Tìm kiếm Đối kháng – Trò chơi

Lê Ngọc Thành
Khoa Công nghệ Thông tin
lnthanh@fit.hcmus.edu.vn

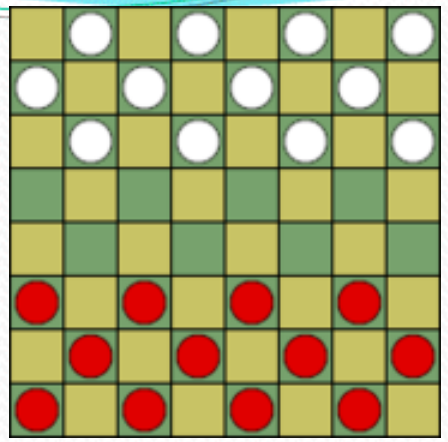
Tổng quan

- Trò chơi
- Quyết định tối ưu trong Trò chơi
- Thuật toán MINIMAX
- Tỉa nhánh α-β
- Hàm lượng giá, Tìm kiếm cắt nhánh

Trò chơi

- Là một trong những đặc tính được xem là "thông minh" của con người
- Các trò chơi ra đời gần như cùng lúc với AI
- Đã dành được những thành tựu đáng kể
- Ở đây ta xem xét các dạng trò chơi trí tuệ (board game)

Trò chơi



- Checkers:
  - Hai người chơi
  - Người chơi lần lượt di chuyển quân của mình theo đường chéo, 1 lần 1 ô
  - Nếu có quân đối phương trước mặt, có thể nhảy qua (nếu có ô trống) và ăn
  - Ván cờ kết thúc khi một trong hai người không còn nước đi

Trò chơi

- Checker
  - 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 hoàn hảo
    - Lưu trữ 444 tỷ vị trí với hàng tetrabyte bộ nhớ

Trò chơi

- Cờ vua
  - 1997, DeepBlue 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 200.000.000 nước đi mỗi giây so với 2 của Kasparov
    - (99.99% nước đi được xem là ngu ngốc)
    - Hàm lượng giá cực kỳ phức tạp

