BUSCA CONTRA ADVERSÁRIO (ADVERSARIAL SEARCH)

Benjamin Grando Moreira

Jogos

- Em ambientes multiagentes, há pouca previsibilidade
  - Ações dos outros agentes
  - É preciso tratar as contingências
- Em ambiente competitivos, há conflito de objetivos
  - Ex. negociação em comércio eletrônico
- Nestes casos, temos “Busca contra adversário, ou simplesmente “jogo”
  - Em economia, na “teoria dos jogos”, estuda-se vários tipos de “jogos”

Jogos

- Histórico
  - Xadrez: desde anos 50, hoje atingiu nível de mestre
  - Damas e Othelo: hoje, melhor que qualquer humano
  - Gamão: hoje, nível de campeão
  - Go, nível amador

Jogos

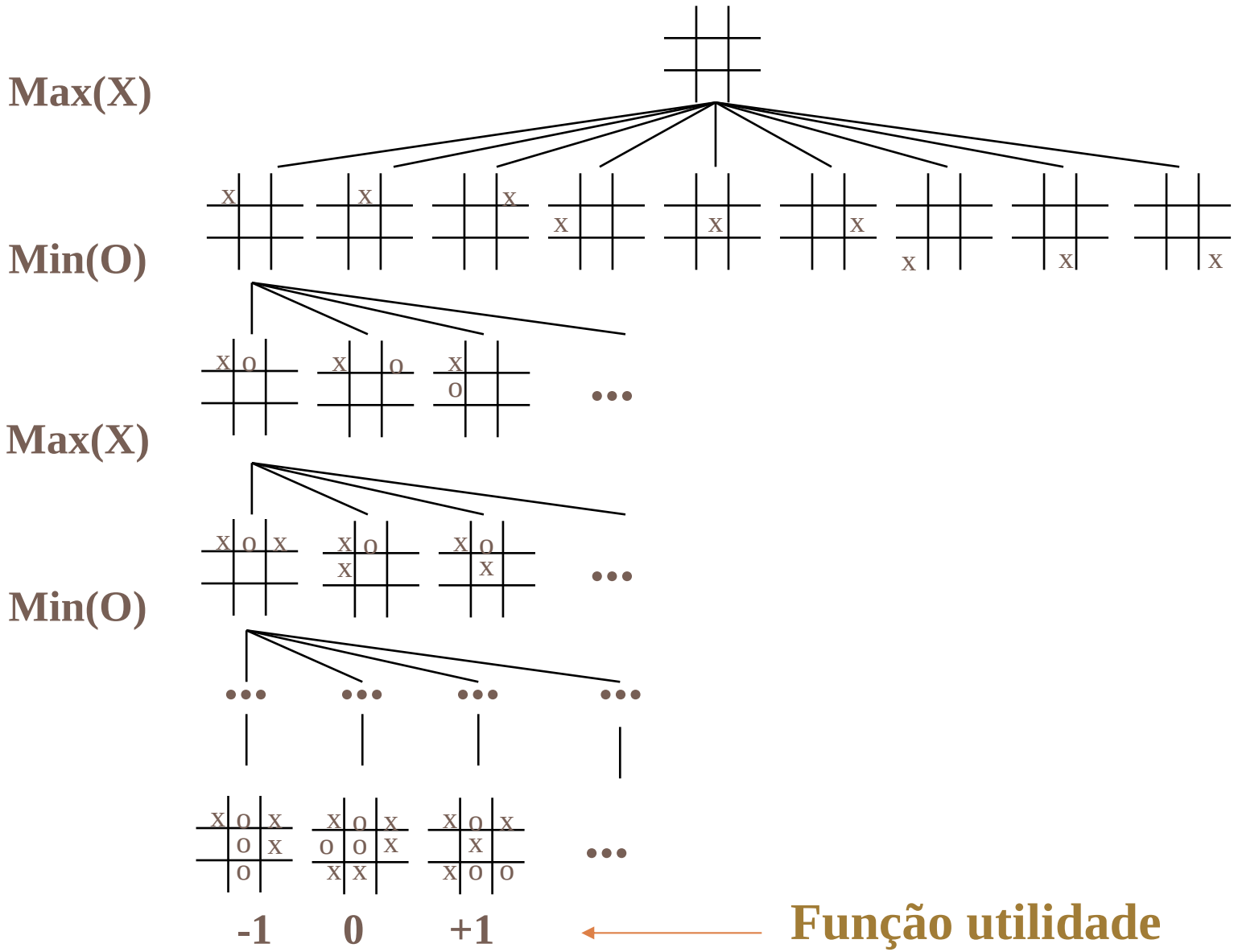
- Aplicações atrativas para métodos IA desde o início.
  - Sinônimo de inteligência
  - Ações bem definidas e ambiente acessível
  - Abstração (representação simplificada de problemas reais)
- Porém desafiador:
  - Complexidade
    - Xadrez: 35 movimentos possíveis por turno, 25 jogadas por jogador por partida => $35^{50} \cong 10^{154}$ (10^{40} nós distintos)
  - Incerteza devido ao outro jogador;
  - Problema “contingencial”: agente deve agir antes de completar a busca

Formulando e resolvendo o problema

- Formulação
  - Estado inicial: posições do tabuleiro + de quem é a vez
  - Estado final: posições em que o jogo acaba
  - Operadores: jogadas legais para um dado estado da partida
  - Função de utilidade (objetivo ou payoff): valor numérico para os estados finais (pontuação)
    - Xadrez = +1, 0, -1; gamão = [-192,+192]

- Busca: algoritmo minimax
  - Idéia: maximizar a utilidade (ganho) supondo que o adversário vai tentar minimizá-la (todos jogam otimamente!)
    - O agente é MAX e o adversário é MIN
  - Minimax faz busca cega em profundidade

Jogo da velha (min-max)



