Trường Đại học Khoa học Tự nhiên
Khoa Công nghệ Thông tin

TÀI LIỆU LÝ THUYẾT TRÍ TUỆ NHÂN TẠO

Chủ đề 4

TÌM KIẾM ĐỐI KHÁNG

Giảng viên: ThS. Vũ Thanh Hưng
Email: vthung@fit.hcmus.edu.vn

Biên soạn: ThS. Nguyễn Ngọc Thảo

HKii/2011-2012

NỘI DUNG

- Tổng quan về trò chơi
- Quyết định tối ưu trong trò chơi
  - Thuật toán MINIMAX
  - Tỉa nhánh $\alpha - \beta$
- Quyết định thời gian thực và không hoàn hảo
  - Hàm lượng giá
  - Tìm kiếm cắt ngang

TỔNG QUAN VỀ TRÒ CHƠI

GIỚI THIỆU TRÒ CHƠI

- Liên kết chặt chẽ với trí tuệ con người.
- Trò chơi phát triển từ những ngày đầu của Trí tuệ nhân tạo.
  - *Ví dụ: từ năm 1950, bài toán đánh cờ đã lần lượt được giải quyết bởi Konrad Zuse, Claude Shannon, Norbert Wiener, và Alan Turing.*
- Phát triển nhanh chóng và đạt đến mức cạnh tranh với con người.

GIỚI THIỆU TRÒ CHƠI

- Trò chơi là bài toán thú vị vì khó tìm được lời giải tốt.
  - *Ví dụ: cờ vua có hệ số phân nhánh khoảng 35 và mỗi người chơi thường đi 50 bước $\Rightarrow$ cây tìm kiếm có khoảng 10^{154} nút.*
- Tương tự như thế giới thực, trò chơi đòi hỏi
  - Khả năng ra quyết định khi không thể tính toán tối ưu
  - Chạy trong thời gian chấp nhận được.

CÁC DẠNG TRÒ CHƠI

- Trò chơi đối kháng
  - Thắng lợi của một bên là thất bại của bên còn lại.
- Trò chơi hợp tác
  - Các bên tham gia có chung mối quan tâm và hàm lợi ích.
- Nhiều trò chơi nằm giữa khoảng cách từ đối kháng đến hợp tác.

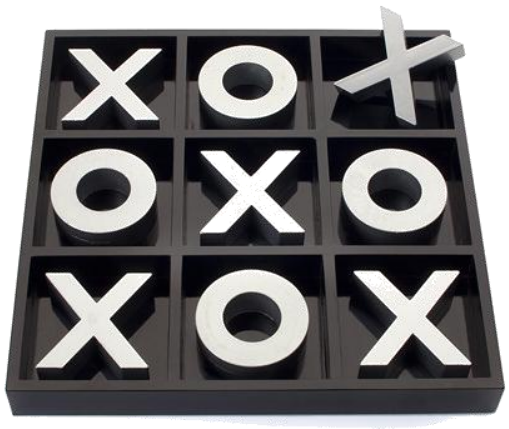
Chúng ta chỉ quan tâm đến trò chơi đối kháng trong bài học này.

TÌM KIẾM TRÒ CHƠI

- Đặc điểm của tìm kiếm trò chơi
  - Thứ tự ra quyết định: điều khiển được
  - Quyết định của đối phương: không điều khiển được
- Tính ngẫu nhiên: hành vi của đối phương là không biết trước $\Rightarrow$ không chắc chắn
- Mục tiêu của tìm kiếm: tối đa hóa hàm lợi ích cho bản thân.

TRÒ CHƠI ĐỐI KHÁNG

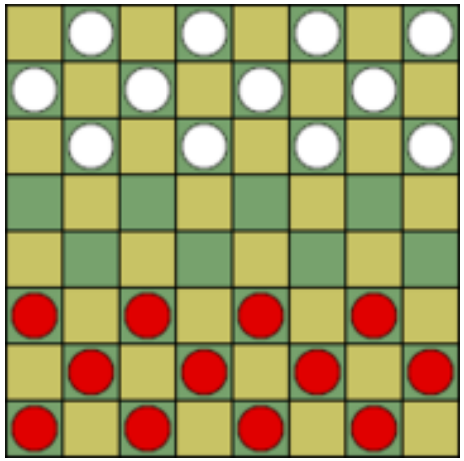
- Tic – Tac – Toe
  - Một bên sử dụng dấu X, bên còn lại dùng dấu O.
  - Người chơi lần lượt đặt dấu X (hoặc O) vào các ô trống trên bảng.



  - Trò chơi kết thúc khi có một bên tạo được một hàng ngang (hoặc hàng dọc, hàng chéo) chỉ chứa toàn kí hiệu X (hoặc O)

TRÒ CHƠI ĐỐI KHÁNG

- Checkers
  - Người chơi lần lượt di chuyển quân của mình theo đường chéo, 1 lần 1 ô.



  - Quân đối phương trước mặt $\Rightarrow$ được nhảy qua nếu có ô trống tại vị trí nhảy tới và ăn quân này.
  - Ván cờ kết thúc khi một trong hai bên không còn quân để đi.

TRÒ CHƠI ĐỐI KHÁNG

- Checkers
  - Năm 1952, Arthur Samuel (IBM) viết các chương trình chơi cờ đầu tiên
  - Năm 1994, Chinook đánh bại Tinsley, vô địch thế giới, thua 3 ván trong 42 năm!
  - Bí quyết:
    - Tìm kiếm tất cả nước đi khi có 8 quân hay ít hơn
    - Tất cả được nhận diện thông tin thắng, thua, hòa
    - Lưu trữ 444 tỷ vị trí với hàng terabyte bộ nhớ
  - Chinook (weblink):

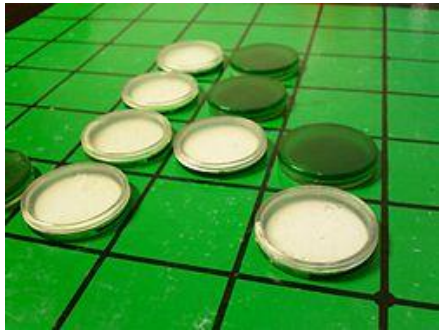
TRÒ CHƠI ĐỐI KHÁNG

- Cờ vua
  - 1997, Deep Blue đánh bại Gary Kasparov trong một trận đấu 6 ván

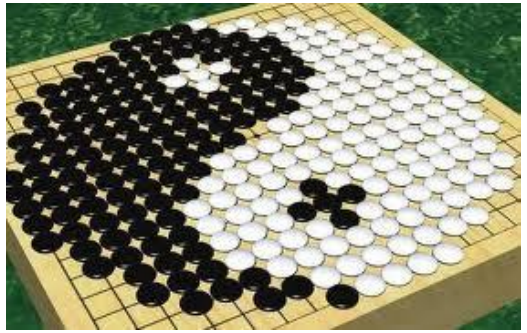
- Bí quyết
  - Tìm kiếm vét cạn với độ sâu cao nhất có thể
  - Tính được 126.000.000 nước đi mỗi giây so với 2 của Kasparov
  - Sử dụng tìm kiếm alpha-beta lặp sâu dần
  - Hàm lượng giá cực kỳ phức tạp (8000 đặc trưng)

TRÒ CHƠI ĐỐI KHÁNG

- Một số trò chơi đối kháng khác
  - Othello (Reversi): năm 1997, chương trình Logistello đánh bại vô địch thế giới
  - Cờ vây (GO): vẫn chưa có chương trình hiệu quả (do độ phân nhánh quá lớn, b > 300)



Othello (Reversi)



