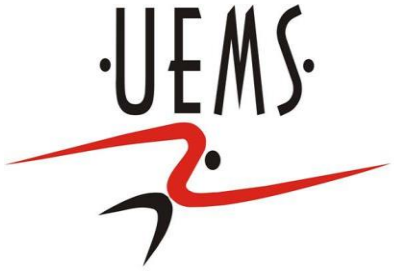


Busca Competitiva e Jogos

Profa. Mercedes Gonzales Márquez

Tópicos

- Jogos
- Decisões ótimas em jogos
- Algoritmo Minimax
- Poda ALFA-BETA
- Decisões imperfeitas em tempo real
- Jogos não determinísticos (estocásticos)

Até agora

- Problemas sem interação com outro agente.
- O agente possui total controle sobre suas ações e sobre o efeito de suas ações.
- Muitas vezes encontrar a solução ótima é factível.

Jogos

- Vamos nos concentrar em jogos como jogo da velha, xadrez, ...
- A natureza simplificada desses jogos é uma vantagem.
- O estado de um jogo é fácil de representar e os agentes geralmente são restritos a um pequeno número de ações, cujos efeitos são definidos por regras precisas.

Jogos

- Jogos físicos, como futebol e basquete, tem descrições mais complicadas, uma gama maior de ações possíveis e regras um tanto precisas que definem a legalidade das ações.

Teoria de Jogos

- Modela-se explicitamente os agentes adversários com as técnicas de busca competitiva da árvore de jogo.
- Começamos com uma classe restrita de jogos e definimos a jogada ótima e um **algoritmo para encontrá-la: busca do valor minimax**.
- Mostramos que a **poda** torna a busca mais eficiente, ignorando partes da árvores de busca que não fazem diferença para a jogada ótima.
- Para jogos não triviais, normalmente teremos que encerrar a busca em algum ponto.

Dois jogadores

- Em IA, geralmente, jogos são problemas de busca competitiva em ambientes determinísticos e completamente observáveis, em que:
  - existem dois agentes cujas ações se alternam;
  - os valores de utilidade final são sempre simétricos.
- Normalmente usamos o termo **jogada** como sinónimo de ação e **posição** como sinónimo de "estado".

Dois jogadores

- Consideraremos jogos com dois jogadores:
  - MAX e MIN
  - MAX faz o primeiro movimento e depois eles se revezam até o jogo terminar.

Dois jogadores

- O jogo pode ser definido como um problema de busca com os seguintes elementos:

Dois jogadores

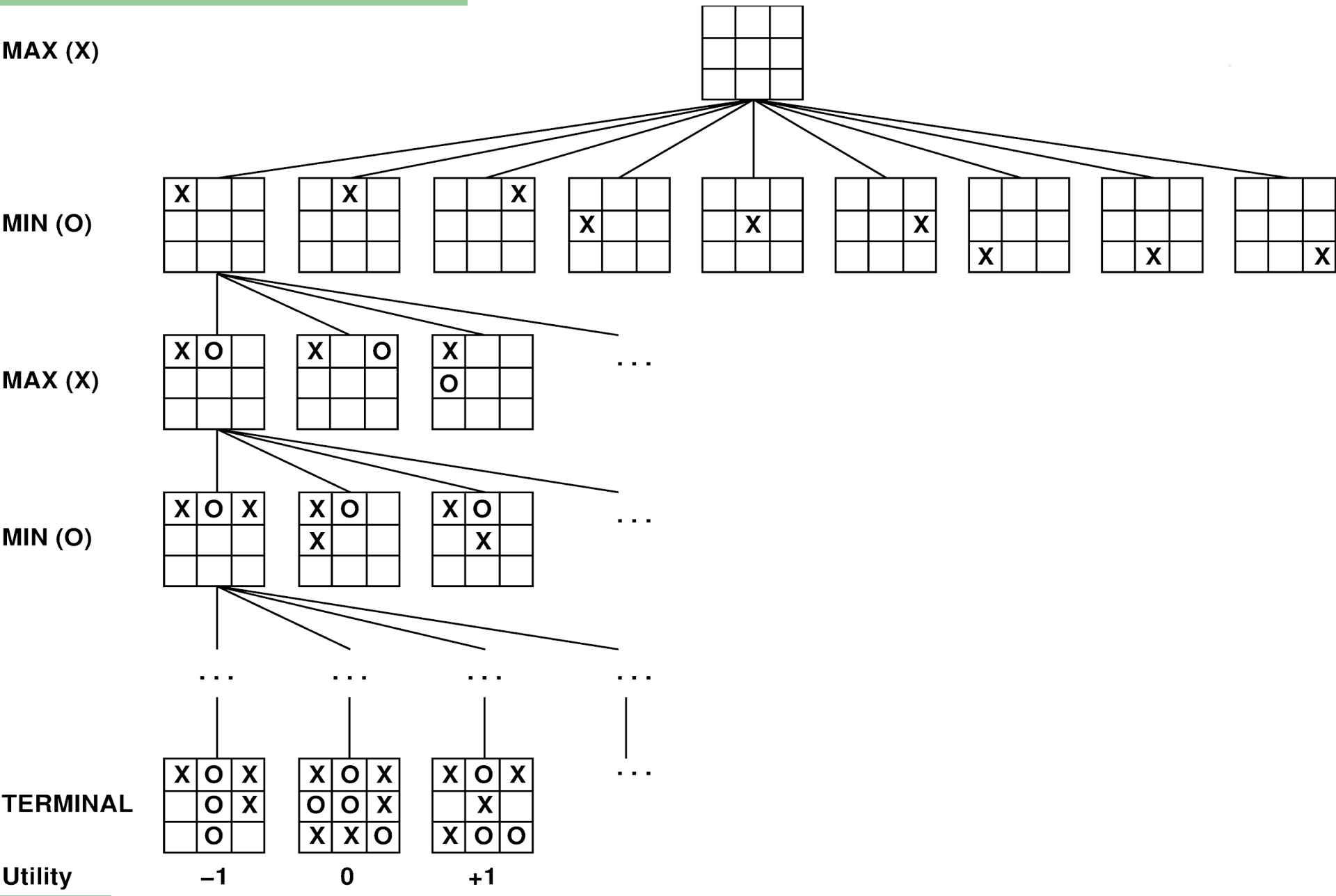
- **S0:** estado inicial
- **Jogador (s):** jogador que se move no estado s
- **Ações (s):** conjunto de movimentos válidos no estado s.
- **Resultado (s,a):** modelo de transição, que define o resultado de realizar a ação a no estado s.
- **teste de término (s),** que é verdadeiro quando o jogo termina.
- **função utilidade:** valor numérico para os estados terminais.

Dois jogadores

- O estado inicial, a função Ações e a função Resultado definem o grafo do espaço de estados, um grafo em que os vértices são estados, as arestas são jogadas e um estado pode ser alcançado por vários caminhos.
- Definimos a árvore de jogo completa como uma árvore de busca que segue cada sequência de jogadas até alcançar um estado terminal.

Dois jogadores

- Na figura mostra-se parte da árvore de jogo para o jogo da velha. A partir do estado inicial MAX tem 9 movimentos possíveis.
- O jogo se alterna entre a colocação de um X por MAX e a colocação de um O por MIN até alcançar nós de folhas correspondentes a estados terminais.
- O número em cada nó folha indica o valor de utilidade do estado terminal , do ponto de vista de MAX; valores altos são bons para MAX e ruins para MIN.


MAX (X)
MIN (O)
MAX (X)
MIN (O)
TERMINAL
Utility
–1
0
+1

Dois jogadores

- Para o jogo da velha a árvore é relativamente pequena – menos de 9! = 362.880 nós terminais. Analisemos agora para o xadrez.
- Xadrez: problema de busca mais complexo que o jogo da velha:

– fator de ramificação médio ≈ 35
– profundidade da solução ≈ 100
– árvore de busca ≈ $35^{100} \approx 10^{154}$

- É melhor pensar na árvore de jogo como sendo uma construção teórica que não podemos perceber no mundo físico.

Estratégia ótima

- MAX deve:

– maximizar sua chance de vitória

– considerar que MIN minimizará sua chance vitória.

- Questão:

– Como usar a árvore de jogo para encontrar uma estratégia ótima que, independentemente das escolhas feitas pelo adversário, garanta que MAX fará sempre o melhor possível?

Estratégia ótima: Minimax

- Uma estratégia ótima pode ser determinada usando-se o valor-minimax de cada nó:

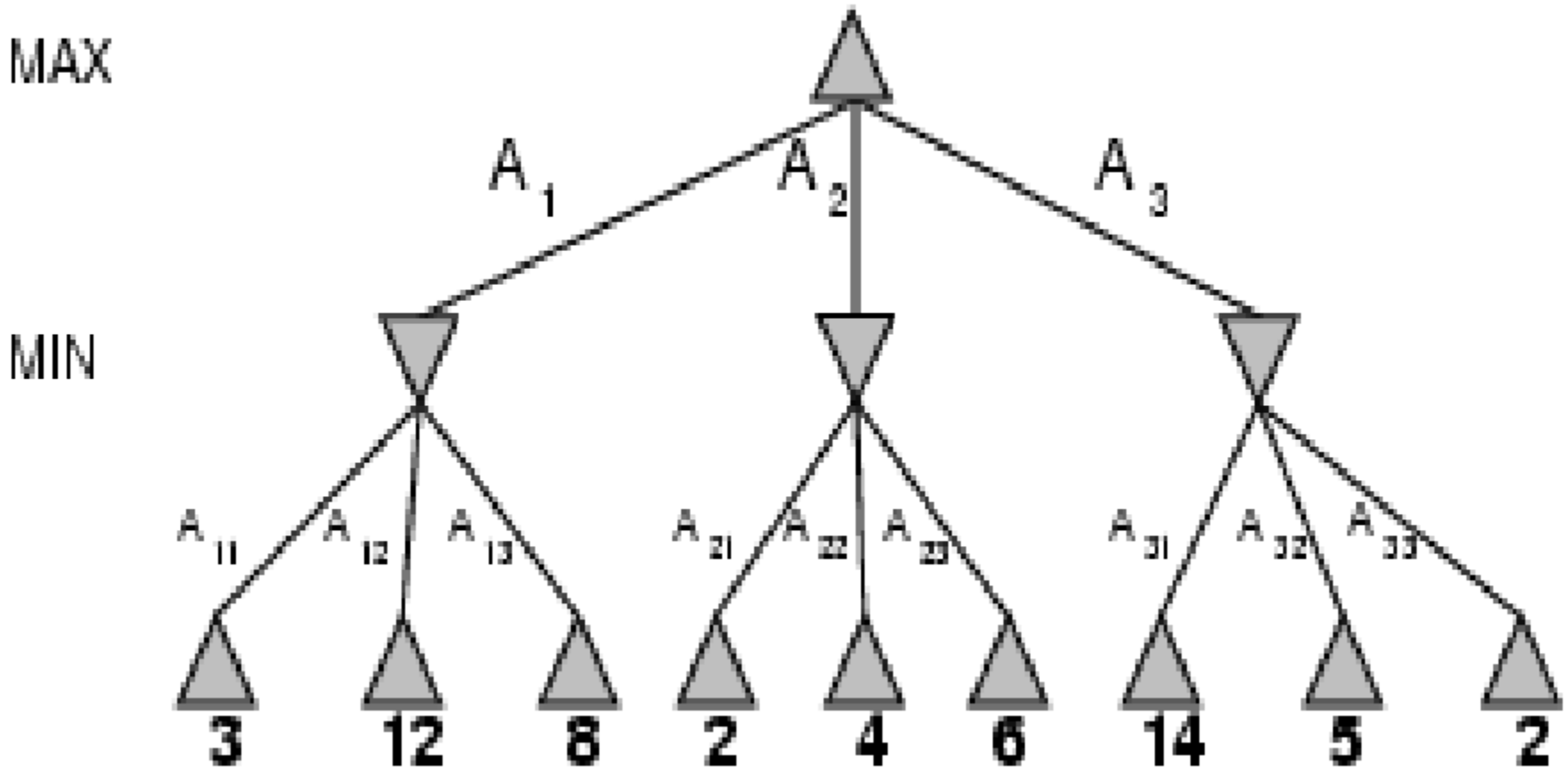
Minimax(s) =

- **UTILIDADE(s) se n é terminal**
- **$\max_{a \in Ações(s)}$ Minimax(Resultado(s,a)) se Jogador(s)=MAX**
- **$\min_{a \in Ações(s)}$ Minimax(Resultado(s,a)) se Jogador(s)=MIN**

