TRƯỜNG ĐẠI HỌC
BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY
OF SCIENCE AND TECHNOLOGY

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
SCHOOL OF INFORMATION AND COMMUNICATIONS TECHNOLOGY

IT3160
Nhập môn Trí tuệ nhân tạo

Artificial Intelligence

PGS.TS Lê Thanh Hương & TS Đỗ Tiến Dũng

ONE LOVE. ONE FUTURE.

Nội dung môn học

- Chương 1 - Giới thiệu về Trí tuệ nhân tạo
- Chương 2 - Tác tử
- Chương 3 - Giải quyết vấn đề
  - Các kỹ thuật tìm kiếm cơ bản
  - Tìm kiếm với tri thức bổ sung (Informed search)
  - Các kỹ thuật tìm kiếm có đối thủ
  - Tìm kiếm dựa trên thỏa mãn ràng buộc
- Chương 4 - Logic và suy diễn
- Chương 5 - Học máy

Tìm kiếm có đối thủ

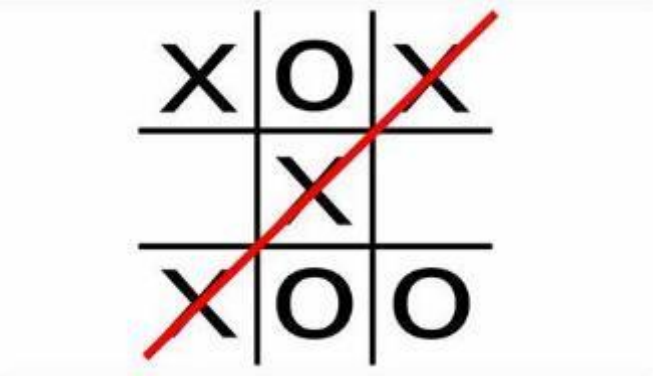
- Các thủ tục tìm kiếm sâu dần (IDS) và tìm kiếm A* hữu dụng đối với các bài toán (tìm kiếm) liên quan đến một tác tử
- Thủ tục tìm kiếm cho các bài toán liên quan đến 2 tác tử có mục tiêu đối nghịch nhau (xung đột với nhau)?
  - Tìm kiếm có đối thủ (Adversarial search)
- Phương pháp tìm kiếm có đối thủ được áp dụng phổ biến trong các trò chơi (games)

Các vấn đề của tìm kiếm trong trò chơi

- Khó dự đoán trước được phản ứng của đối thủ
  - Cần xác định (xét) một nước đi phù hợp đối với mỗi phản ứng (nước đi) có thể của đối thủ
- Giới hạn về thời gian (trò chơi có tính giờ)
  - Thường khó (hoặc không thể) tìm được giải pháp tối ưu $\rightarrow$ Xấp xỉ
- Tìm kiếm có đối thủ đòi hỏi tính hiệu quả (giữa chất lượng của nước đi và chi phí thời gian)
  - Đây là một yêu cầu khó khăn
- Nguyên tắc trong nhiều trò chơi đối kháng
  - Một người chơi thắng = Người chơi kia thua
  - Mức độ (tổng điểm) thắng của một người chơi = Mức độ (tổng điểm) thua của người chơi kia

