Busca com Adversários

- Múltiplos Agentes
- Ambiente Competitivo
  - Objetivos Conflitantes

- Jogos Clássicos
  - Xadrez, Damas, Jogo da Velha, Gamão, Go, Reversi (Othelo), etc...
  - Sempre foram um grande desafio em IA

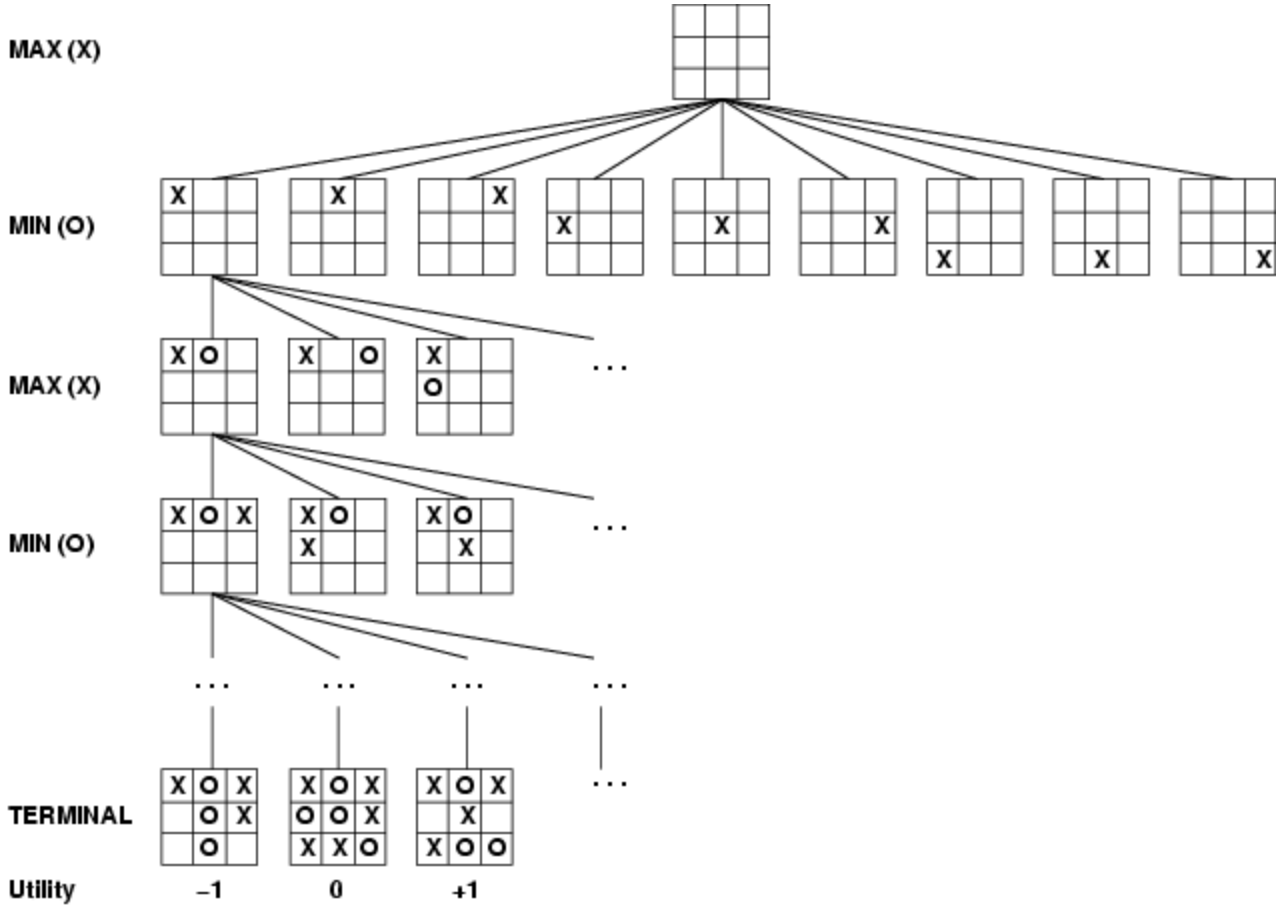
Jogos Clássicos em IA

- Características "gerais" dos Jogos Clássicos
  - Determinísticos, completamente observáveis
  - 2 Jogadores, *turn taking*
  - *Zero-Sum Games*
    - A função de utilidade dos jogadores adversários no final do jogo tem valor igual e oposto.
  - Um caso especial do conceito matemático de **Teoria dos Jogos** – (Jonh Nash *"Beautiful Mind")*
- Exemplos
  - Xadrez, Damas, Jogo da Velha, Gamão, Go, Reversi (Othelo), etc...

Jogos Clássicos em IA

- Em geral, são problemas difíceis.
  - Xadrez:
    - Jogadas possíveis: 35 (em média)