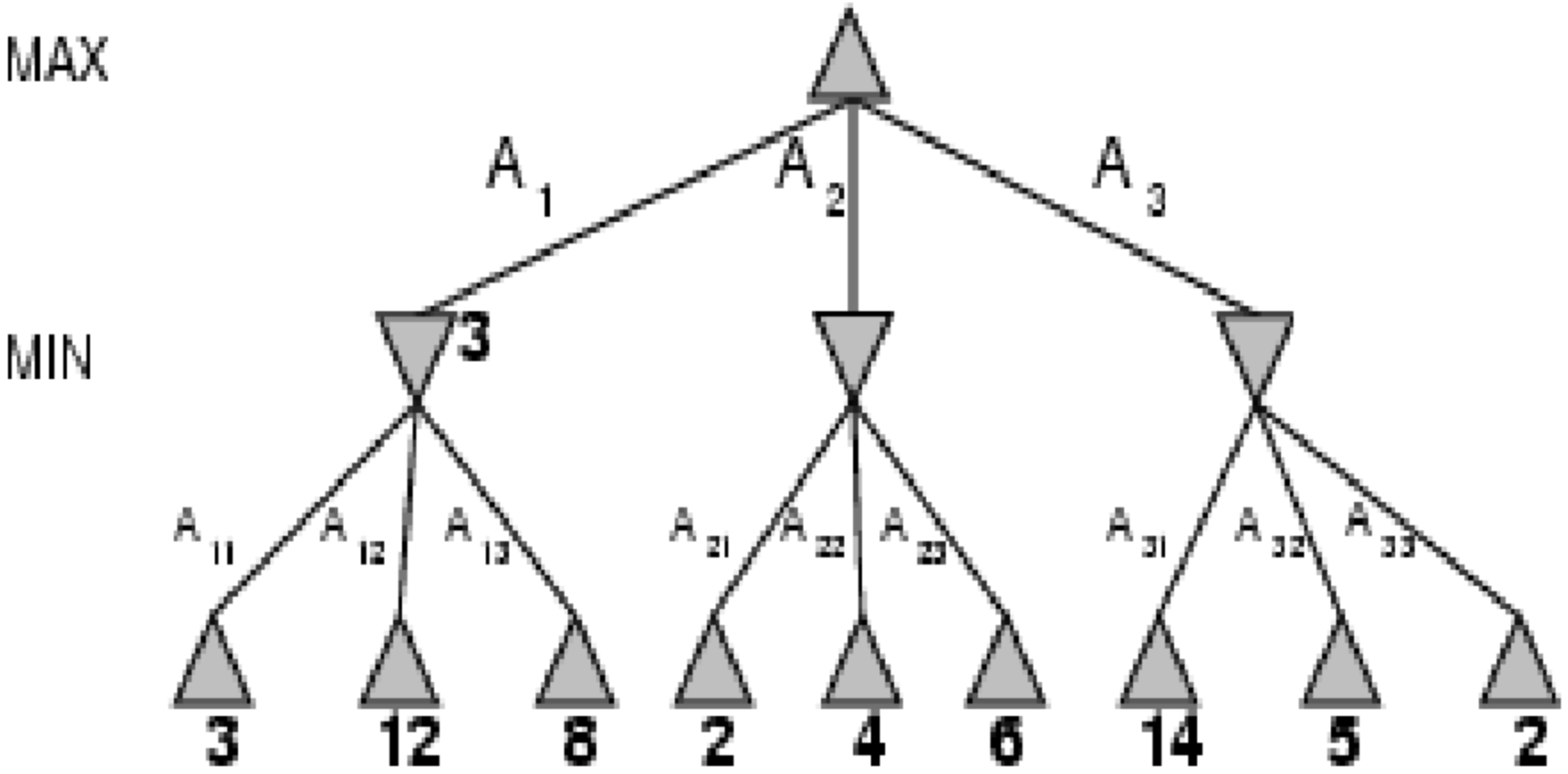
Estratégia ótima

- gerar a árvore de busca completa, a partir de s
- aplicar a função utilidade a cada estado terminal
- propagar os valores de utilidade, a partir das folhas, até a raiz s (MIN escolhe mínimo e MAX escolhe máximo)