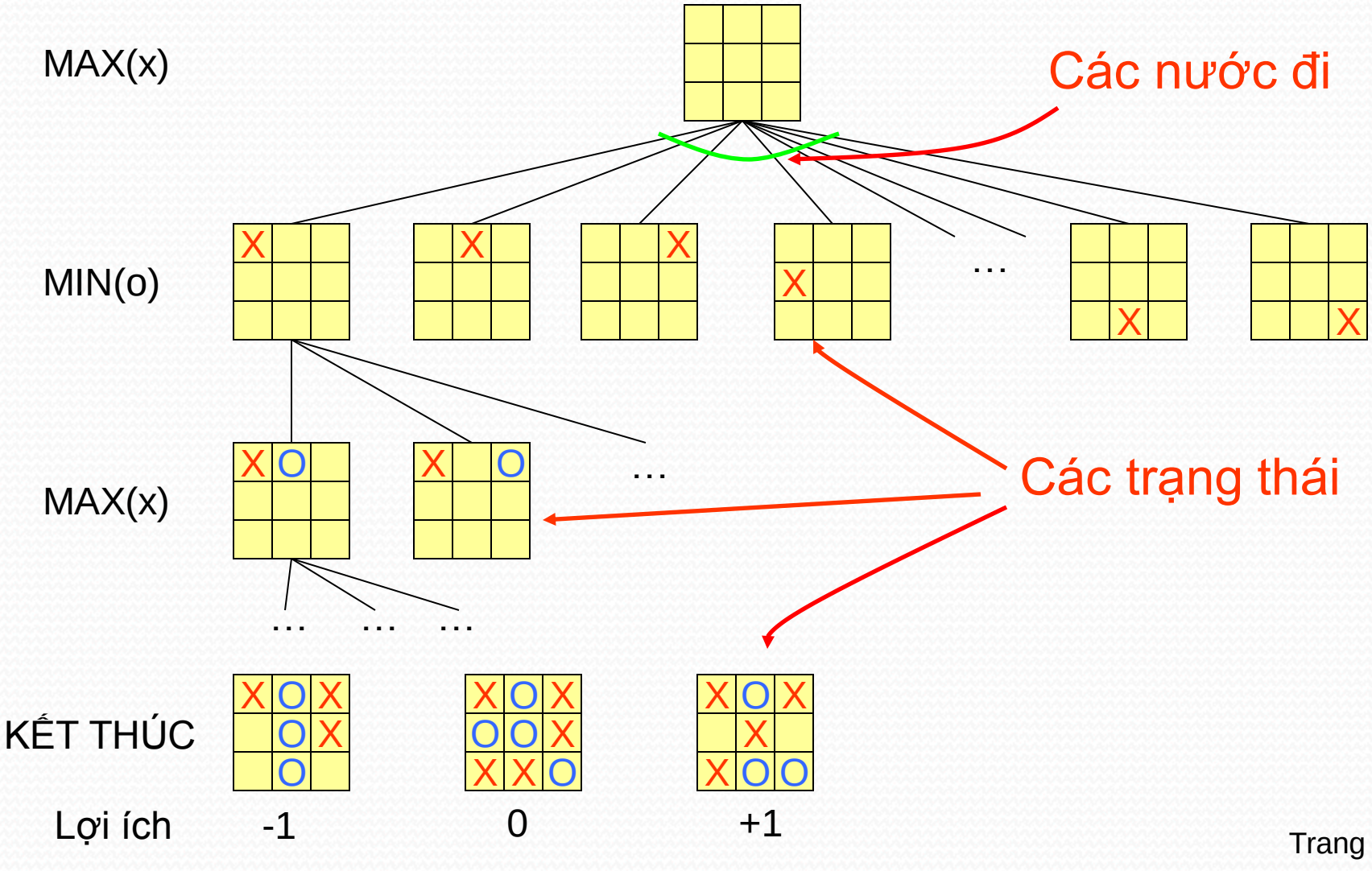
Trò chơi

- Một số khác:
  - Othello: 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)

