Universidade Federal do Espírito Santo
Centro de Ciências Agrárias – CCA UFES
Departamento de Computação

Árvore de Jogos
Minimax e Poda Alfa-Beta

Inteligência Artificial

Site: http://jeiks.net
E-mail: jacsonrcsilva@gmail.com

Motivação
Deep Blue

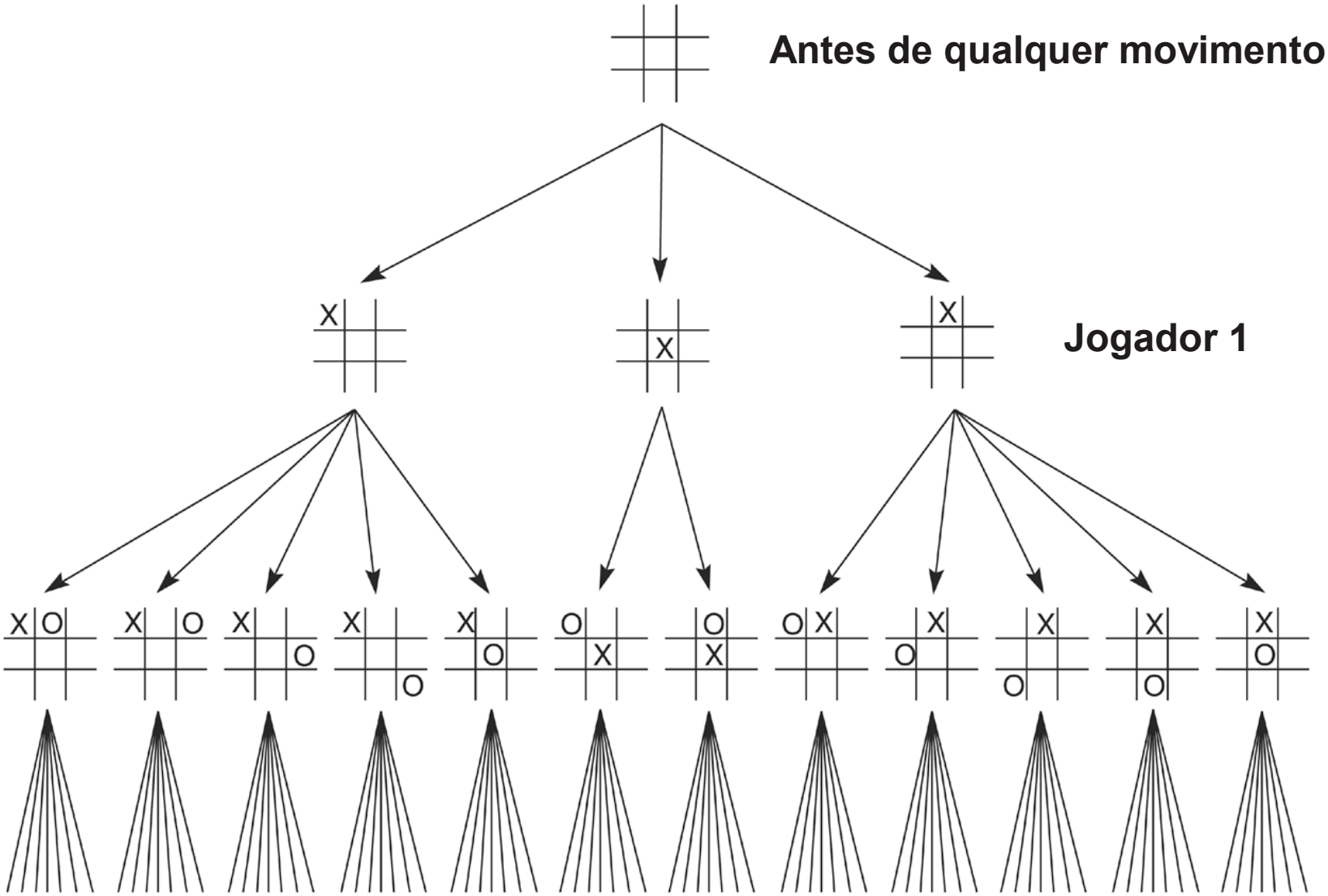
- Supercomputador e um software criados pela IBM especialmente para jogar xadrez;
- Possuía **256** processadores. Analisava aproximadamente **200** milhões de posições por segundo;
- Em *fevereiro de 1996*, Garry Kasparov ganhou 3 partidas, empatou 2 e perdeu 1 contra o *Deep Blue*. Kasparov declarou que era o último humano a ser campeão de xadrez;
- Em *maio de 1997*, após uma severa atualização, *Deep Blue* venceu Kasparov em um novo confronto de 6 partidas, sendo 2 vitórias, 3 empates e 1 derrota.

Jogos com Inteligência Artificial

- Costumam utilizar diversas técnicas de busca;
- Utilizam heurísticas agregadas;
- E possuem uma uma detalhada base de conhecimento.

Árvores de Jogos

- É a representação de jogos para dois jogadores utilizando árvores, onde:
  - **Nó raiz**: estado antes de qualquer movimento do jogo;
  - **Nós da árvore**: possíveis estados do jogo;
  - **Arcos** (arestas não contidas em círculos): movimentos.
- Os movimentos dos dois jogadores são os **níveis** da árvore:
  - Arcos do *primeiro nível*: movimentos do 1º Jogador;
  - Arcos do *segundo nível*: movimentos do 2º Jogador;
  - e assim por diante.
- Folhas da árvore: **estados finais** (vitória, perda ou empate);
- O **nó objetivo** pode ser um estado final onde o computador ganhou;


Antes de qualquer movimento
Jogador 1
Jogador 2

Árvore de Jogos

- Uma abordagem possível:
  - Utilizar **busca** em *profundidade* ou *largura*.
  - Porém:
    - Não funciona para alcançar o objetivo;
    - Pois não é o computador que define todos as jogadas;
    - Existe um adversário inteligente.
- Para utilizar uma árvore de jogos, o computador precisa de uma *função de avaliação*.
  - Assim, pode-se tratar estados de vitória como nós objetivos e conduzir as buscas normalmente.

Premissas em Árvores de Jogos

- Para melhor explicação, assumiremos que:
  - Não existe *elemento de azar:*
    - Jogadas de sorte/azar, como dados e/ou cartas.
  - Tem total conhecimento do estado do jogo
    - Os jogadores não escondem informação de ninguém.