Trò chơi cờ ca-rô

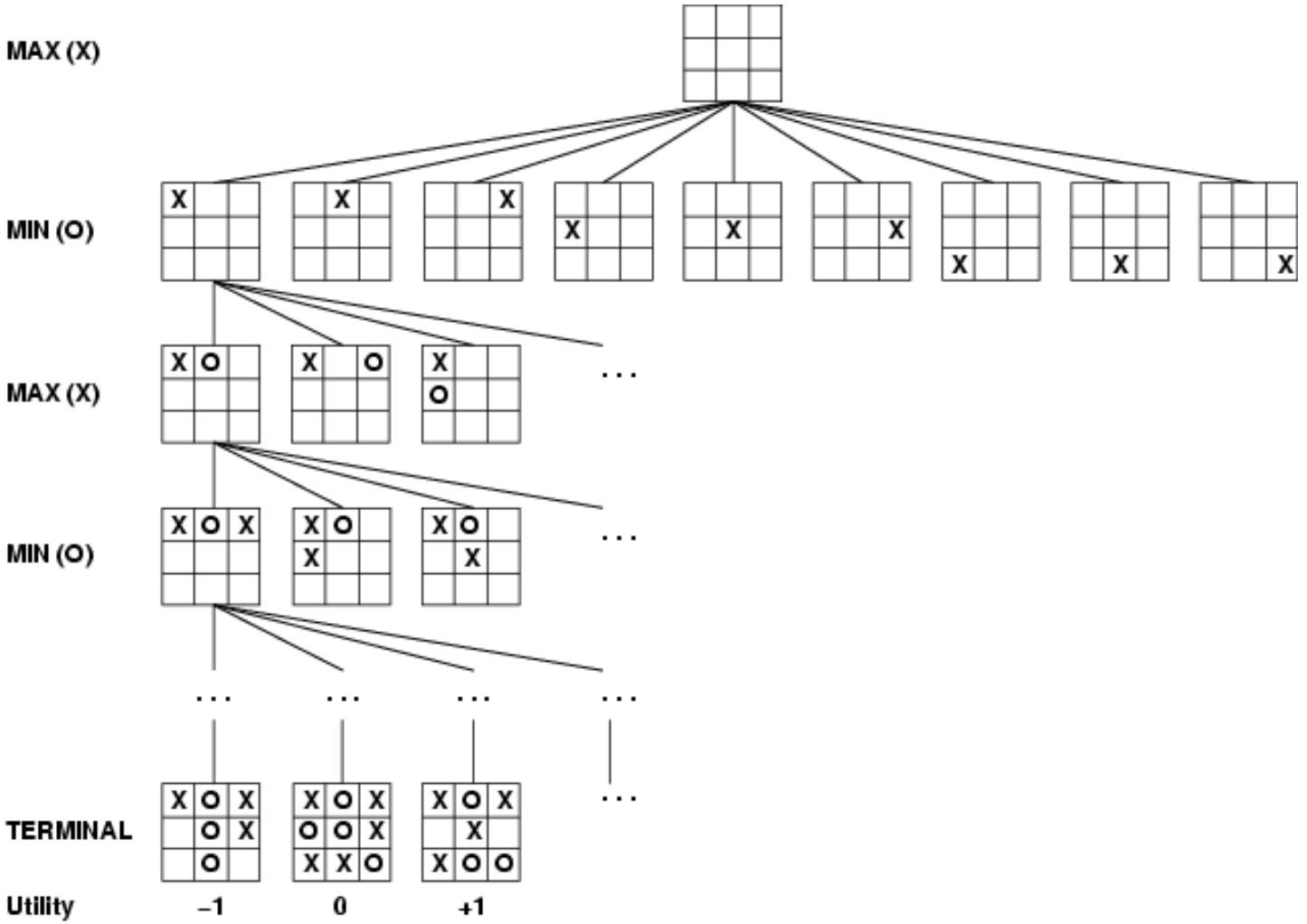
- Trò chơi cờ ca-rô là một ví dụ phổ biến trong AI để minh họa về tìm kiếm có đối thủ
  - Vd: http://www.ourvirtualmall.com/tictac.htm
- Là trò chơi đối kháng giữa 2 người
  - Thay phiên nhau đi các nước (moves) (X và O)
  - Kết thúc trò chơi: Người thắng đánh dấu được vào các ô thẳng hàng