Quyết định tối ưu trong Trò chơi

- Lời giải tối ưu: một 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

Ví dụ cây tìm kiếm trò chơi - TicTacToe



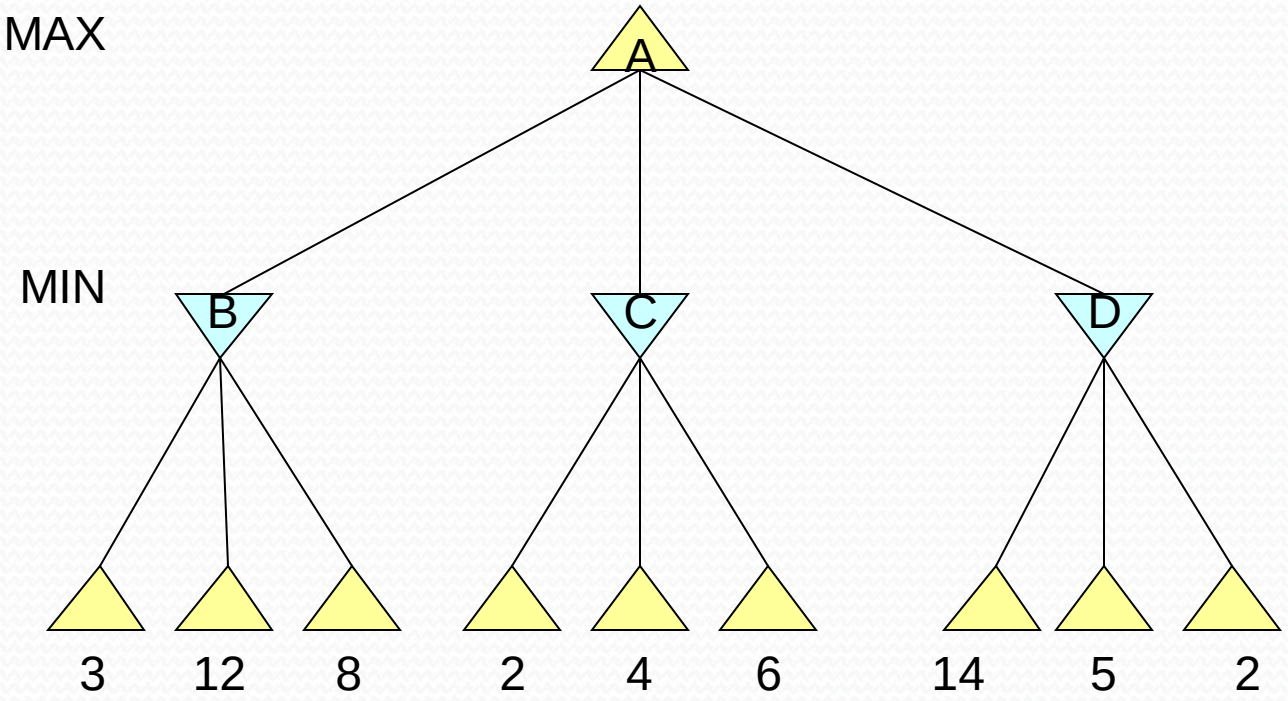
Thuật toán MINIMAX

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

Giá trị MINIMAX

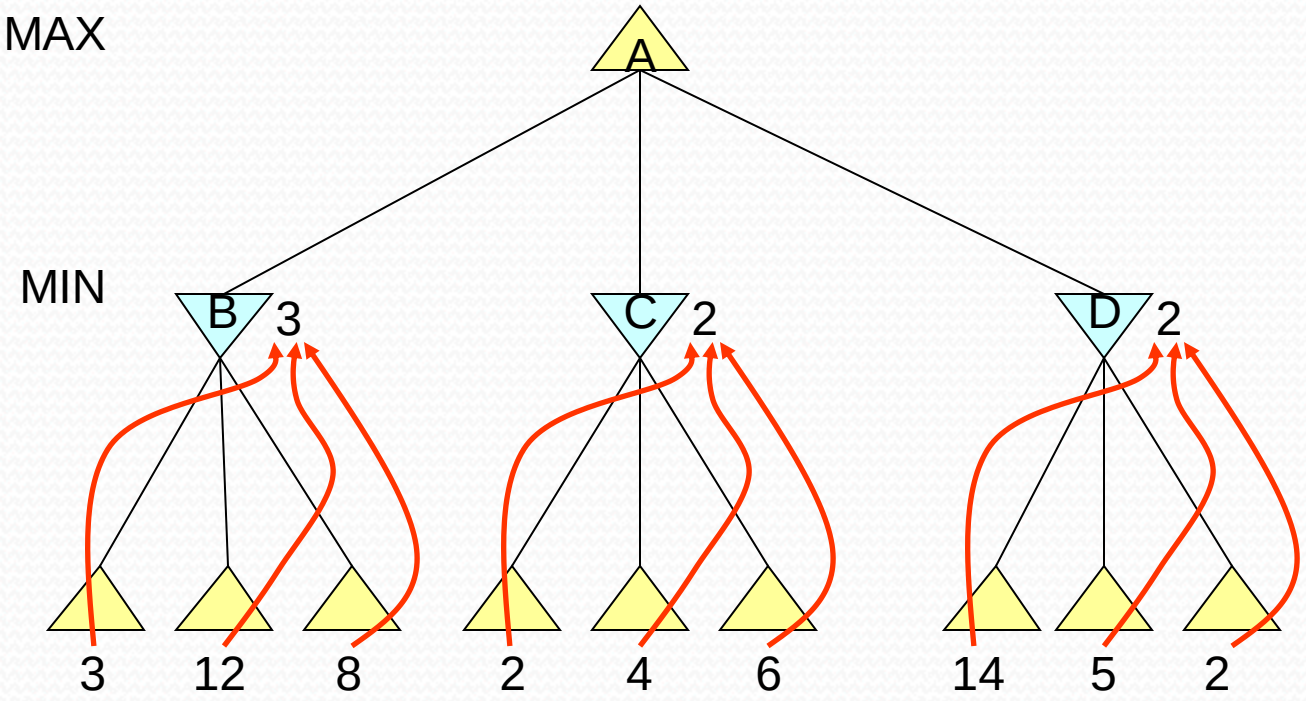
- MINIMAX-VALUE(n) =
  - Utility(n) *nếu n là trạng thái kết thúc*
  - max{MINIMAX-VALUE(s) | s∈succs(n)} *nếu n là một nút MAX*
  - min{MINIMAX-VALUE(s) | s∈succs(n)} *nếu n là một nút MIN*

