CC7711

Inteligência Artificial e Robótica

Prof. Dr. Flavio Tonidandel

Mecanismos de Busca – parte 2

CC7711 - Inteligência Artificial e Robótica

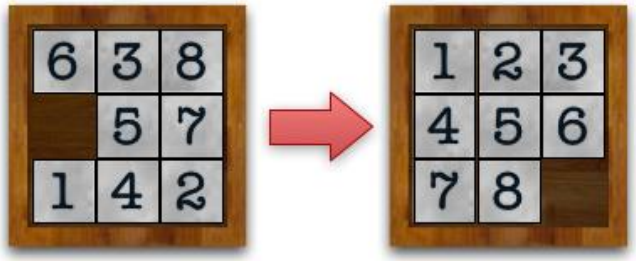
Heurística Admissível

- Considere:
- h(n) → heurística (estimativa)
- h*(n) → custo real do objetivo (meta) a partir de *n*

- heurística admissível:
  - *h não deve superestimar o custo real h* para se alcançar uma solução*.
  - Ou seja: deve ser otimista → $h(n) \leq h^*(n)$

- Caso contrário a heurística será não-admissível

Heurísticas Admissíveis

Estado Inicial  Objetivo (meta)

- **Heurística 1: Número de pastilhas que estão fora de lugar: h = 7 (não inclui espaço)**
- **Heurística 2: Distância de Manhattan: distância horizontal + vertical de cada pastilha para o seu lugar correto: h = 15 (não inclui espaço)**

As duas heurísticas são admissíveis para o problema do jogo dos oito.

Se incluir espaço na heurística, ela não será admissível
h* = máximo de movimentos = 3
h = 4 (numero de pastilhas fora do lugar incluindo o espaço)

Para se encontrar uma boa heurística, geralmente *relaxa-se* o problema

Heurística Admissível

- **Qual é melhor, h1 ou h2** ?
- É sempre melhor usar uma função heurística com valores mais altos, contanto que ela seja *admissível.*
  - ex. *h2* melhor que *h1*

- h_i <u>domina</u> $h_k \Rightarrow h_i(n) \geq h_k(n) \; \forall n$ no espaço de estados
  - h_2 domina h_1 no exemplo anterior

- Caso existam muitas funções heurísticas para o mesmo problema, e nenhuma delas domine as outras, usa-se uma *heurística composta*:
  - $h(n) = max\,(h_1(n), h_2(n),\ldots,h_m(n))$
  - Assim definida, *h* é *admissível* e *domina* cada função h_i individualmente

Qualidade da Heurística

- Qualidade da função heurística:
  medida através do fator de expansão efetivo (*b).**
  - *b** é o fator de expansão de uma árvore uniforme com *N* nós e nível de profundidade *d*
  - $N = 1 + b^* + (b^*)^2 + \ldots + (b^*)^d$, onde
    - N = total de nós expandidos para uma instância de problema
    - d = profundidade da solução;
- Mede-se empiricamente a qualidade de *h* a partir do conjunto de valores experimentais de *N* e *d*.
  - uma boa função heurística terá o *b** muito próximo de 1.
- Se o custo de execução da função heurística for maior do que expandir nós, então ela *não* deve ser usada.
  - uma boa função heurística deve ser *eficiente* e *econômica*, no que se refere ao custo computacional

Função de Avaliação (incluindo custo nos ramos)

- Além da heurística, pode-se ter o custo de cada ramo da árvore de busca.
  Exemplo:
  - Suponha que o seu sistema de IA deva lidar com:
    - Planejamento de rotas (ir de um ponto a outro na cidade)
    - Cada rota gasta combustível de forma diferente
    - Quero, portanto, atingir meu objetivo (heurística) e minimizar o gasto de combustível (custo)

- Função de avaliação:
  - $f(n) = g(n) + h(n)$
  - $g(n)$ = função de custo de n ao nó inicial
  - $h(n)$ = função heurística
- Se f(n) = h(n), então a busca, guiada pela função de avaliação,