Biểu diễn bài toán trò chơi đối kháng

- Trò chơi bao gồm các thông tin
  - Trạng thái bắt đầu (Initial state): Trạng thái của trò chơi + Người chơi nào được đi nước đầu tiên
  - Hàm chuyển trạng thái (Sucessor function): Trả về thông tin gồm (nước đi, trạng thái)
    - Tất cả các nước đi hợp lệ từ trạng thái hiện tại
    - Trạng thái mới (là trạng thái chuyển đến sau nước đi)
  - Kiểm tra kết thúc trò chơi (Terminal test)
  - Hàm tiện ích (Utility function) để đánh giá các trạng thái
- Trạng thái bắt đầu + Các nước đi hợp lệ = Cây biểu diễn trò chơi (Game tree)

Cây biểu diễn trò chơi cờ ca-rô

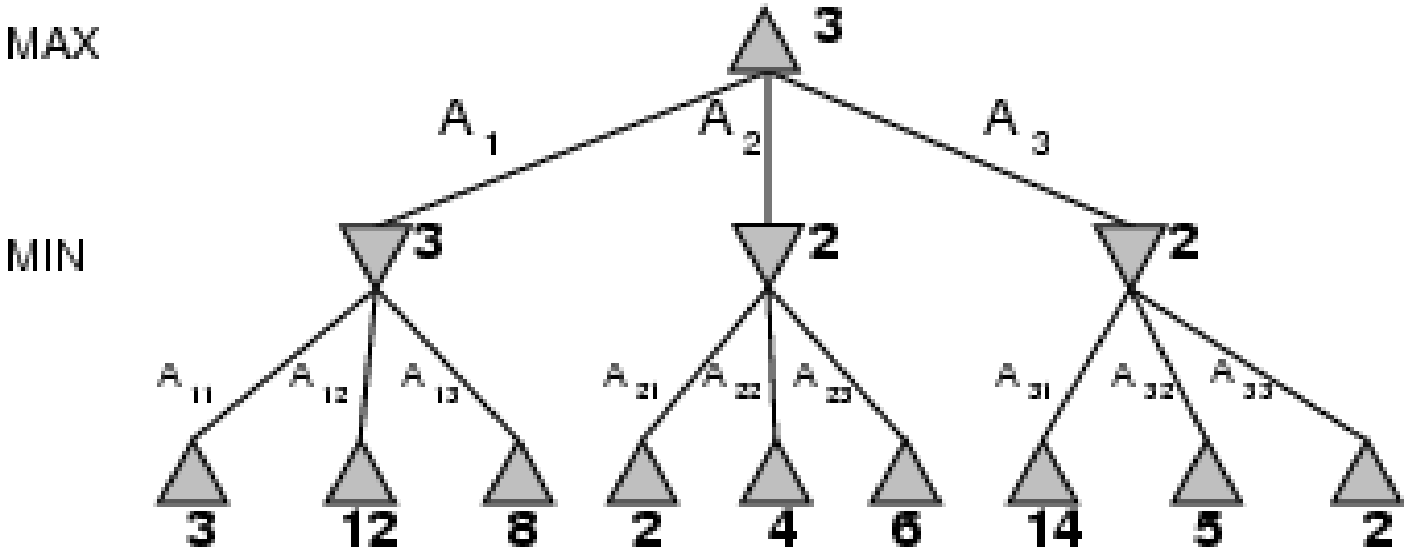


Các chiến lược tối ưu

- Một chiến lược tối ưu là một chuỗi các nước đi giúp đưa đến trạng thái đích mong muốn (vd: chiến thắng)
- Chiến lược của MAX bị ảnh hưởng (phụ thuộc) vào các nước đi của MIN – và ngược lại
- MAX cần chọn một chiến lược giúp cực đại hóa giá trị hàm mục tiêu – với giả sử là MIN đi các nước đi tối ưu
  - MIN cần chọn một chiến lược giúp cực tiểu hóa giá trị hàm mục tiêu
- Chiến lược này được xác định bằng việc xét giá trị MINIMAX đối với mỗi nút trong cây biểu diễn trò chơi
  - Chiến lược tối ưu đối với các trò chơi có không gian trạng thái xác định (deterministic states)