    - Tempo de jogo: 50 jogadas, cada jogador
    - Árvore com 35^100 nodos (=10^154). 10^40 distintos
- Árvore do Jogo
  - Estado inicial: tabuleiro, primeiro jogador
  - Função sucessora: <jogada, estado>
  - Teste de fim: chegou a um estado terminal?
  - Função de utilidade: valor do estado terminal

Árvore do Jogo da Velha



Estratégias Ótimas

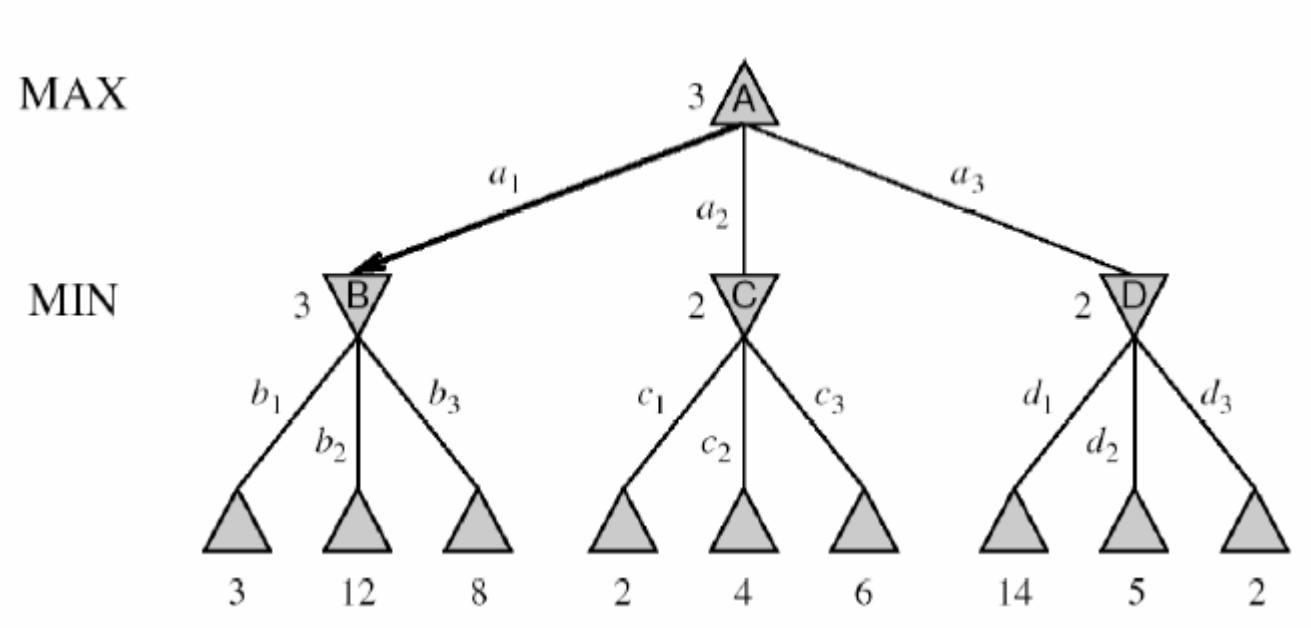
- O movimento do adversário é imprevisível
- Portanto a estratégia ótima **leva ao melhor resultado considerando que o adversário sempre faz a melhor jogada possível**
- Algoritmo **MiniMax**
  - Jogadas alternam entre Max e Min
  - Valor MiniMax: utilidade daquele estado considerando jogadas ótimas até o final
    - Utilidade(s) se s é um nodo terminal
    - Maior MiniMax dos sucessores de s, se s é Max
    - Menor MiniMax dos sucessores de s, se s é Min

