    else for mỗi đỉnh v là con của u do
        {α = max [α, MinVal(v, α, β)];
        if α≥β then exit }// cắt bỏ cây con từ các đỉnh v còn lại
    MaxVal = α
End;
```

```
Function MinVal(u, α, β);
Begin
    if u là lá của cây hạn chế hoặc u là đỉnh kết thúc
    then MinVal = eval(u)
    else for mỗi đỉnh v là con của u do
        {β = min[β, MaxVal(v, α, β)];
         if α≥β then exit }// cắt bỏ cây con từ các đỉnh v còn lại
     MinVal = β
End;
```

```
Procedure Alpha_beta(u,v)
Begin
    α= -∞; β= +∞
    for mỗi đỉnh w là con của u do
        if α<= MinVal(w,α,β) then
            {α = MinVal(w,α,β);
             v = w;}
End;
```