Busca A*

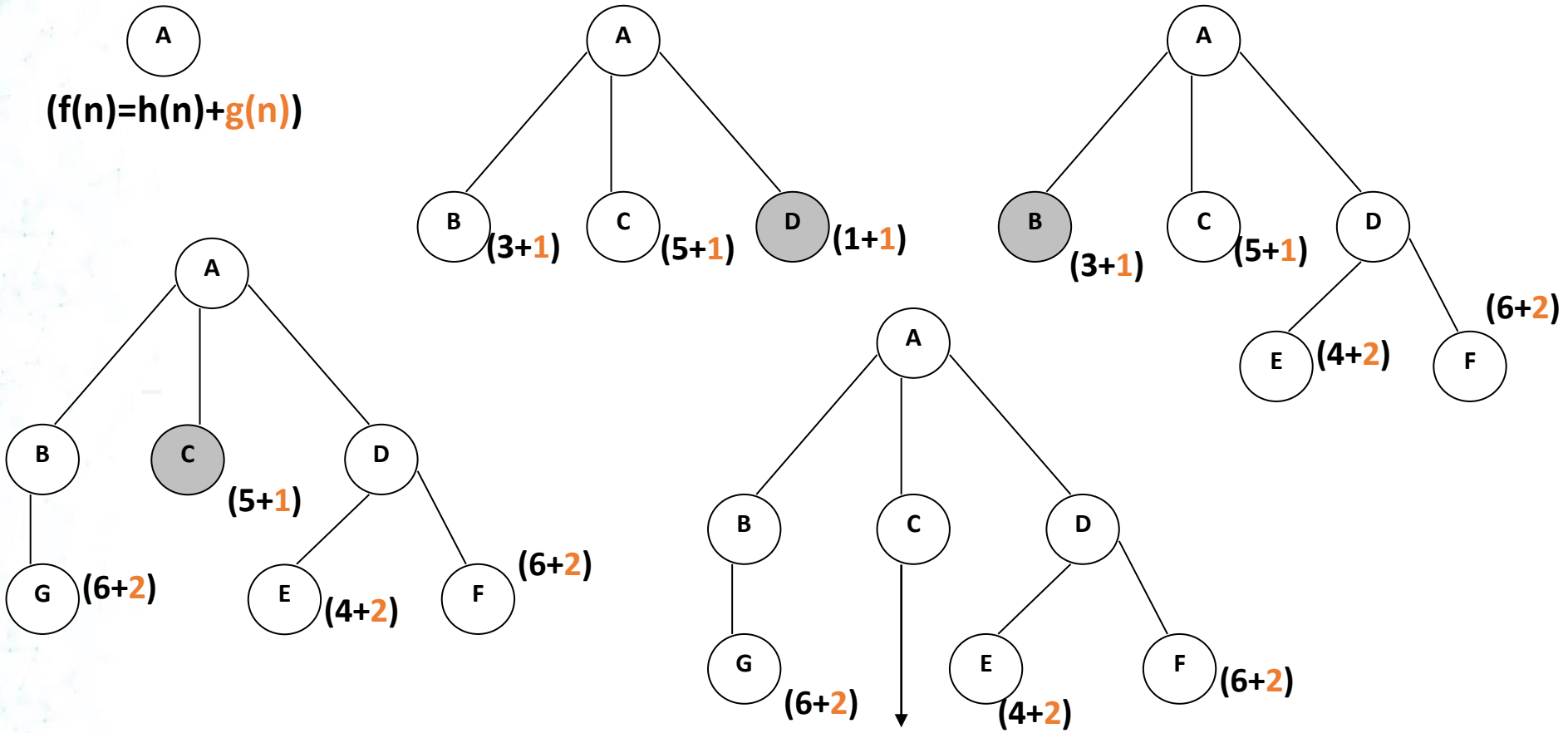
- A busca A* é a busca Melhor-Escolha guiada pela função de avaliação
  - Ou seja, a diferença entre A* e Melhor-Escolha é que o A* usa o custo de cada ramo

- A* expande o nó de menor valor de *f* na fronteira do espaço de estados.

- Interessante:
  - Se *h* é *admissível*, *f (n)* nunca irá superestimar o custo real da melhor solução através de *n.* Logo, o A* encontrará a solução ótima em termos de custo.

Buscas A*

- Exemplo considerando cada ramo custo=1



Análise Busca A*

Semelhante à busca Melhor-Escolha porém levando em consideração o custo de cada ramo

É completa ? Sim.