Algoritmo

```
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
```

Críticas

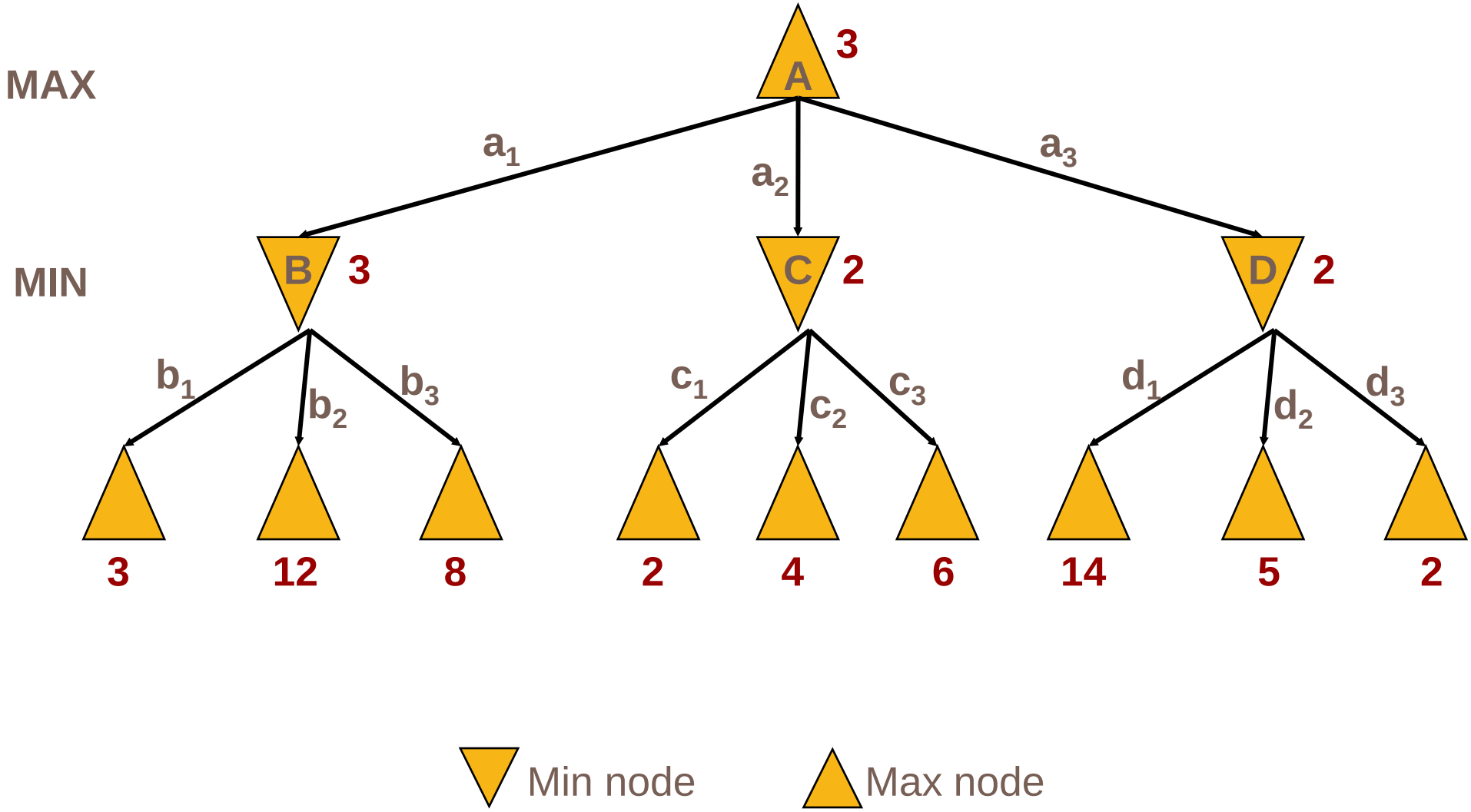
- É completa e ótima mas...
- Problemas
  - Tempo gasto é totalmente impraticável, porém o algoritmo serve como base para outros métodos mais realísticos.
  - Complexidade: $O(b^m)$.
- Para melhorar (combinar duas técnicas)
  - Podar a arvore onde a busca seria irrelevante: **poda alfa-beta (alfa-beta pruning)**
  - Substituir a profundidade *n* de min-max(n) pela estimativa de min-max(n): **função de avaliação**

Função de avaliação

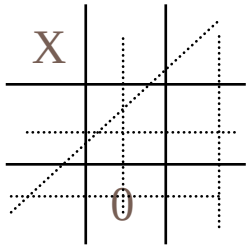
Funções de Avaliação

- Reflete as chances de ganhar: baseada no valor material
  - ex. valor de uma peça independentemente da posição das outras
- Função Linear de Peso de propriedade do nó:
  - $w_1f_1+w_2f_2+...+w_nf_n$
  - Ex. Os pesos (w) no xadrez poderiam ser o tipo de pedra do xadrez (Peão-1, ..., Rainha-9) e os (f) poderiam ser o número de cada peça no tabuleiro.
- Escolha crucial: compromisso entre precisão e eficiência