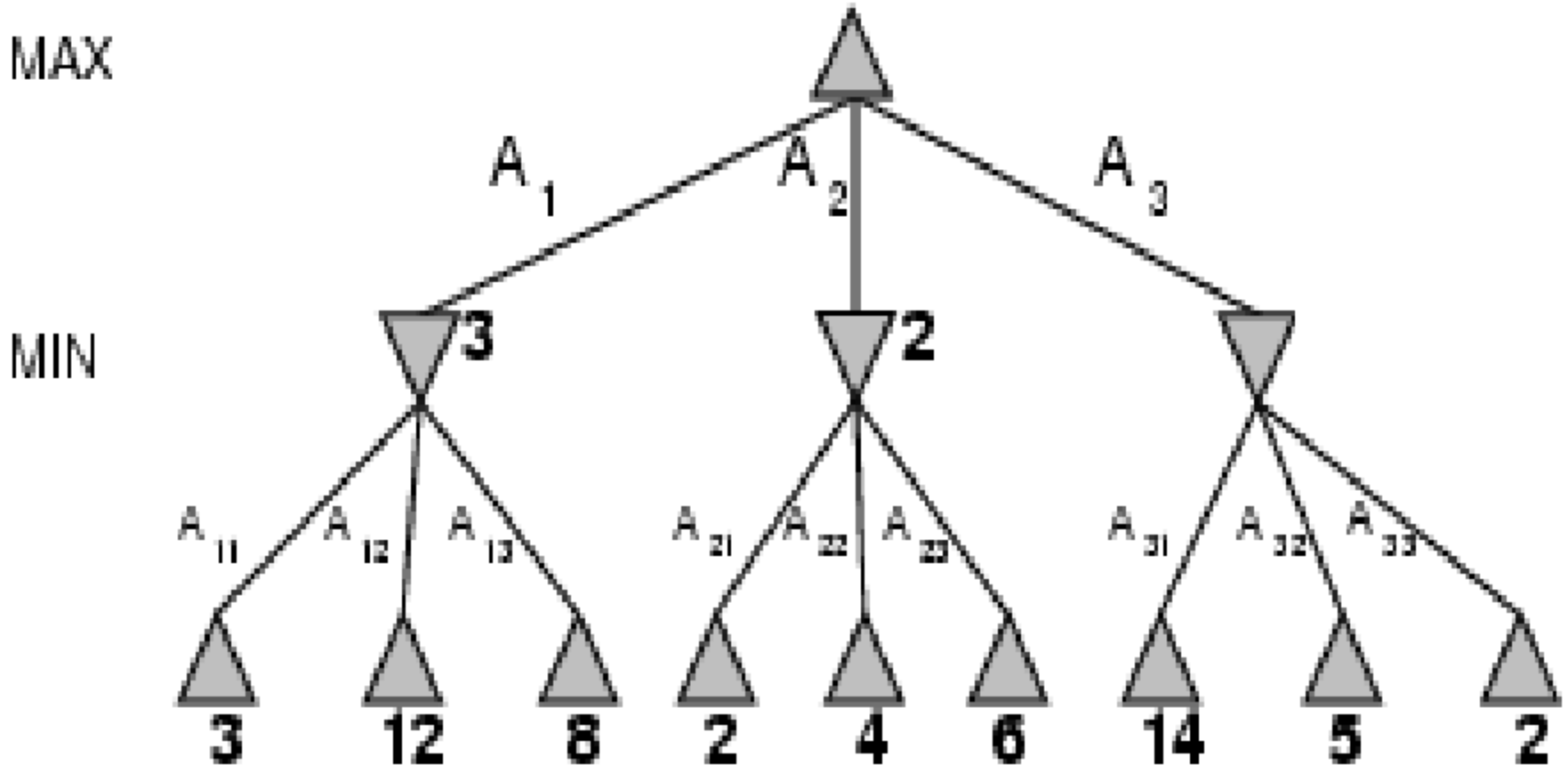
Exemplo: Um jogo de duas jogadas



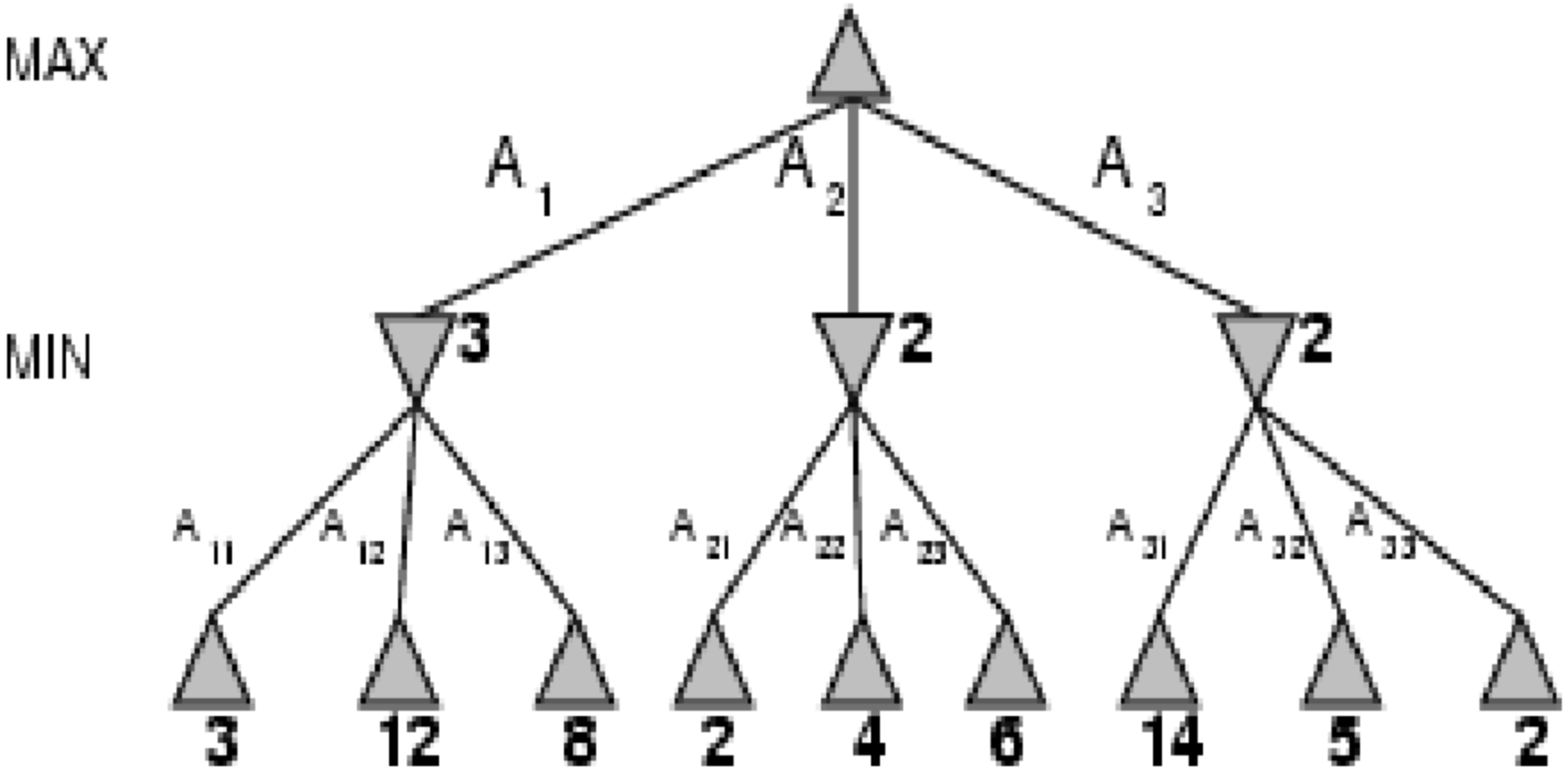
Exemplo: Um jogo de duas jogadas



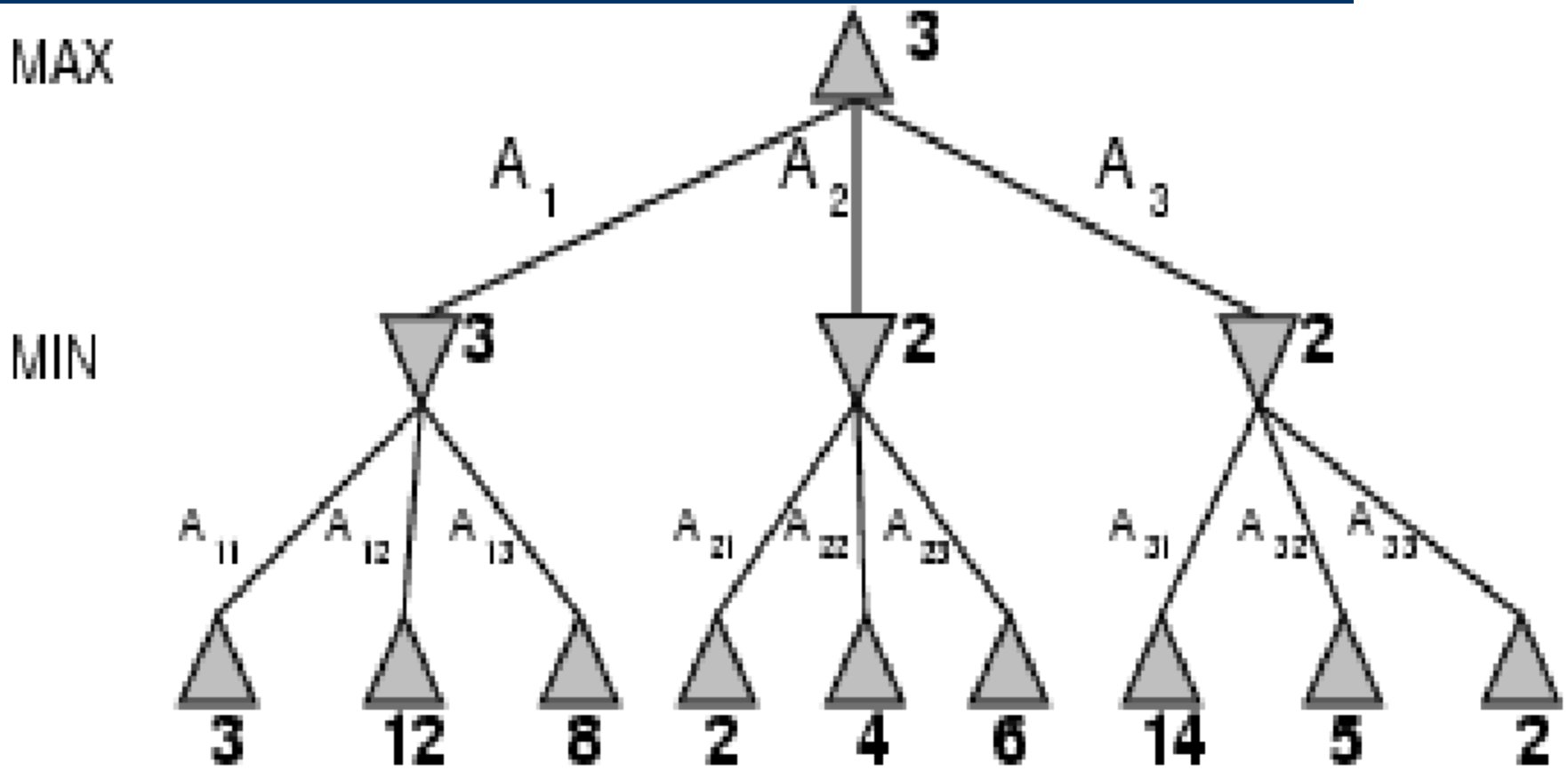
Exemplo: Um jogo de duas jogadas



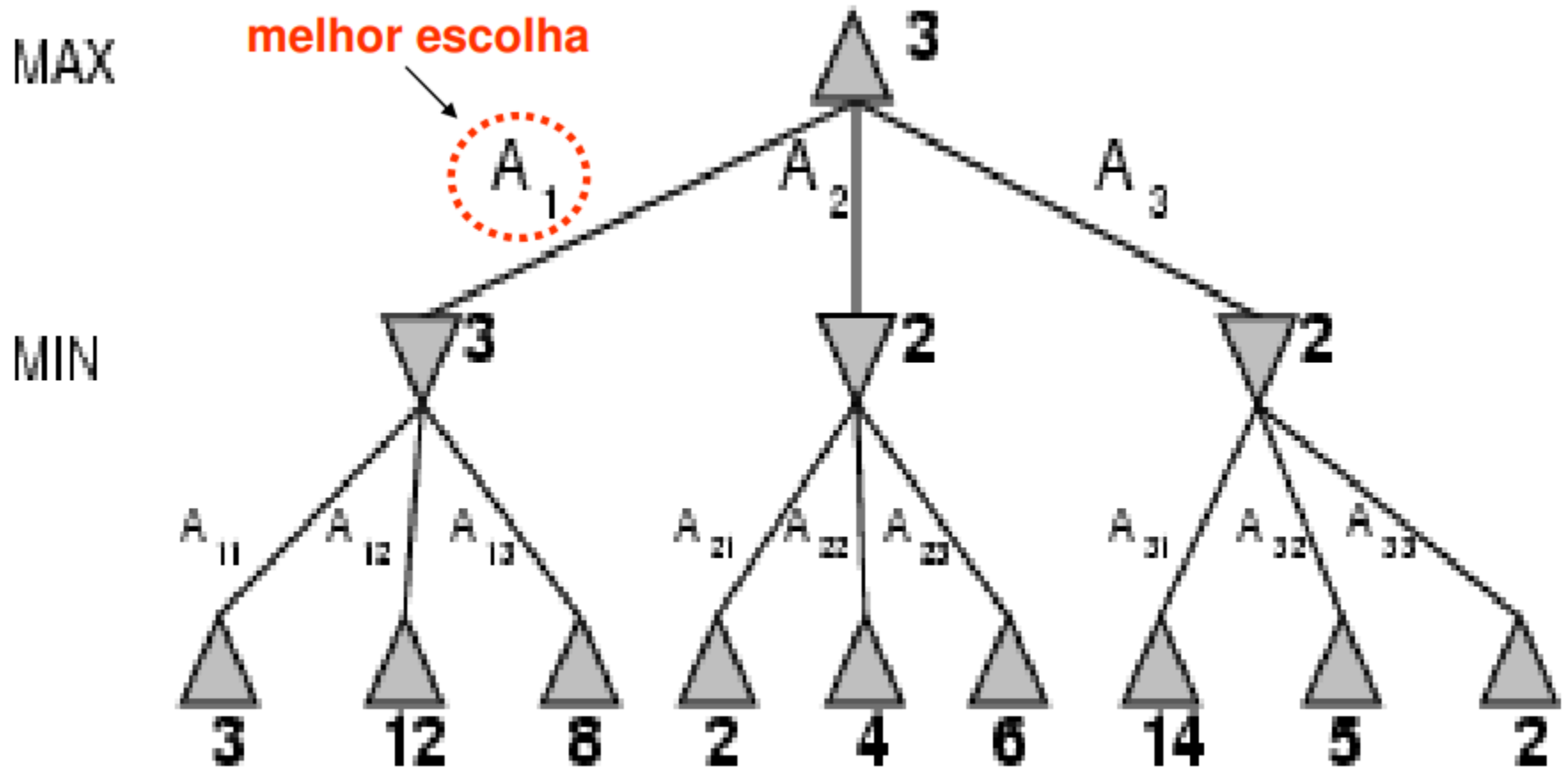
Exemplo: Um jogo de duas jogadas



Exemplo: Um jogo de duas jogadas



Exemplo: Um jogo de duas jogadas



Melhor escolha: maximiza o resultado para MAX no pior caso!!!

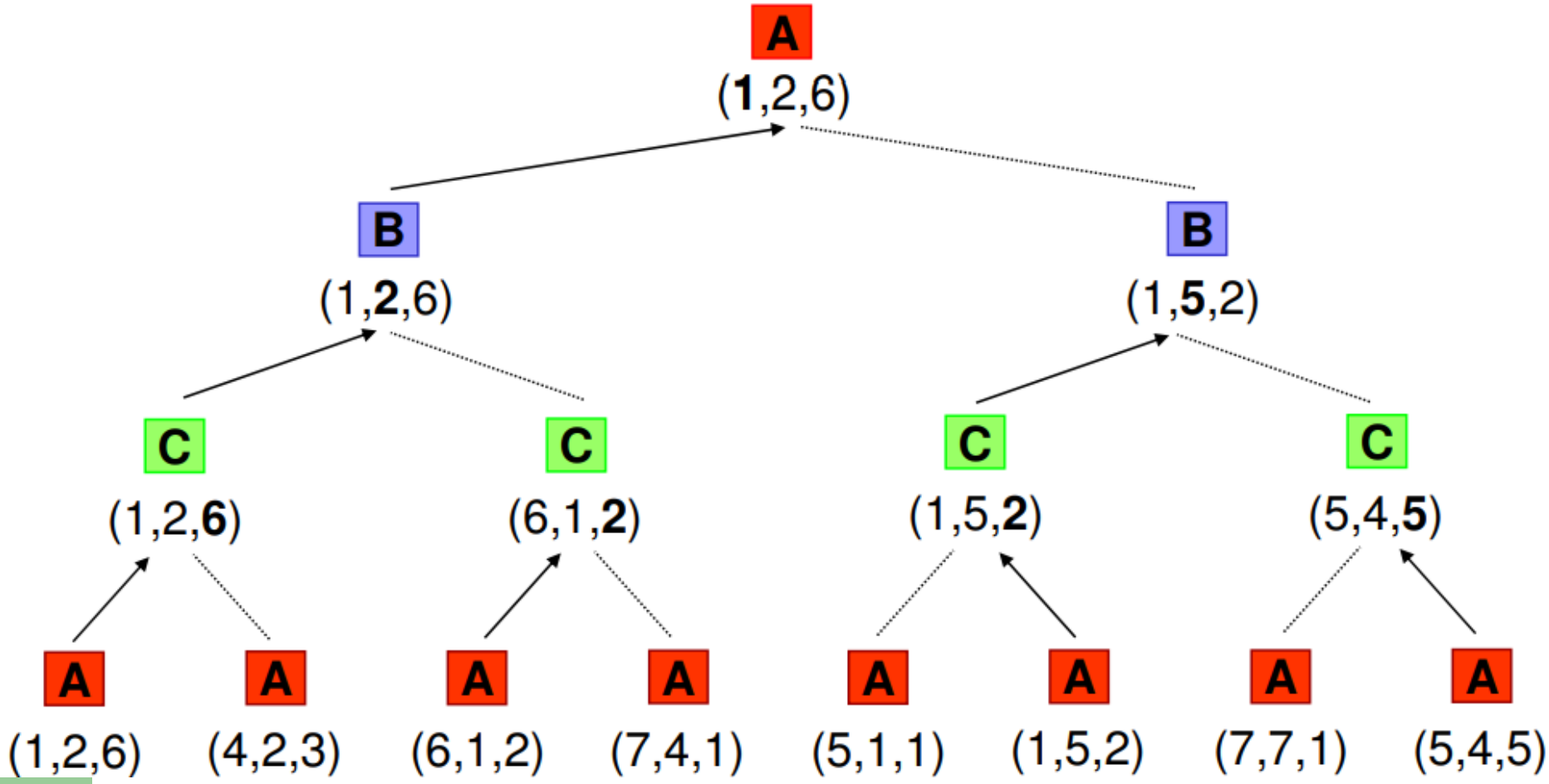
Propriedades do Minimax

- completo? sim, se a árvore for finita
- ótimo? sim, se o adversário for ótimo
- tempo? $O(b^m)$
- espaço? $O(b \times m)$ ou $O(m)$

Jogos com vários jogadores

- muitos jogos permitem mais de dois jogadores
- valor utilidade ⇒ vetor de valores utilidades)