É ótima ? Sim, com relação ao custo, i.e., minimiza o custo quando a heurística é *admissível*.

Fator de Ramificação: no pior caso todos os nós gerados a partir de um nó (b)

Custo de Tempo: no pior caso $O(b^d)$.

Custo de Memória: no pior caso $O(b^d)$.

Embora seja um algoritmo caro computacionalmente, é o melhor algoritmo para encontrar soluções ótimas com relação ao custo.

Comparação de Heurísticas

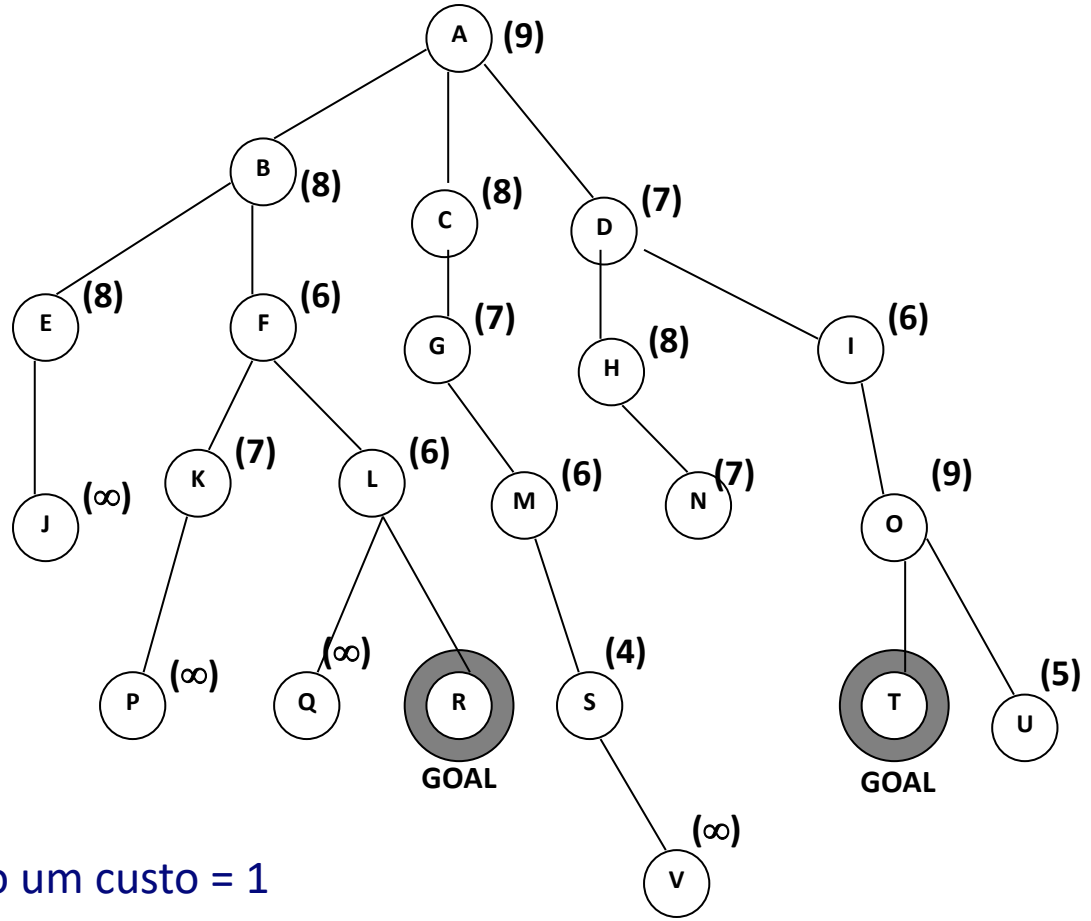
Resolvendo o problema do jogo-dos-oito usando A* e duas heurísticas:

h1: Número de pastilhas fora de lugar e h2: Distância de Manhattan

Profundidade da Solução	Número médio de nós expandidos		Fator de ramificação efetiva b*	
d	A*(h1)	A*(h2)	A*(h1)	A*(h2)
2	6	6	1,79	1,79
4	13	12	1,48	1,45
6	20	18	1,34	1,30
8	39	25	1,33	1,24
10	93	39	1,38	1,22
12	227	73	1,42	1,24
14	539	113	1,44	1,23
16	1301	211	1,45	1,25
18	3056	363	1,46	1,26
20	7276	676	1,47	1,27
22	18094	1219	1,48	1,28
24	39135	1641	1,48	1,26