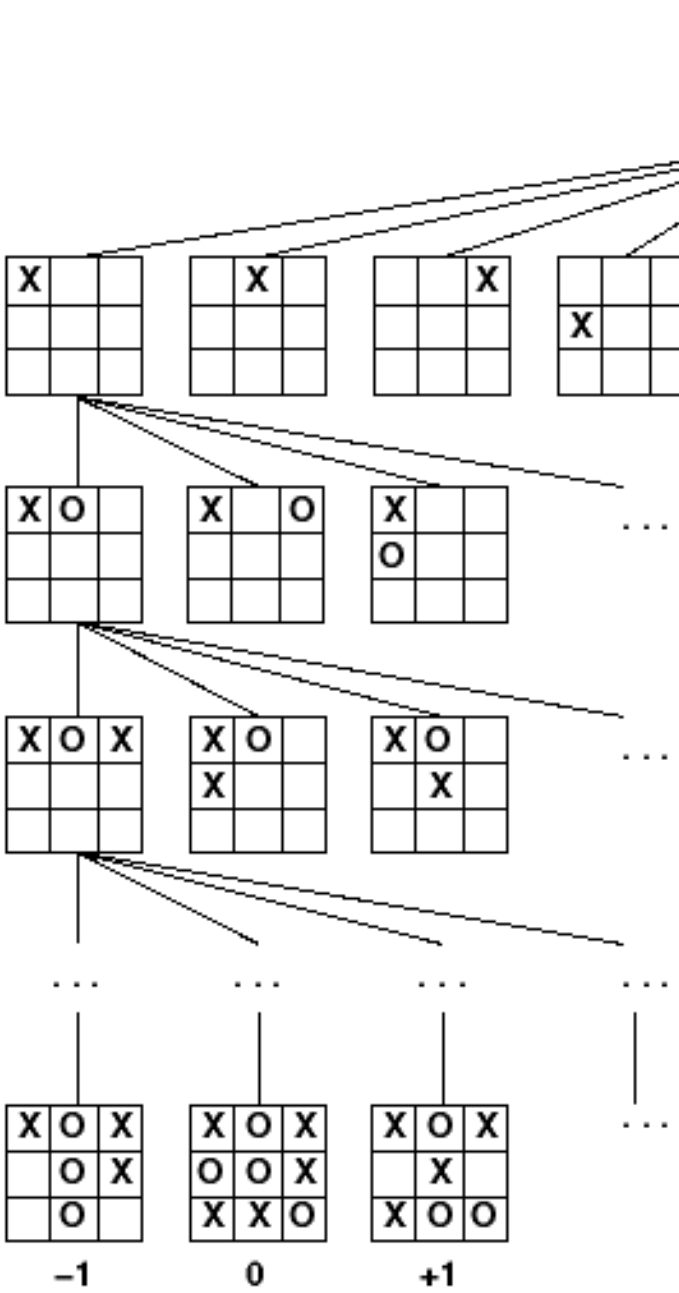
Soma Zero

- O placar geral ao final de um jogo, para cada jogador,
  - Pode ser 1 (vitória); ou
  - Pode ser 0 (empate); ou
  - Pode ser -1 (derrota).
- Então, o placar total:
  - para ambos os jogadores e
  - para qualquer jogo,
  - deve **sempre** ser **zero**.
  - Ou seja:
    - (Jogador *A* ganha) 1 pto. **+** (Jogador B perde) -1 pto. **=** 0 pontos;
    - (Jogador *B* ganha) 1 pto. **+** (Jogador A perde) -1 pto. **=** 0 pontos;
    - (Jogador A empata) 0 pto. **+** (Jogador B empata) 0 pto. **=** 0 pontos.

Soma Zero

- As técnicas de busca são métodos *Adversários*
  - Cada jogador não está apenas tentando ganhar,
  - Mas também provocando o oponente perder.
- O computador sempre assume que o oponente está jogando para ganhar.
- Vez:
  - Refere-se à profundidade da árvore;
  - Representa um único nível de escolha na árvore.
- O Movimento:
  - Representa duas escolhas na árvore, sendo uma para cada jogador.

Utilização da Soma Zero



E a função de avaliação?

Funções de Avaliação

- Também conhecidas como *avaliadores estáticos*,
  - Pois avaliam um jogo a partir apenas de uma posição estática.
- São vitais para a maioria dos programas,
  - pois quase sempre é impossível realizar a busca em toda a árvore de jogo, devido ao seu tamanho.
- Uma busca raramente deve atingir o nó folha;
- O programa necessita ser capaz de *cortar* a busca e *avaliar* a posição do tabuleiro naquele nó.
- Uma função de avaliação é utilizada
  - para examinar uma posição específica do tabuleiro;
  - para estimar quão bem o computador está indo, ou qual a probabilidade de ganhar a partir desta posição.
- Uma função de avaliação deve ser rápida e eficiente.

Funções de Avaliação

Como utilizar uma função de avaliação para comparar duas posições?

Funções de Avaliação

Funções de Avaliação geralmente são

funções lineares ponderadas

- Onde:
  - Diferentes pontuações são determinadas para uma dada posição e simplesmente reunidas de forma ponderada.
- Exemplo em um jogo de xadrez:

  r = número de rainhas t = número de torres

  c = número de cavalos b = número de bispos

  p = número de peões
  - pontuação = $9r + 5t + 3c + 3b + p$
- Se dois computadores jogassem, o computador vencedor seria o que possuísse a melhor função de avaliação.

Como realizar e melhorar a busca?

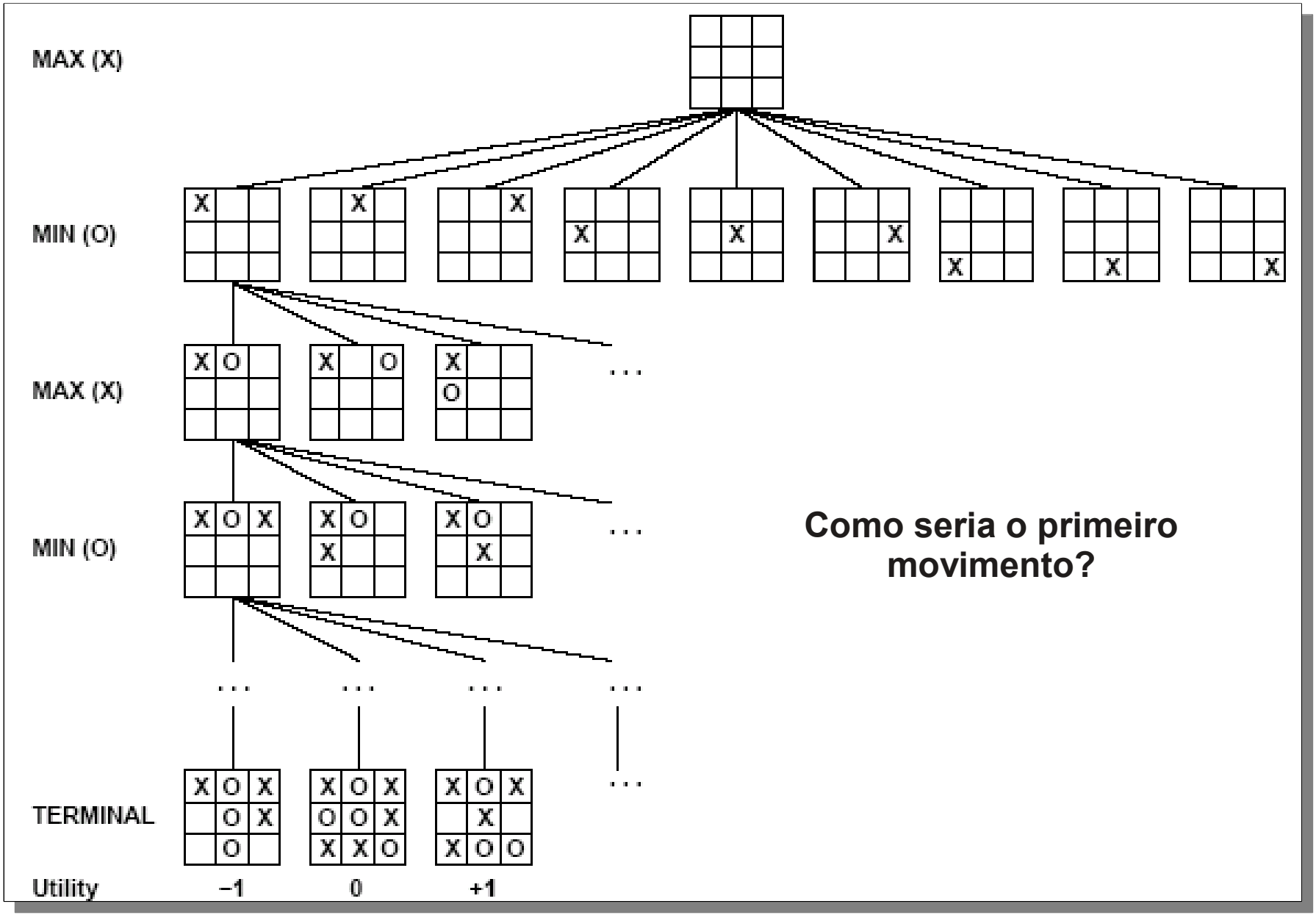
MiniMax;

Antecipação Limitada, Poda Alfa-Beta.

MiniMax

- Dois jogadores : MAX e o MIN;
- MAX começa e os jogadores alternam até que o jogo termine.
- O vencedor leva uma recompensa, o perdedor uma penalidade.
- Então possuímos:
  - Estado Inicial: onde ninguém fez nenhuma jogada ainda;
  - Função Sucessora: que fornece a lista de movimentos legais.
  - Teste Terminal: que verifica se o jogo acabou (se é um estado terminal).
  - Função de Utilidade: Dá um valor aos estados terminais. No jogo da velha, por exemplo: 1 para *vitória*, -1 para *derrota* e 0 para *empate*.
- O MAX então utiliza uma árvore de busca para determinar seu próximo movimento.