Minimax com três jogadores



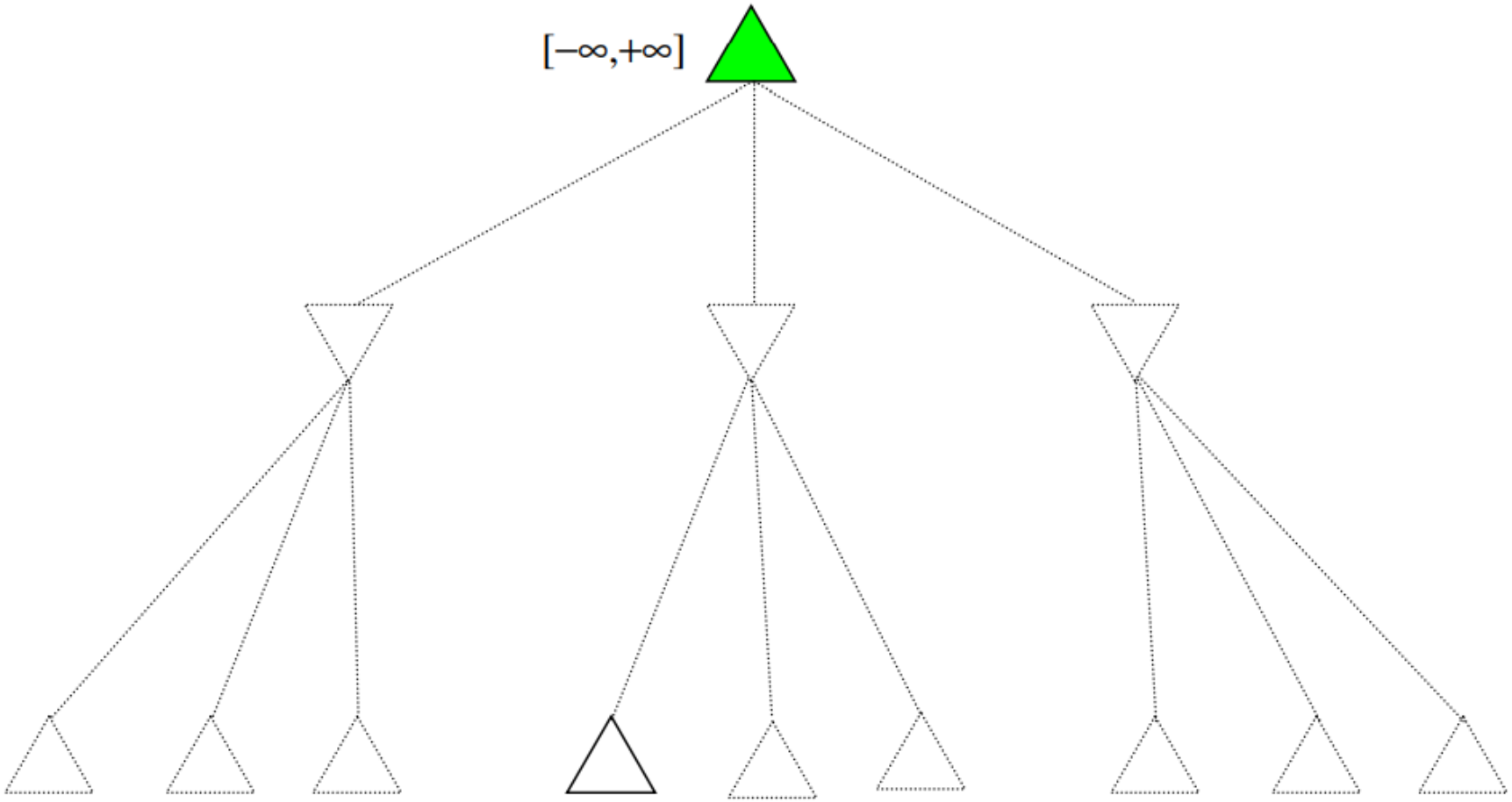
Poda α-β

- O número de estados é exponencial na profundidade da árvore.
- Nenhum algoritmo pode eliminar completamente o expoente, mas, ás vezes, podemos cortá-lo ou podar grandes partes da árvore, sem causar diferenças no resultado.

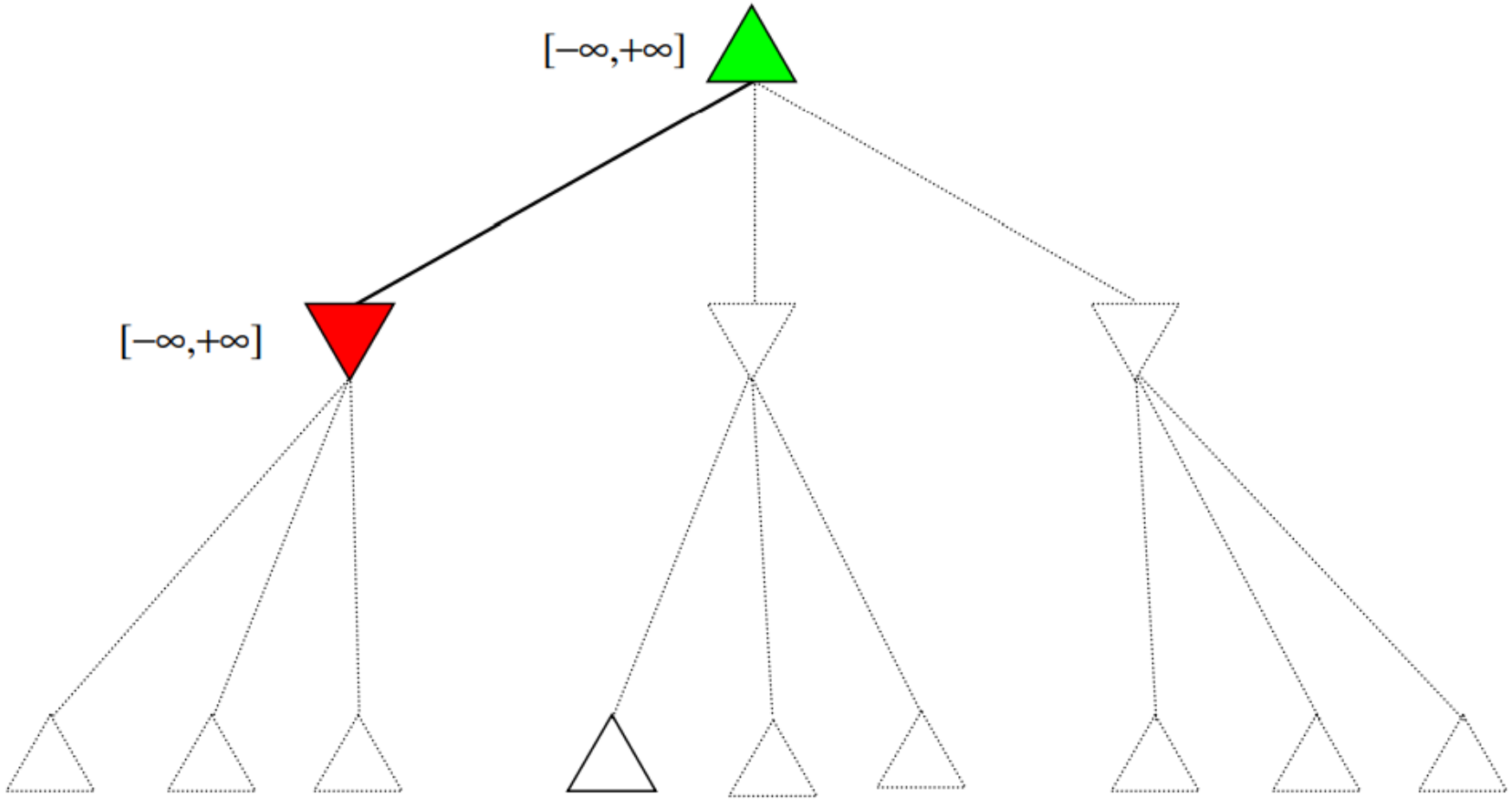
Poda α-β

- poda α-β devolve a mesma ação que MINIMAX, mas poda ramificações que não influenciam na escolha dessa ação
- poda α-β pode reduzir a complexidade da busca, dependendo da ordem em que os nós são examinados.
- Vejamos um exemplo