Giá trị MINIMAX

- MAX chọn nước đi ứng với giá trị MINIMAX cực đại (để đạt được giá trị cực đại của hàm mục tiêu)
- Ngược lại, MIN chọn nước đi ứng với giá trị MINIMAX cực tiểu



Giải thuật 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
```

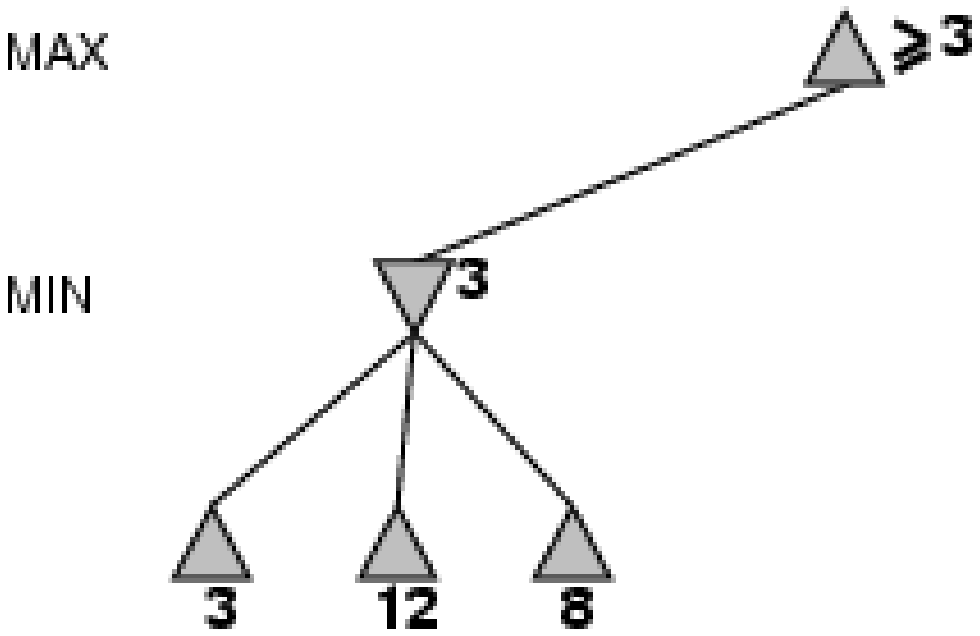
Giải thuật MINIMAX: các đặc điểm

- Tính hoàn chỉnh
  - Có (nếu cây biểu diễn trò chơi là hữu hạn)
- Tính tối ưu
  - Có (đối với một đối thủ luôn chọn nước đi tối ưu)
- Độ phức tạp về thời gian
  - $O(b^m)$
- Độ phức tạp về bộ nhớ
  - O(bm) (khám phá theo chiến lược tìm kiếm theo chiều sâu)