Giá trị MINIMAX (vd)



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

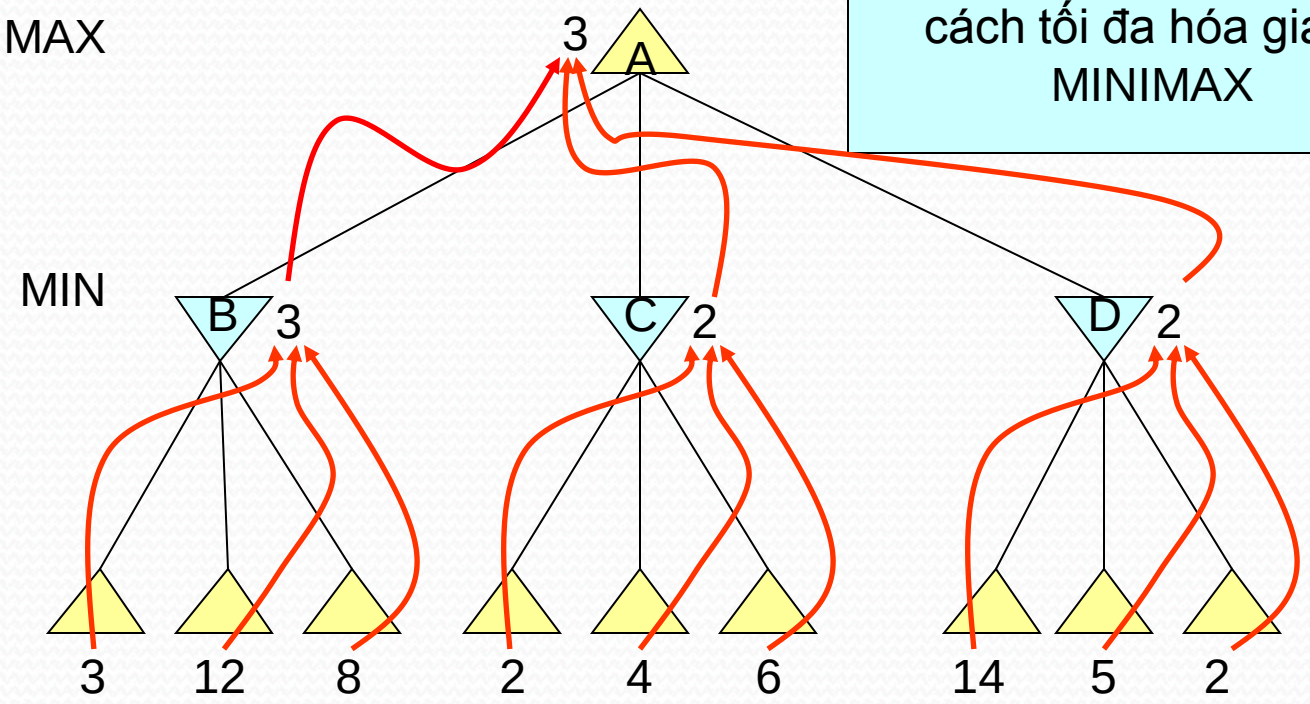
Giá trị MINIMAX (vd)



Tại mỗi trạng thái có thể, MIN luôn chọn đường đi tối thiểu hóa giá trị tiện ích ở trạng thái kết thúc

Giá trị MINIMAX (vd)

Đến lượt mình, MAX tìm cách tối đa hóa giá trị MINIMAX



Và MAX chọn chiến lược đi đến B ứng với giá trị MINIMAX tối đa

Thuật toán MINIMAX

```
function MINIMAX-DECISION(state) returns an action
    v ← MAX-VALUE(state)
    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
        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
        v ← MIN(v, MAX-VALUE(s))
    return v
```