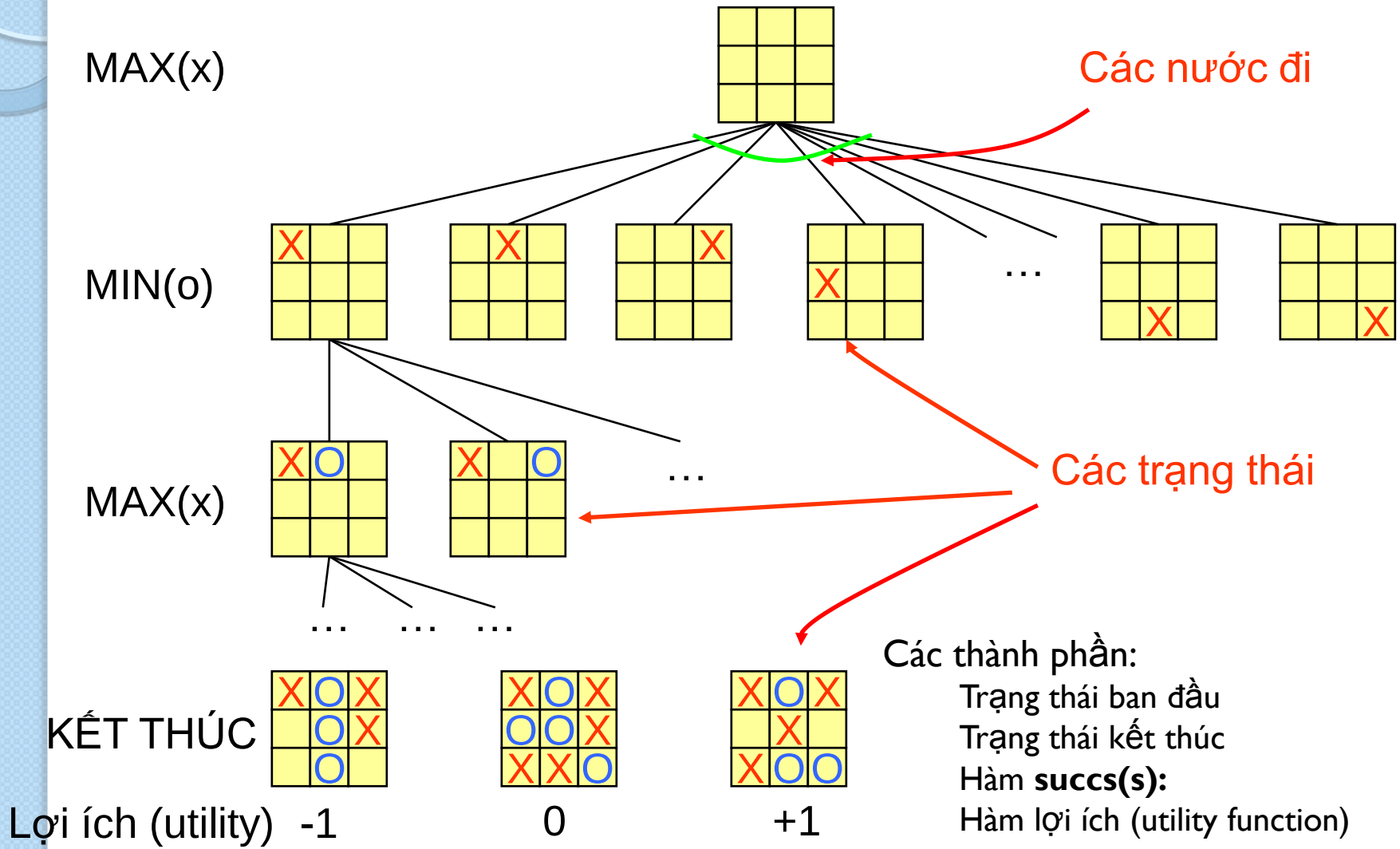
GO

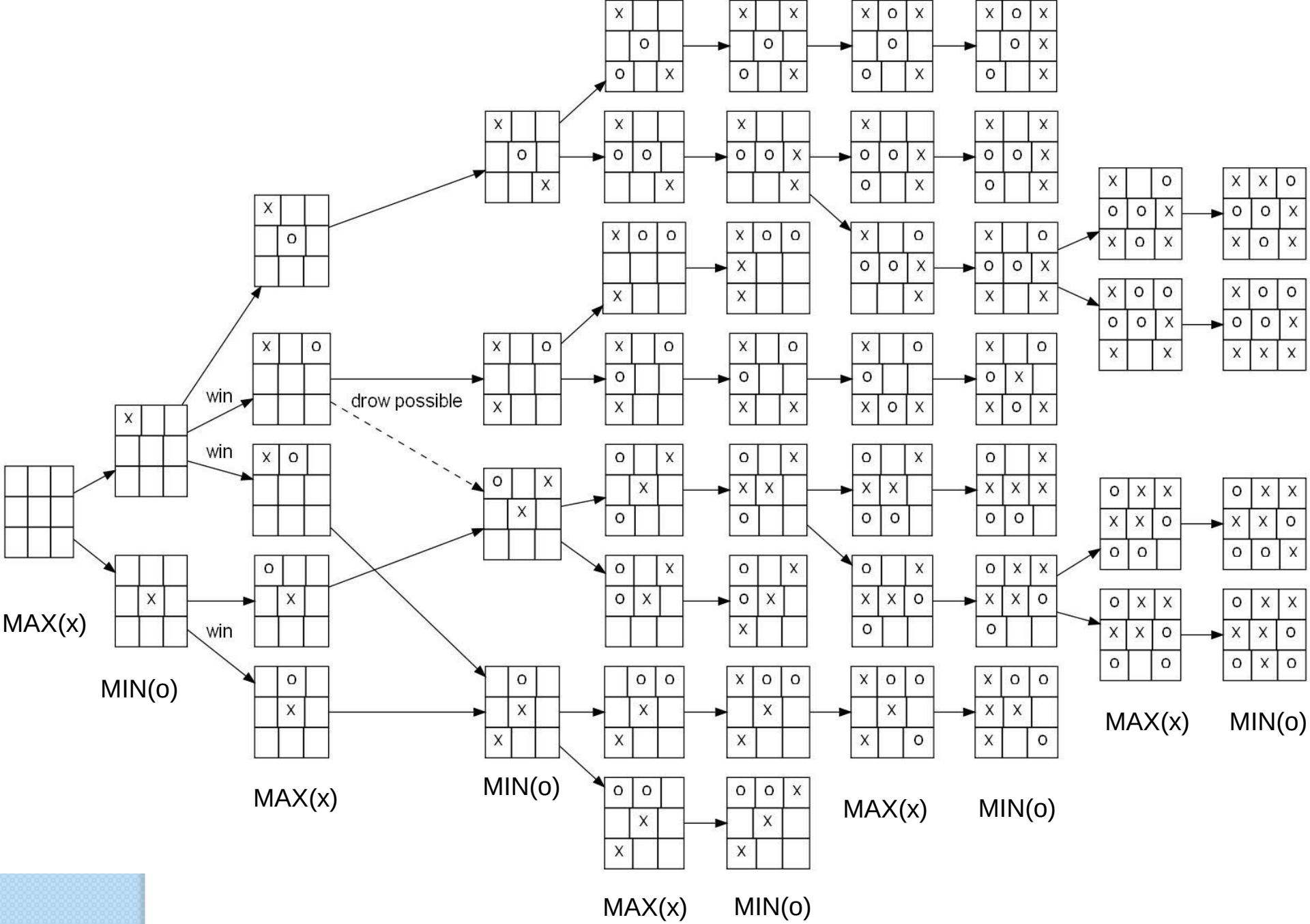
QUYẾT ĐỊNH TỐI ƯU TRONG TRÒ CHƠI

QUYẾT ĐỊNH TỐI ƯU

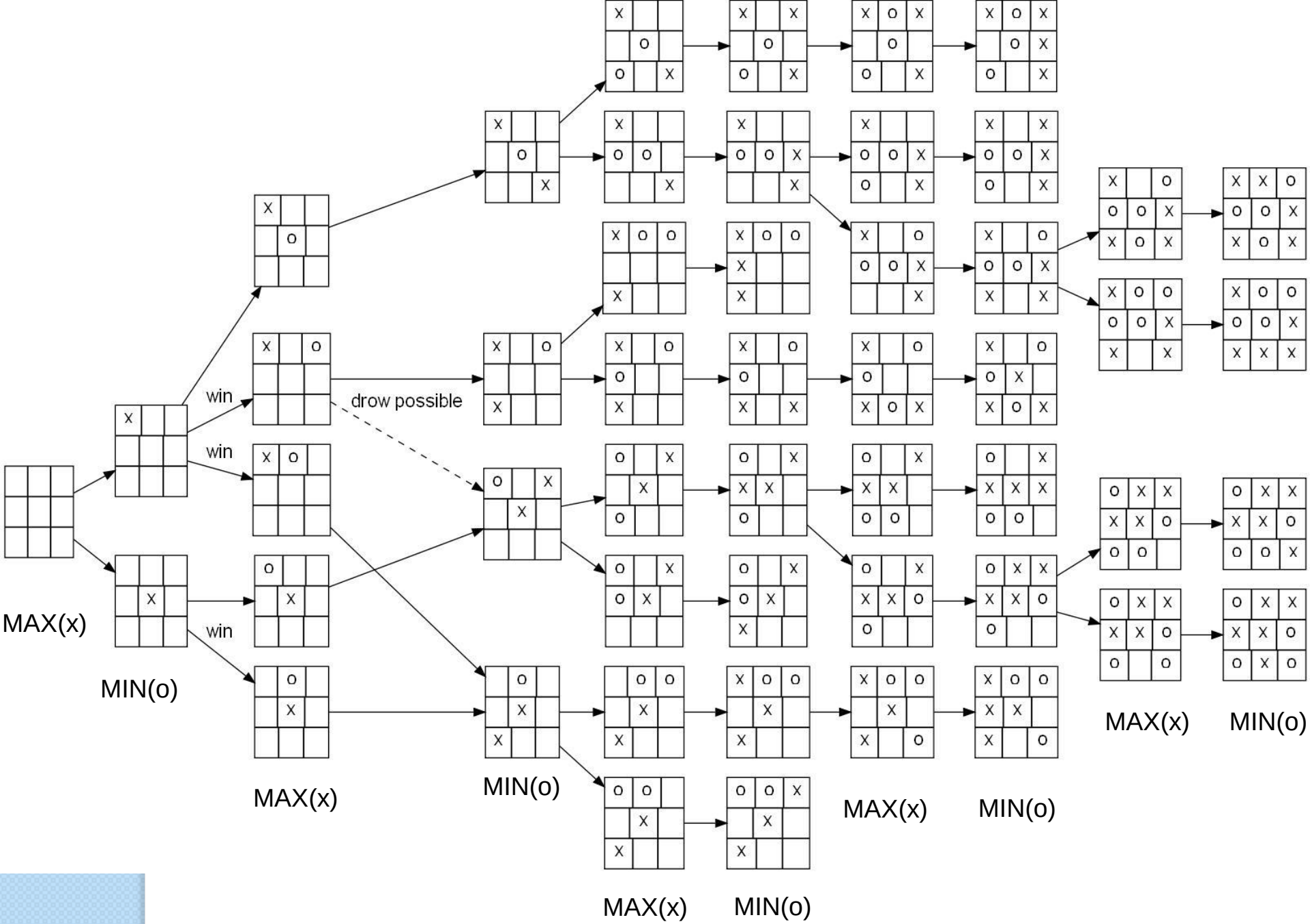
- Lời giải tối ưu:
  - đường đi bảo đảm chiến thắng cho người chơi
- Hai người chơi: MAX vs MIN
- Các thành phần:
  - Trạng thái ban đầu (initial state)
  - Trạng thái kết thúc (terminal state)
  - Hàm **succs(s):** các nước đi hợp lệ
  - Hàm lợi ích (utility function): đánh giá trạng thái kết thúc

TIC-TAC-TOE – CÂY TÌM KIẾM





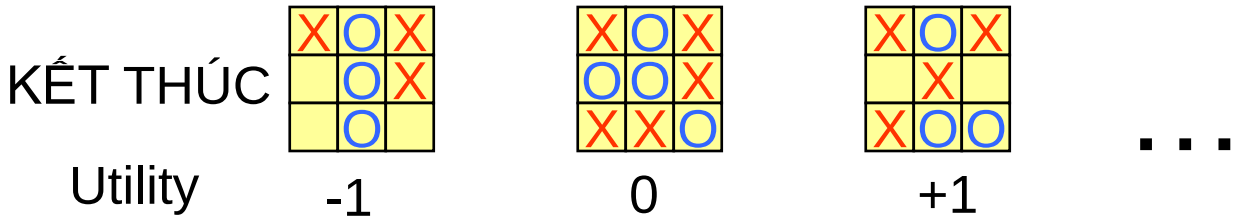
Cây tic tac toe với một số trường hợp MAX thắng hoặc hòa



Cây tic tac toe với một số trường hợp MAX thắng hoặc hòa

Ý tưởng

- Xét nước đi tốt nhất có thể có của đối phương, thuật toán MINIMAX quyết định nước đi cho người chơi.
  - MAX tối đa hóa hàm lợi ích
  - MIN tối thiểu hóa hàm lợi ích
  - Chiến lược của MAX phụ thuộc vào chiến lược của MIN ở bước sau



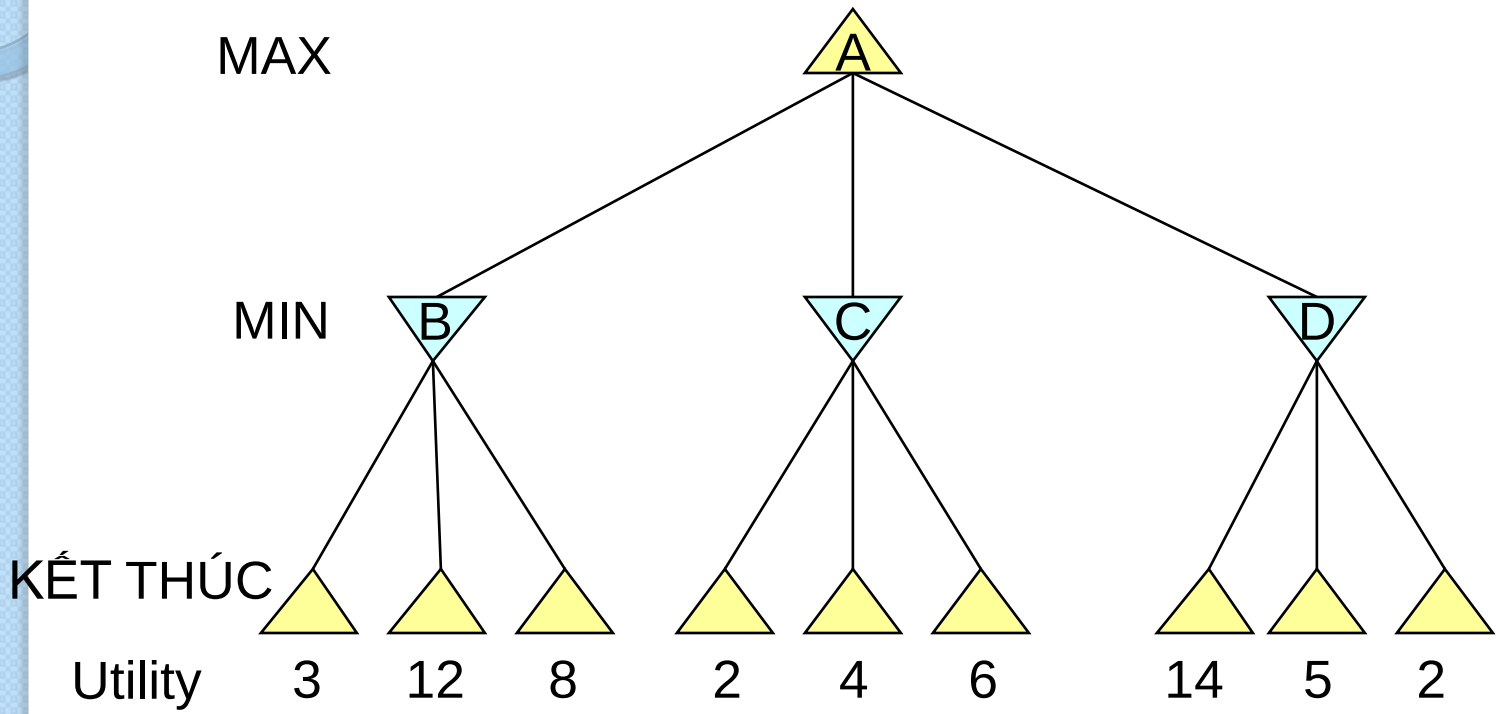
MINIMAX-VALUE

Giá trị MINIMAX-VALUE: lợi ích ở trạng thái kết thúc tương ứng của đường đi, giả sử những người chơi luôn tối ưu

MINIMAX-VALUE(n) =

- Utility(n) *nếu n là trạng thái kết thúc*
- max{MINIMAX-VALUE(s) | s∈succs(n)} *nếu n là một nút MAX*
- min{MINIMAX-VALUE(s) | s∈succs(n)} *nếu n là một nút MIN*