Estratégias Ótimas

- Mais formalmente, função recursiva

$$
\begin{aligned}
&\text{MinimaxValue}(n) = \\
&\quad \text{Utility}\ (n) && \text{if } n \text{ is a terminal state} \\
&\quad \max_{s \in \ Successors(n)} \text{MinimaxValue}(s) && \text{if } n \text{ is a MAX node} \\
&\quad \min_{s \in \ Successors(n)} \text{MinimaxValue}(s) && \text{if } n \text{ is a MIN node}
\end{aligned}
$$

Exemplo: jogo com 2 jogadas



- A estratégia ótima para Max, na verdade é a melhor solução para o pior caso
- O que ocorre se Min não jogar otimamente?
  - Max poderá jogar ainda melhor…

Algoritmo MiniMax

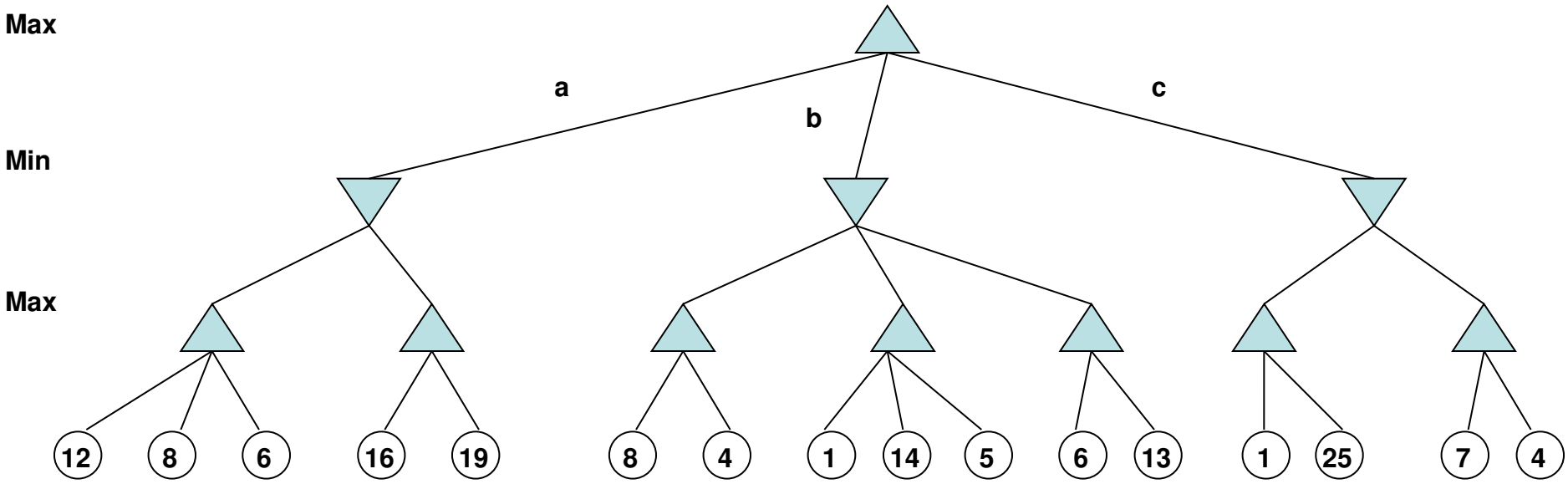
```
function MINIMAX-DECISION(state) returns an action
    v ← MAX-VALUE(state)
    return the action in SUCCESSORS(state) with value v

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
```

Exercício

- Qual jogada deve ser feita pelo jogador max (a, b ou c)? Porque?