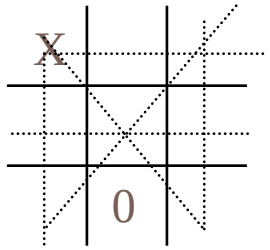
Exemplo de função de avaliação



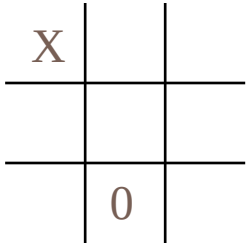
Função para o jogo da velha



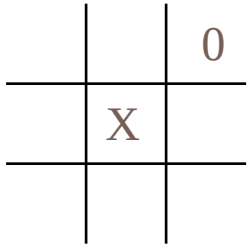
0 tem 5 possibilidades



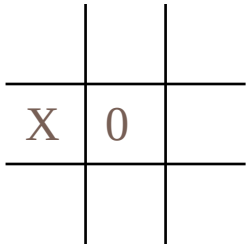
X tem 6 possibilidades



h = 6 - 5 = 1

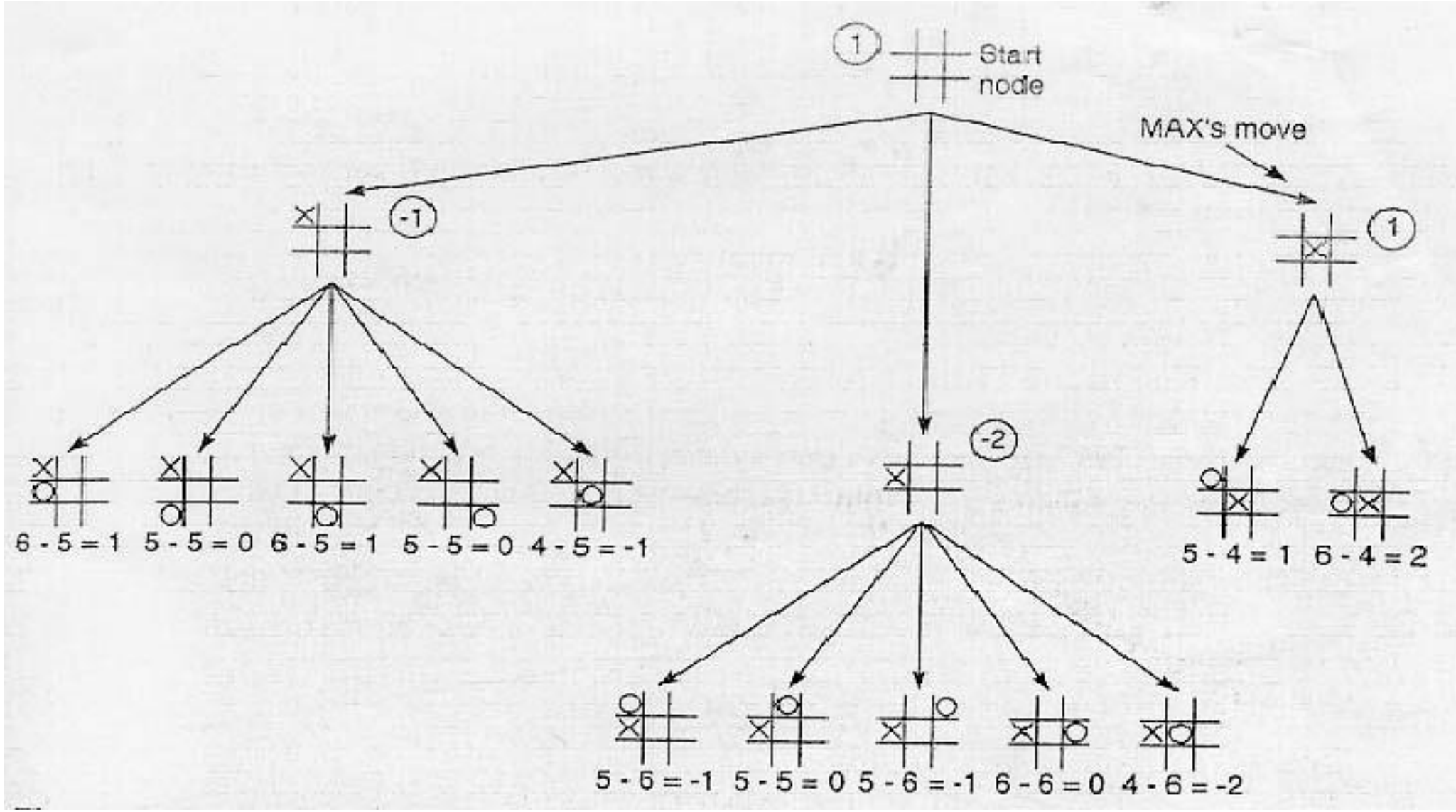


h = 5 - 4 = 1



h = 4 - 6 = = -2

Uso da Funções de Avaliação



Minimax de duas jogadas (two-ply) aplicado à abertura do jogo da velha

Quando aplicar a função de avaliação?

- Definir uma profundidade máxima não funciona devido à incerteza inerente ao problema

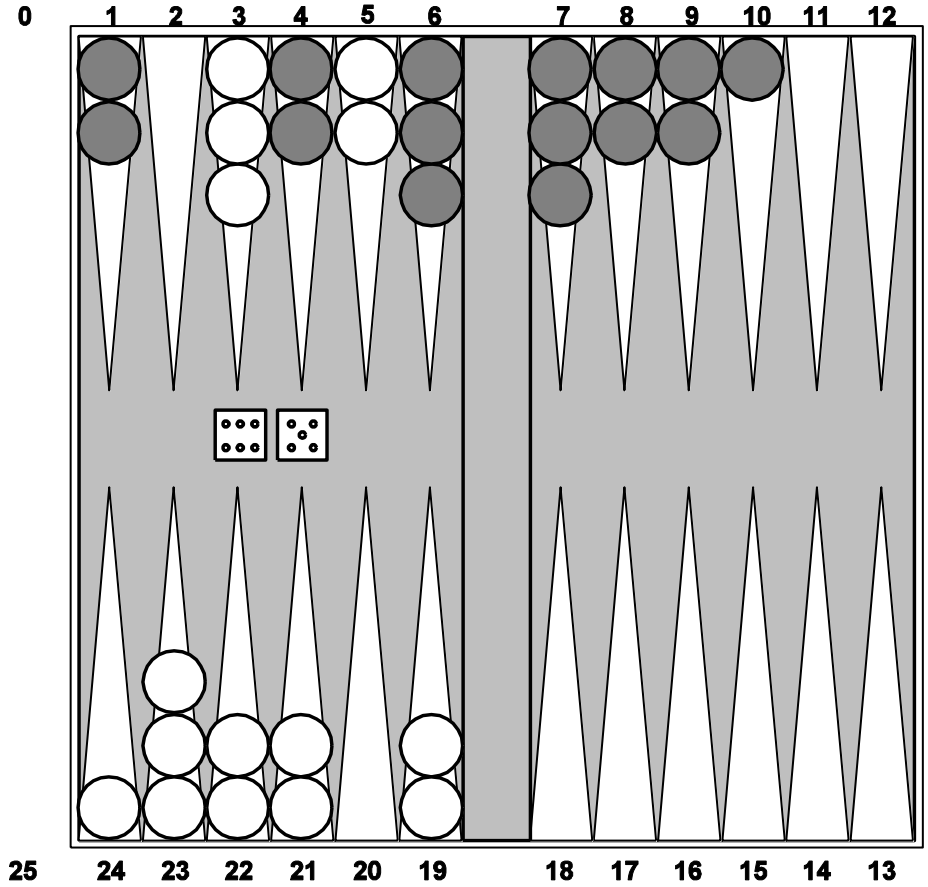
- Solução: Procura Tranquila (*quiescence search*):
  - **Ideia:** evitar avaliação em situações a partir das quais pode haver mudanças bruscas
    - No caso do jogo da velha, toda posição é tranquila mas no xadrez não (ex. um peça de xadrez a ser comida)
  - **Algoritmo:** Se a situação (nó) é “tranquila”, então aplica a função de avaliação, senão busca até encontrar uma situação “tranquila”