VÍ DỤ MINIMAX

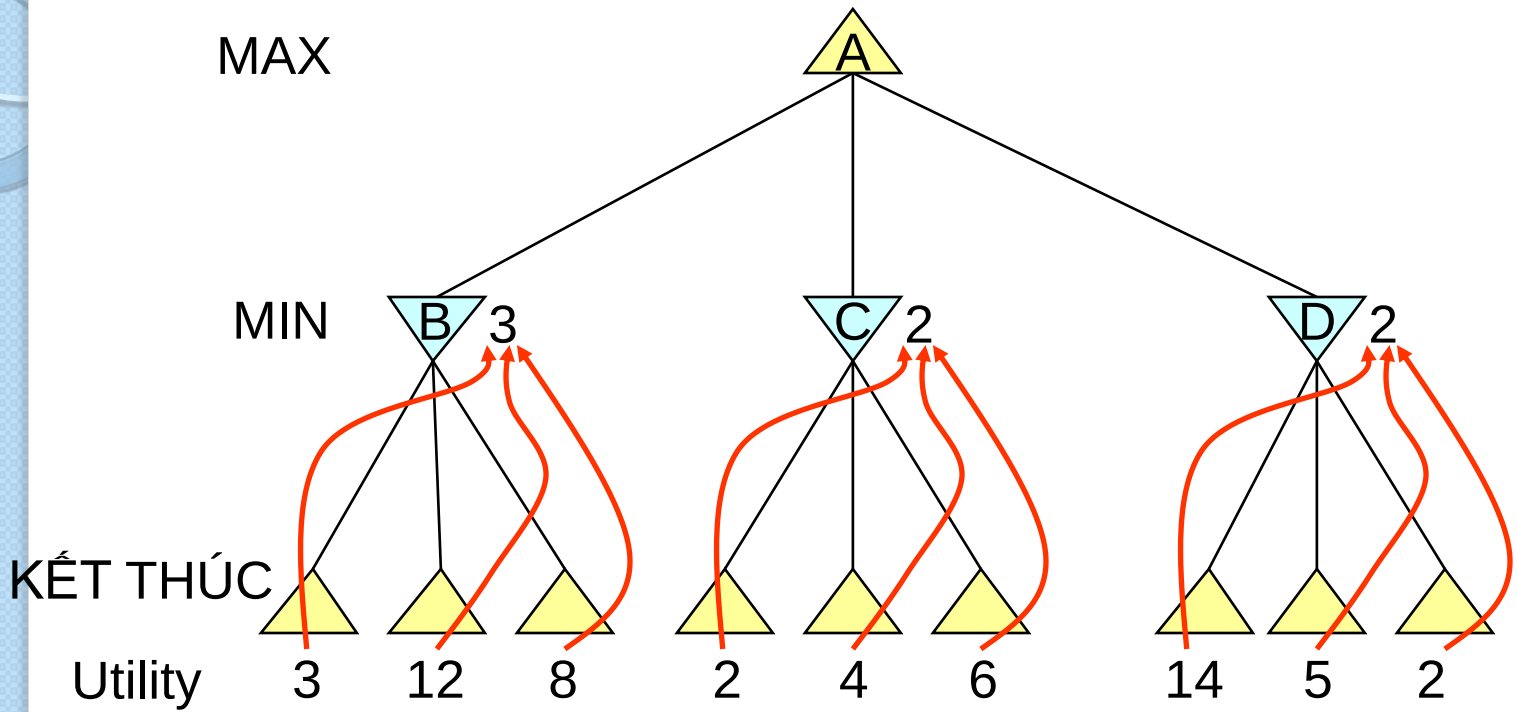


MINIMAX-VALUE(n) =

- **Utility(n)** **if *n là kết thúc***
- max{MINIMAX-VALUE(s)} if *n là MAX*
- min{MINIMAX-VALUE(s)} if *n là MIN*

Ở trạng thái kết thúc, giá trị MINIMAX-VALUE(n) = Utility(n)

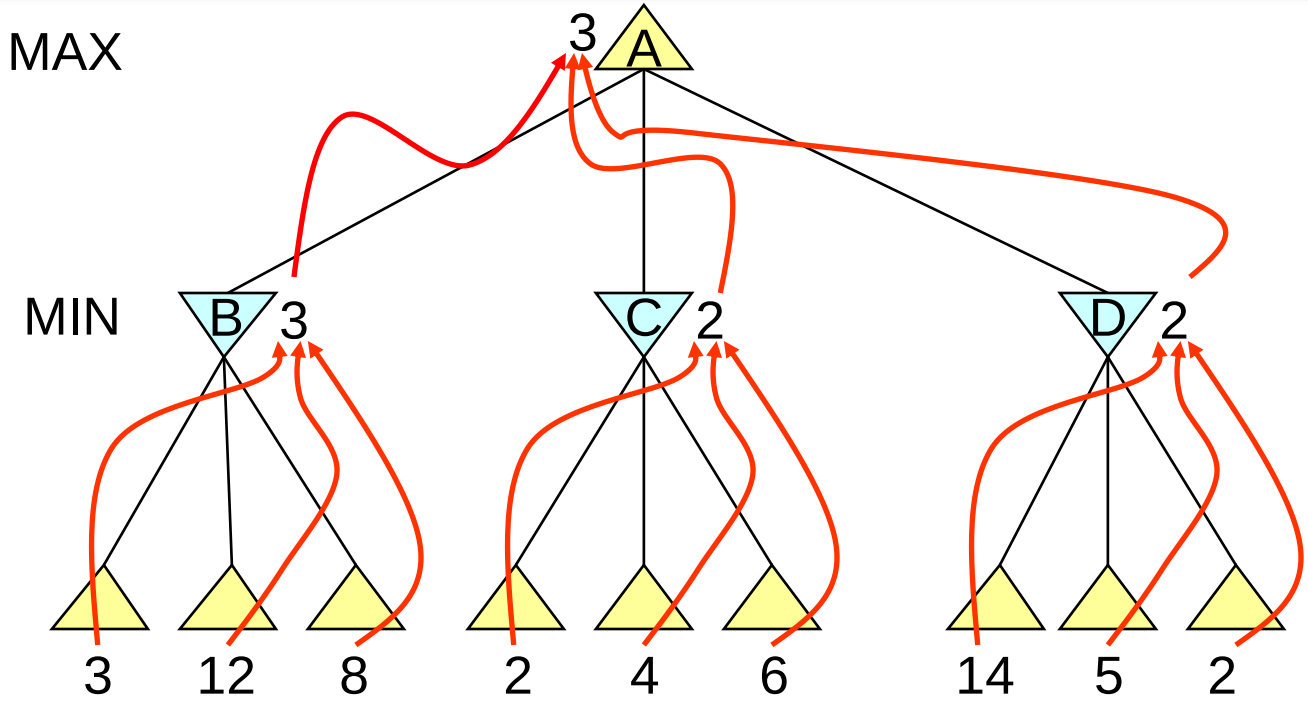
VÍ DỤ MINIMAX



MINIMAX-VALUE(n) =

- Utility(n) if n là kết thúc
- max{MINIMAX-VALUE(s)} if n là MAX
- min{MINIMAX-VALUE(s)} if n là MIN

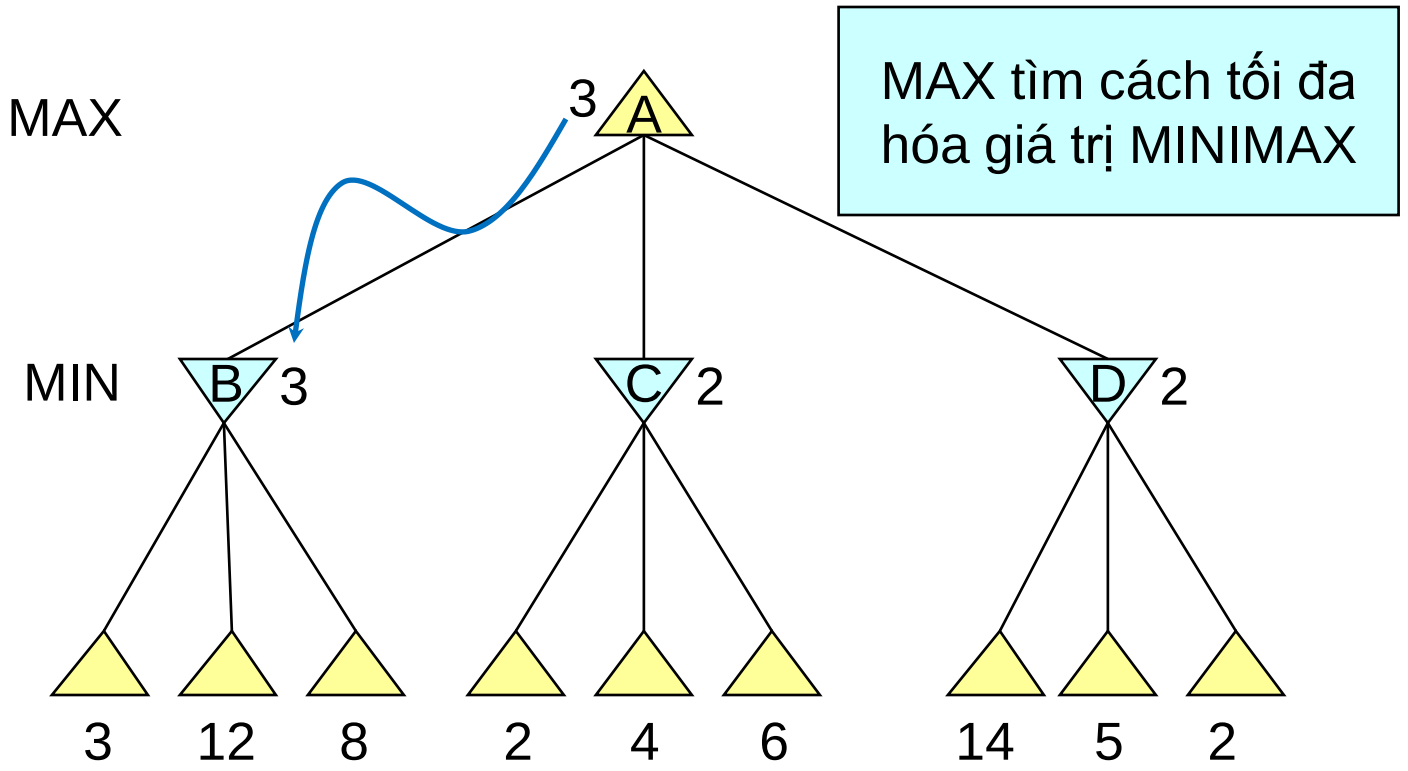
VÍ DỤ MINIMAX



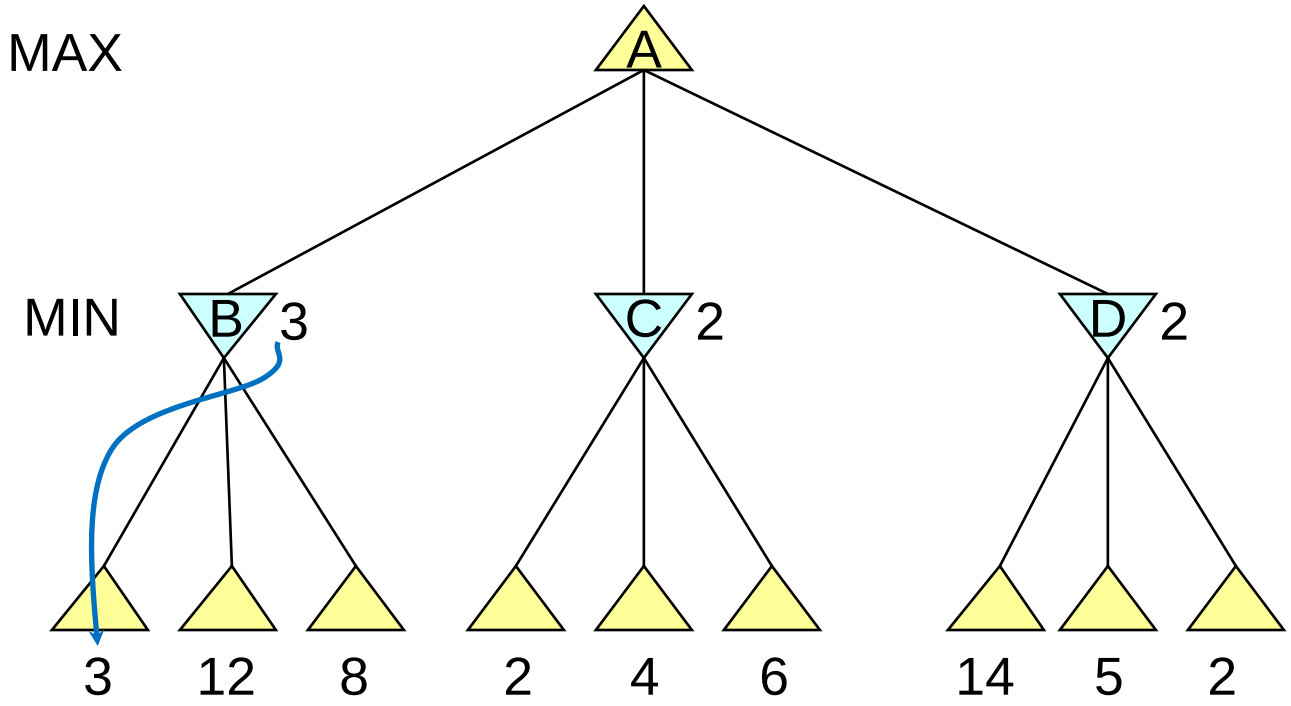
MINIMAX-VALUE(n) =

- Utility(n) if n là kết thúc
- **max{MINIMAX-VALUE(s)} if n là MAX**
- min{MINIMAX-VALUE(s)} if *n là MIN*