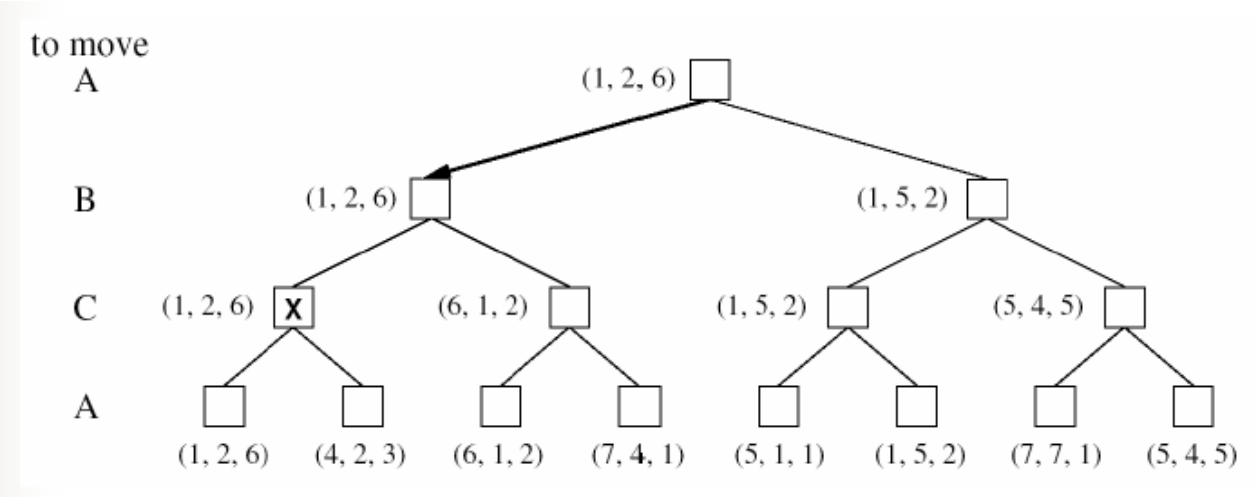


Algoritmo MiniMax

- Faz uma caminhamento em profundidade completo da árvore!
- Se a profundidade é **m** e em cada estado existem **b** jogadas possíveis, a ordem de complexidade é $\mathbf{O(b^m)}$
- Ou seja esse algoritmo não é prático para jogos reais
  - Mas serve para análise matemática e como base para algoritmos mais eficientes

Multiplayer games

- O MiniMax pode ser estendido para jogos com múltiplos jogadores
  - Uso de um vetor de utilidades