MiniMax: Árvore parcial



Estratégia Ótima

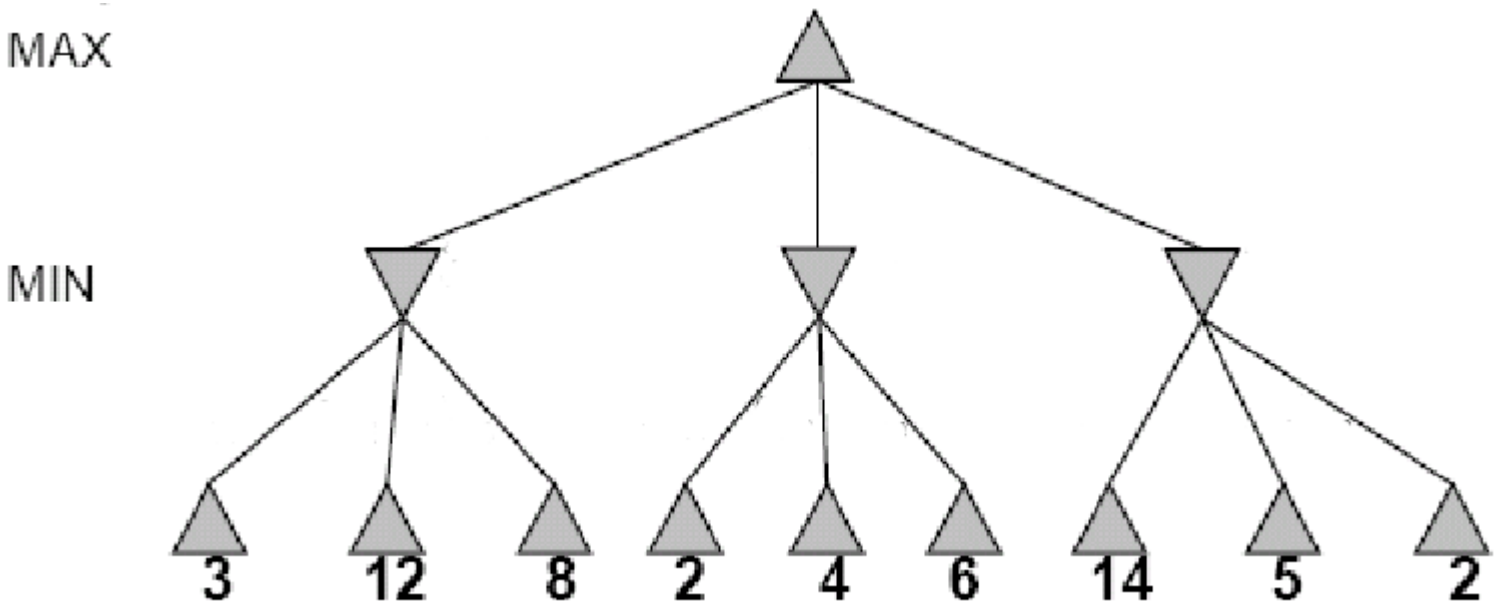
- Deve-se encontrar a estratégia para MAX, mas sempre assumindo que MIN é infalível.
- Então nossa premissa é:

  Os dois jogadores jogam de forma ótima.

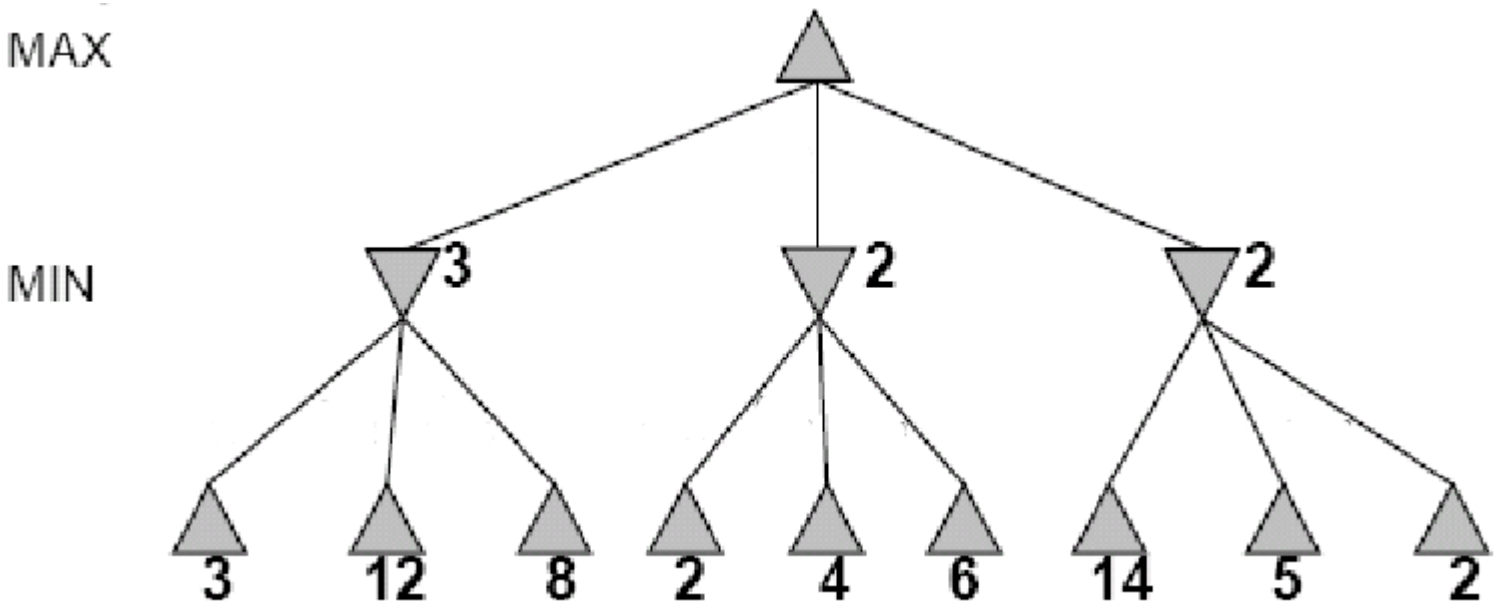
- Dada uma árvore de jogo, a estratégia ótima pode ser determinado usando o valor minimax de cada nó:

$$\text{VALOR-MINIMAX}(n) = \begin{cases} \text{UTILIDADE}(n) & \text{Se } n \text{ é terminal} \\ \max_{s \in \text{sucessores}(n)} \text{VALOR-MINIMAX}(s) & \text{Se } n \in \text{MAX} \\ \min_{s \in \text{sucessores}(n)} \text{VALOR-MINIMAX}(s) & \text{Se } n \in \text{MIN} \end{cases}$$