VÍ DỤ MINIMAX



VÍ DỤ MINIMAX - Chơi



Tới lượt mình, MIN luôn chọn đường đi
tối thiểu hóa giá trị tiện ích ở trạng thái kết thúc

THUẬT TOÁN MINIMAX

```
function MINIMAX-DECISION(state) returns an action
    v ← MAX-VALUE(state)                                   MAX đi đầu tiên
    return the action in SUCCESSORS(state) with value v

function MAX-VALUE(state) returns a utility value
    if TERMINAL-TEST(state) then return UTILITY(state)
    v ← −∞
    for a, s in SUCCESSORS(state) do                       nếu n là MAX
        v ← MAX(v, MIN-VALUE(s))
    return v

function MIN-VALUE(state) returns a utility value
    if TERMINAL-TEST(state) then return UTILITY(state)
    v ← ∞
    for a, s in SUCCESSORS(state) do                       nếu n là MIN
        v ← MIN(v, MAX-VALUE(s))
    return v
```

ĐÁNH GIÁ THUẬT TOÁN

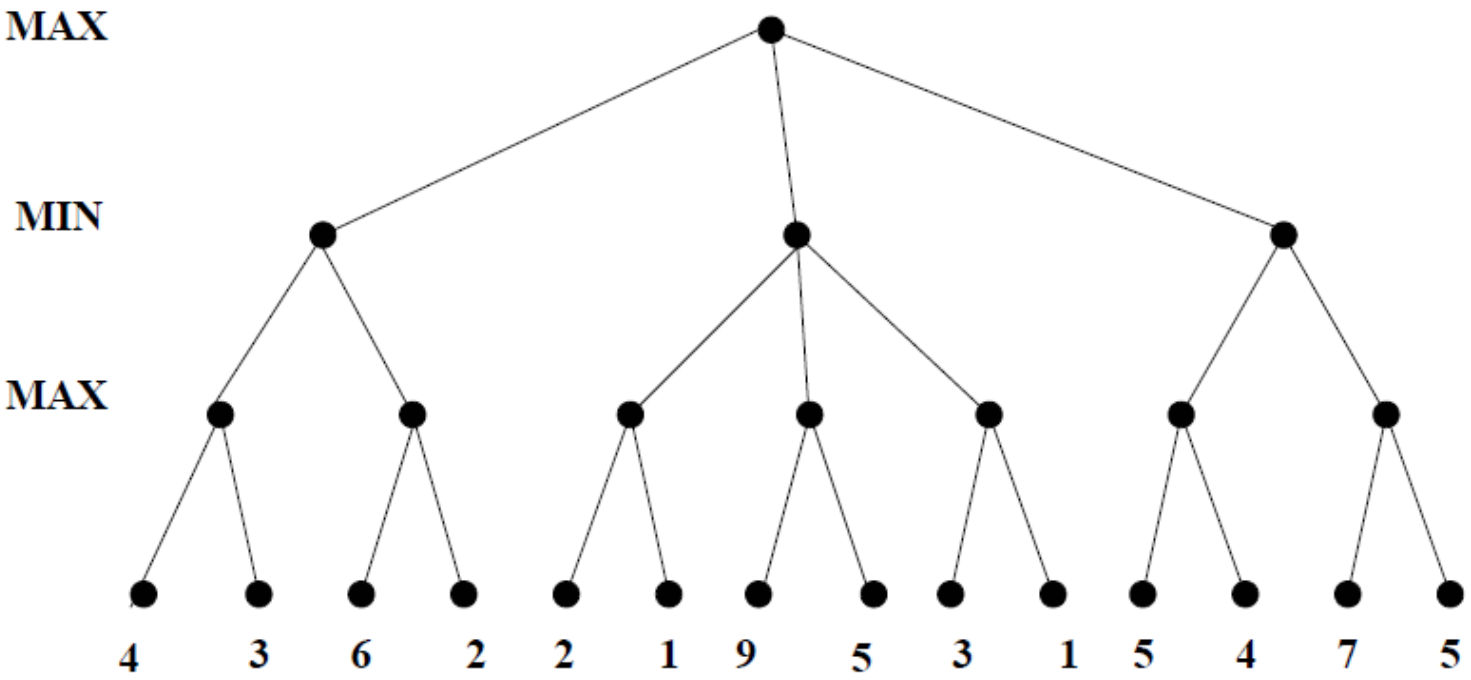
- Cần phải xây dựng toàn bộ cây trò chơi trước khi ra quyết định.
- Đầy đủ? Có (nếu cây tìm kiếm hữu hạn)
- Tối ưu? Có (với một đối thủ tối ưu)
- Độ phức tạp thời gian? $O(b^m)$
- Độ phức tạp không gian? O(m) (tìm kiếm theo chiều sâu)

- Với cờ vua, b ≈ 35 và m ≈100 với một ván thông thường

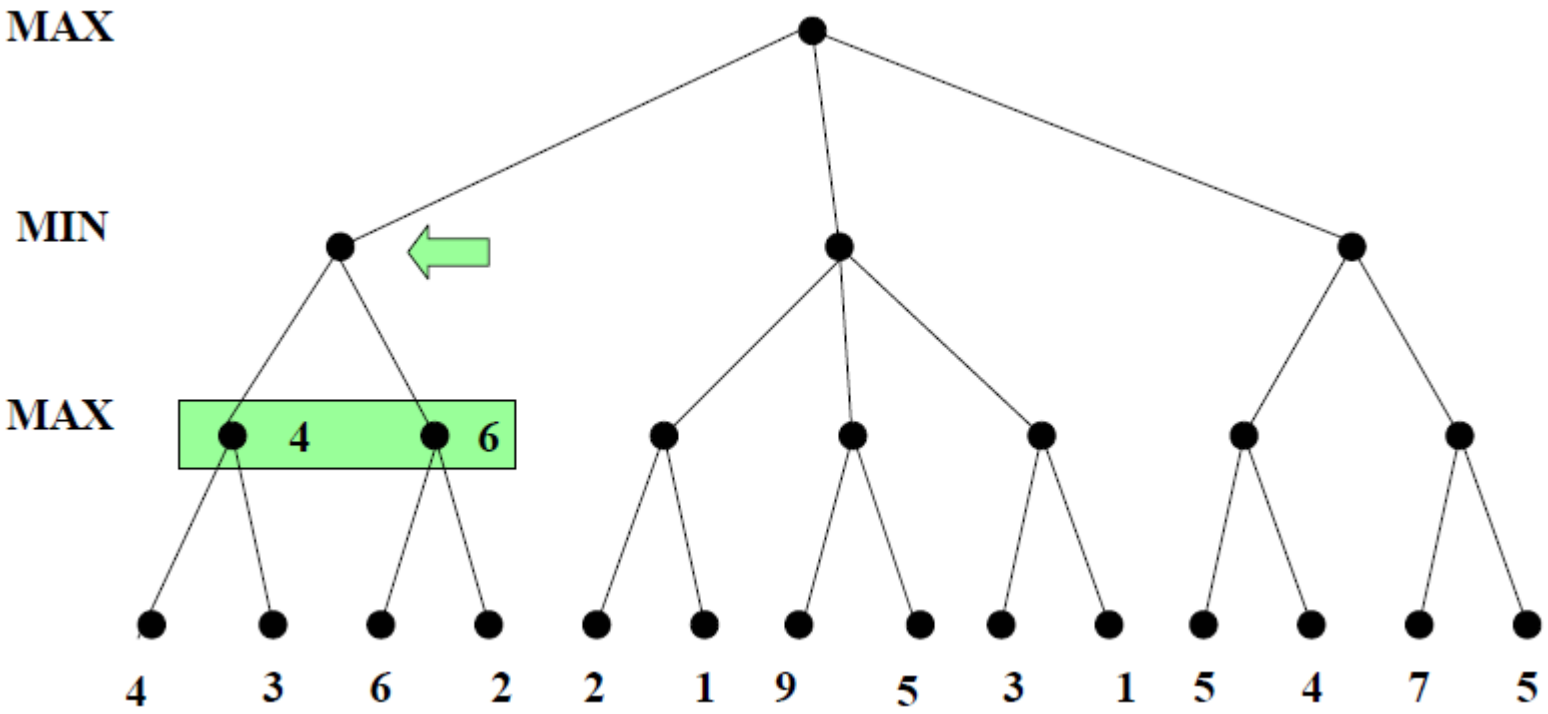
  ⇒ hoàn toàn không thể tìm được lời giải tối ưu

BÀI TẬP VÍ DỤ

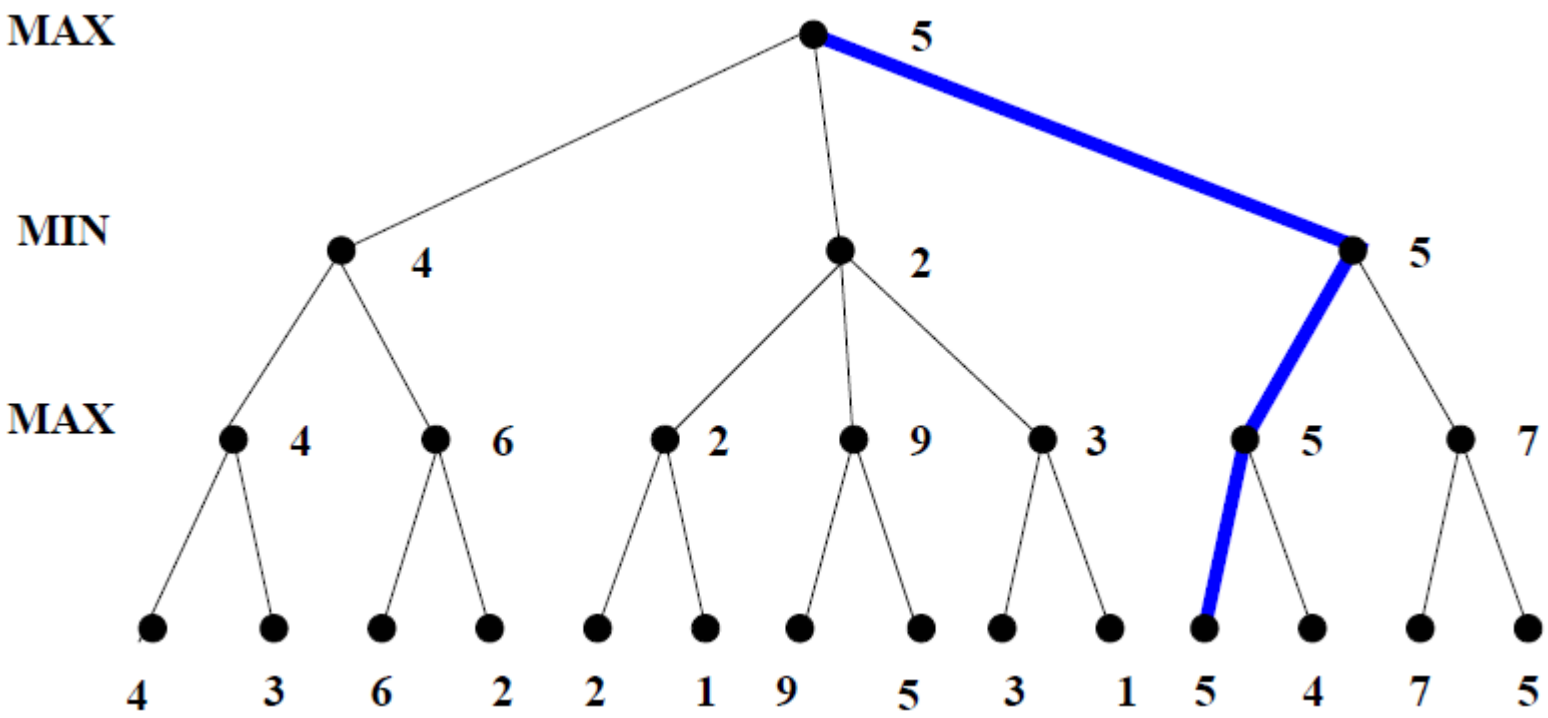
- Cho cây trò chơi như bên dưới.
  - Hãy tính giá trị tiện ích cho nước đi còn lại.
  - Để đạt kết quả tốt nhất, MAX và MIN sẽ lần lượt đi những nước nào?