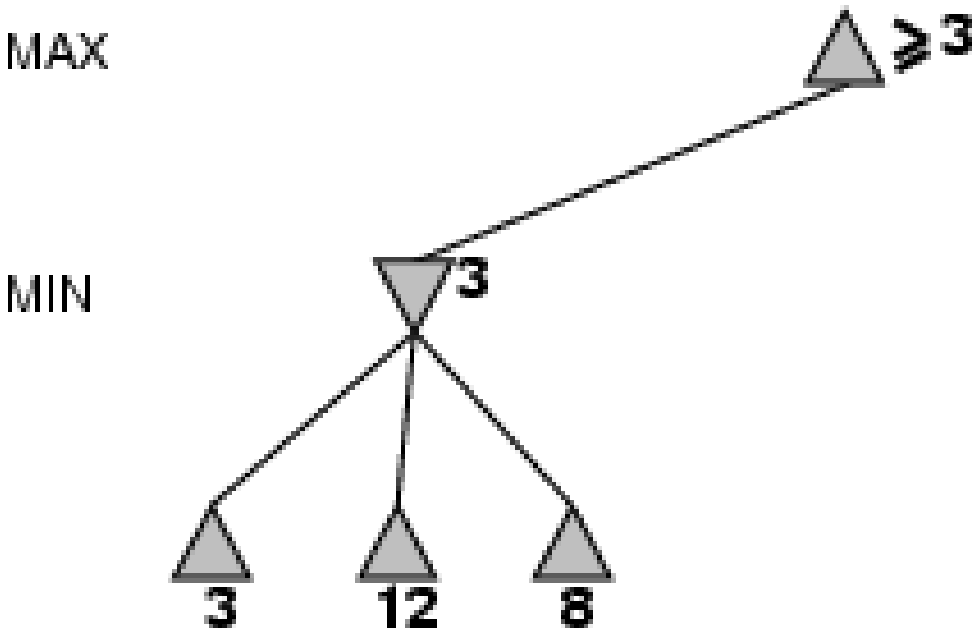


- Alianças
  - Podem ocorrer naturalmente no processo de maximizar a função de utilidade

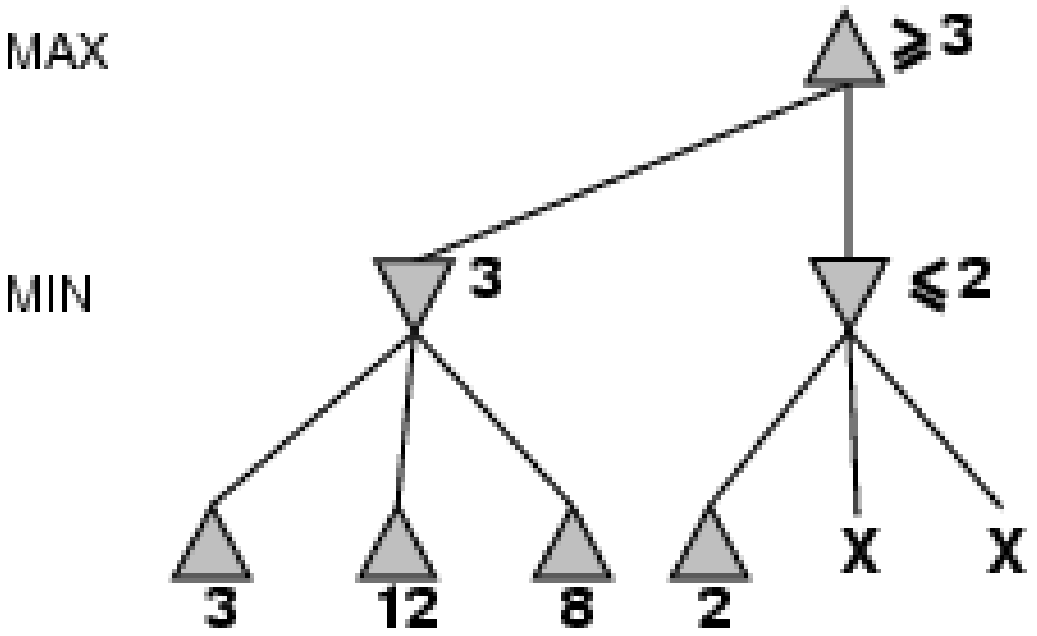
Alpha-Beta Pruning

- Problema com o MiniMax
  - # de estados é exponencial
- *Pruning* = Poda
  - Não examinar grandes partes da árvore, diminuindo assim o custo
  - Pode causar a "perda" da solução
- Alpha-Beta Prunning
  - Retorna a mesma ação do MiniMax, mas elimina caminhos que não influenciam a decisão
  - NÃO altera a solução