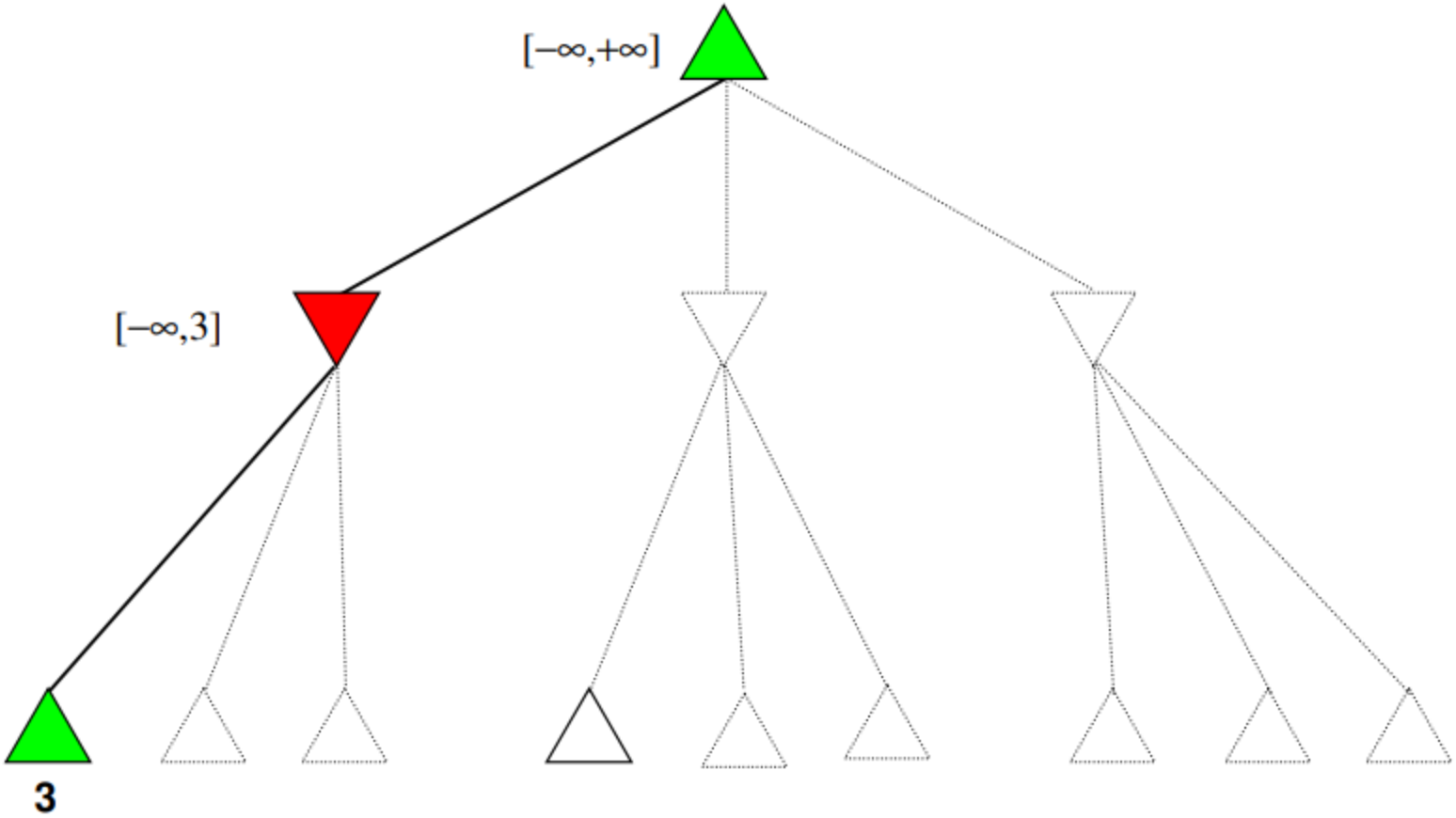
Poda α-β



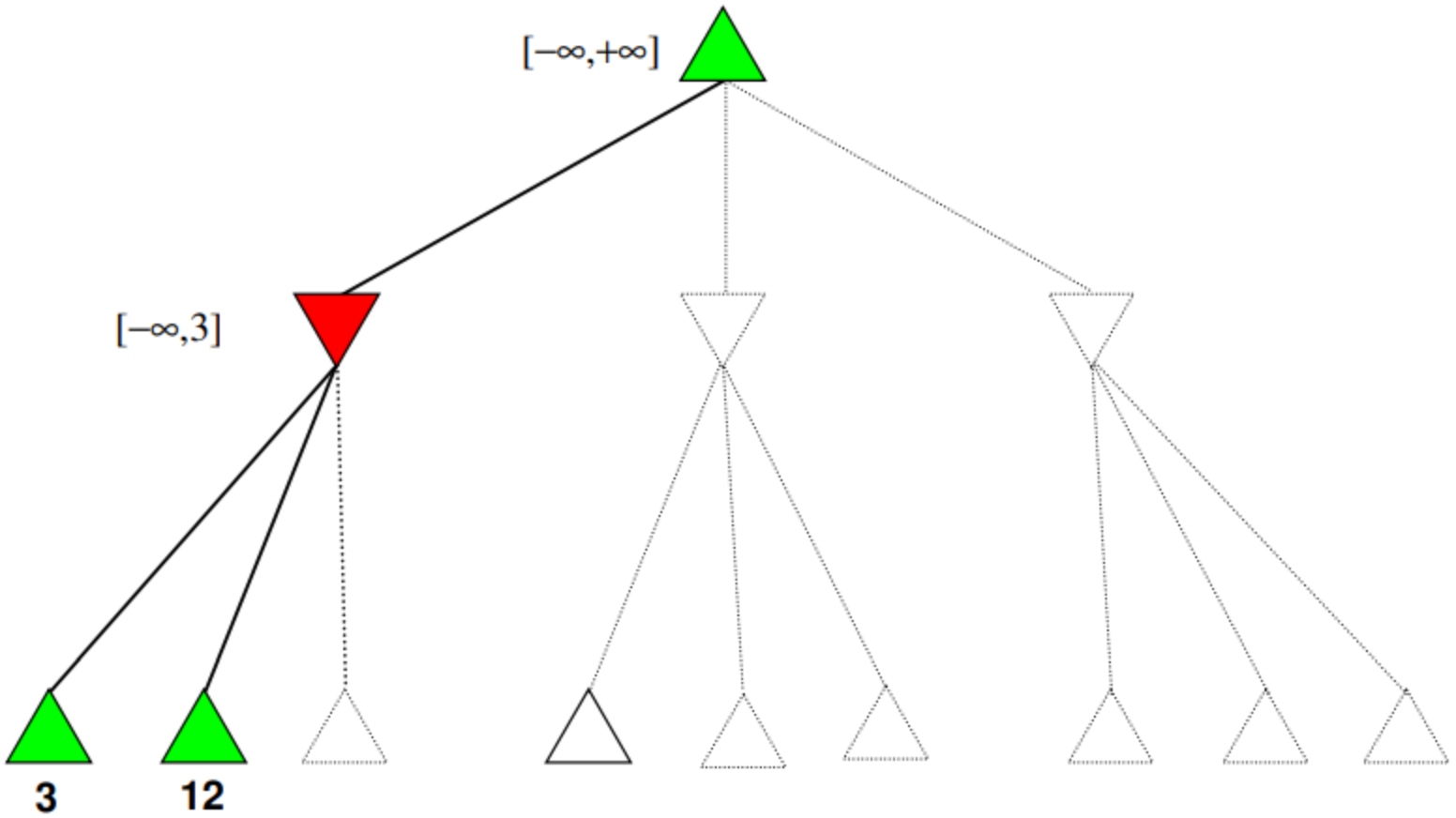
Poda α-β



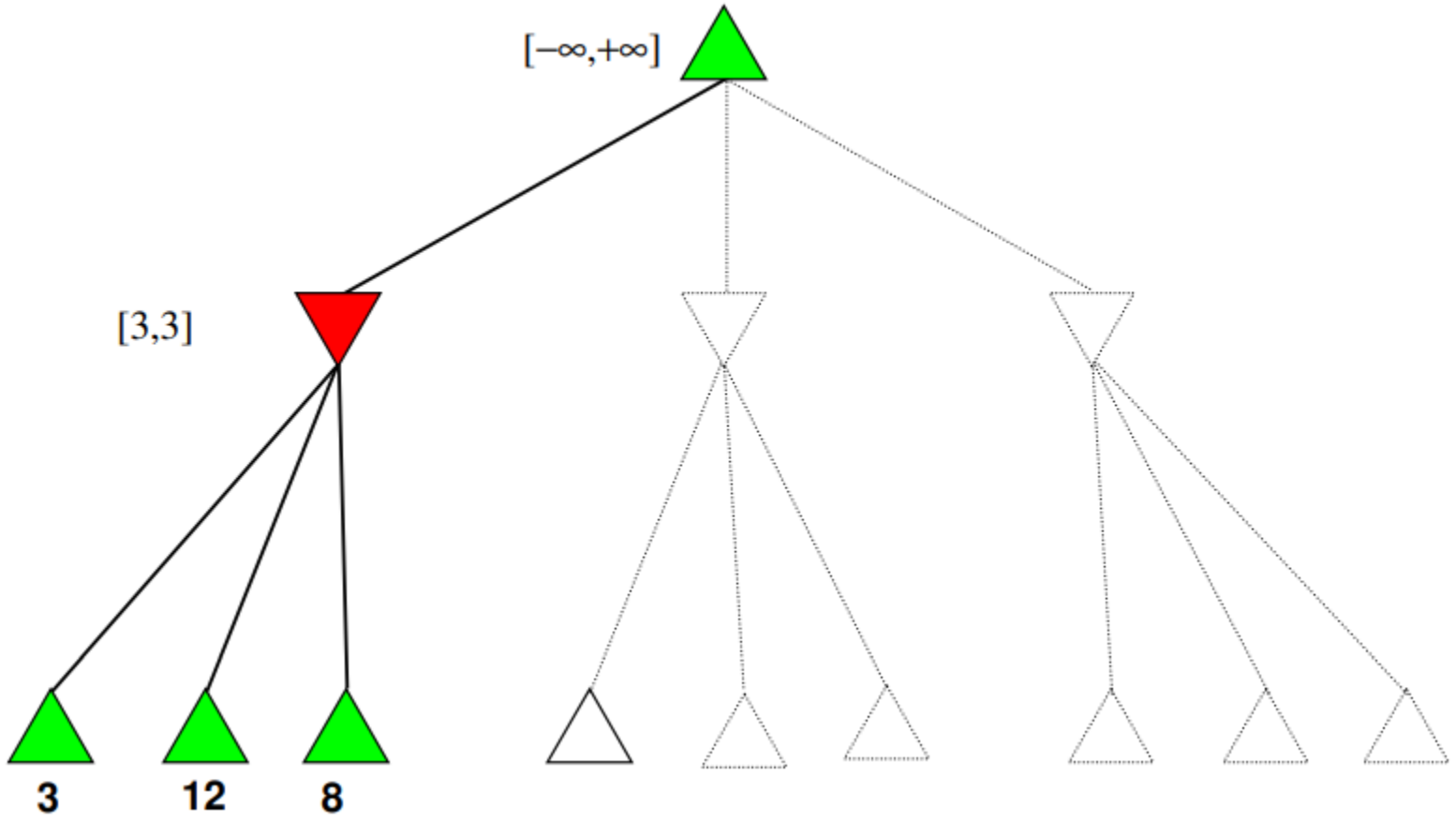
Poda α-β



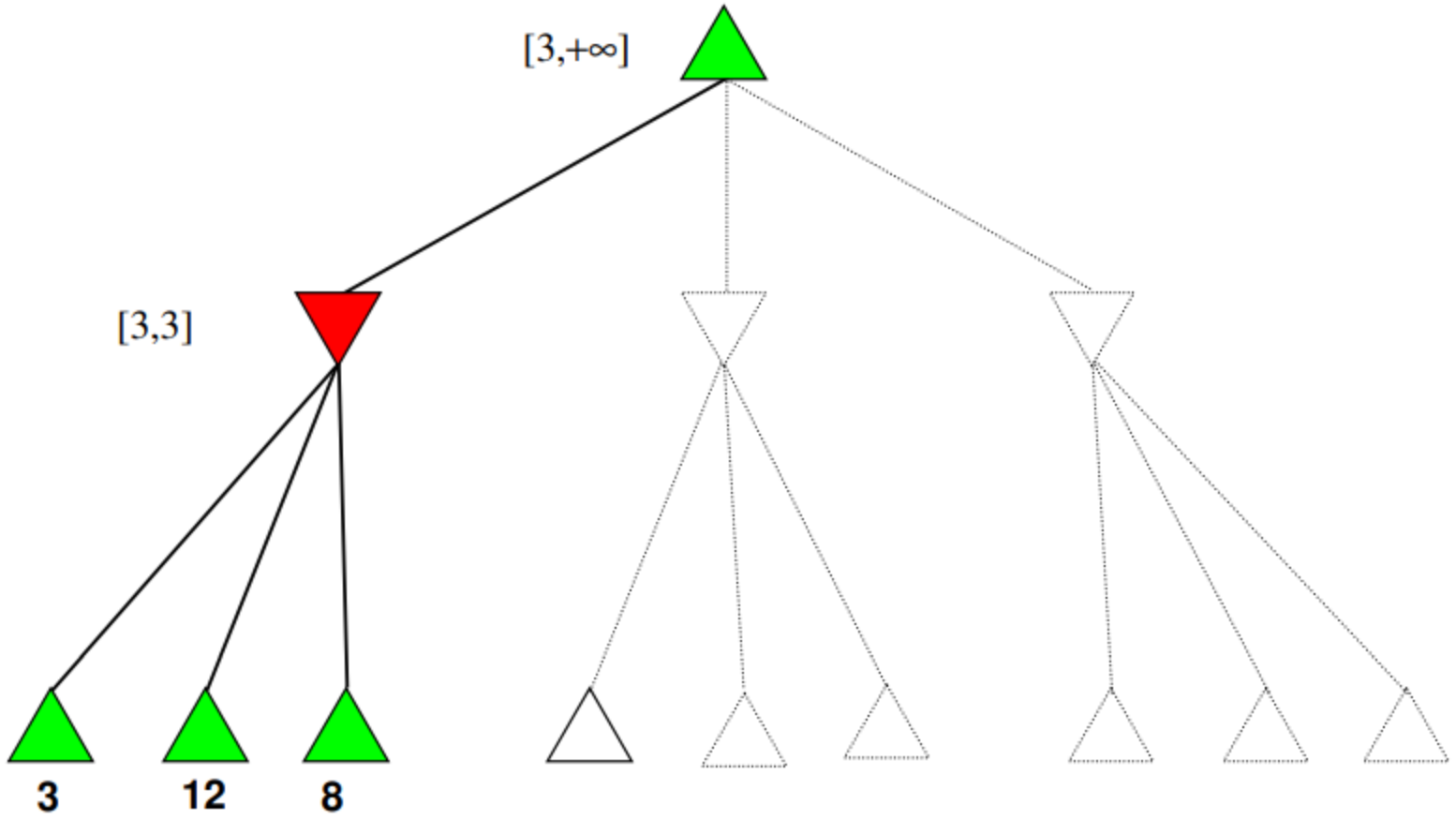
Poda α-β



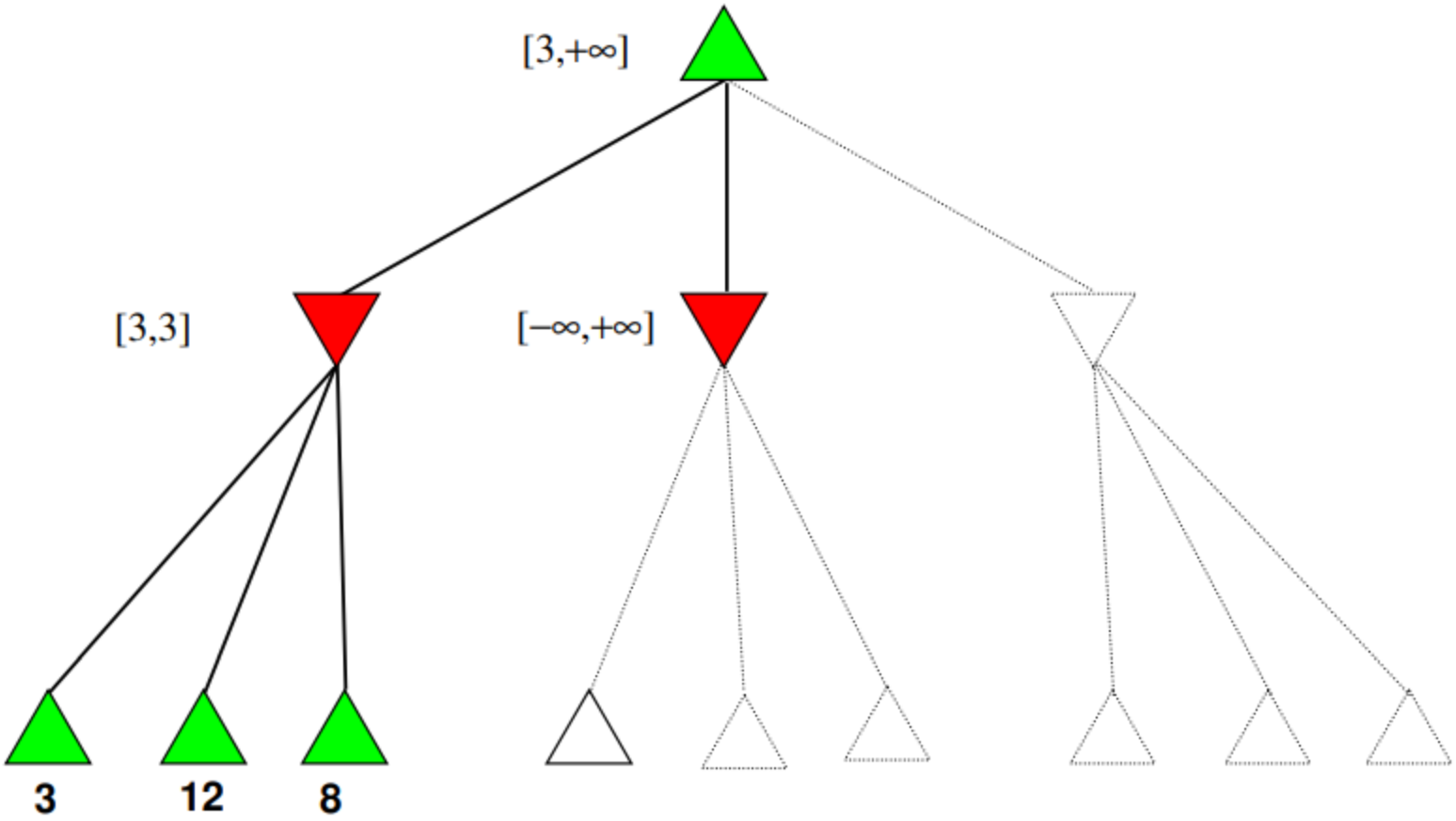
Poda α-β



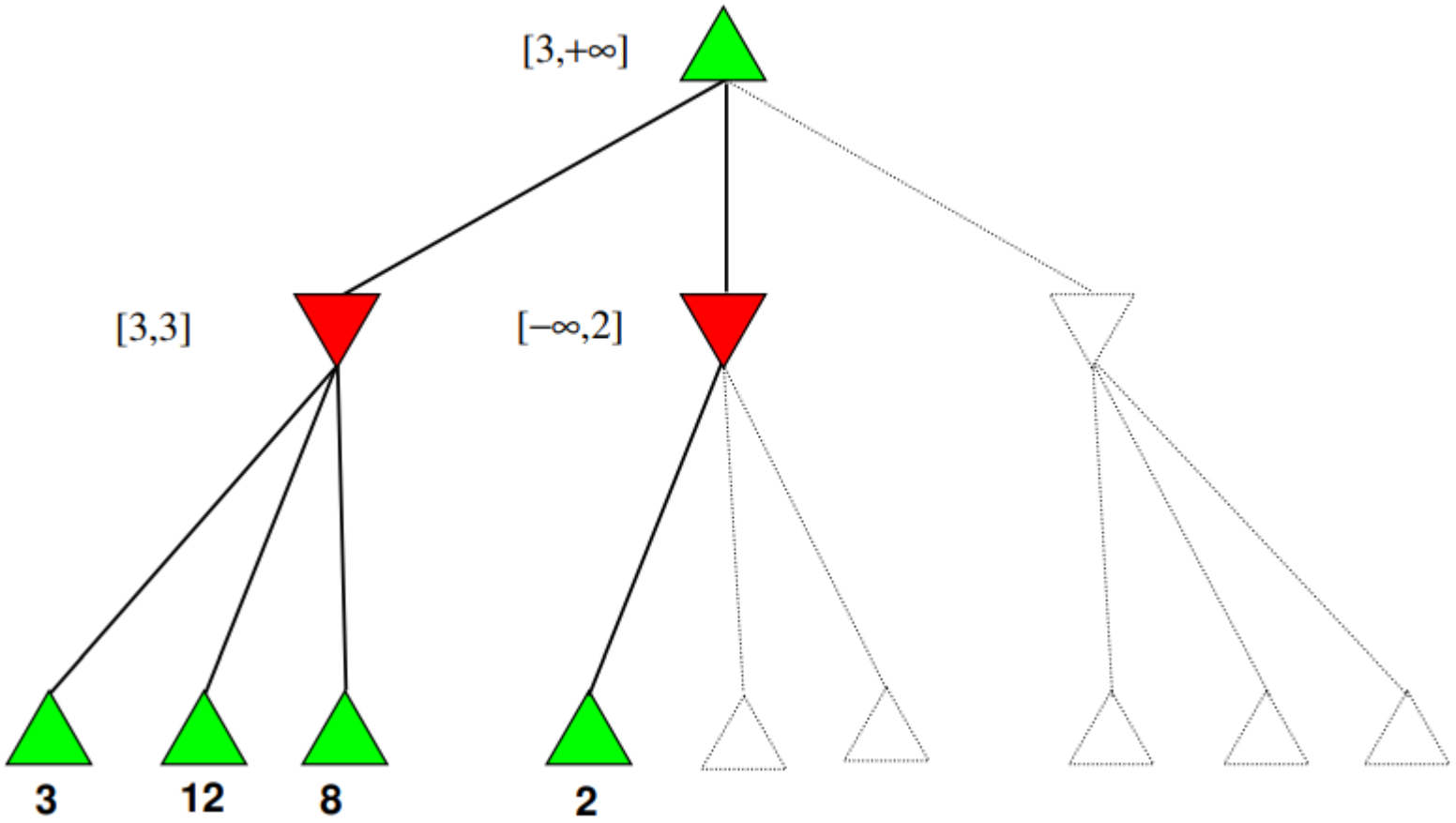