E aí: é útil mesmo?

- Ainda pode ser complexo...
  - $b^m = 10^6$, b=35 → m=4
- Olhar para frente 4-ply (quatro lances) é pouco para um nível profissional no xadrez!
  - 4-ply ≈ novato humano
  - 8-ply ≈ PC típico, mestre humano
  - 12-ply ≈ Deep Blue, Kasparov

Jogos com elementos de acaso

- Há jogos com elementos de imprevisibilidade
  - Ex. gamão: não sabemos quais são as jogadas legais do adversário.

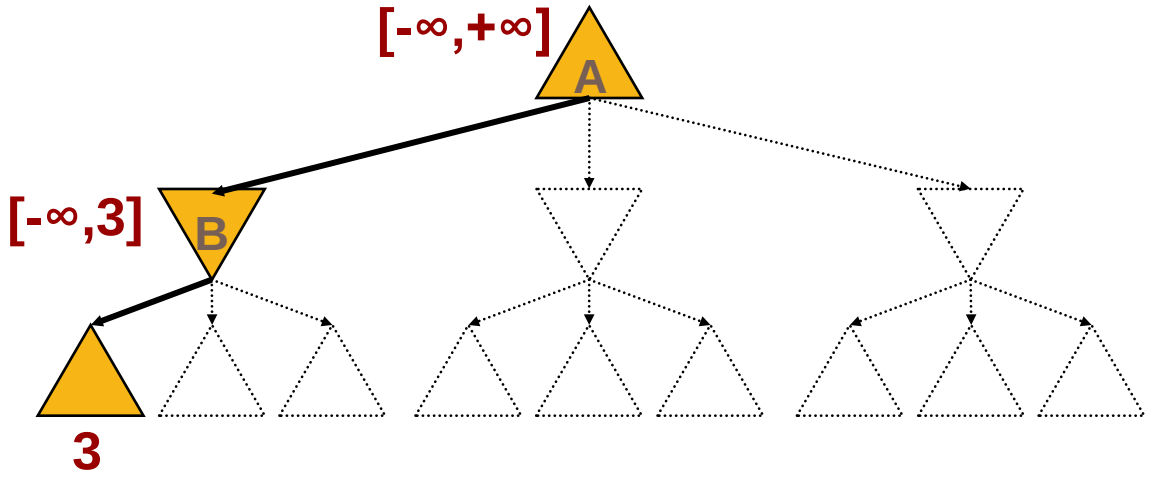


Poda alfa-beta

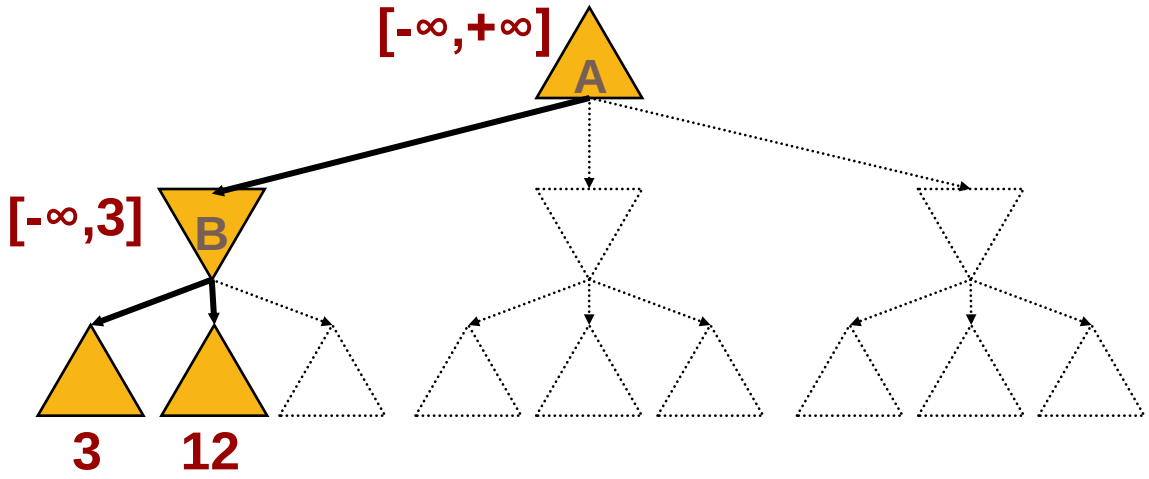
Alpha-Beta Pruning (poda alfa-beta)

- **Função:**
  - Não expandir desnecessariamente nós durante o minimax (mas devolvendo o mesmo resultado)
- **Ideia:** não vale a pena piorar, se já achou algo melhor
- **Mantém 2 parâmetros**
  - α - melhor valor (mais alto) encontrado até então para MAX
  - β - melhor valor (mais baixo) encontrado até então para MIN
- **Teste de expansão:**
  - α não pode diminuir (não pode ser menor que um ancestral)
  - β não pode aumentar (não pode ser maior que um ancestral)