Đánh giá Thuật giải MINIMAX

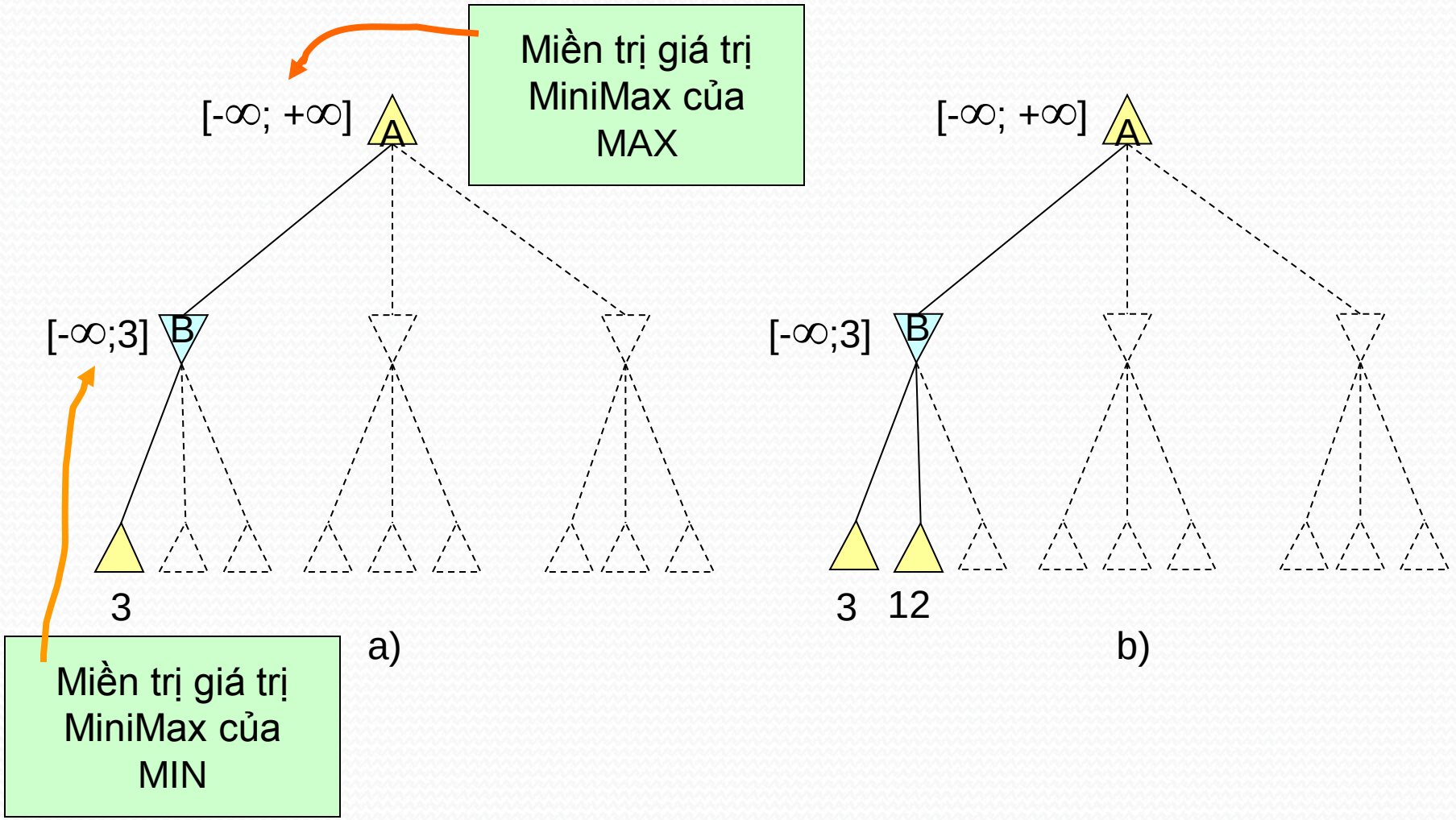
- Đầy đủ? Có (nếu cây tìm kiếm hữu hạn)
- Tối ưu? Có (với một đối thủ tối ưu)
- Độ phức tạp thời gian? $O(b^m)$
- Độ phức tạp không gian? $O(b^m)$ (tìm kiếm theo chiều sâu)