Poda α-β



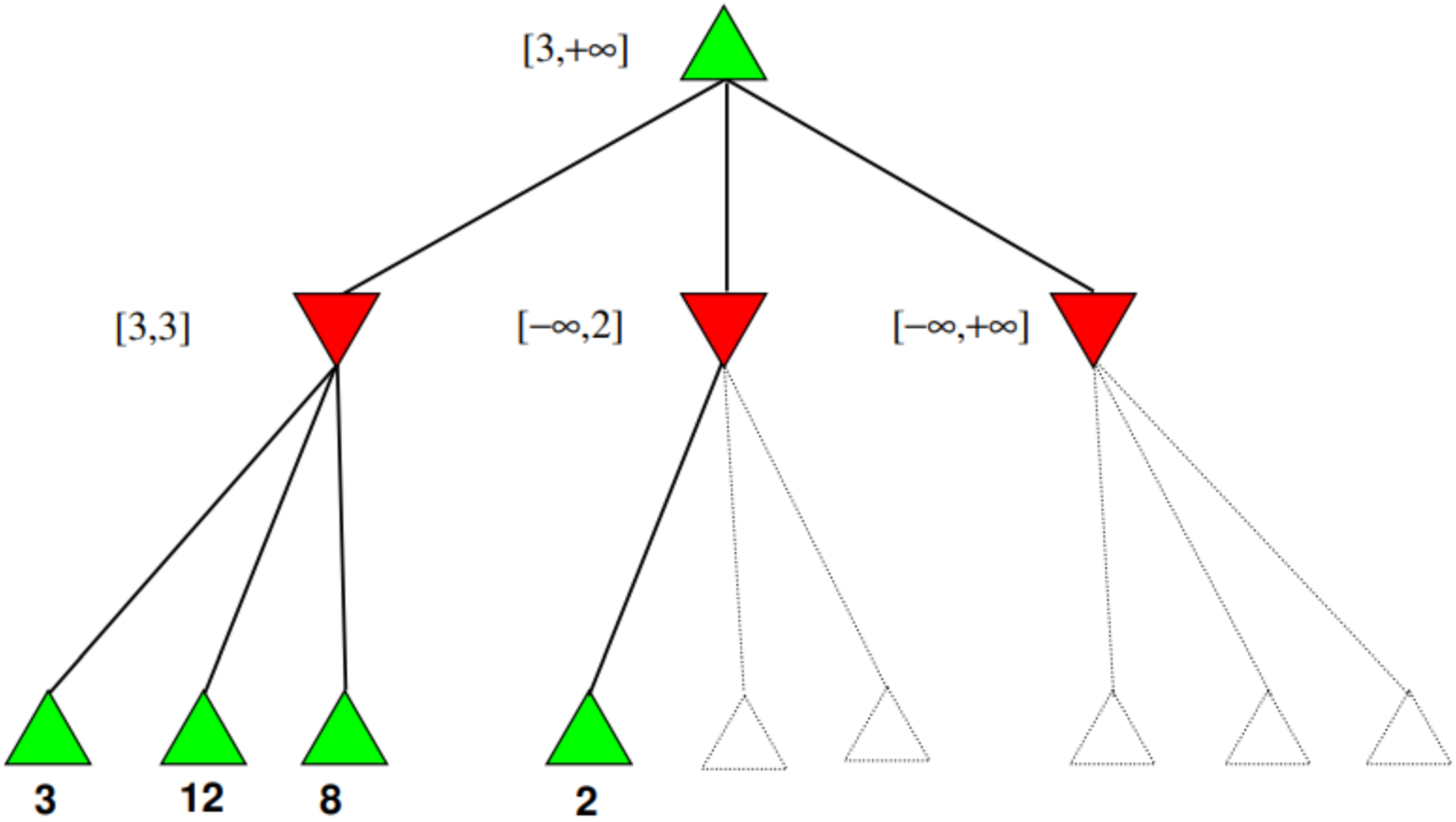
Poda α-β



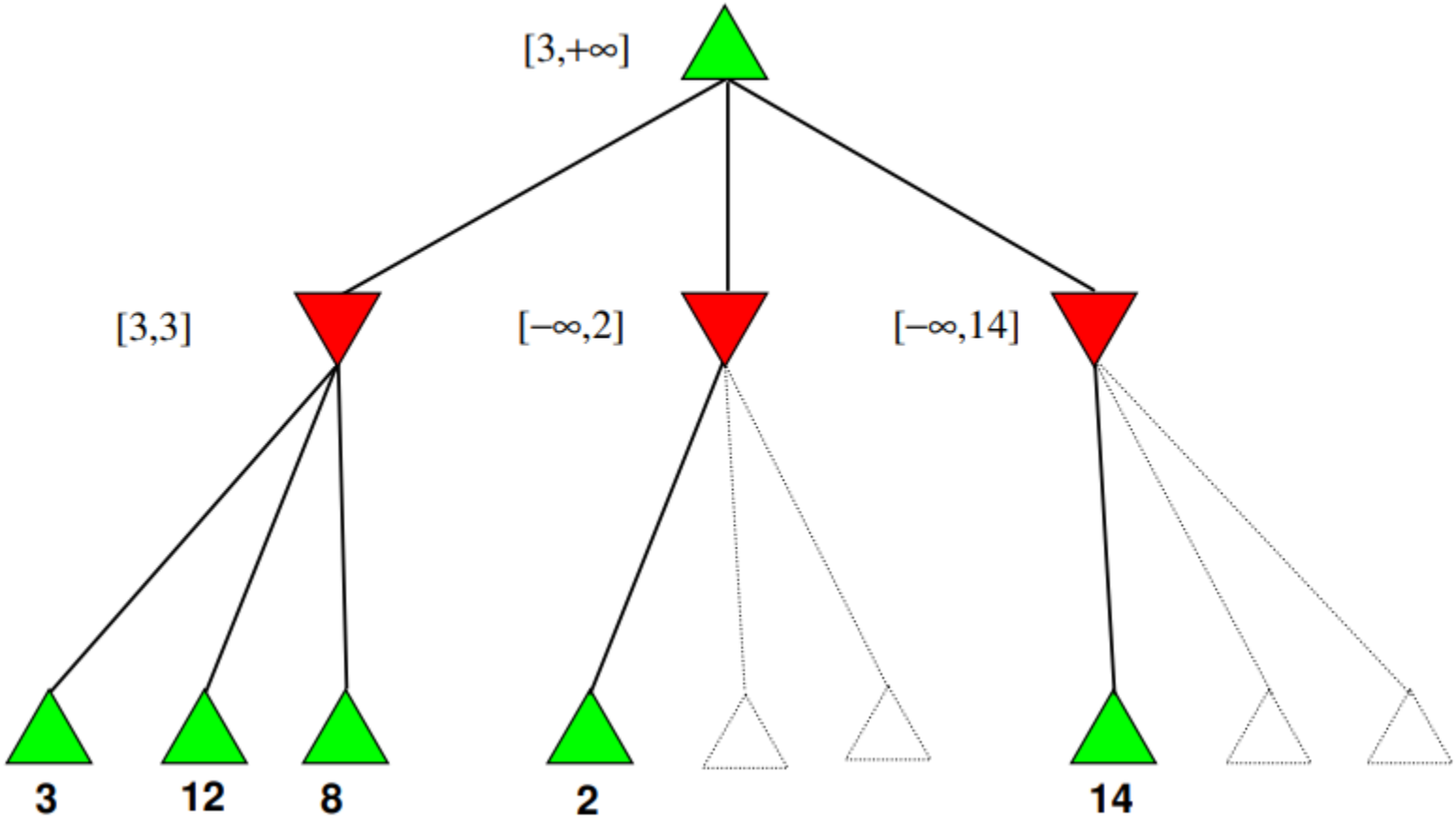
Poda α-β



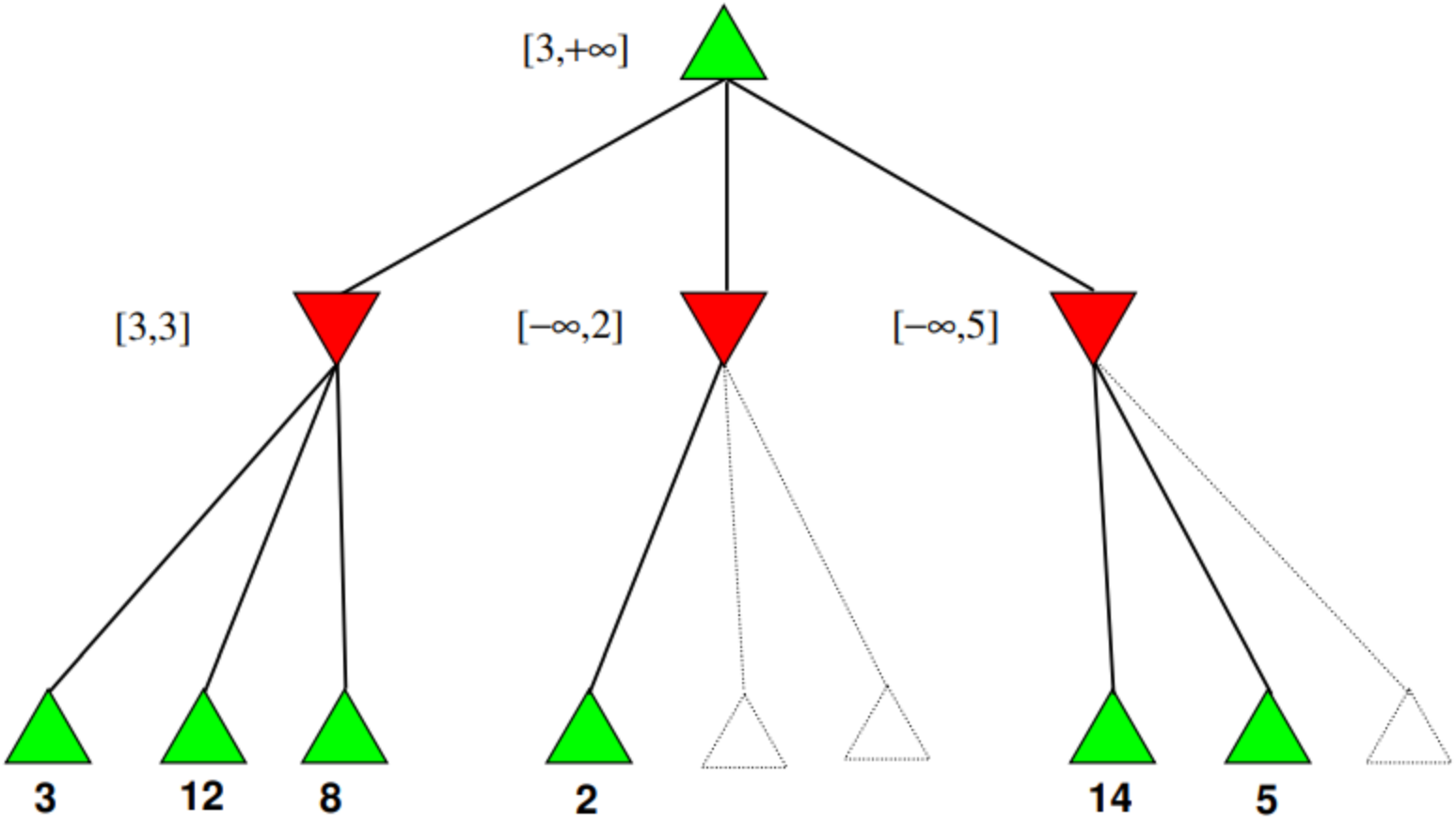
Poda α-β



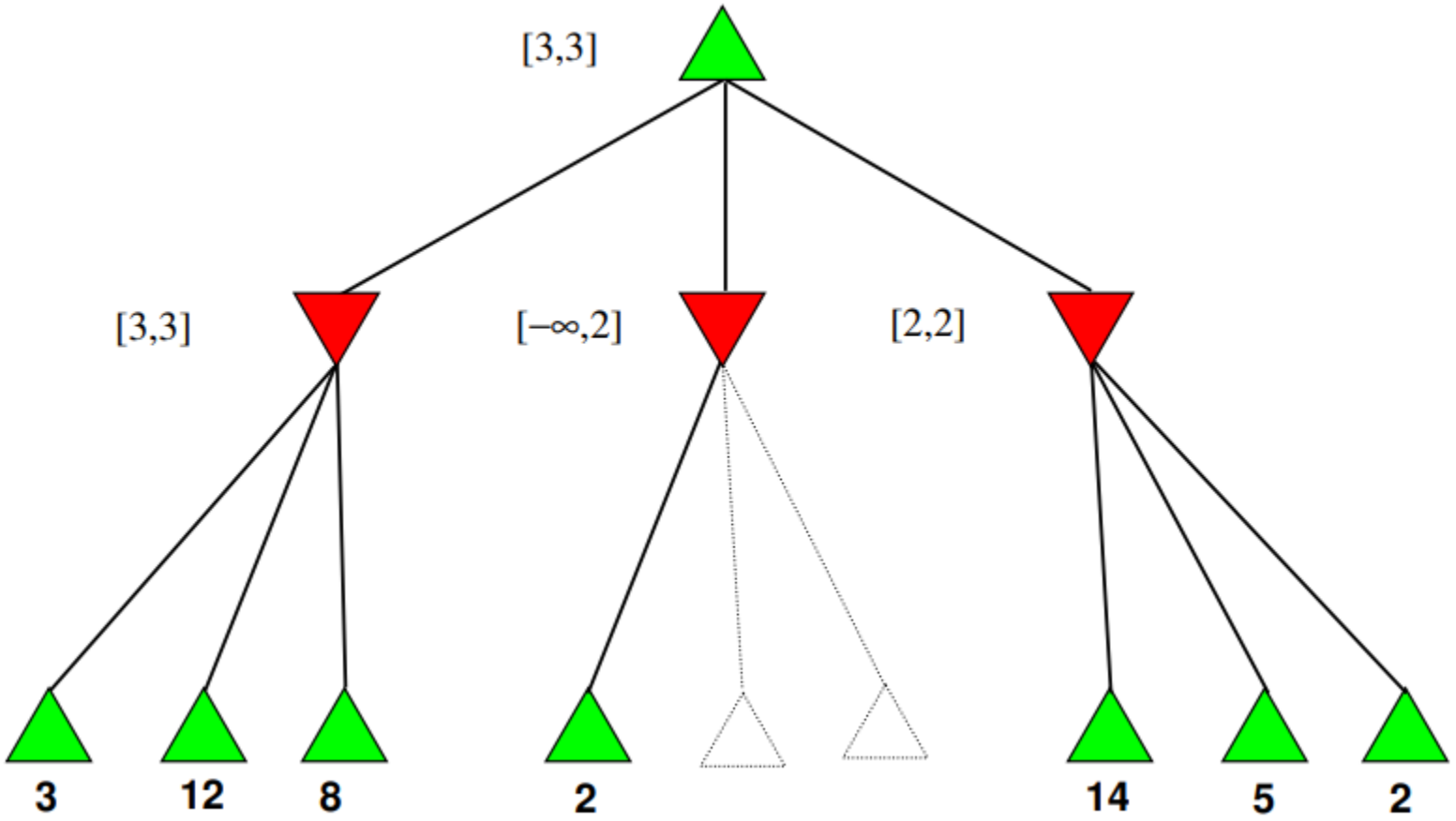
Poda α-β



Poda α-β



Poda α-β



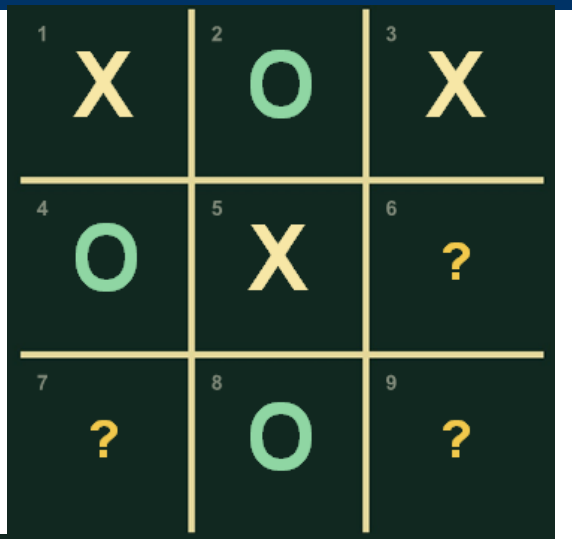