Lembrar que min-max faz busca em profundidade

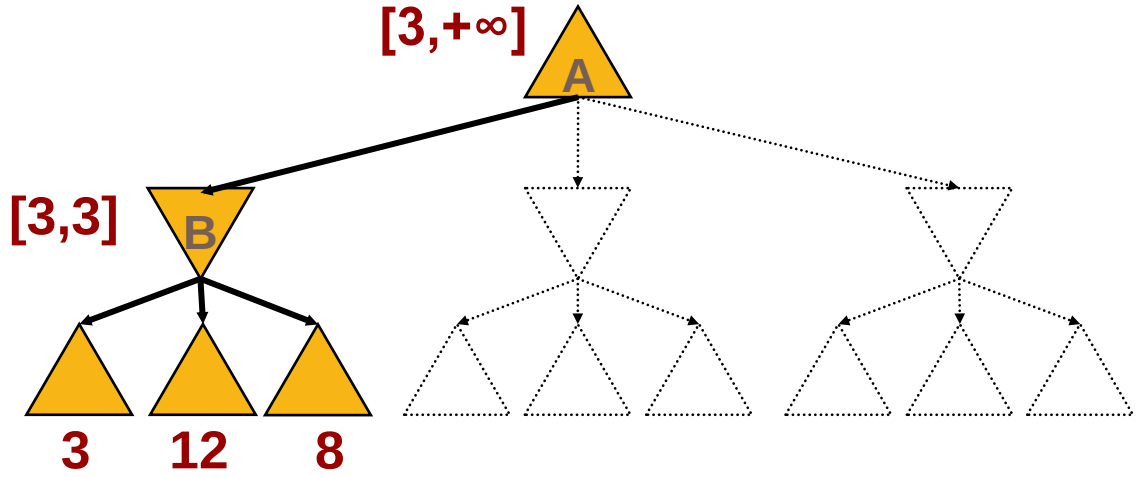


O melhor para MIN é 3 e para MAX é +∞

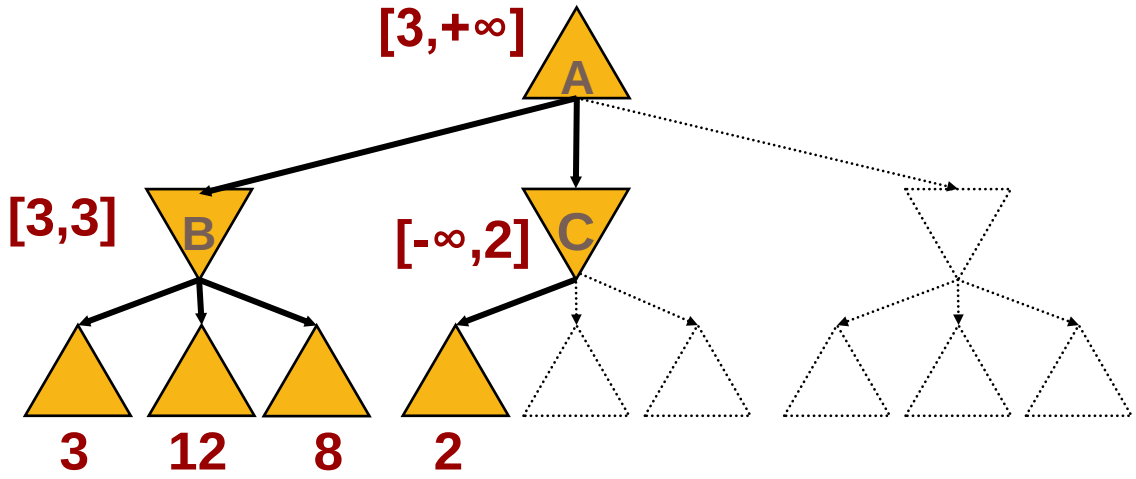


Nada mudou...