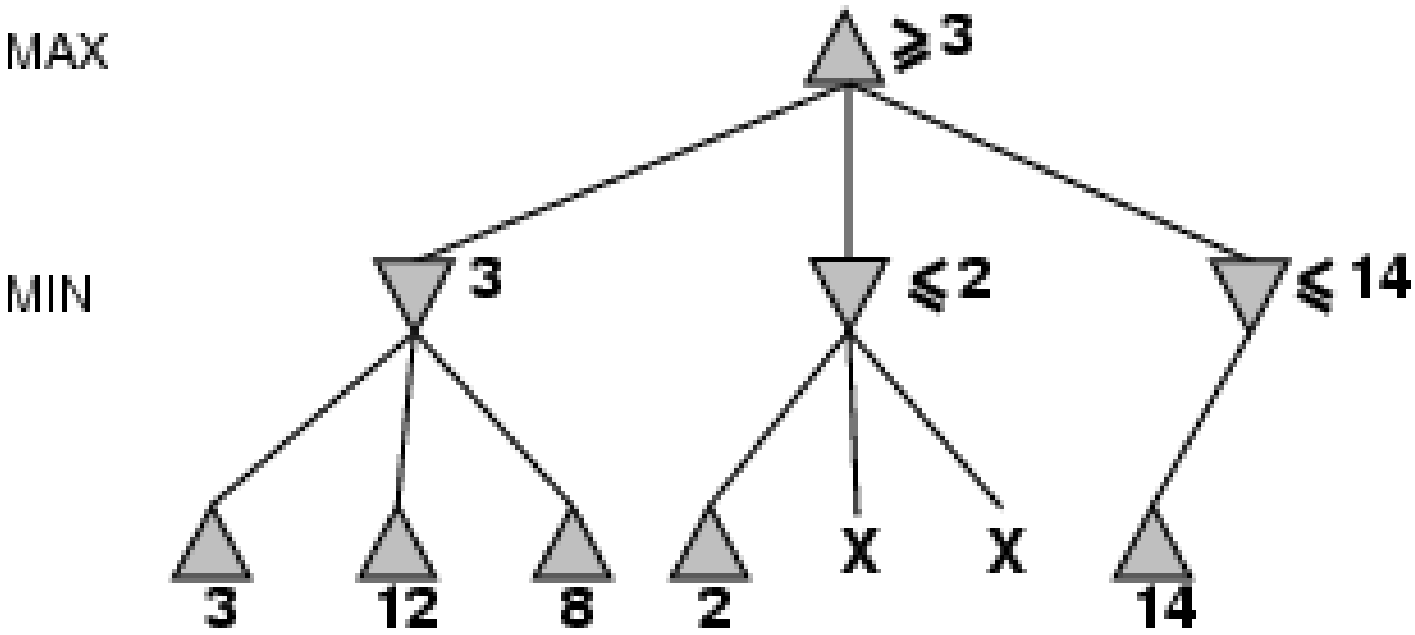
α-β pruning example



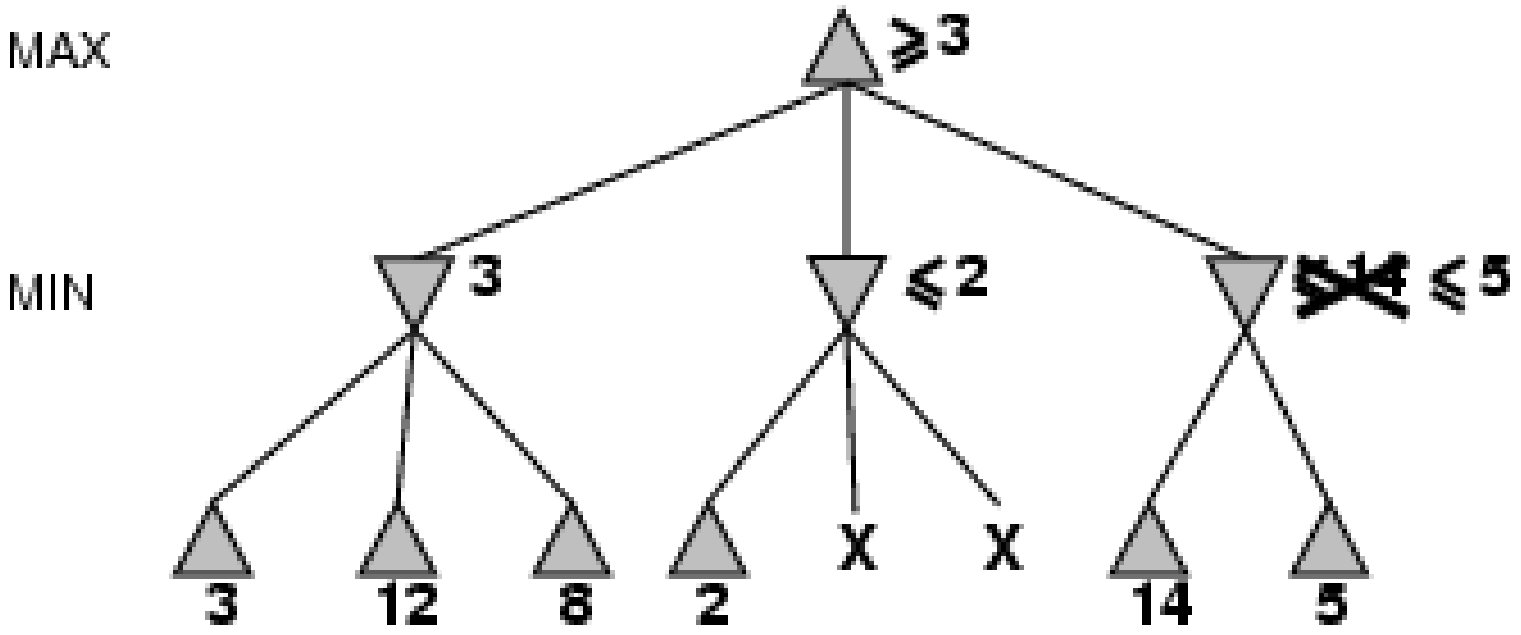
α-β pruning example



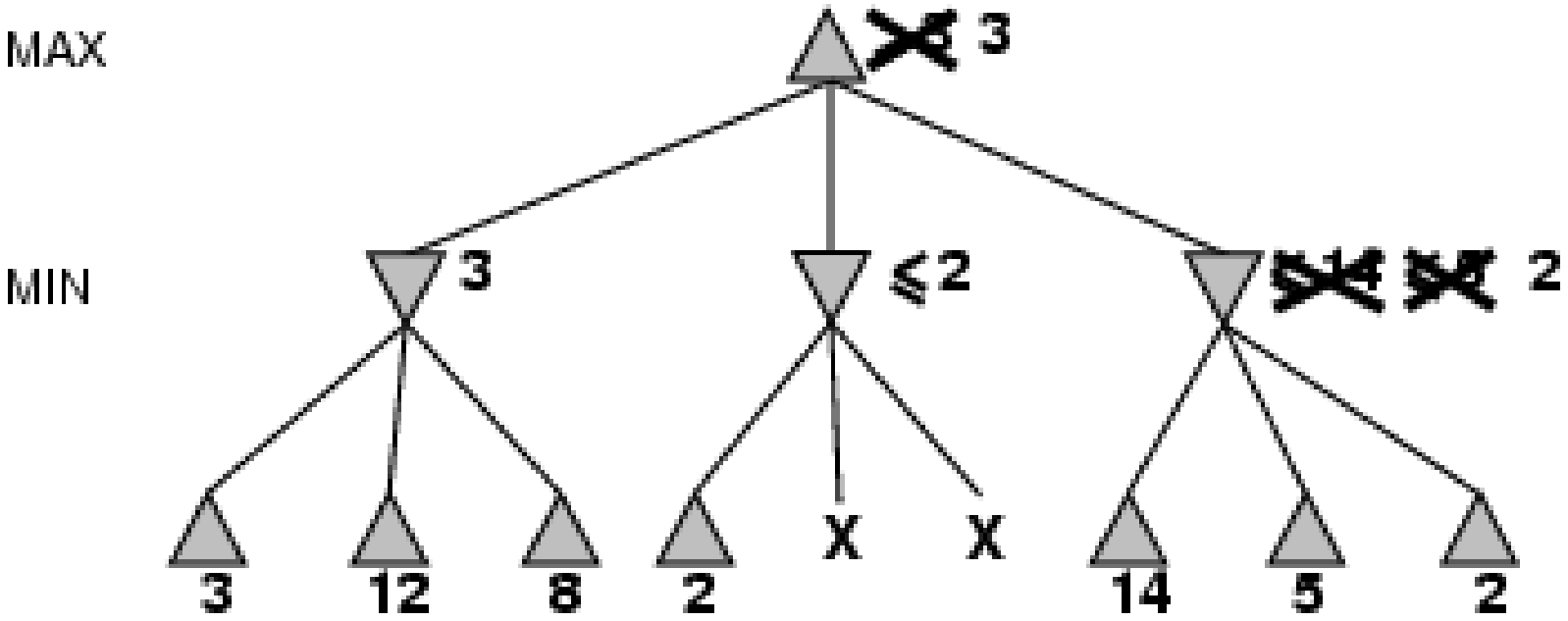
α-β pruning example



α-β pruning example

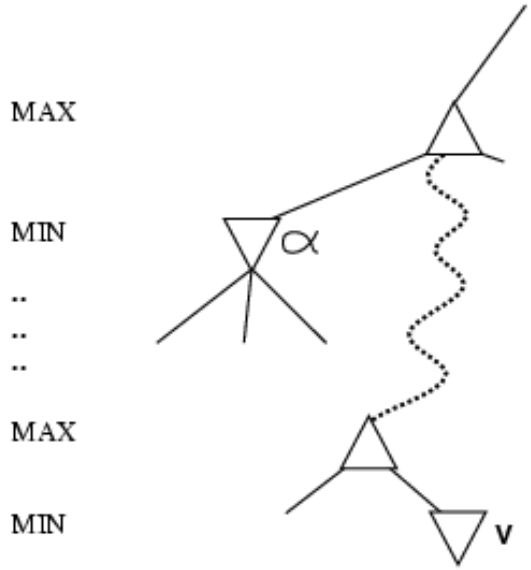


α-β pruning example



Alpha-Beta Pruning

- α = o maior valor já encontrado no caminho (melhor alternativa p/ MAX)
- β = o menor valor já encontrado no caminho (melhor alternativa p/ MIN)
- Se v é pior que alpha, MAX vai evitá-lo



Alpha-Beta Pruning

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

Alpha-Beta Pruning

```
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
```

Alpha-Beta Pruning

- A eficiência do algoritmo depende da ordem em que os nodos são examinados

- Na média, considerando que uma ordenação boa pode ser feita, o número de nodos a ser examinado é $\mathbf{O(b^{m/2})}$

Estados Repetidos

- Uma causa do crescimento exponencial de estados e a ocorrência de estados repetidos
  - <a1,b1,a2,b2> = <a1,b2,a2,b1>
- Uma forma de evitar isso é construir uma "Tabela de Transposição", que guarda a utilidade de estados já computados
- *Tradeoff*: Espaço x Tempo
  - Tentar guardar apenas os mais significativos

Exercício

- Qual jogada deve ser feita pelo jogador max (a, b ou c)? Porque?
- Se fosse utilizada a poda alfa-beta esse resultado seria alterado? Quais nodos folha não precisariam ser examinados?