BÀI TẬP VÍ DỤ



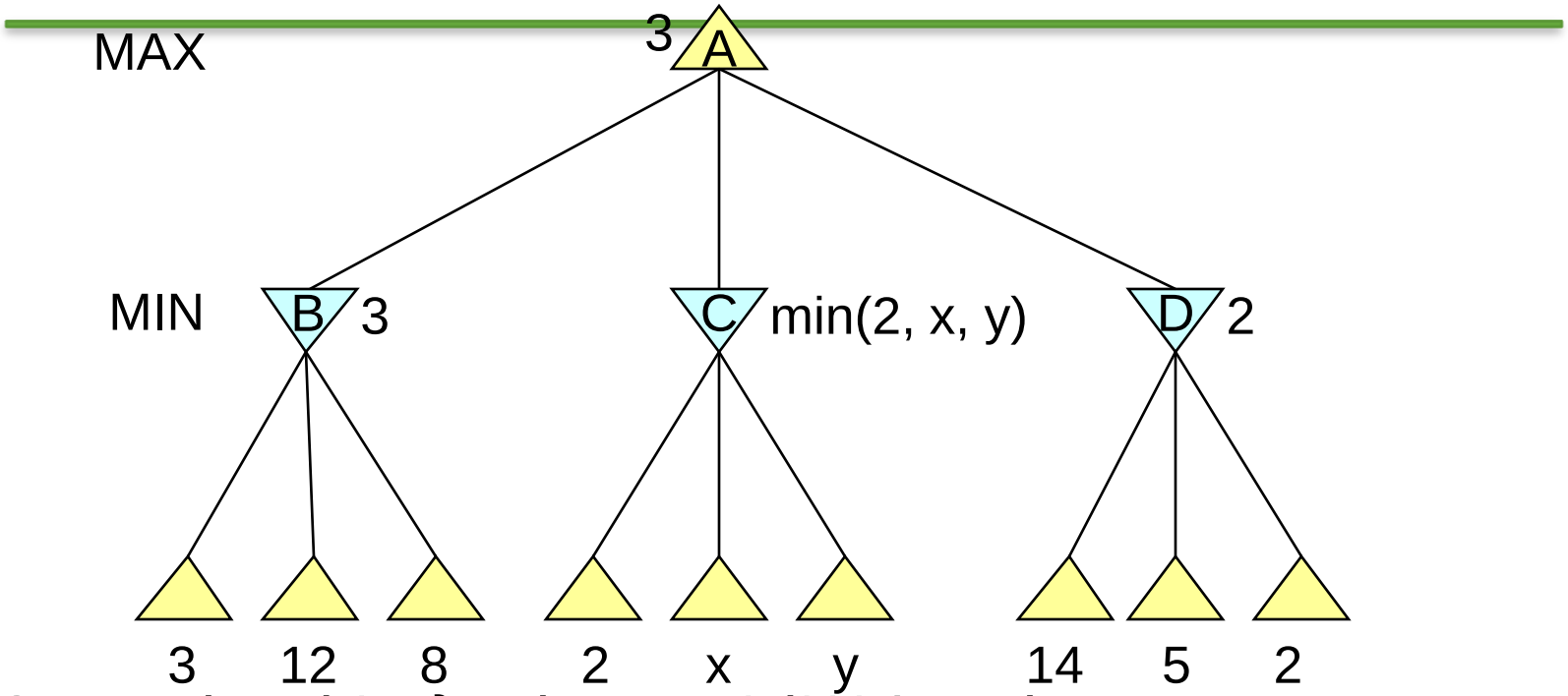
BÀI TẬP VÍ DỤ



TỈA NHÁNH α – β

- Mục tiêu: ra quyết định minimax chính xác mà không cần xét mọi nút trong cây trò chơi.
- Mượn ý tưởng "tỉa nhánh" để loại bỏ những nhánh không có khả năng ảnh hưởng đến kết quả cuối cùng.

VÍ DỤ TỈA NHÁNH α-β



- Gọi x, y là lợi ích của các trạng thái không xét.
- MINIMAX-VALUE(gốc)

  = max(min(3, 12, 8), min(2, x, y),min(14, 5, 2))

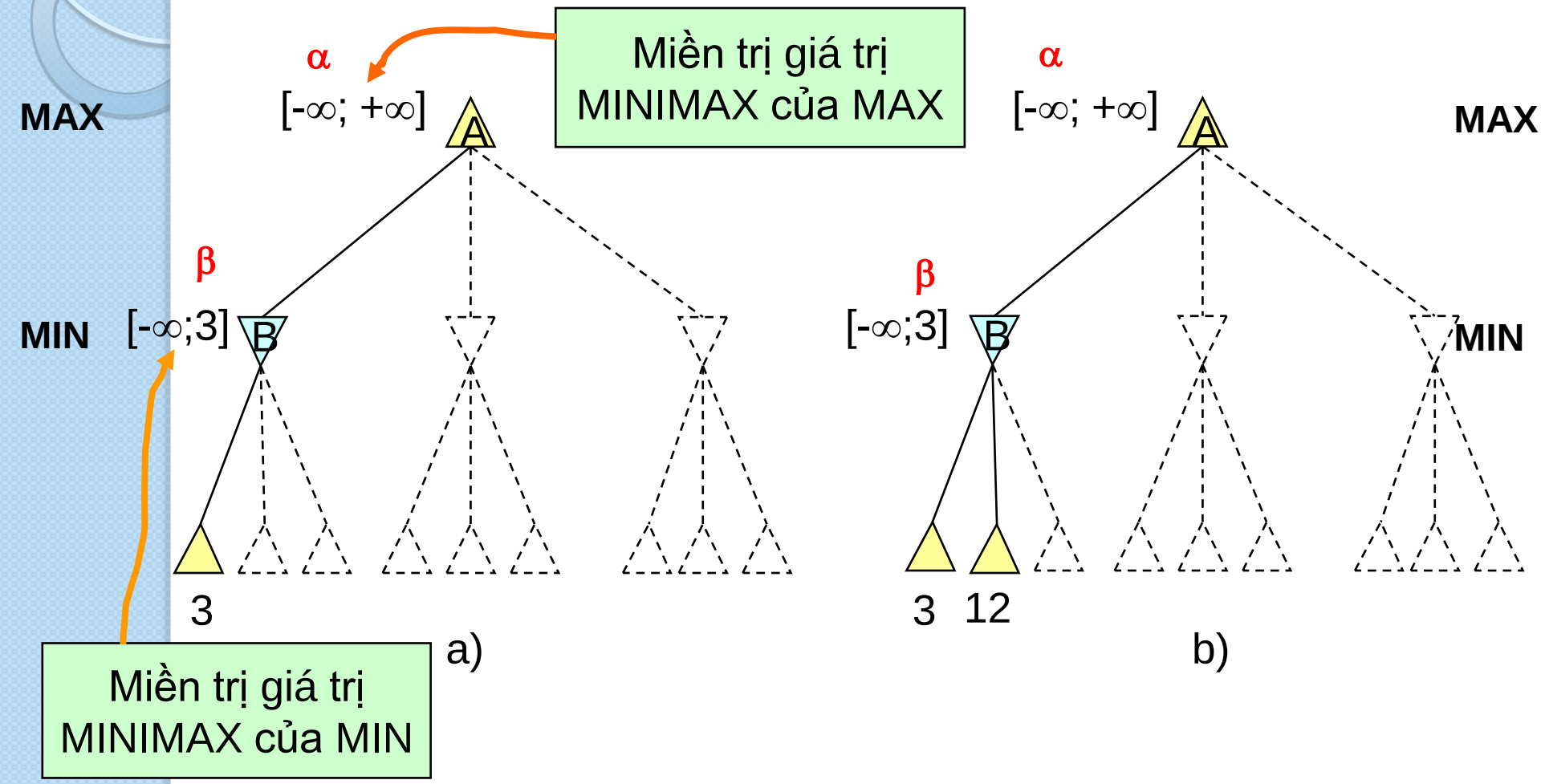
  = max(3, min(2, x, y), 2) → đặt z = min(2, x, y)

  = max(3, z, 2) với z ≤2

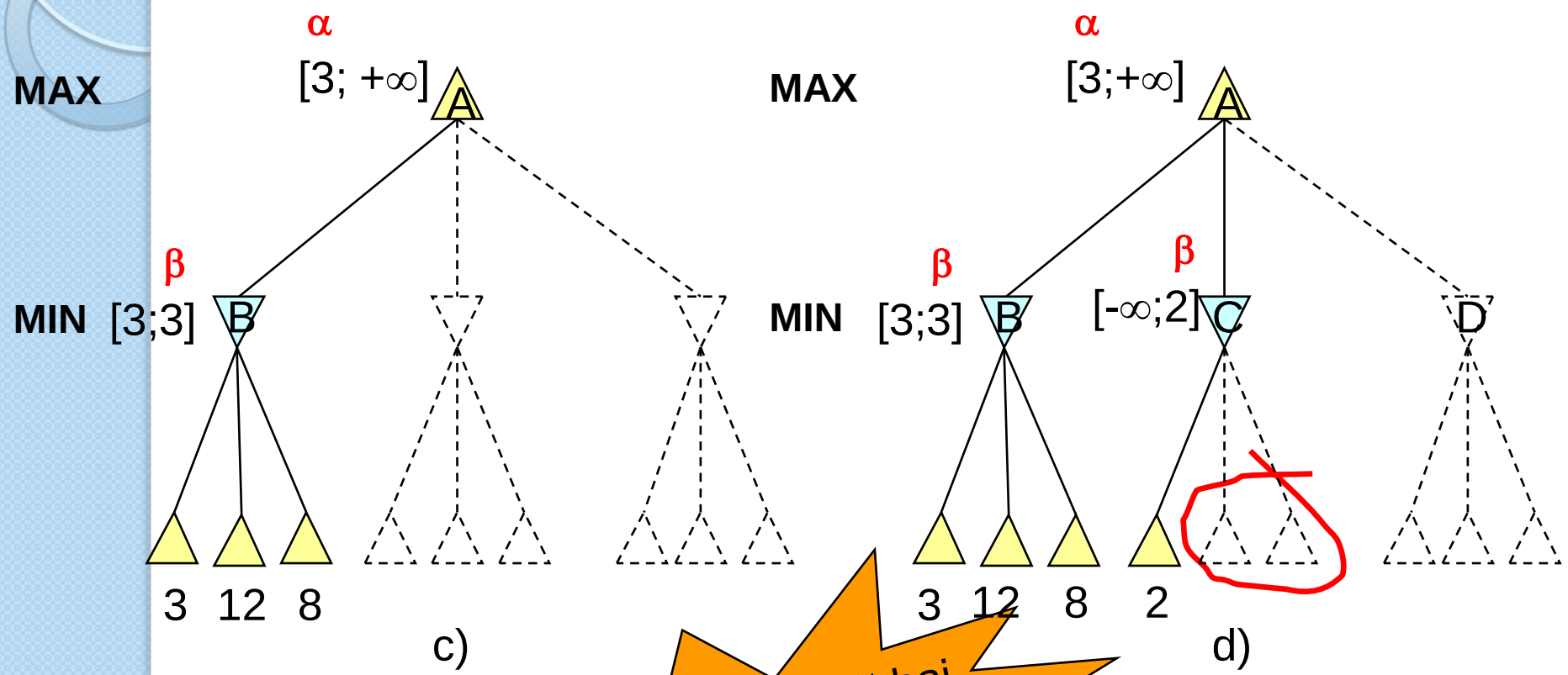
  = 3

Giá trị MINIMAX tại gốc không phụ thuộc vào x, y.