Alfa-Beta

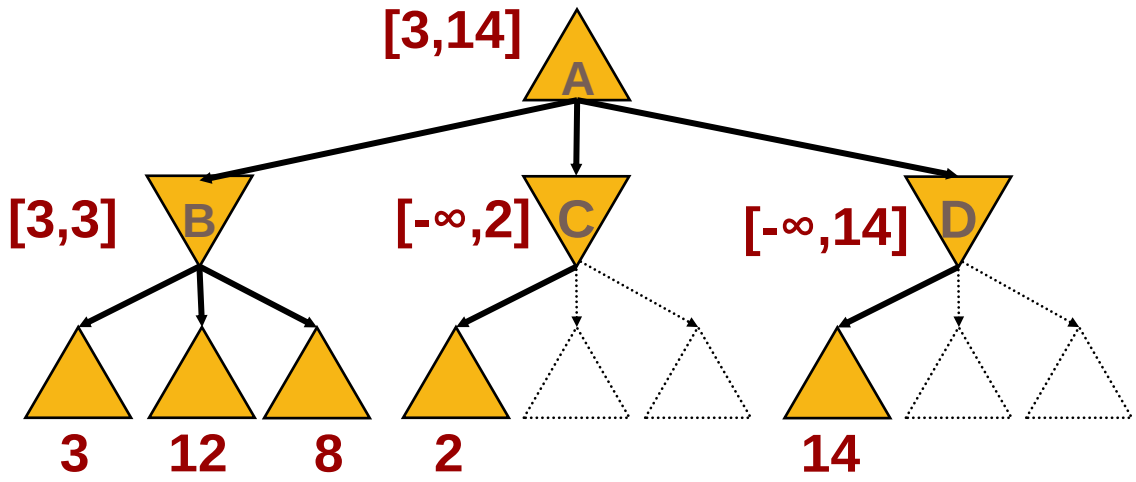


Agora dá para saber que MIN vai escolher no máx. 3

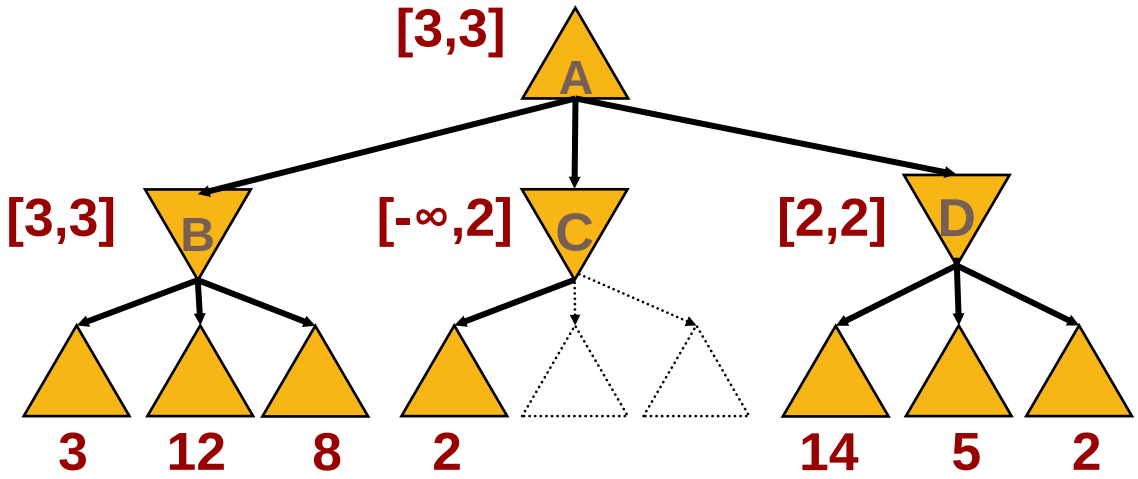


Não vale mais a pena para MAX explorar C, porque MIN vai escolher no máx. 2 e MAX já tem 3