Poda α-β

- VALOR-MINIMAX(raiz)

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

Exemplo de uma sub-árvore do Jogo da Velha

É a vez de MAX: X joga

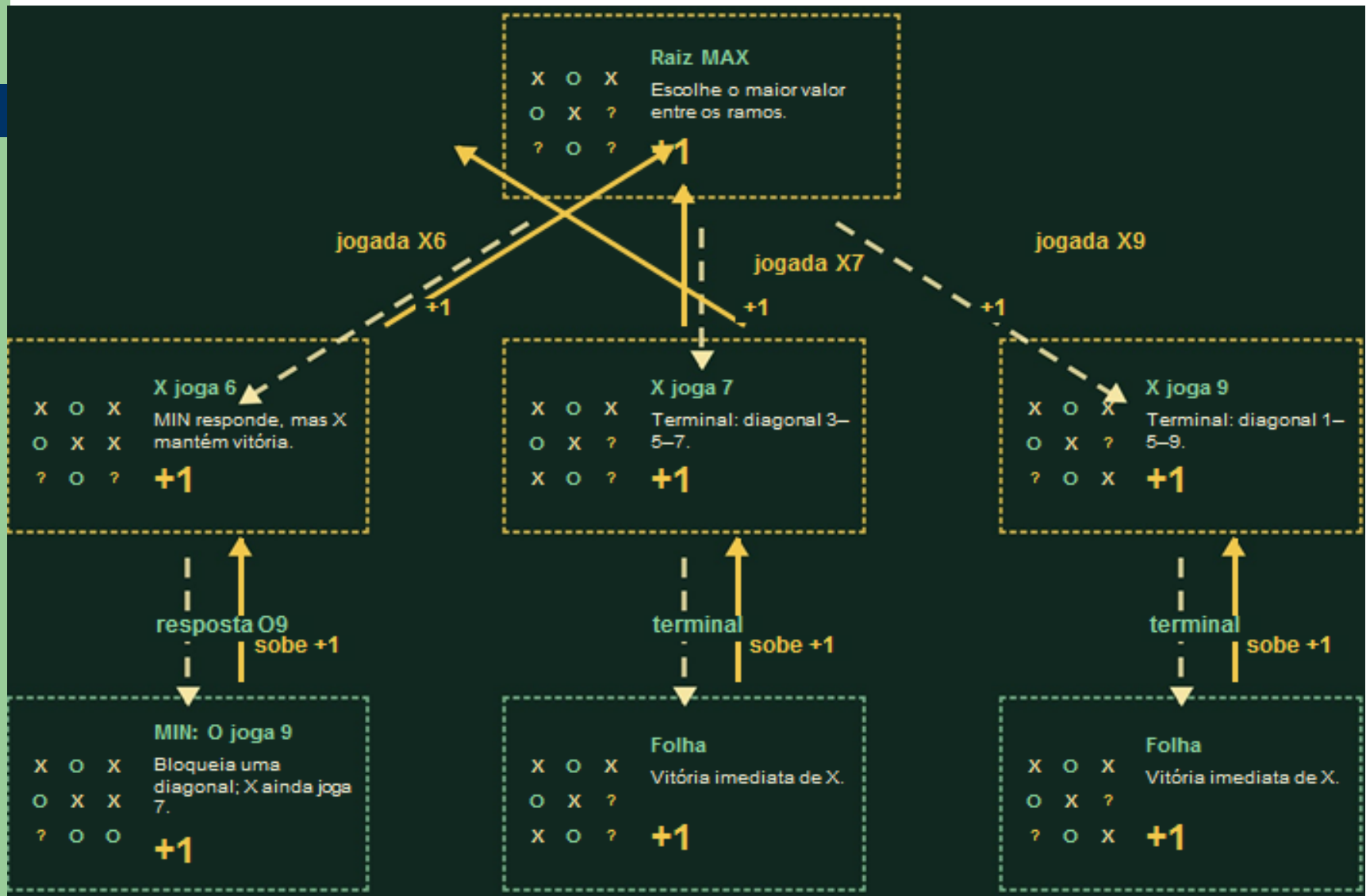


O minimax perguntará: **qual jogada garante o melhor resultado contra uma resposta ótima de O?**