- Đối với trò chơi cờ vua, hệ số phân nhánh b ≈35 và hệ số mức độ sâu của cây biểu diễn m≈100
  - Chi phí quá cao – Không thể tìm kiếm chính xác nước đi tối ưu

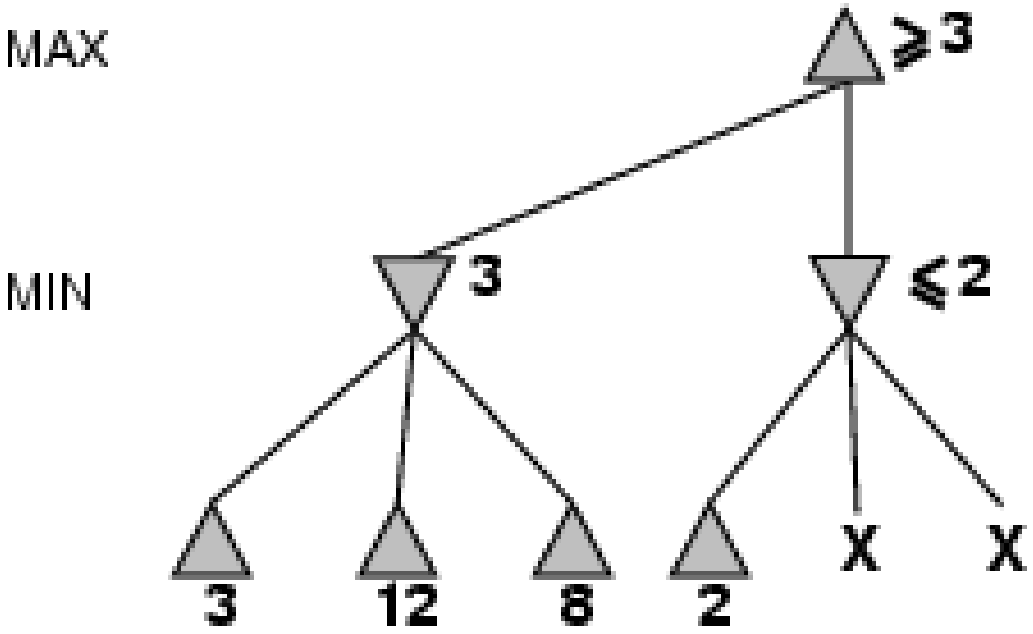
Cắt tỉa tìm kiếm

- Vấn đề: Giải thuật tìm kiếm MINIMAX vấp phải vấn đề bùng nổ (mức hàm mũ) các khả năng nước đi cần phải xét → không phù hợp với nhiều bài toán trò chơi thực tế
- Chúng ta có thể cắt tỉa (bỏ đi – không xét đến) một số nhánh tìm kiếm trong cây biểu diễn trò chơi
- Phương pháp cắt tỉa α-β (Alpha-beta prunning)