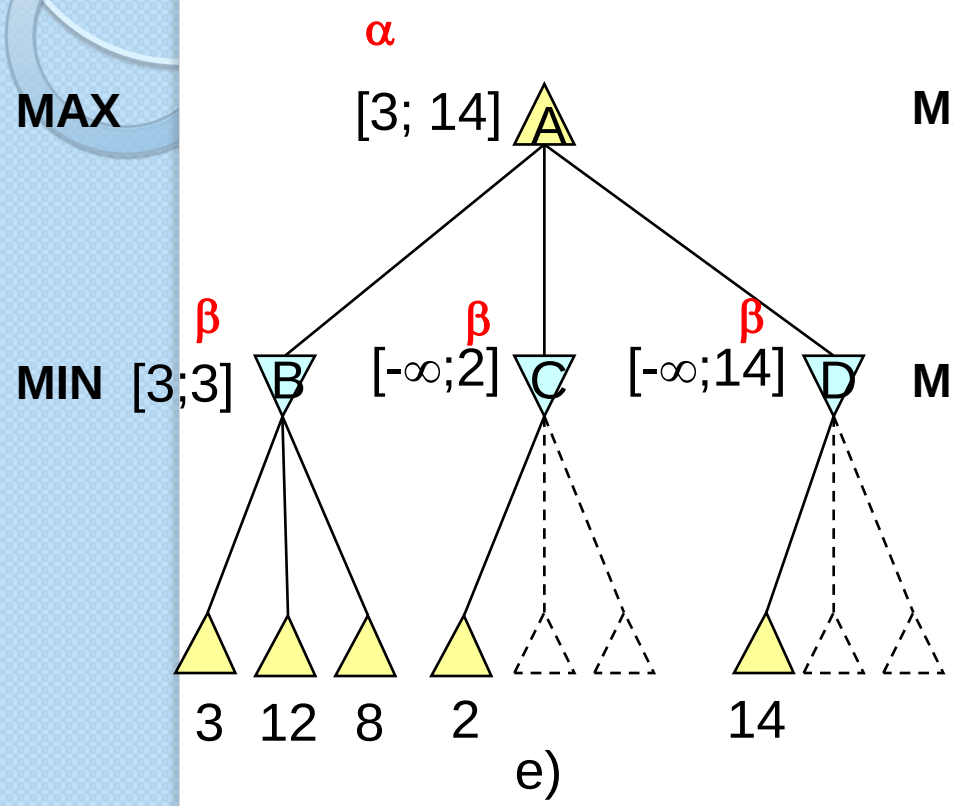
VÍ DỤ TỈA NHÁNH α-β

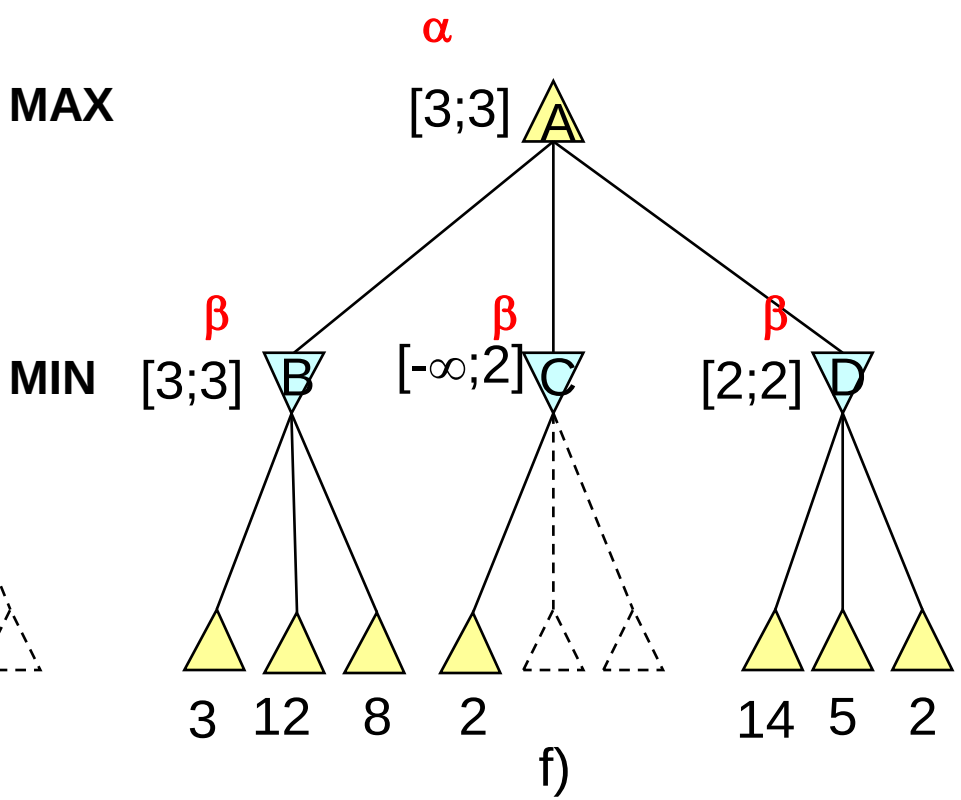


VÍ DỤ TỈA NHÁNH α-β



VÍ DỤ TỈA NHÁNH α-β



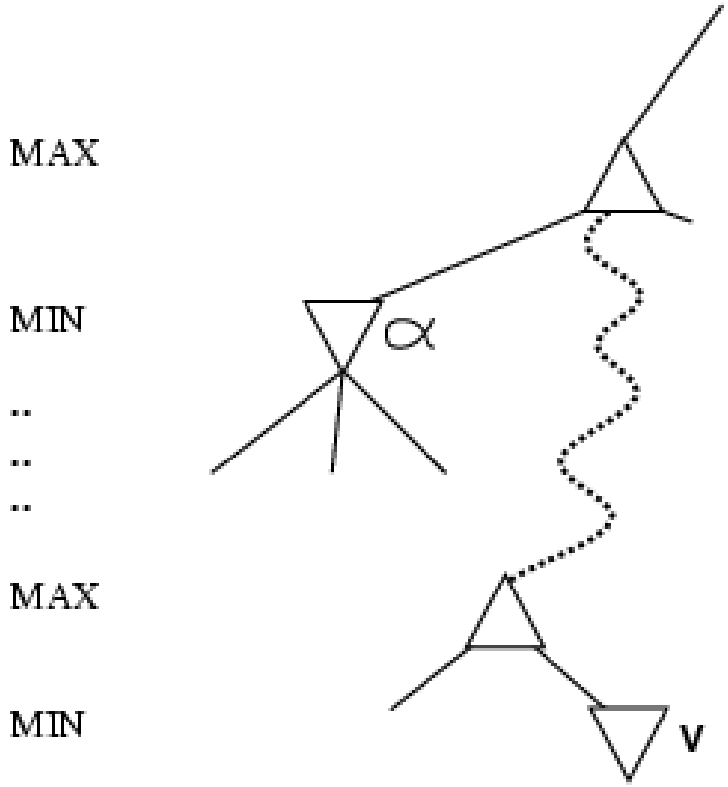


ĐÁNH GIÁ TỈA NHÁNH α-β

- Tỉa nhánh không ảnh hưởng đến kết quả cuối cùng
- Thứ tự các nước đi tốt có thể cải thiện hiệu quả của tỉa nhánh (trong ví dụ, hãy xem xét nhánh D)
- Với "thứ tự hoàn hảo", độ phức tạp thời gian là $O(b^{m/2})$ (cho phép tìm với độ sâu gấp đôi)

TẠI SAO GỌI LÀ α – β?

- α là giá trị của lựa chọn tốt nhất (giá trị cao nhất) cho đến hiện tại ở một điểm bất kỳ trên một đường đi cho *MAX*
- Nếu v xấu hơn α, MAX sẽ tránh nó và tỉa nhánh này
- Định nghĩa β tương tự cho MIN



THUẬT TOÁN α – β

Nếu n là MIN

```
function MIN-VALUE(state, α, β) returns a utility value
  inputs: state, current state in game
          α, the value of the best alternative for MAX along the path to state
          β, the value of the best alternative for MIN along the path to state
  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← +∞
  for a, s in SUCCESSORS(state) do
    v ← MIN(v, MAX-VALUE(s, α, β))
    if v ≤ α then return v
    β ← MIN(β, v)
  return v
```

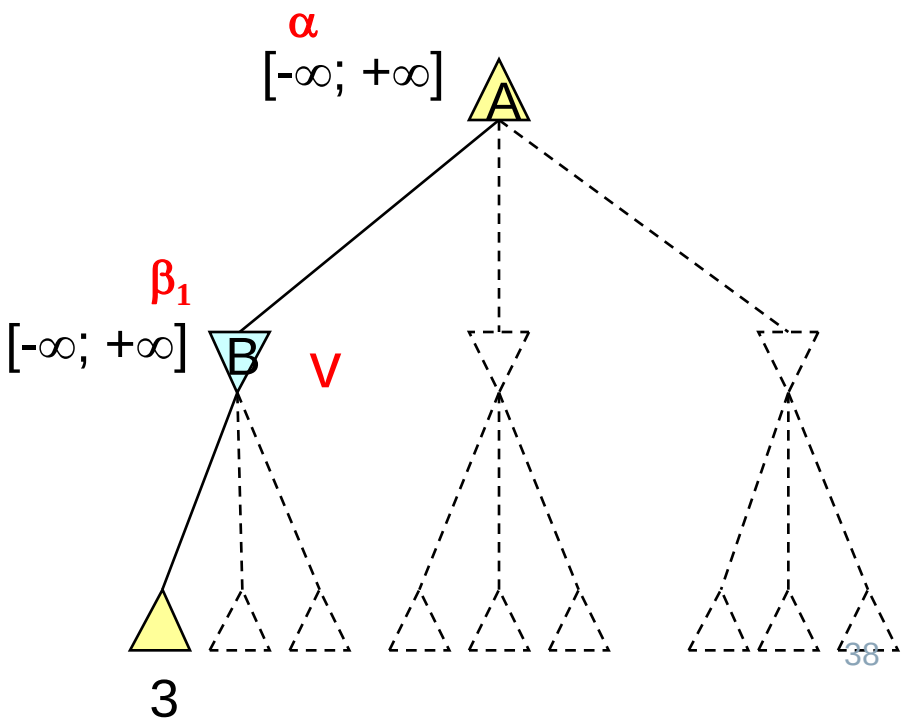
Ví dụ 1: State là node B

State là node B

- $\alpha = -\infty$
- $\beta_1 = +\infty$
- v = $+\infty$
- s = node 3
  - v = 3
  - v > α
  - $\beta_1 = \min(\beta_1, v) = 3$

Xử lý của MIN

$v \leftarrow +\infty$
for a, s in SUCCESSORS($state$) **do**
  $v \leftarrow$ MIN(v, MAX-VALUE(s, α, β))
  if $v \leq \alpha$ **then return** v
  $\beta \leftarrow$ MIN(β, v)
return v



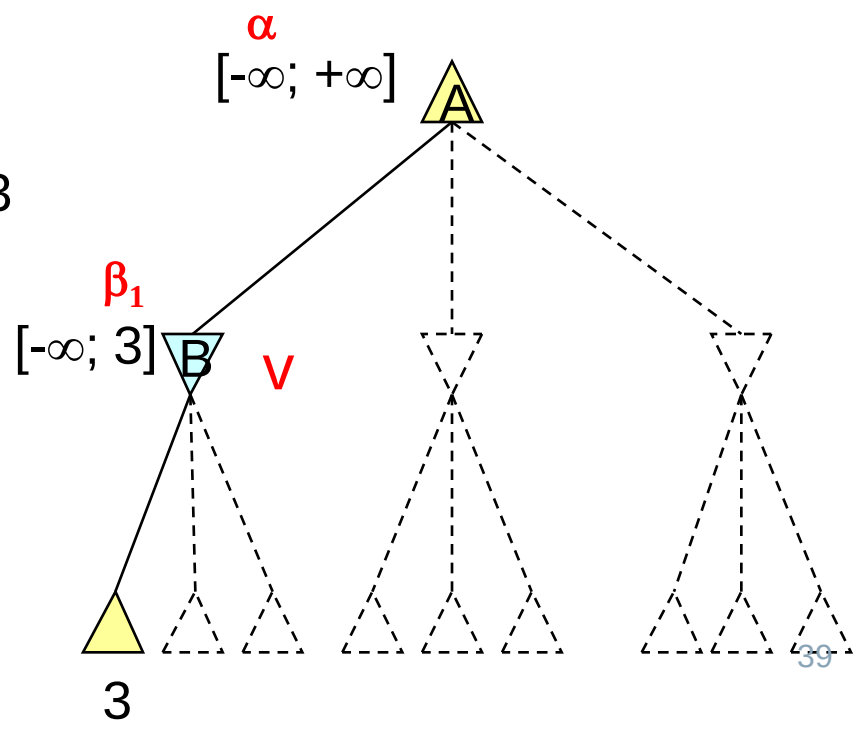
Ví dụ 1: State là node B

State là node B

- $\alpha = -\infty$
- $\beta_1 = +\infty$
- v = $+\infty$
- s = node 3
  - MAX-VALUE(s, α, β_1) = 3
  - v = min($+\infty$, 3) = 3
  - v > α
  - β_1 =min(β_1,v) = 3