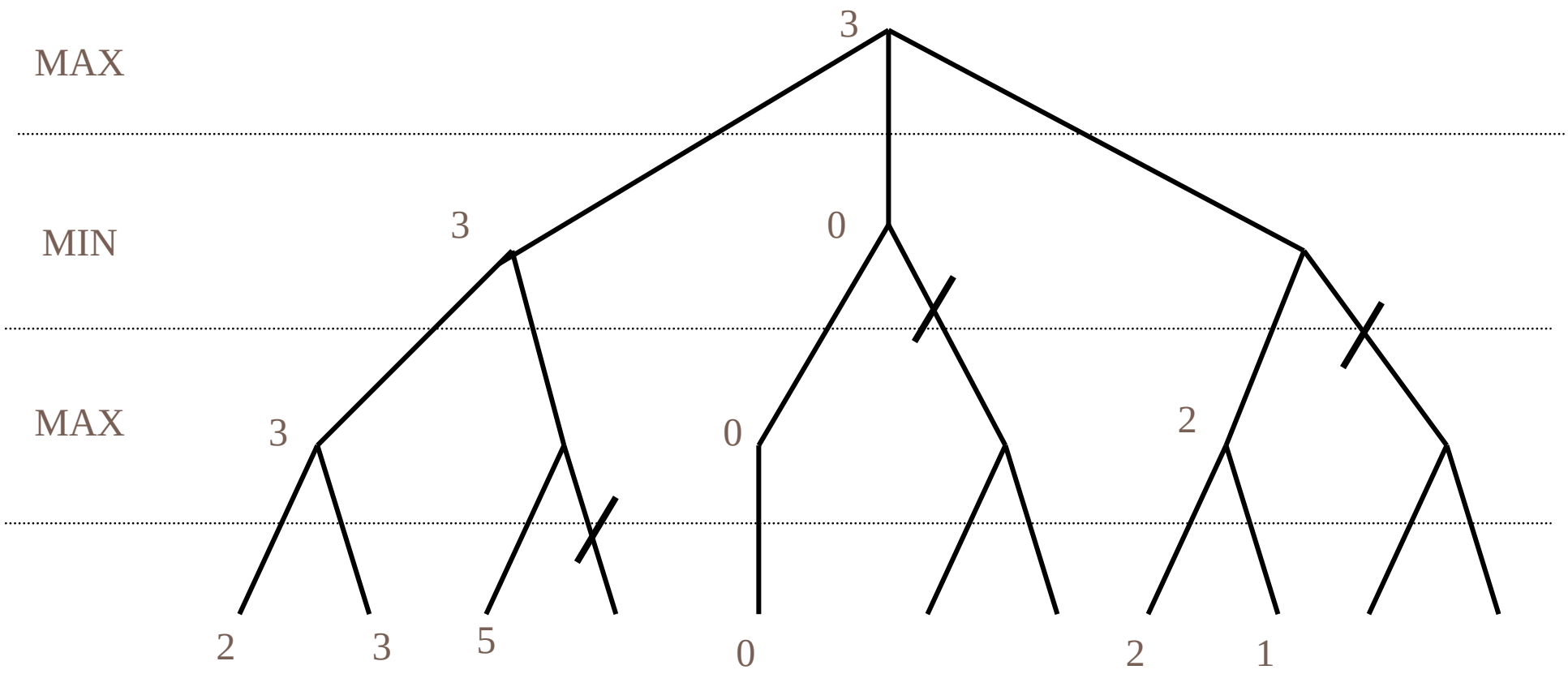
Alfa-Beta



Realisticamante, 14 é o melhor para max por enquanto



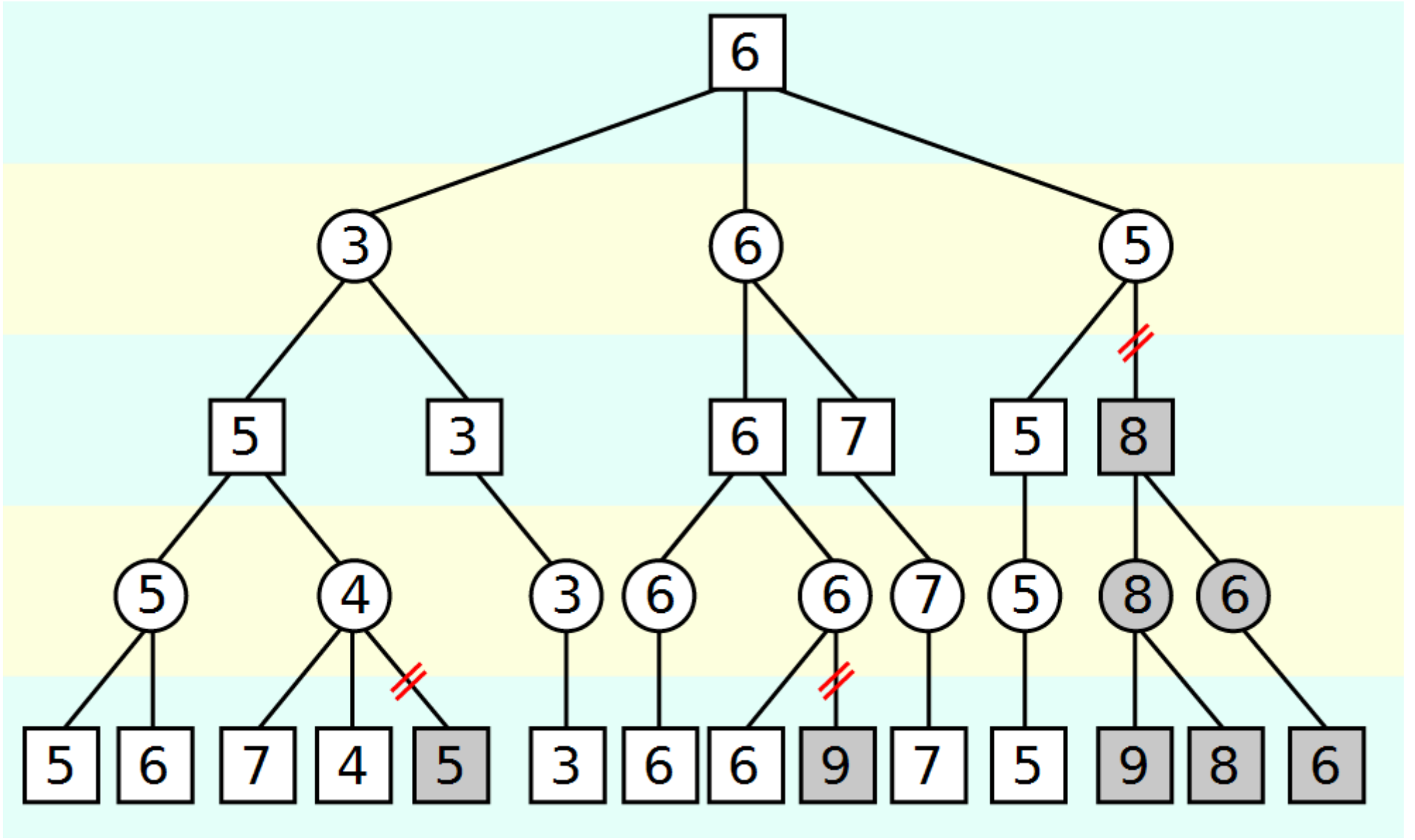
Alpha-Beta Pruning: outro exemplo



Links

http://www.youtube.com/watch?v=6tFqd15YjM0

- Aplicado ao jogo Otello
  - http://www.youtube.com/watch?v=bYMA0y2j9ZM
  - http://www.youtube.com/watch?v=gs6bS25hBsA