  - Ý tưởng: Nếu một nhánh tìm kiếm nào đó không thể cải thiện đối với giá trị (hàm tiện ích) mà chúng ta đã có, thì không cần xét đến nhánh tìm kiếm đó nữa!
  - Việc cắt tỉa các nhánh tìm kiếm ("tồi") không ảnh hưởng đến kết quả cuối cùng

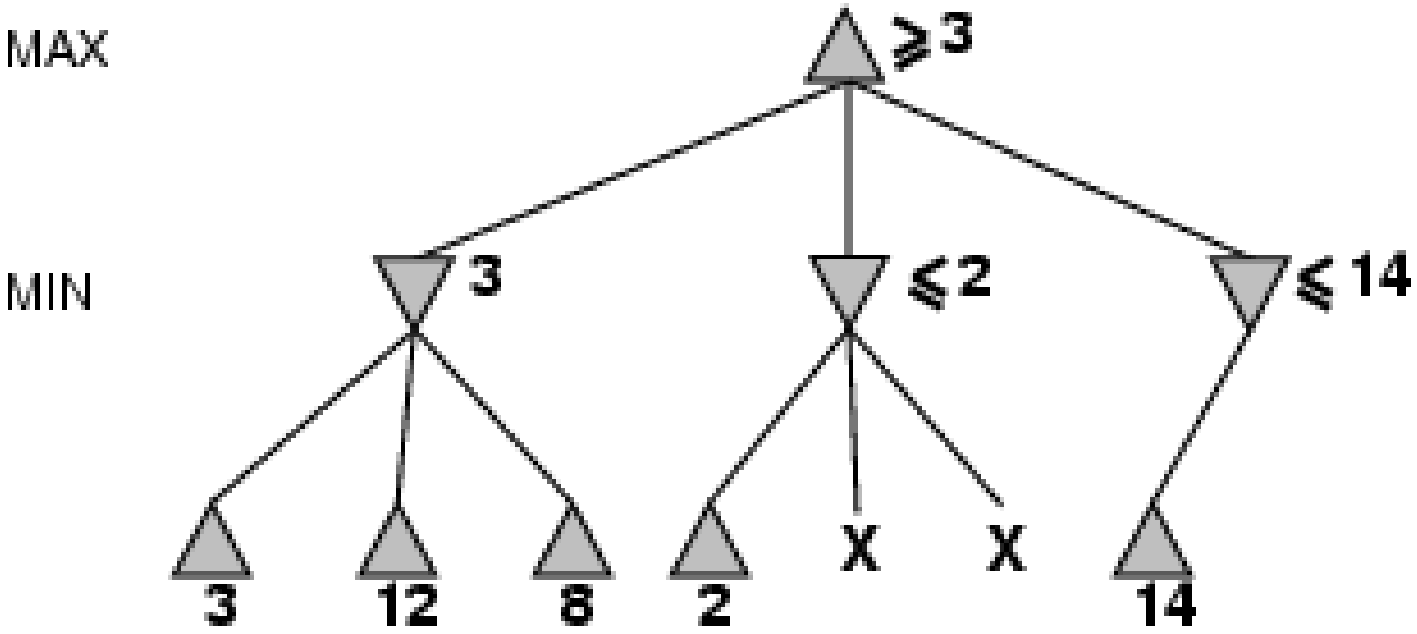
Cắt tỉa α-β – Ví dụ (1)



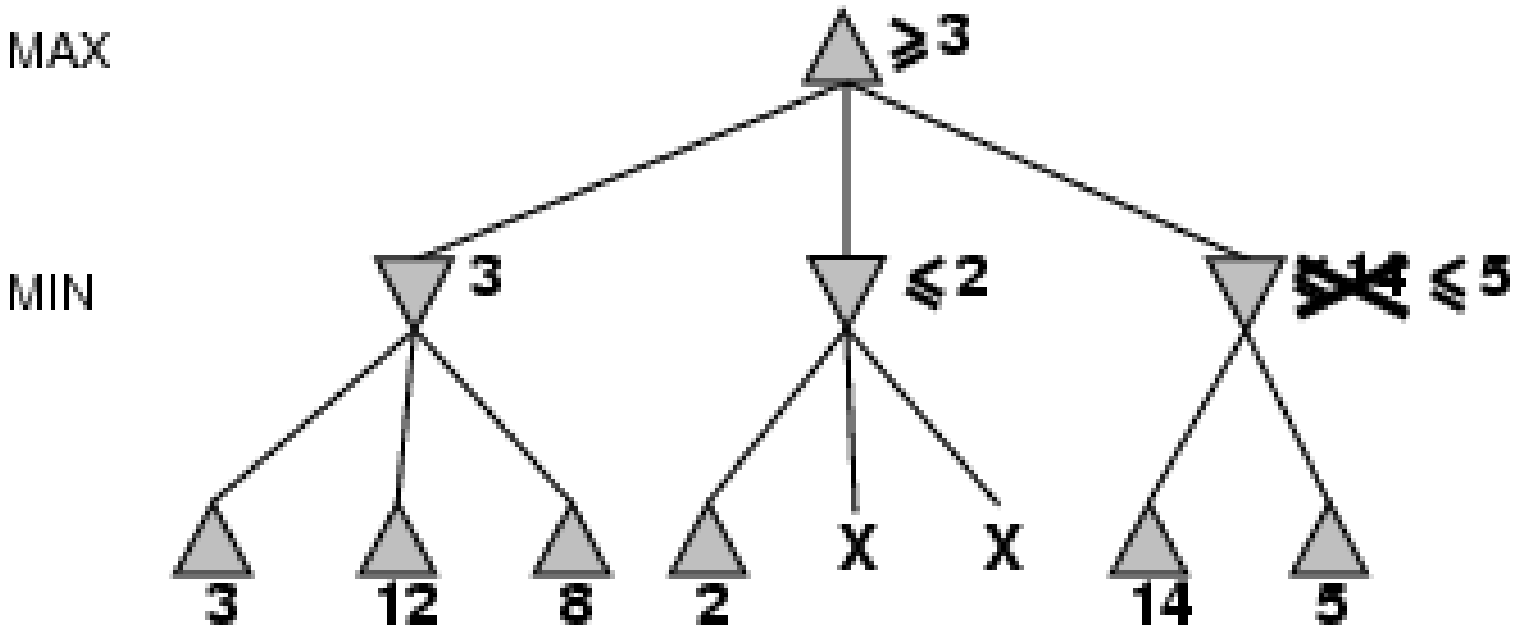
Cắt tỉa α-β – Ví dụ (2)



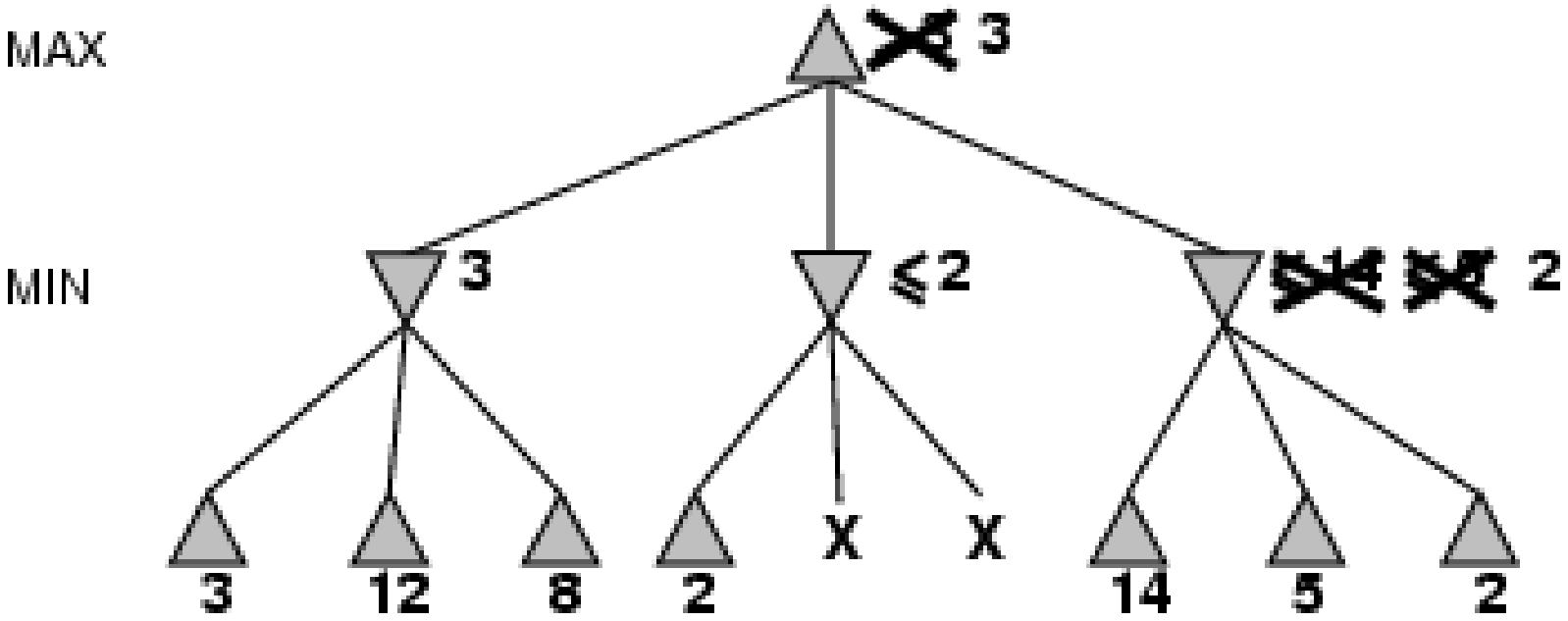
Cắt tỉa α-β – Ví dụ (3)



Cắt tỉa α-β – Ví dụ (4)

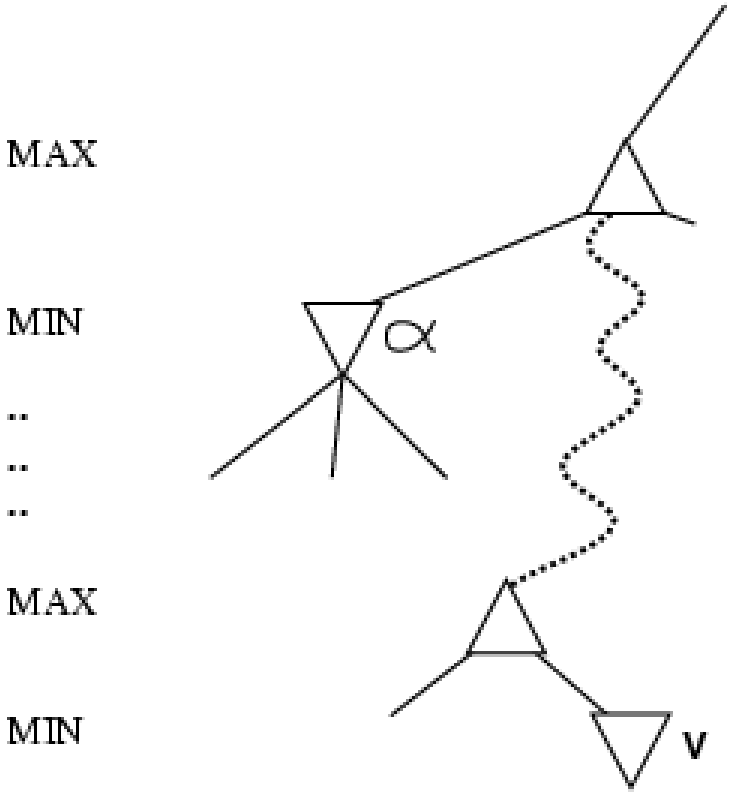


Cắt tỉa α-β – Ví dụ (5)



Tại sao được gọi là cắt tỉa α-β?