6 7 9

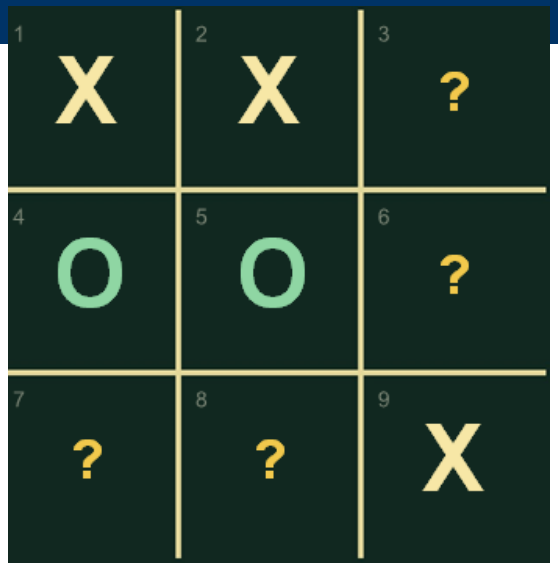
Avaliação das folhas:
+1 se X vence, 0 se empata e −1 se X perde.

Exemplo de uma sub-árvore do Jogo da Velha



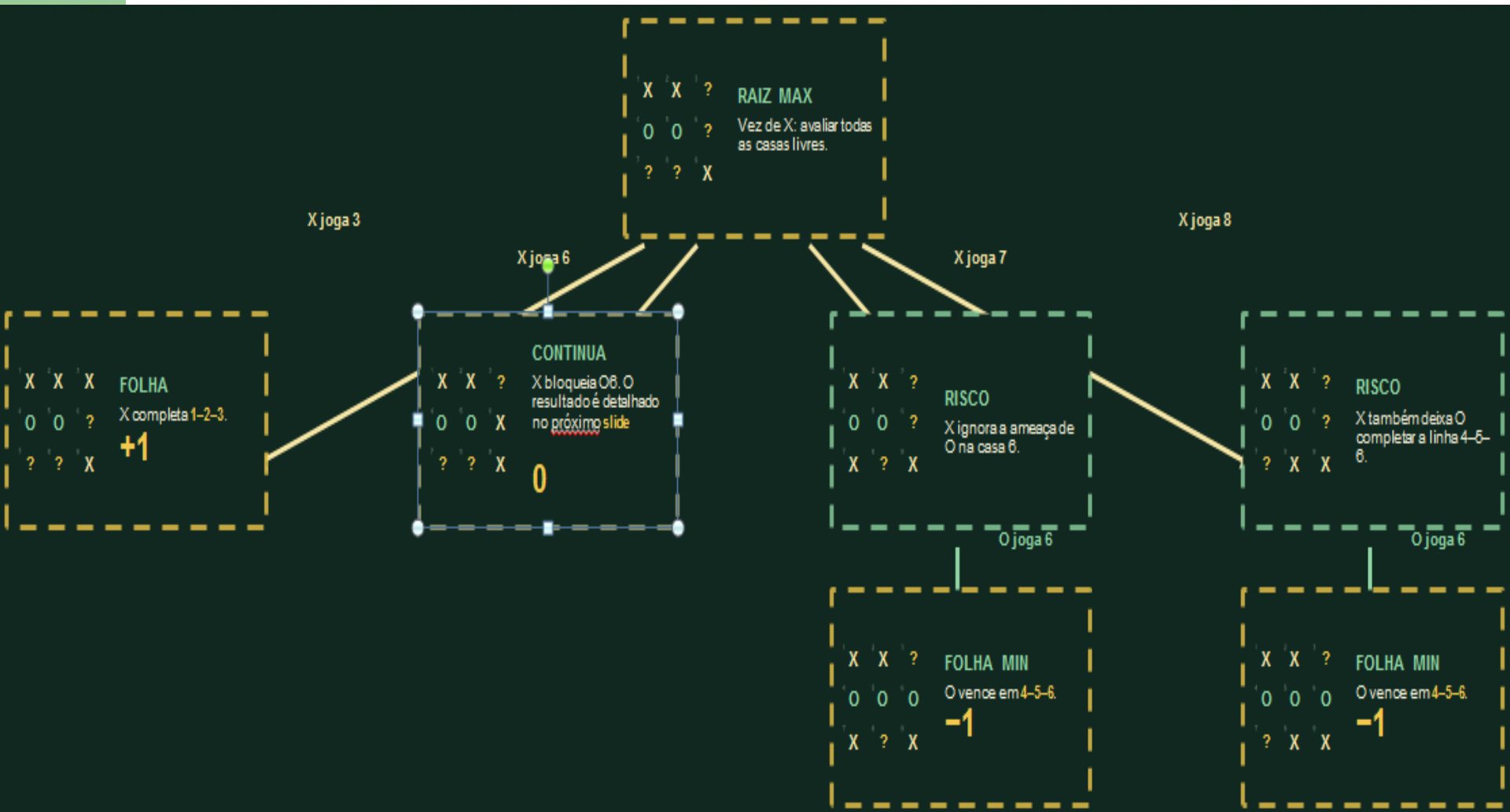
Exemplo Sub-árvore do Jogo da Velha - Uma posição com disputa real

É a vez de MAX: X joga

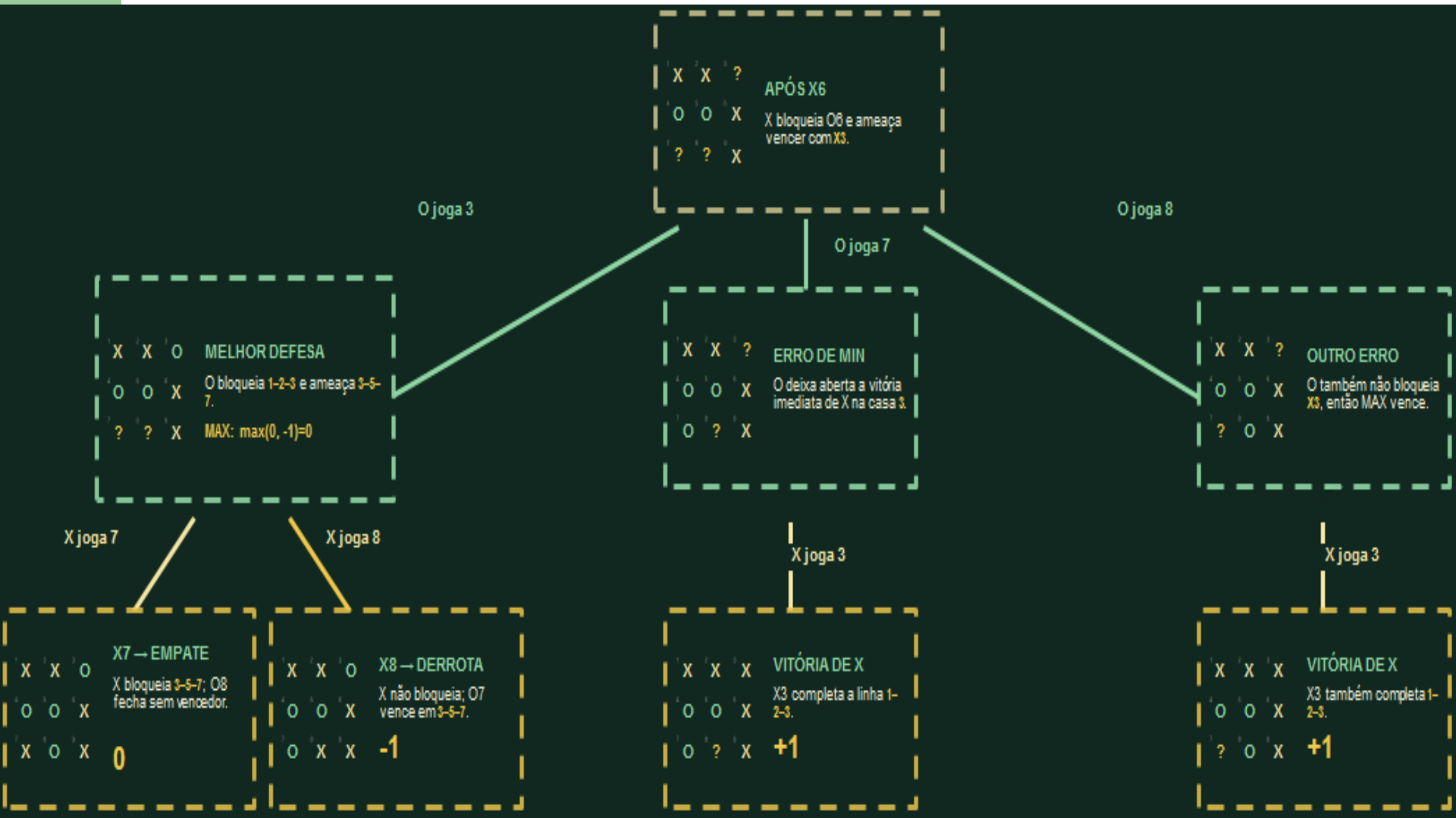


O minimax perguntará: **qual jogada garante o melhor resultado contra uma resposta ótima de O?**

Exemplo Sub-árvore do Jogo da Velha - Uma posição com disputa real

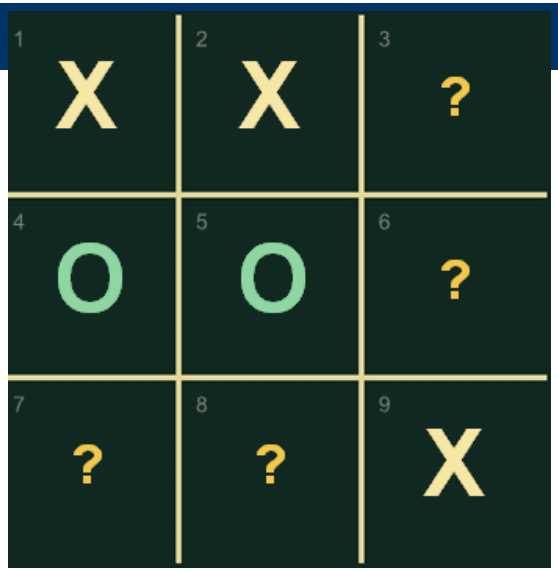


Exemplo Sub-árvore do Jogo da Velha - Uma posição com disputa real



Exemplo Sub-árvore do Jogo da Velha – Com poda alpha-beta

É a vez de MAX: X joga



Ao avaliar primeiro X joga 3, X encontra vitória imediata e fixa **α = +1**. No ramo **X joga 6**, basta MIN achar **O3 = 0** para concluir que esse ramo não supera a vitória já conhecida

Exemplo Sub-árvore do Jogo da Velha – Com poda alpha-beta