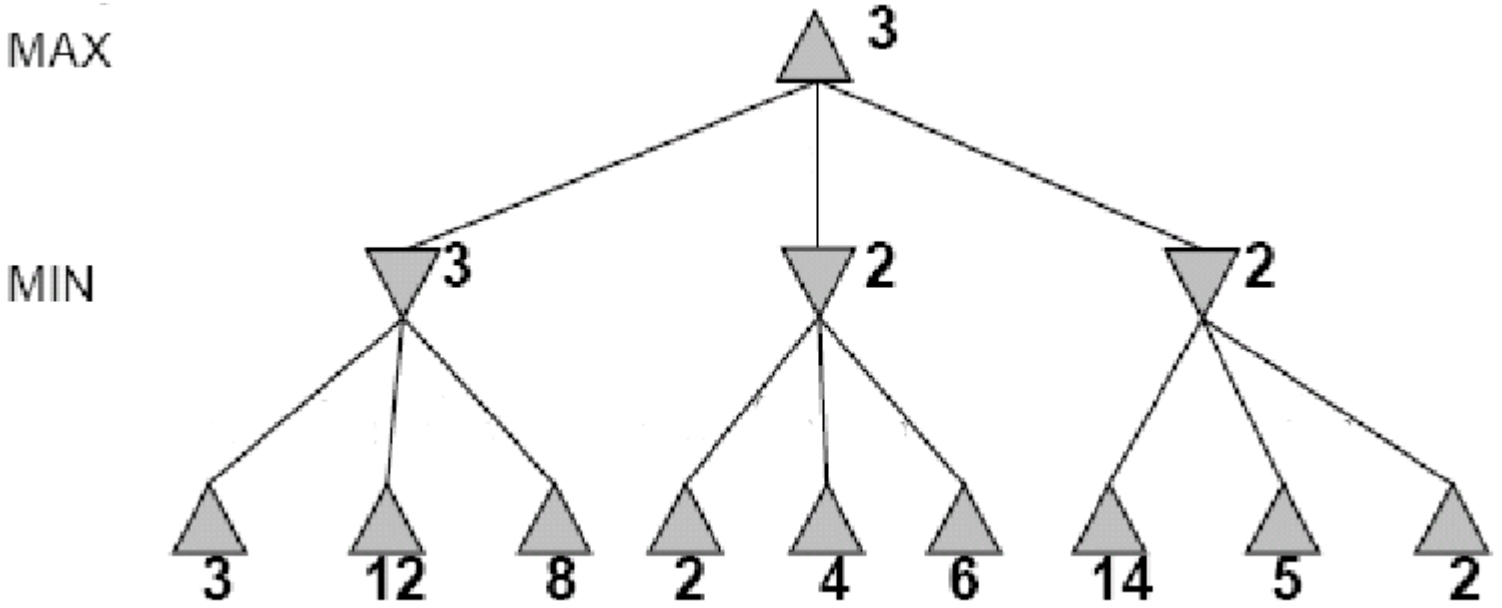
Exemplo - Passo 1



Exemplo - Passo 2

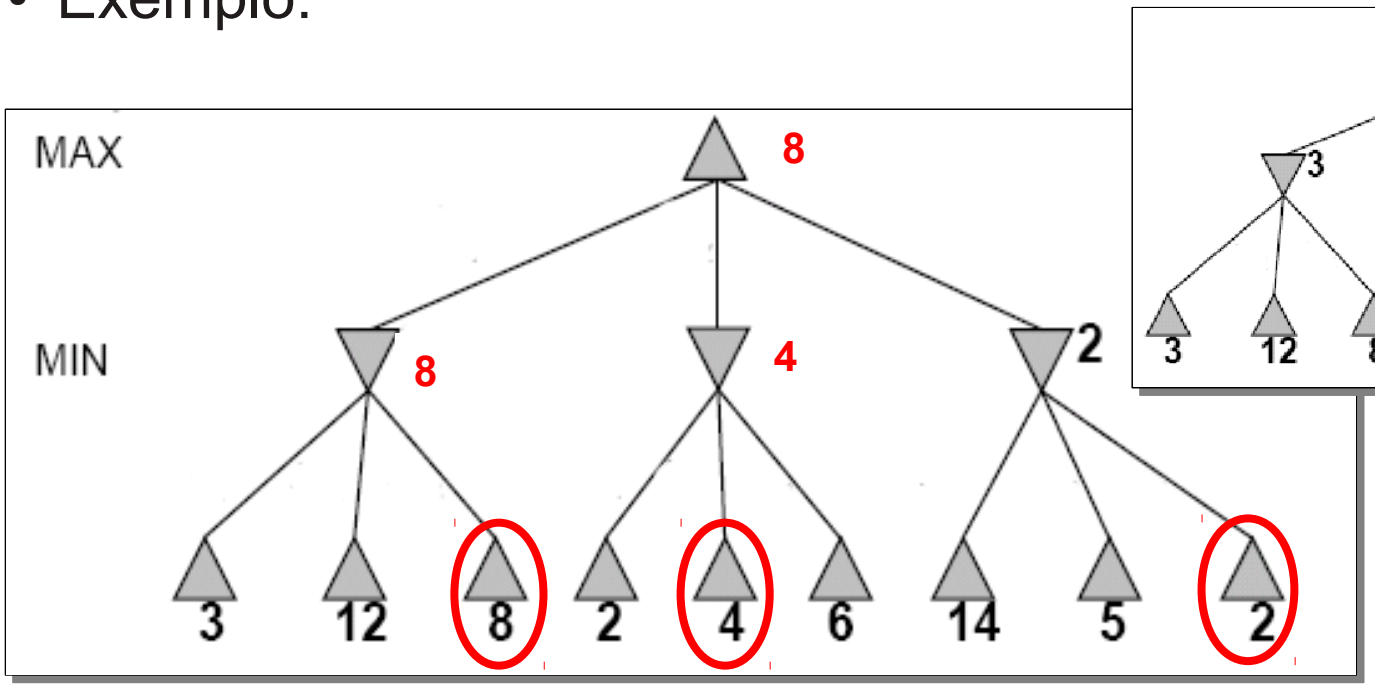


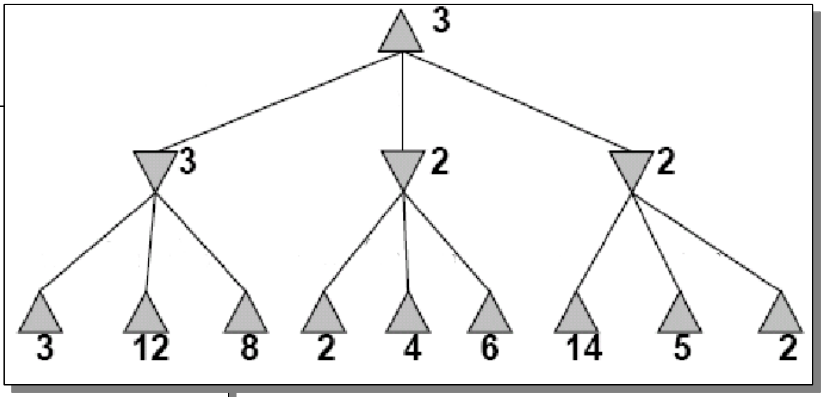
Exemplo - Passo 3



E se o MIN não for um bom jogador?

- A definição de melhor jogada para MAX assume que MIN joga de forma ótima.
- Logo, estamos maximizando o resultado em um cenário de pior caso para MAX.
- Se MIN não jogar otimamente, MAX pode se dar ainda melhor.
- Exemplo:





Metodologia

- Implementando como uma busca em profundidade;
- Tem que avaliar todos os filhos antes de poder avaliar um nó;
- Logo, para avaliar a raiz, tem que avaliar a árvore inteira!
- Isto pode ser extremamente custoso,
  - principalmente se o fator de ramificação for muito alto.

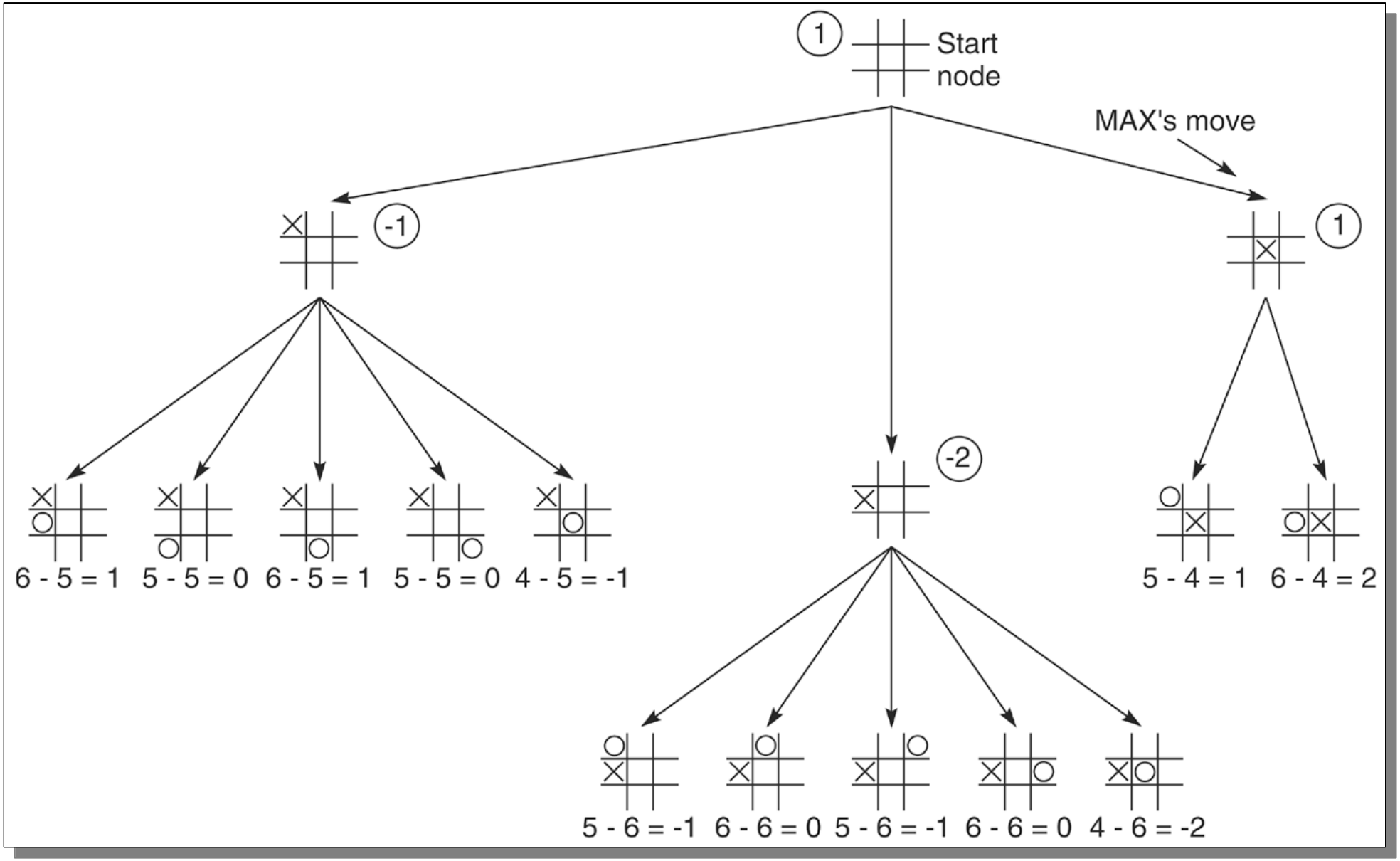
Otimizando o método

- Métodos para não examinar todos os nós:
  - Antecipação Limitada;
  - Poda Alfa-Beta.