- α là giá trị của nước đi tốt nhất đối với MAX (giá trị tối đa) tính đến hiện tại đối với nhánh tìm kiếm
- Nếu v là giá trị tồi hơn α, MAX sẽ bỏ qua nước đi ứng với v
  - Cắt tỉa nhánh ứng với v
- β được định nghĩa tương tự đối với MIN



Giải thuật cắt tỉa α-β (1)

```
function ALPHA-BETA-SEARCH(state) returns an action
    inputs: state, current state in game

    v ← MAX-VALUE(state, −∞, +∞)
    return the action in SUCCESSORS(state) with value v

---

function MAX-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 ← MAX(v, MIN-VALUE(s, α, β))
        if v ≥ β then return v
        α ← MAX(α, v)
    return v
```

Giải thuật cắt tỉa α-β (2)

```
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
```

Cắt tỉa α-β

- Đối với các trò chơi có không gian trạng thái lớn, thì phương pháp cắt tỉa α-β vẫn không phù hợp
  - Không gian tìm kiếm (kết hợp cắt tỉa) vẫn lớn
- Có thể hạn chế không gian tìm kiếm bằng cách sử dụng các tri thức cụ thể của bài toán
  - Tri thức để cho phép đánh giá mỗi trạng thái của trò chơi
  - Tri thức bổ sung (heuristic) này đóng vai trò tương tự như là hàm ước lượng *h(n)* trong giải thuật tìm kiếm A*