6
3
6
5
5
3
6
7
5
8
5
4
3
6
6
7
5
8
6
5
6
7
4
5
3
6
6
9
7
5
9
8
6

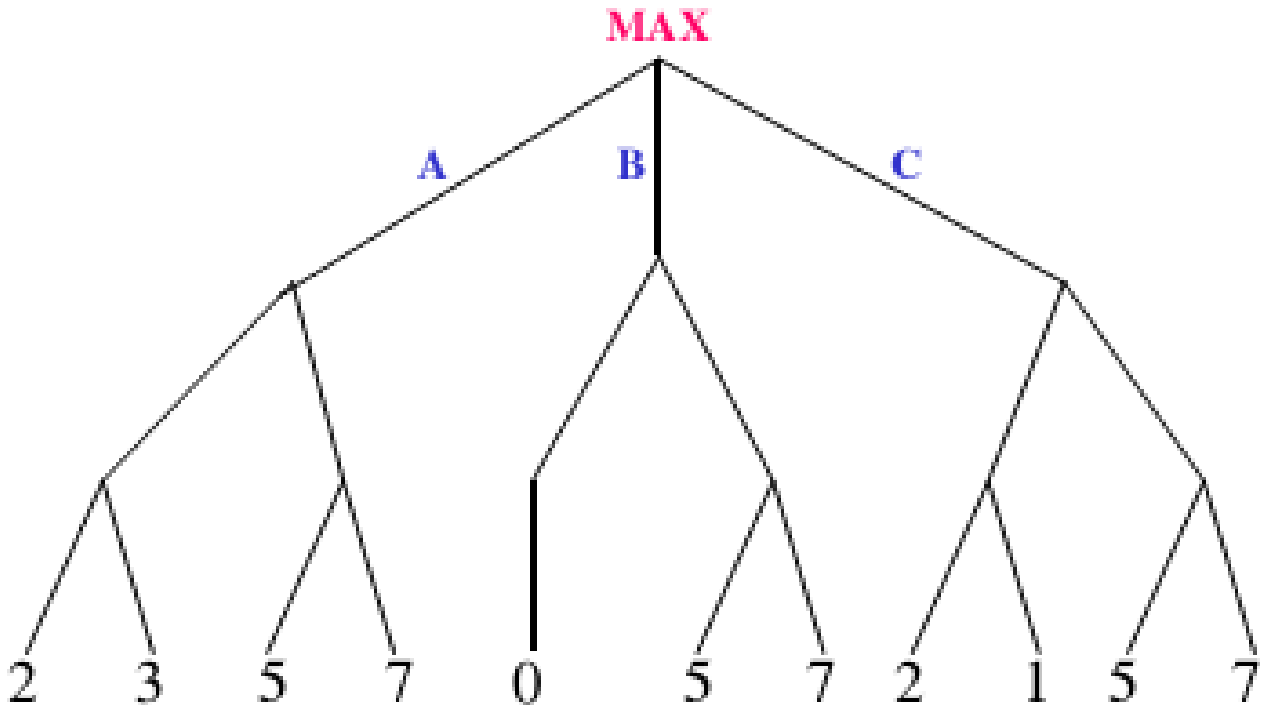
Conclusões

- Poda não afeta o resultado final
- Bom exemplo de raciocínio sobre a relevância de se calcular coisas (forma de meta-raciocínio)
- A perfeição é impossível => é preciso aproximar

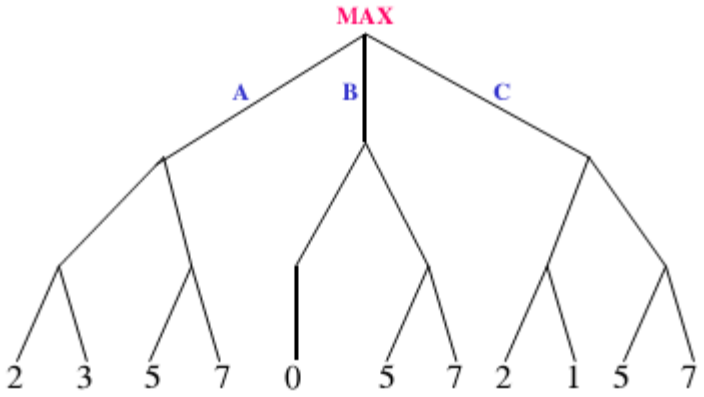
Atividades

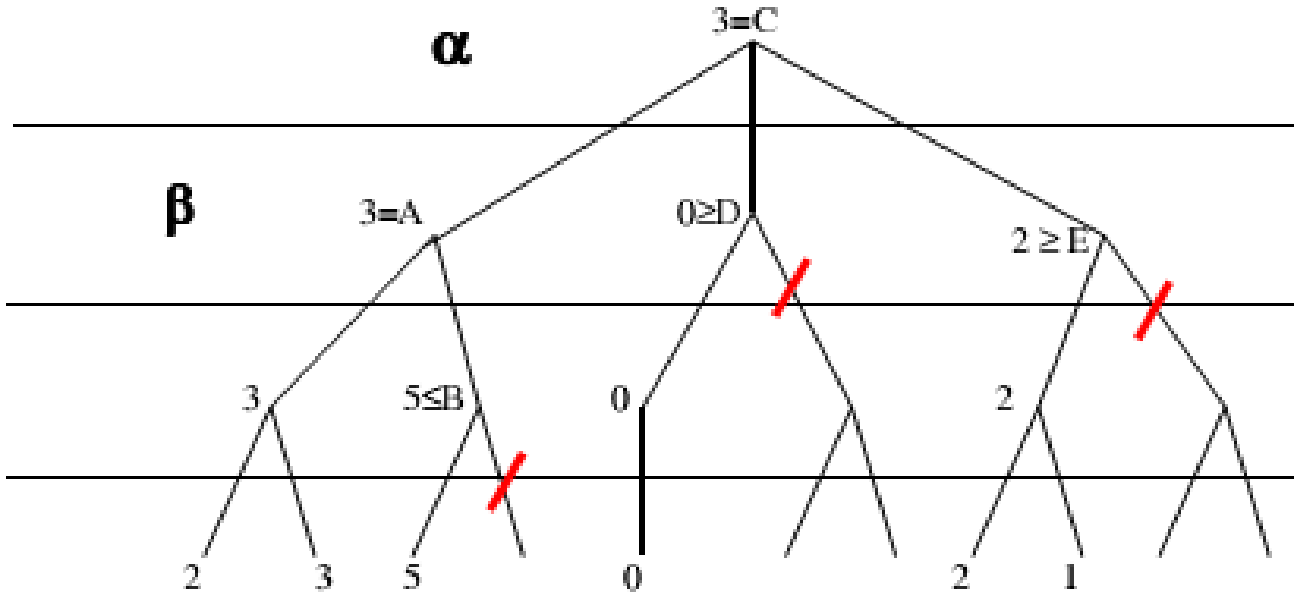
Exercício

- Decidir a jogada de MAX (A, B ou C) considerando as utilidades fornecidas nas folhas.



Exercício

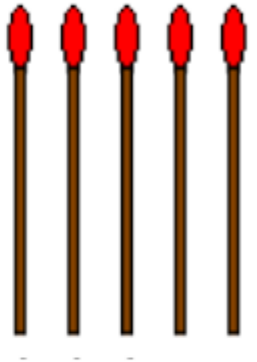




A tem β=3; B será podado por β, já que 5>3
D é podado por α, já que 0<3
E é podado por α, já que 2<3
C é 3.

Exercício – jogo dos 5 palitos

- **Objetivo**: de um conjunto de 5 palitos, pegar 1 ou 2 palitos e não ser o último a jogar;
  - **Cenário 1:**
    - Jogador 1: Retira 2 palitos
    - Jogador 2: Retira 2 palitos
    - Jogador 1: Retira o último (perde)
  - **Cenário 2:**
    - Jogador 1: Retira 1 palito
    - Jogador 2: Retira 2 palitos
    - Jogador 1: Retira 1 palito
    - Jogador 2: Retira o último (perde)



- *Exercício: Construir a árvore minimax com os payoffs indicados nos nodos*

Exercício – jogo dos 5 palitos