Antecipação Limitada

- Consiste em executar uma busca no espaço de estados até um número de níveis **predefinido**.
- A Função de Avaliação é substituída por uma F*unção Heurística*.
- O teste terminal é substituído por um teste de corte.
- Exemplo no jogo da velha:
  - H(n) = X(n) – O(n) ,sendo:
    - H(n) a função heurística (avaliação de um estado);
    - X(n) Total de linhas vitoriosas de X;
    - O(n) Total de linhas vitoriosas de O.

MiniMax: antecipação limitada



Antecipação Limitada

- Porém, é possível que a antecipação limitada encontre um caminho promissor que seja ruim mais tarde (problema do horizonte).
- Além disso, avaliações heurísticas muito profundas podem ser influenciadas por sua profundidade excessiva.
  - Assim, estimativa de minimax (que desejamos) pode ser diferente do minimax das estimativas (o que fazemos).

Poda Alfa-Beta

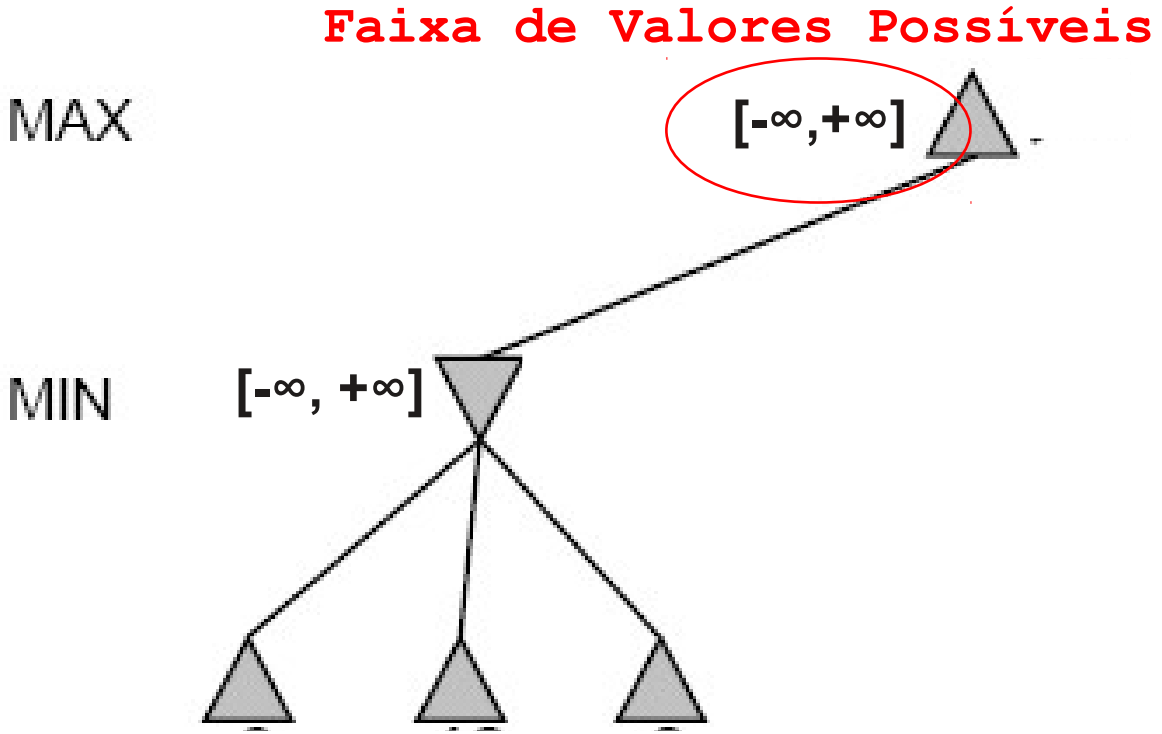
- Inicialmente, a árvore de jogo é percorrida em ordem de profundidade.
- A cada nó que não seja folha, é armazenado um valor, sendo:
  - Alpha: para os MAX
    - É o máximo (melhor) valor encontrado até então nos descendentes dos nós MAX
  - Beta: para os MIN
    - É o mínimo (melhor) valor encontrado até então nos descendentes dos nós min
- Se:
  - Se é nó MAX: Se valor_alfa_de(nó) >= valor_beta_ancestral;
  - Se é nó MIN: Se valor_beta_de(nó) <= valor_alfa_de_ancestral;
  - ENTÃO corta_busca_abaixo(nó)

Poda Alfa-Beta

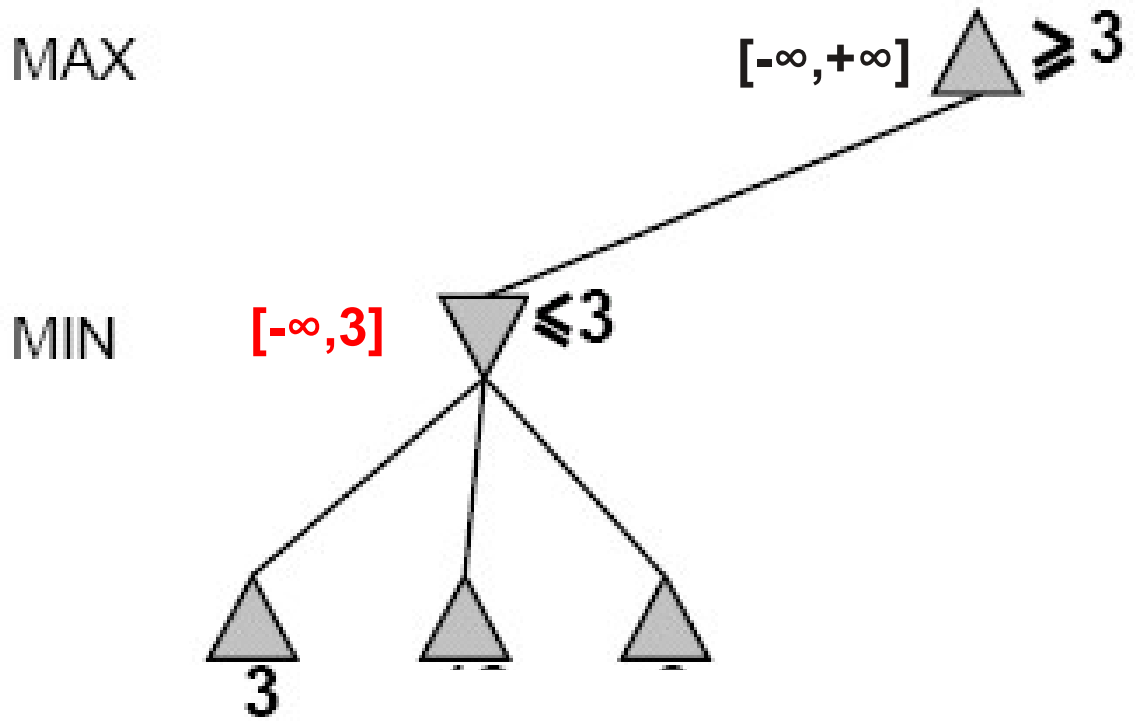
Para implementar isto, nós manipulamos os valores alfa-beta, armazenando-os em cada nó interno da árvore de busca....