Xử lý của MIN

$v \leftarrow +\infty$
for a, s in SUCCESSORS($state$) **do**
  $v \leftarrow$ MIN(v, MAX-VALUE(s, α, β))
  if $v \leq \alpha$ **then return** v
  $\beta \leftarrow$ MIN(β, v)
return v



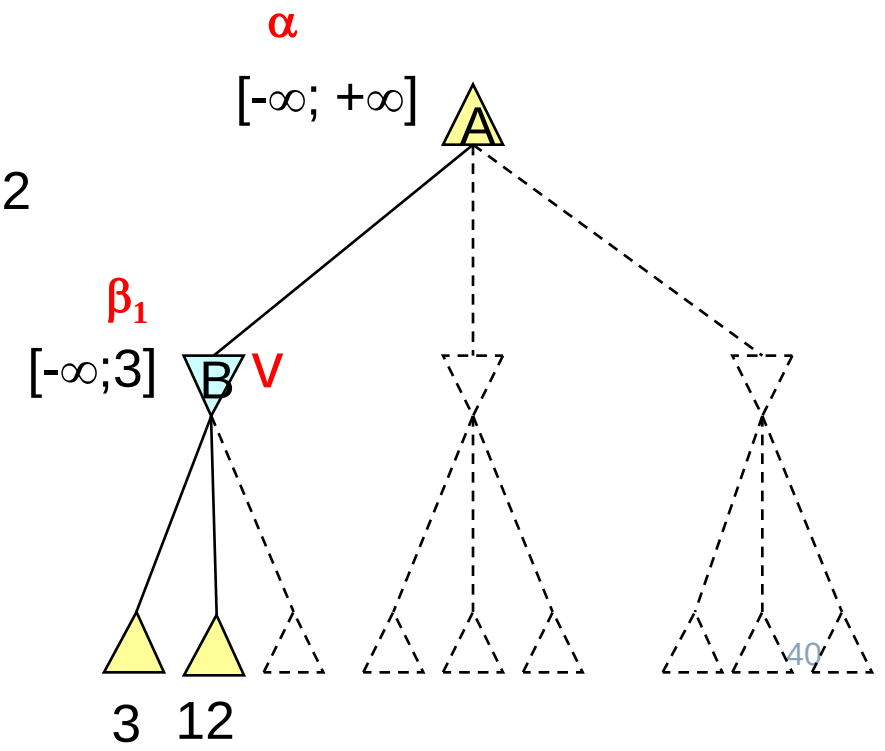
Ví dụ 1: State là node B

State là node B

- $\alpha = -\infty$
- $\beta_1 = 3$
- v = 3
- s = node 12
  - MAX-VALUE(s, α, β_1) = 12
  - v = min(3, 12) = 3
  - v > α
  - β_1 =min(β_1,v) = 3

Xử lý của MIN

$v \leftarrow +\infty$
for a, s in SUCCESSORS($state$) **do**
  $v \leftarrow$ MIN(v, MAX-VALUE(s, α, β))
  if $v \leq \alpha$ **then return** v
  $\beta \leftarrow$ MIN(β, v)
return v



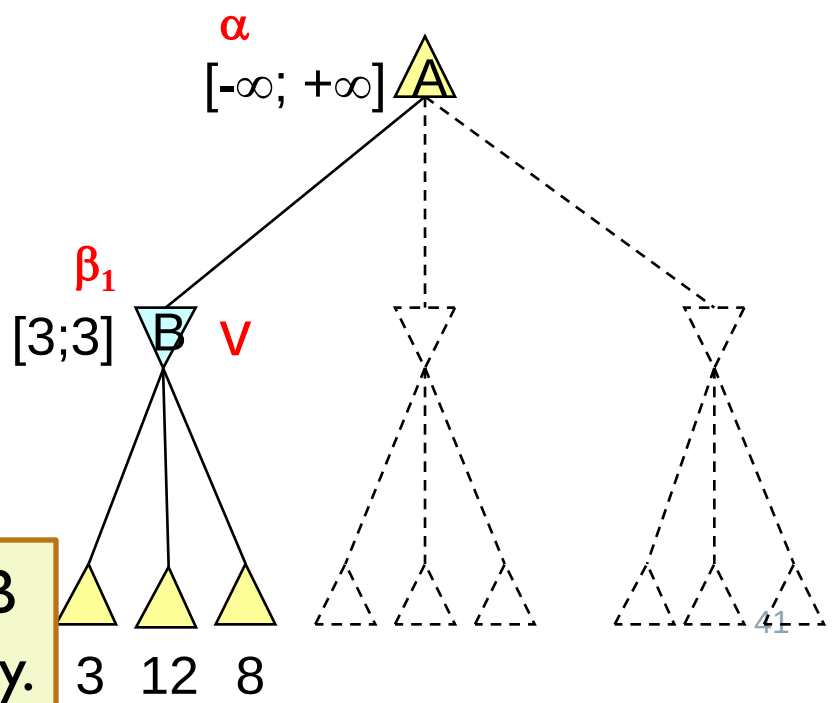
Ví dụ 1: State là node B

Xử lý của MIN

$v \leftarrow +\infty$
for a, s **in** SUCCESSORS($state$) **do**
  $v \leftarrow$ MIN(v, MAX-VALUE(s, α, β))
  if $v \leq \alpha$ **then return** v
  $\beta \leftarrow$ MIN(β, v)
return v

State là node B

- $\alpha = -\infty$
- $\beta_1 = 3$
- v = 3
- s = node 8
  - MAX-VALUE(s, α, β_1) = 8
  - v = min(3, 8) = 3
  - v > α
  - β_1 =min(β_1,v) = 3



c)

Thuật toán xử lý hết các node con của B
--> Không có "tỉa" trong trường hợp này.

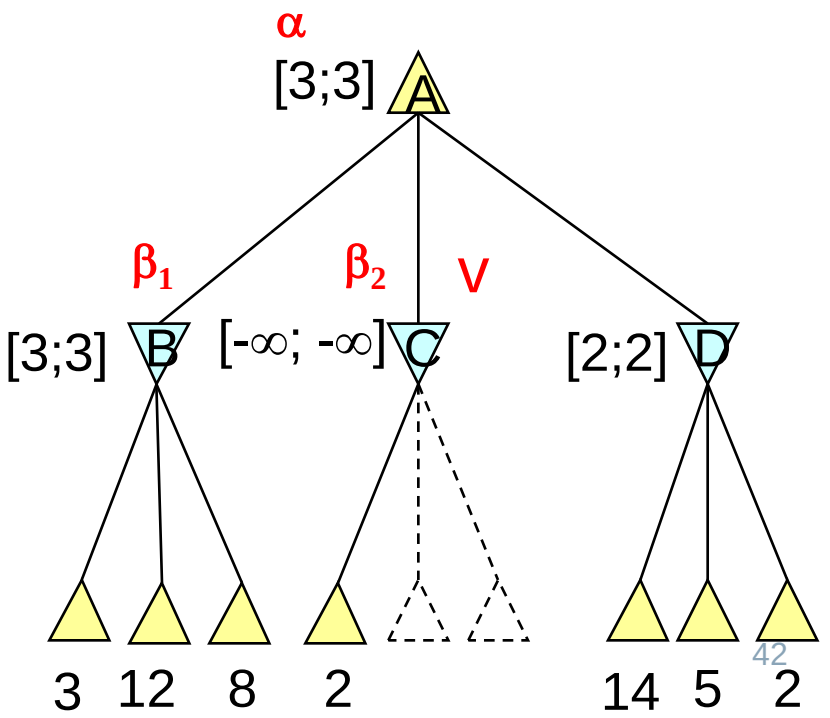
Ví dụ 2: State là node C

State là node C

- $\alpha = 3$
- $\beta_2 = +\infty$
- $v = +\infty$
- s = node 2
  - MAX-VALUE(s, α, β_1) = 2
  - v = min(+∞, 2) = 2
  - $v < \alpha$ (2<3) nên dừng

Xử lý của MIN

$v \leftarrow +\infty$
for a, s in SUCCESSORS($state$) **do**
  $v \leftarrow$ MIN(v, MAX-VALUE(s, α, β))
  if $v \leq \alpha$ **then return** v
  $\beta \leftarrow$ MIN(β, v)
return v



Không xử lý hết các node con của C
--> Có "tỉa" trong trường hợp này.

THUẬT TOÁN α – β

function ALPHA-BETA-SEARCH(*state*) **returns** *an action* — ***MAX đi đầu tiên***

inputs: *state*, current state in game

$v \leftarrow$ MAX-VALUE($state, -\infty, +\infty$)

return the *action* in SUCCESSORS(*state*) with value v

---

function MAX-VALUE($state, \alpha, \beta$) **returns** *a utility value* — ***Nếu n là MAX***

inputs: *state*, current state in game

α, the value of the best alternative for MAX along the path to *state*

β, the value of the best alternative for MIN along the path to *state*

if TERMINAL-TEST(*state*) **then return** UTILITY(*state*)

$v \leftarrow -\infty$

for a, s in SUCCESSORS(*state*) **do**

  $v \leftarrow$ MAX(v, MIN-VALUE(s, α, β))

  if $v \geq \beta$ **then return** v

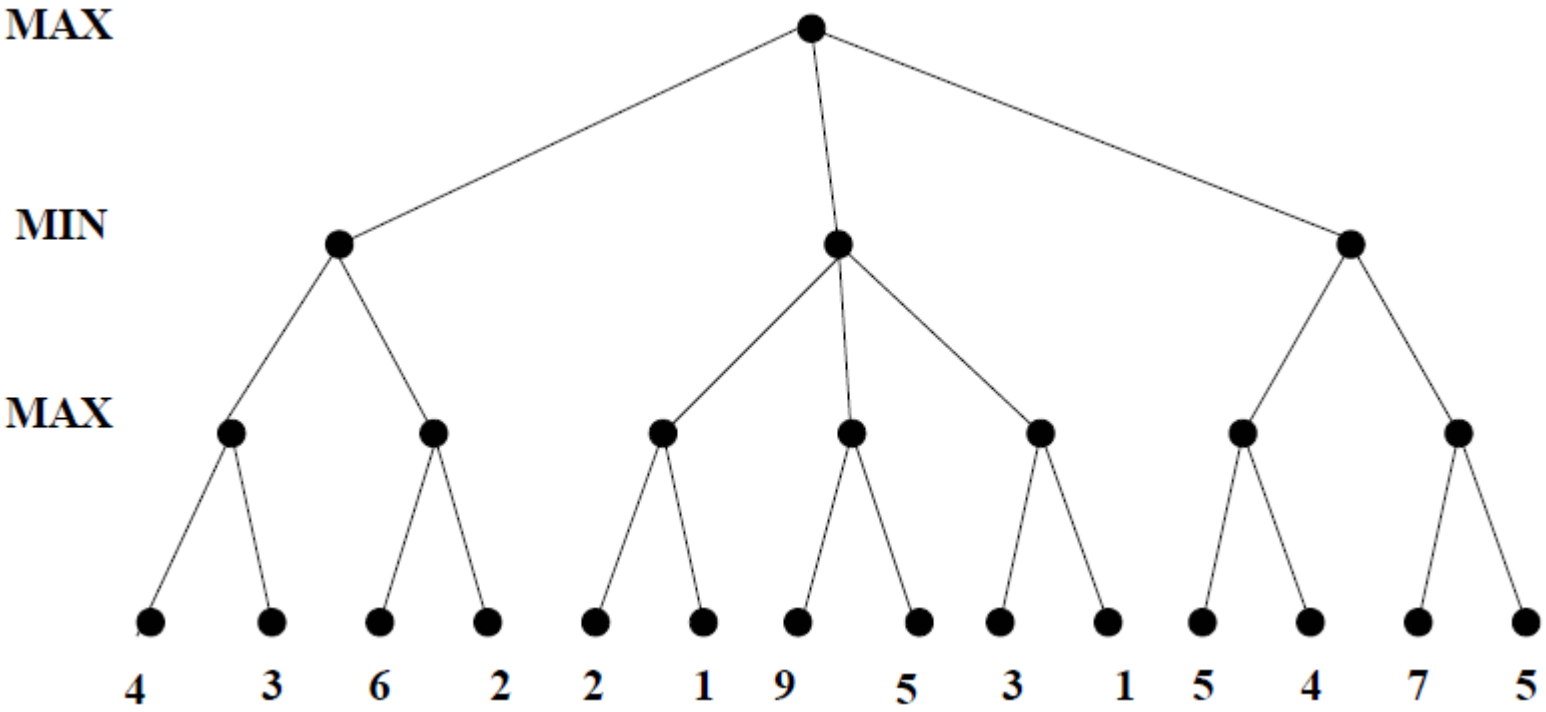
  $\alpha \leftarrow$ MAX(α, v)