- Với cờ vua, b ≈ 35, m ≈100 với một ván thông thường → hoàn toàn không thể tìm được lời giải tối ưu

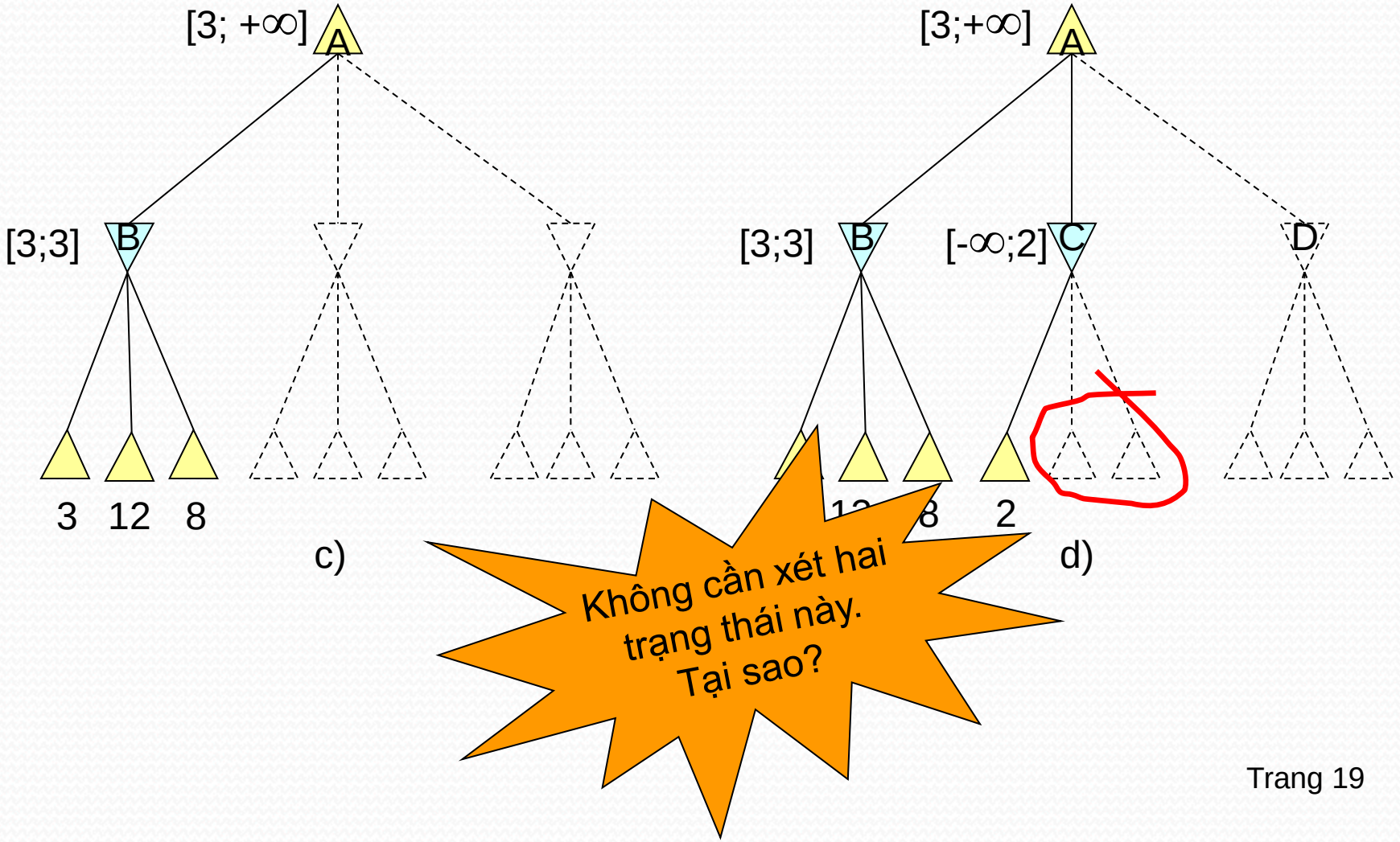
Tỉa nhánh α-β

- Ta có thể làm gì để giảm số trạng thái phải kiểm tra?
- Mẹo: ta có thể tính đúng giá trị quyết định minimax mà không cần duyệt mọi đỉnh.
- Hãy xem xét chi tiết từng bước quá trình tính giá trị minimax.
- Ghi nhớ: thuật toán MINIMAX duyệt theo chiều sâu.