Exemplo

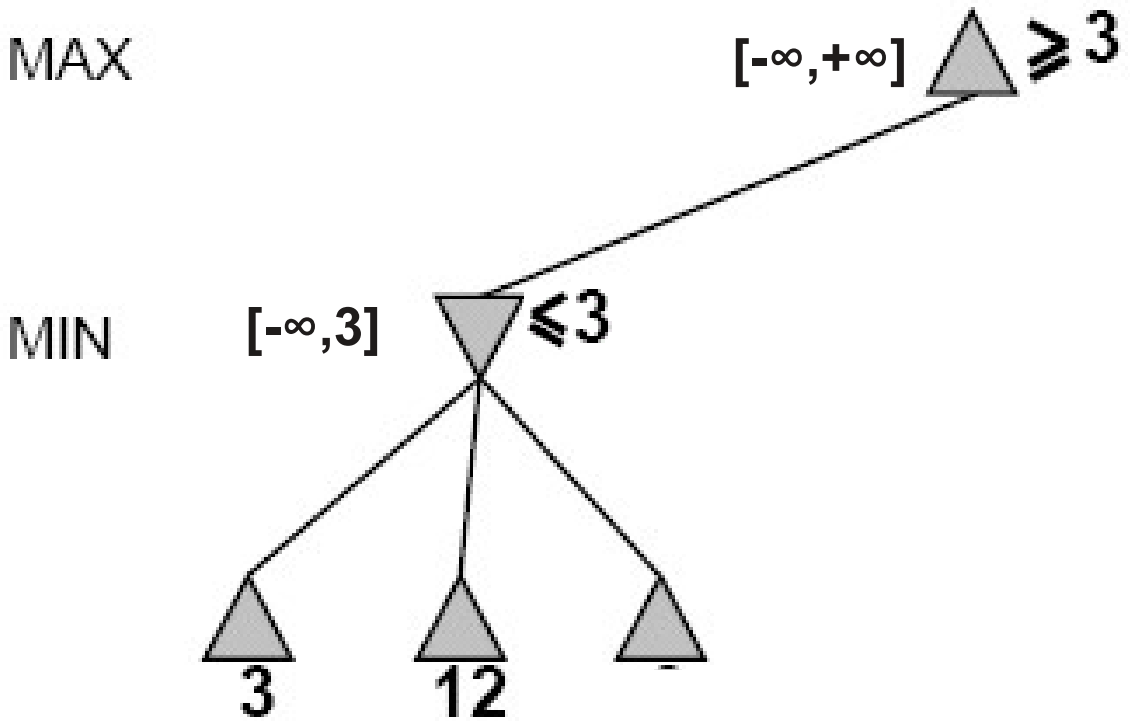
Faça a busca pelo primeiro mais fundo até a primeira folha