return v

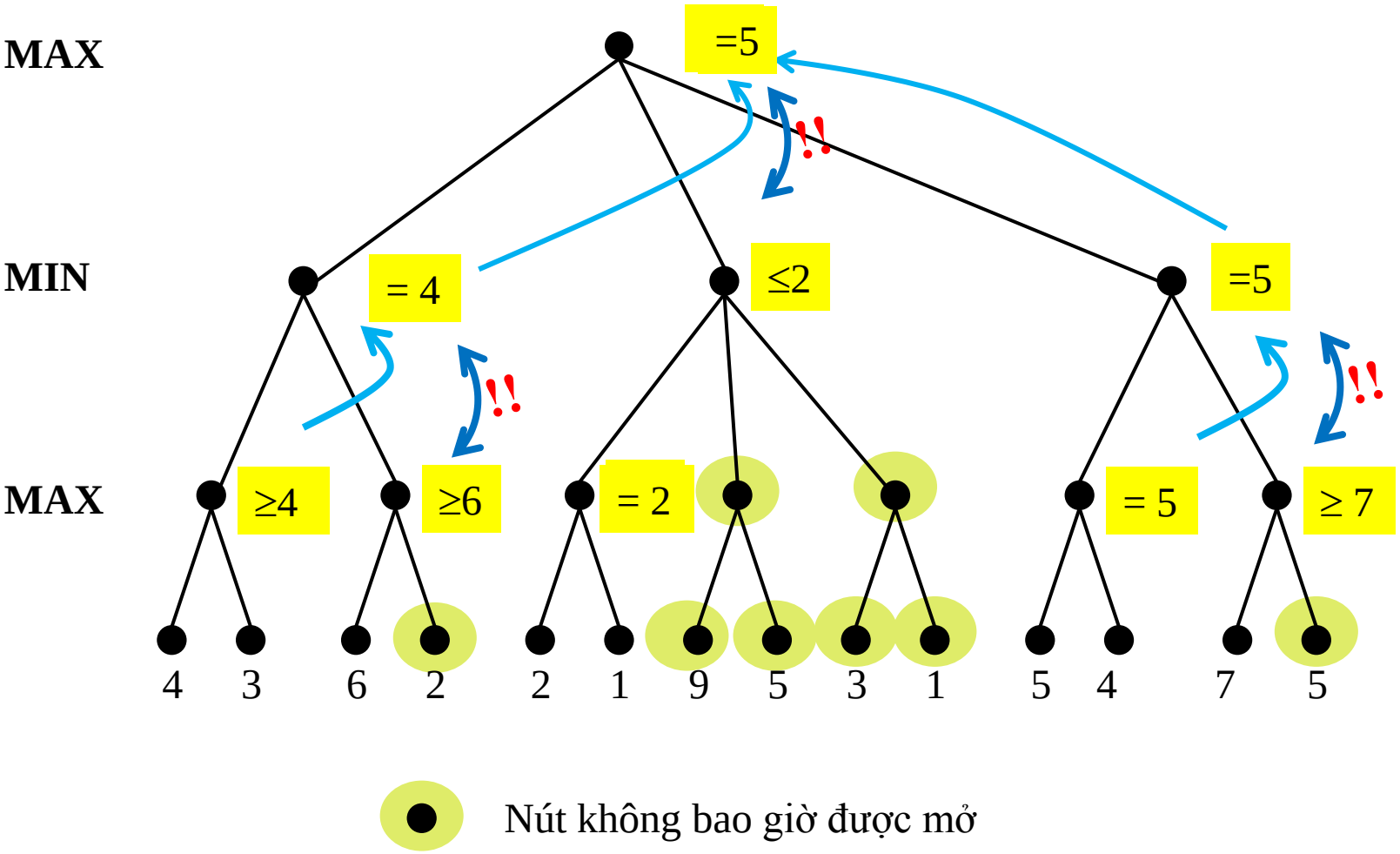
Tương tự cho xử lý của MAX(ngược lại với của MIN).
Thuật toán bắt đầu với MAX đi trước.

BÀI TẬP VÍ DỤ

- Cho cây trò chơi như bên dưới. Hãy tỉa các nhánh không cần duyệt.



BÀI TẬP VÍ DỤ


MAX
MIN
MAX
=5
= 4
≤2
=5
≥4
≥6
= 2
= 5
≥ 7
!!
!!
!!
4 3 6 2 2 1 9 5 3 1 5 4 7 5
Nút không bao giờ được mở

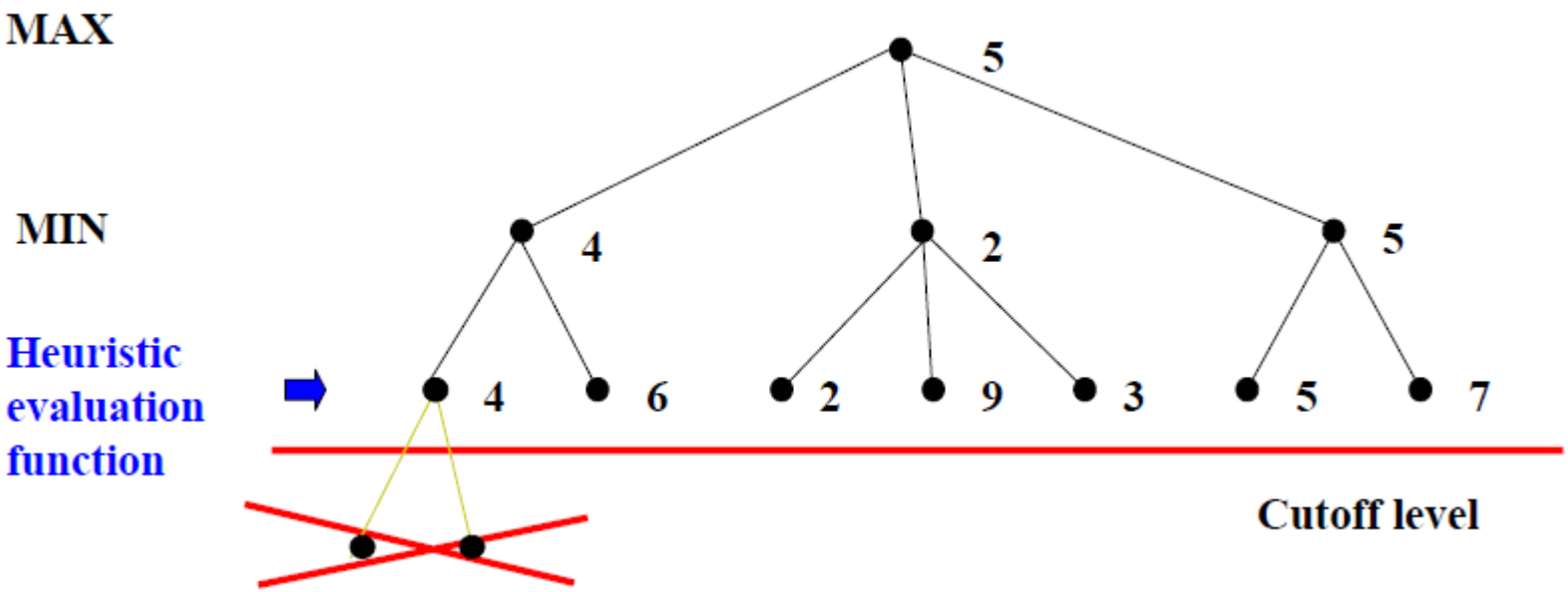
QUYẾT ĐỊNH THỜI GIAN THỰC VÀ KHÔNG HOÀN HẢO

QUYẾT ĐỊNH THỜI GIAN THỰC

- Các trò chơi thường có độ sâu lớn (> 35 đối với cờ vua)
- Trong thời gian thực, không thể đi đến trạng thái kết thúc để đánh giá một nước đi $\Rightarrow$ tìm kiếm giới hạn (cut-off search)

HÀM LƯỢNG GIÁ

- Ý tưởng
  - Giới hạn cây tìm kiếm trước khi đến trạng thái kết thúc.
  - Sử dụng hàm lượng giá các trạng thái không kết thúc thay cho hàm đánh giá lợi ích của trạng thái kết thúc.



HÀM LƯỢNG GIÁ

- Đánh giá khả năng thành công của một nước đi (thắng, thua, hòa?)
- Ví dụ xây dựng hàm lượng giá
  - Đối với trò cờ vua, checkers: gán giá trị cho mỗi quân trên bàn cờ, vị trí của nó và tổng hợp lại.
  - Tổng quát: hàm lượng giá tuyến tính tổng các đặc trưng có được của một đối thủ

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

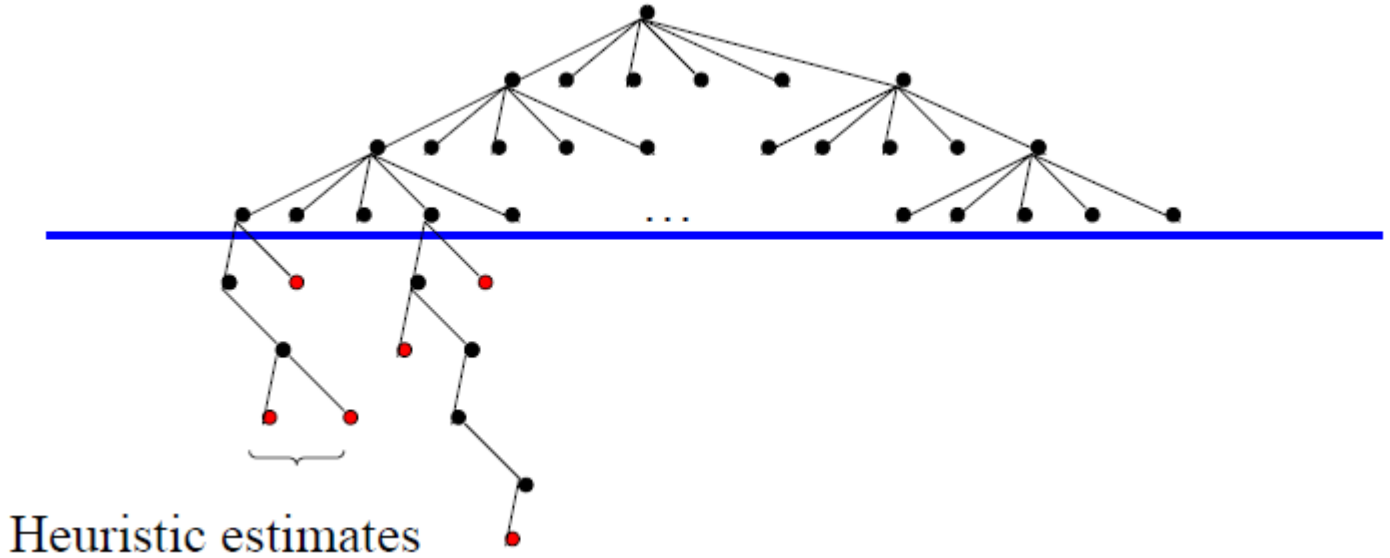
    - w_i: trọng số gán cho quân thứ i (ví dụ: hậu $w = 9$, mã $w = 3$…)
    - f_i: số quân còn lại

HÀM LƯỢNG GIÁ

- *MINIMAXCutoff* giống hệt tìm kiếm *MINIMAXValue* trừ:
  - Thay *Terminal?* bằng *Cutoff?*
  - Thay *Utility()* bằng *Eval()*

MỞ RỘNG Ý TƯỞNG

- Đối với trò chơi thực, giới hạn tập nước đi cần xét **dưới mức cắt ngang** để giảm phân nhánh và tăng hiệu quả hàm lượng giá.
  - Ví dụ: chỉ xét những nước ăn quân trong cờ vua



TỔNG KẾT

- Biết khái niệm trò chơi đối kháng, các thành phần MIN và MAX trong trò chơi.
- Nắm vững quyết định tối ưu trò chơi: thuật toán MINIMAX, tỉa nhánh α-β
- Nắm vững hàm lượng giá

TÀI LIỆU THAM KHẢO

- Tài liệu bài giảng môn học
- Chapter 5. S. Russel and P.Norvig, Artificial Intelligence – A Modern Approach. *Third Edition.* 2010
- Milos Hauskrecht's lecture

www.cs.pitt.edu/~milos/courses/cs2710/lectures/Class8.pdf

KẾT THÚC CHỦ ĐỀ