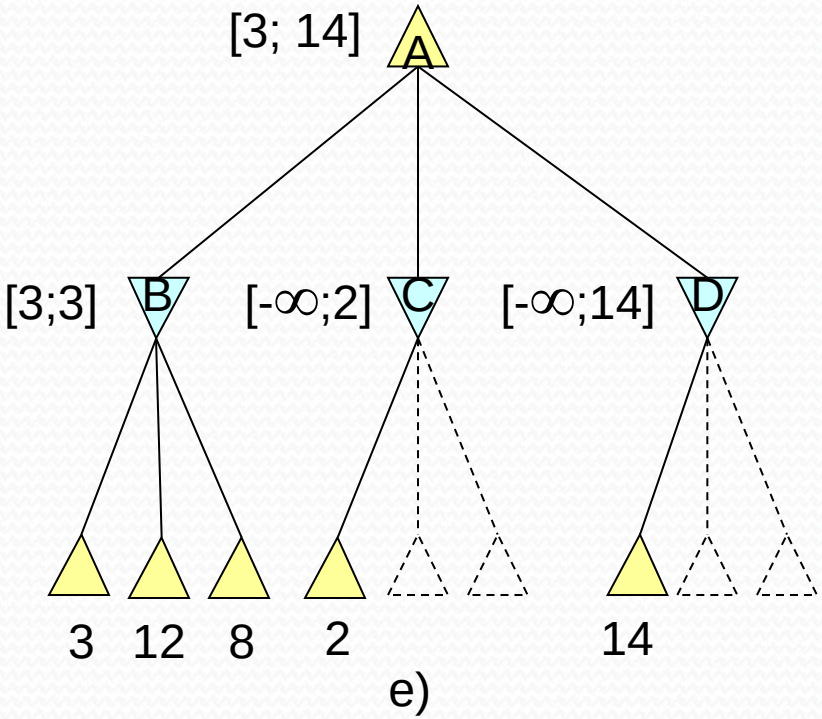
Tỉa nhánh α-β (vd)



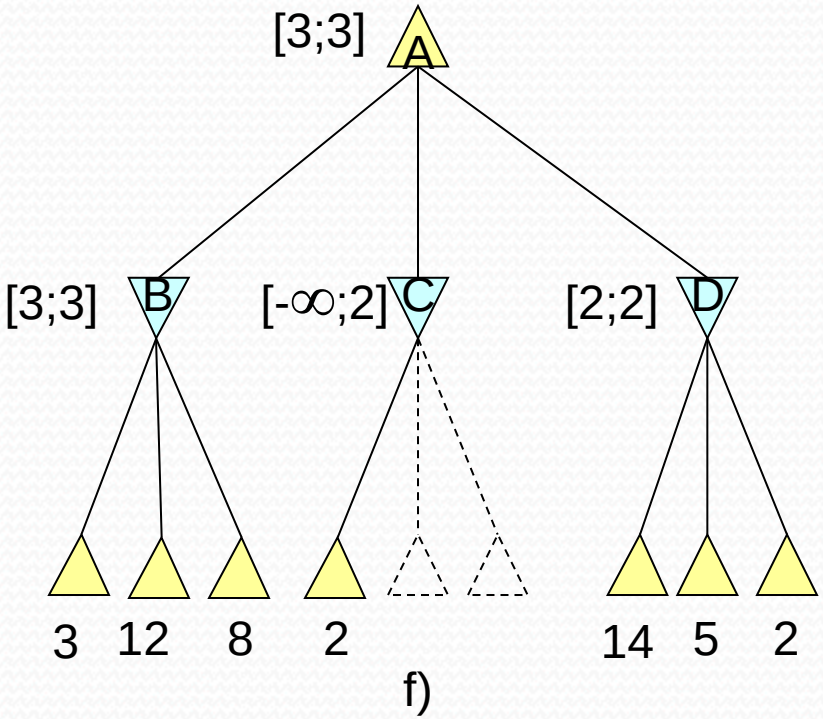
Tỉa nhánh α-β (vd)