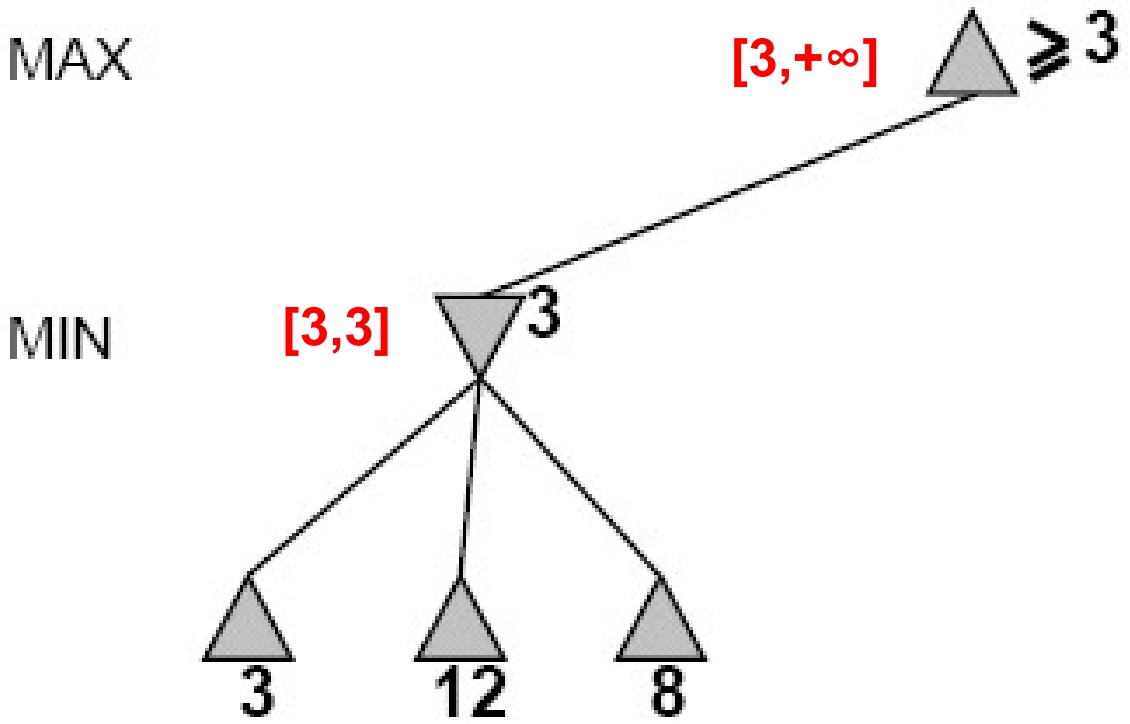
Exemplo



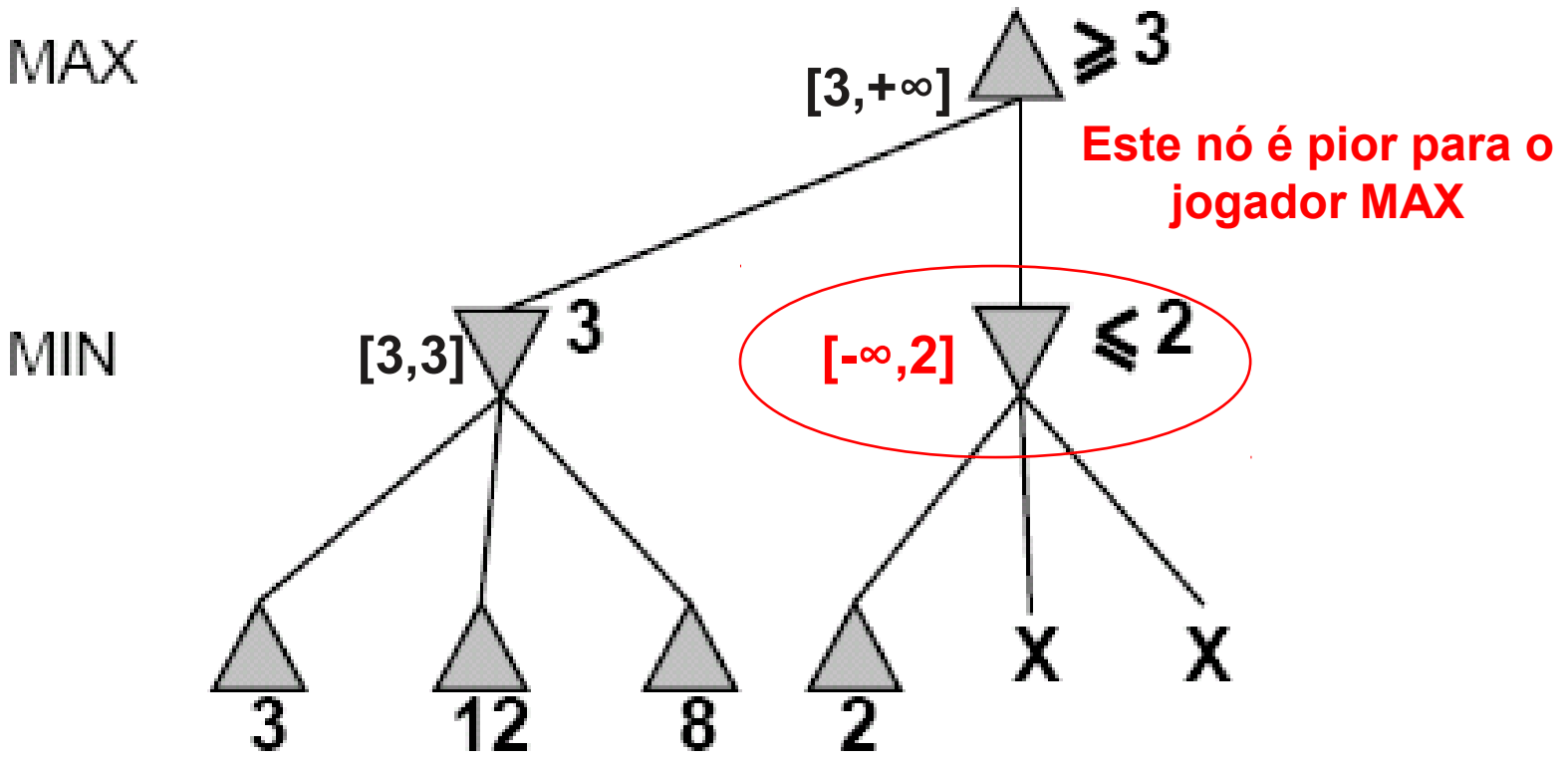
Exemplo



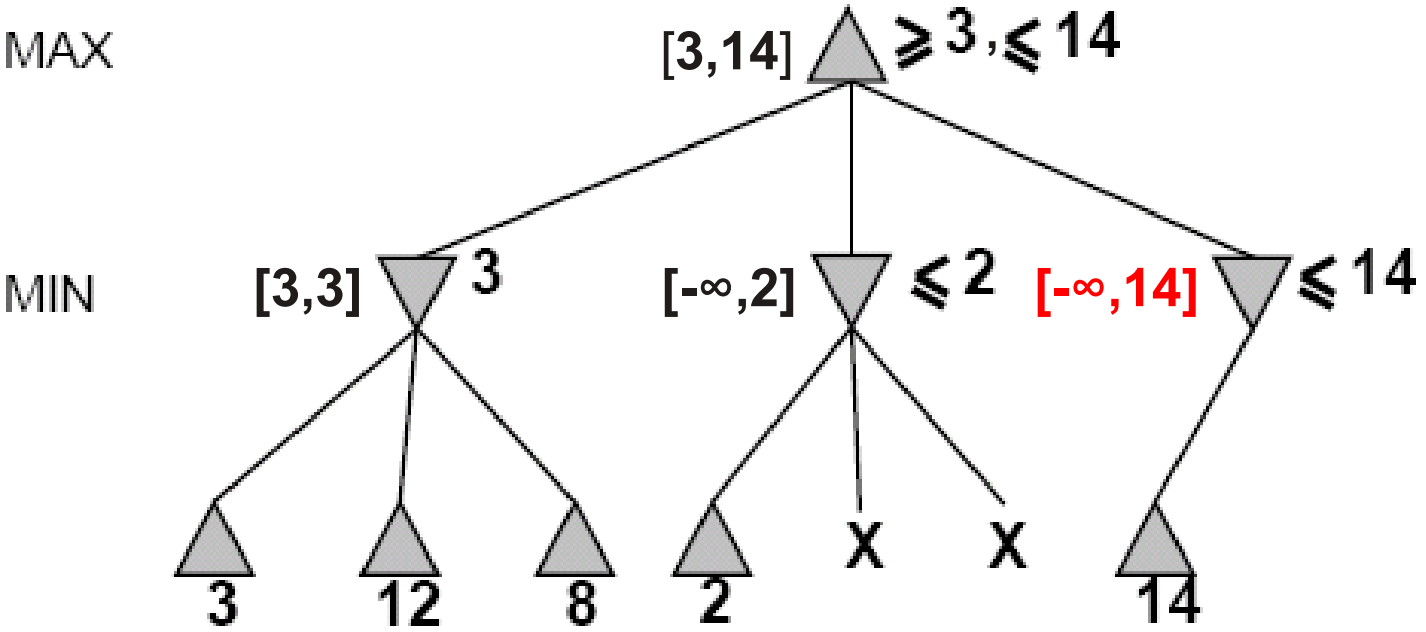
Exemplo



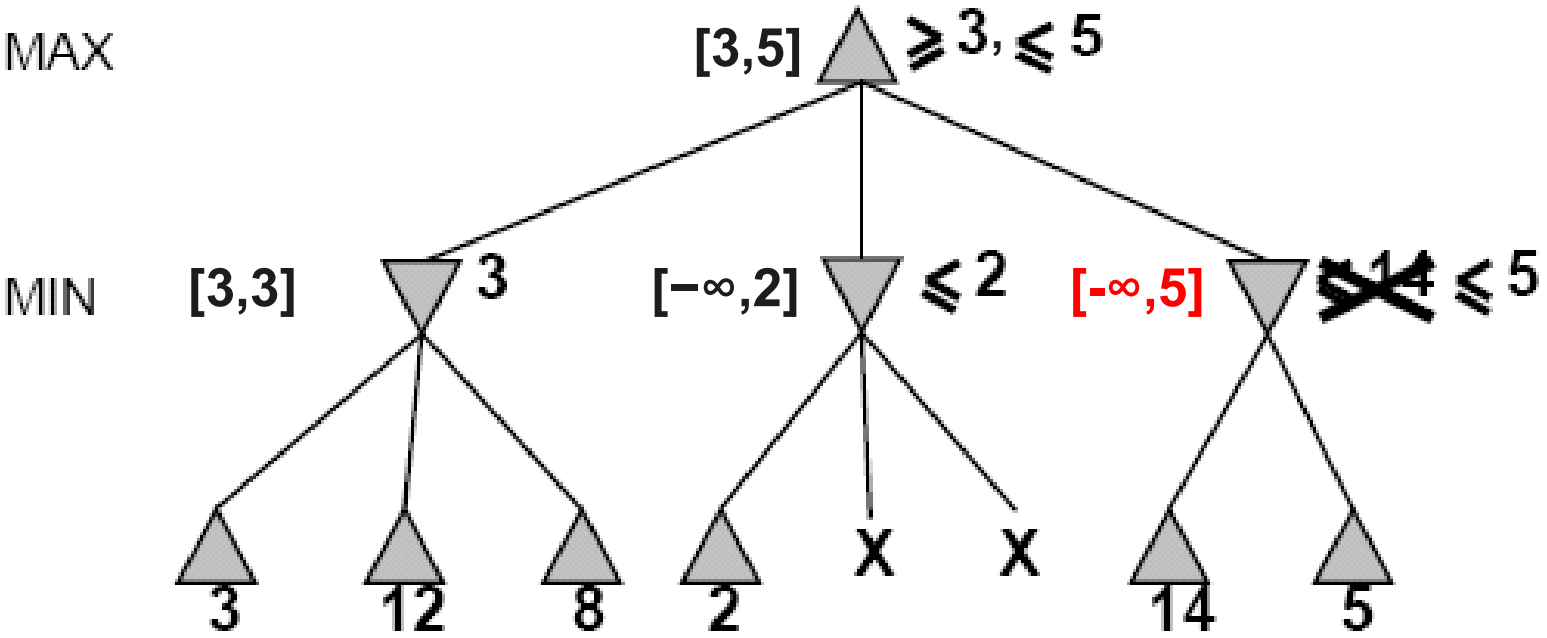
Exemplo



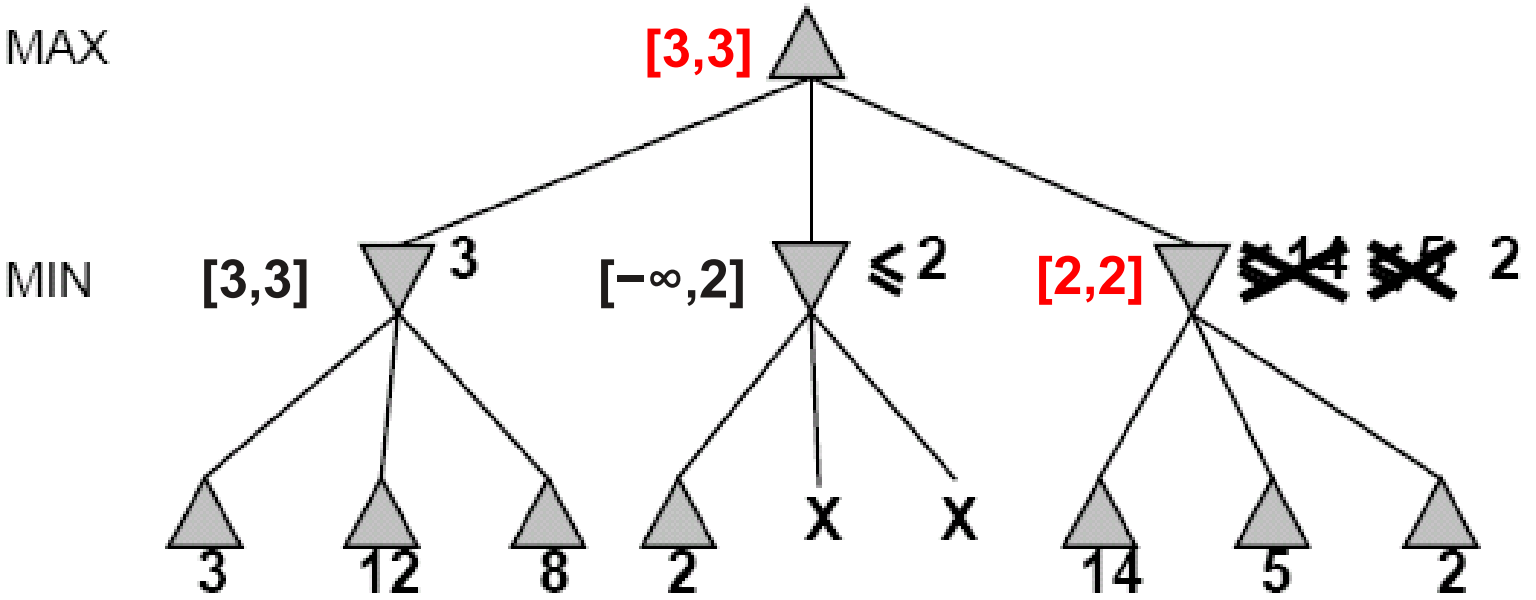
Exemplo



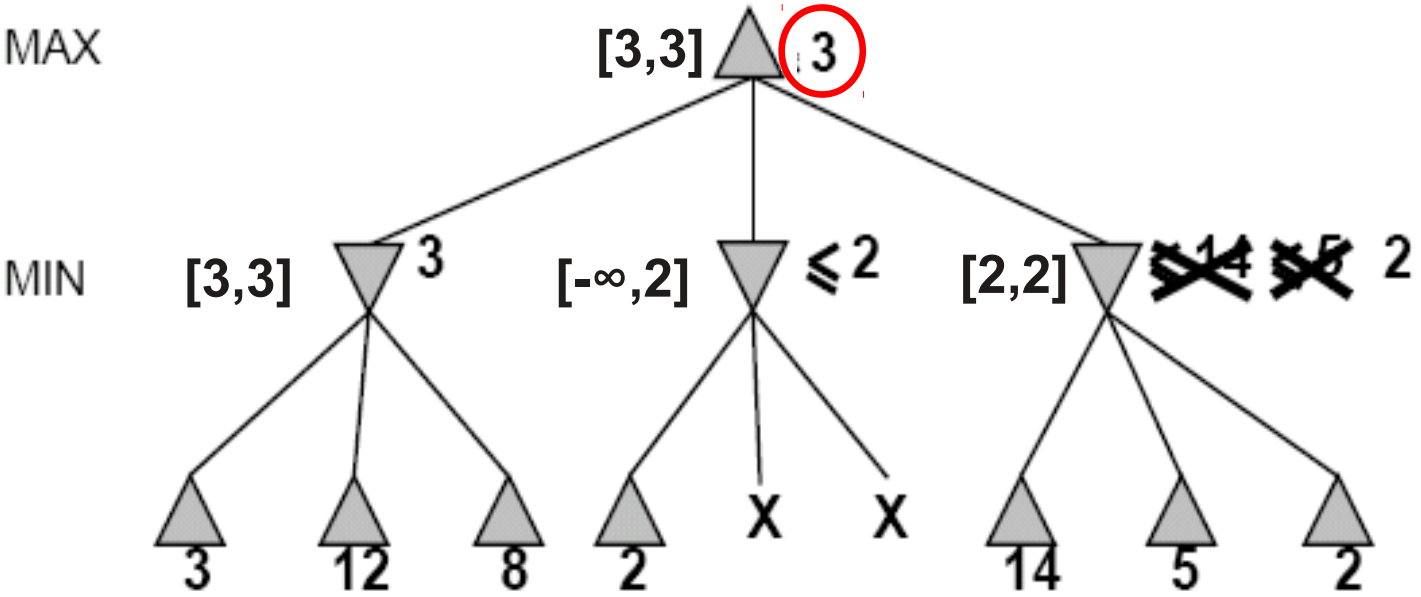
Exemplo



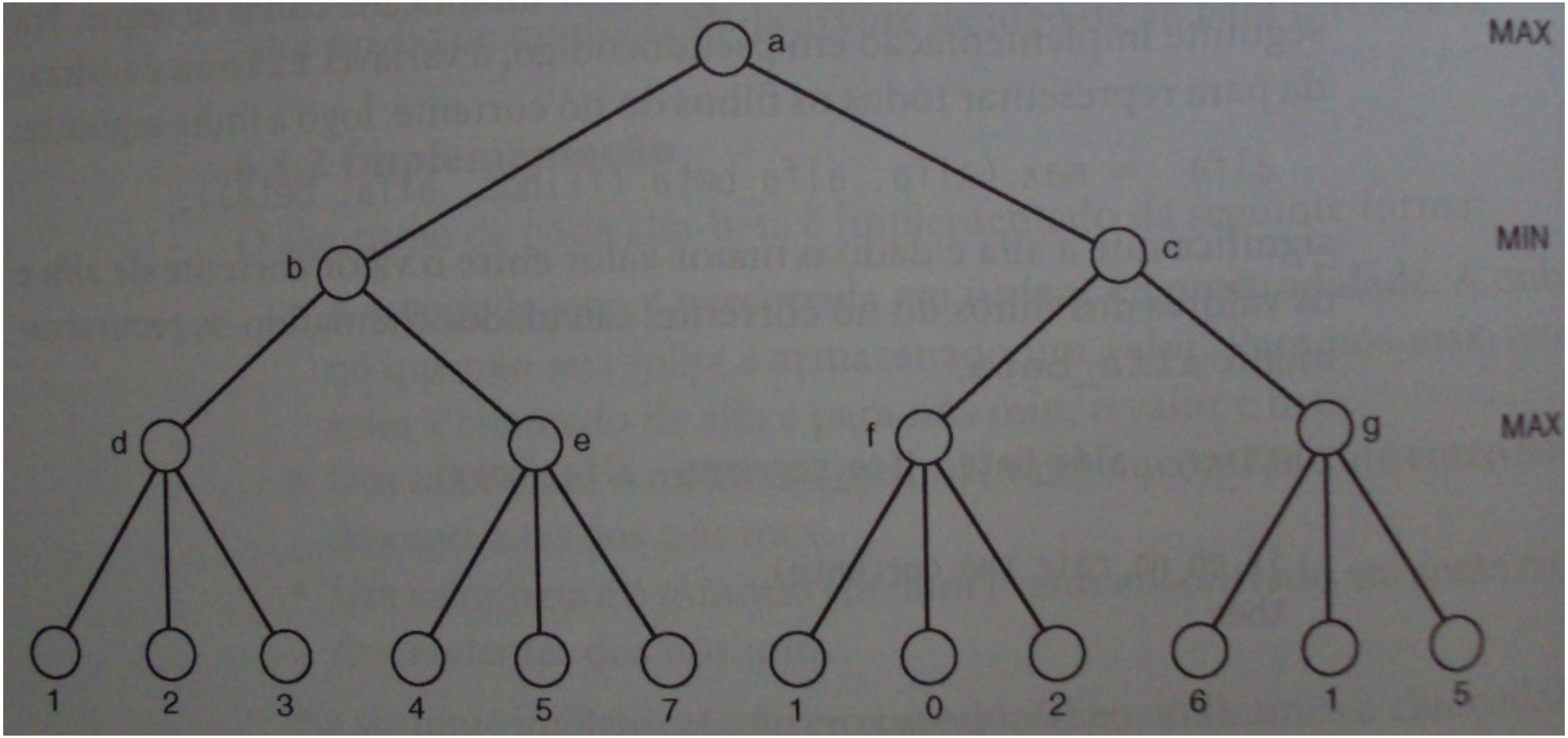
Exemplo



Exemplo



Exercício



Conceitos da poda Alfa-Beta

- Seja um nó **n** na árvore
- Se um jogador tem uma jogada melhor em:
  - Um nó pai de **n**; ou em
  - Um nó mais alto do que **n.**
- **n** nunca será atingido em um jogo real.
- Logo, quando sabemos o suficiente sobre **n**, ele pode ser ignorado.