Tỉa nhánh α-β (vd)



e)



f)

Tỉa nhánh α-β (vd)

- Gọi x, y là lợi ích của các trạng thái không xét. Ta có:

MINIMAX-VALUE(gốc) = max(min(3,12,8),
min(2,x,y),min(14,5,2))
= max(3, min(2,x,y), 2)
= max(3, z, 2) với z <= 2
= 3

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

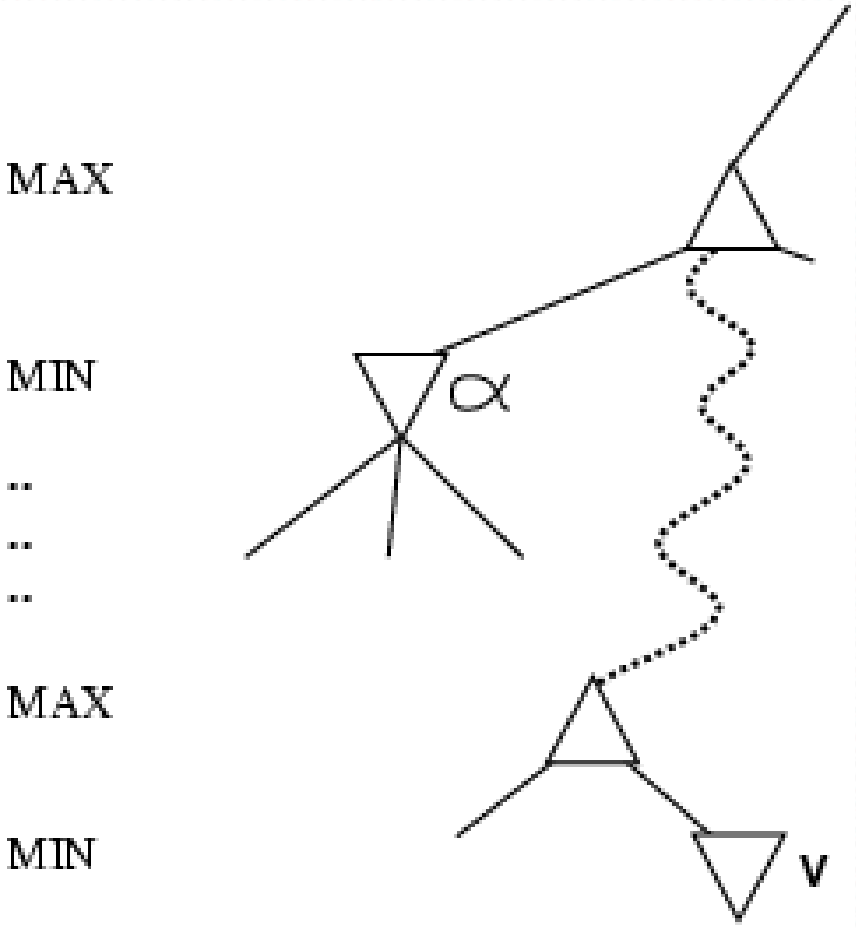
Đánh giá α-β

- 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 = $O(b^{m/2})$ (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) tại một điểm bất kỳ trên một đường đi cho *MAX*
- Nếu v xấu hơn α, MAX sẽ tránh nó

  →Tỉa nhánh này
- Định nghĩa β tương tự cho MIN



Thuật toán α-β

function ALPHA-BETA-SEARCH(*state*) **returns** *an action*
  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*
  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

Thuật toán α-β (tt)

```
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
```

Hàm lượng giá

- 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 -> tìm kiếm giới hạn (cut-off search)
- Cần một 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?)
- Đánh giá tuyến tính tổng các đặc trưng có được của một đối thủ

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

  trong đó: w_i: trọng số gán cho quân thứ I

  (ví dụ: hậu w=9, ngựa w= 3…)

  f_i: số quân còn lại

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

Điều cần nắm

- Các thành phần trò chơi, MIN, MAX
- Thuật toán MINIMAX, thuật toán α-β
- Đánh giá của các thuật toán
- Hàm lượng giá