Exercício

Para a seguinte árvore:

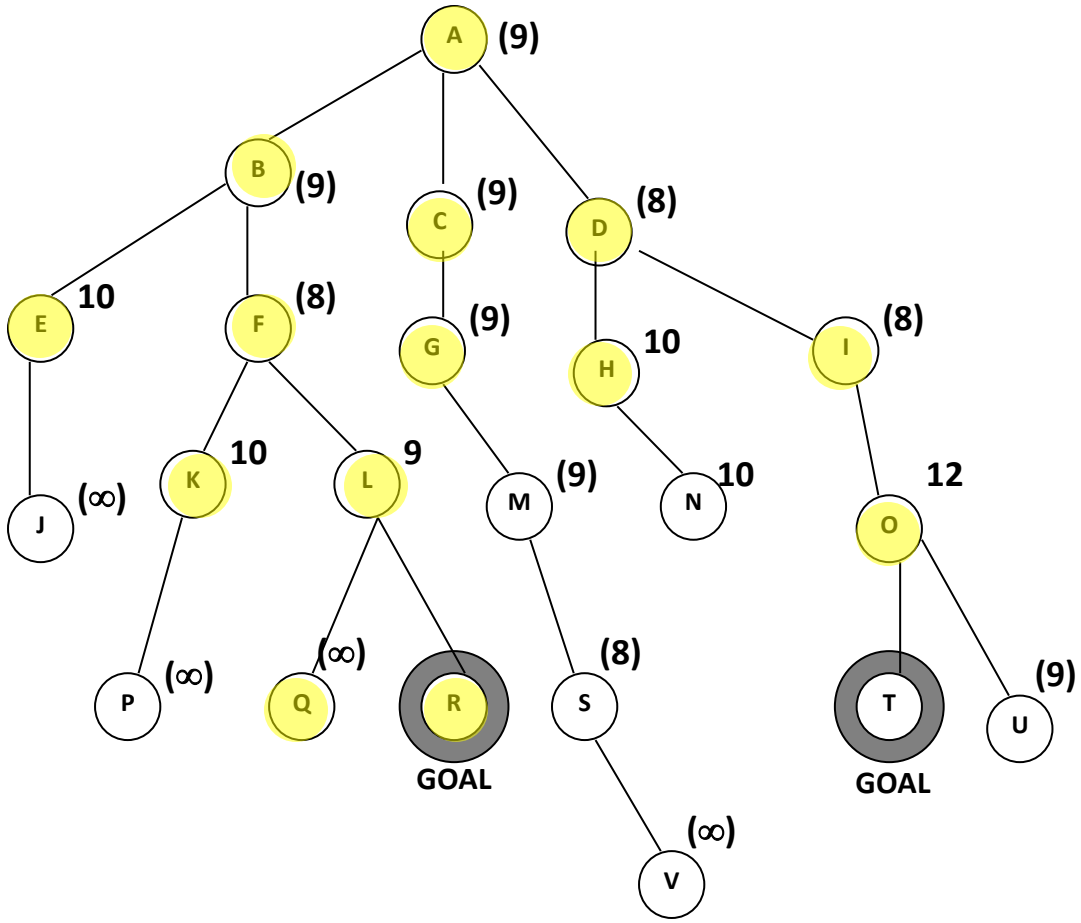


Ache a sequência de nós:

- A*
- Considere cada ramo um custo = 1

Resposta Exercício:

- Sequência de Expansão de Nós para:
- Resposta: A*



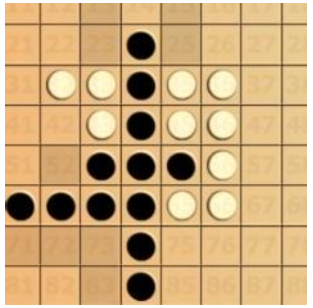
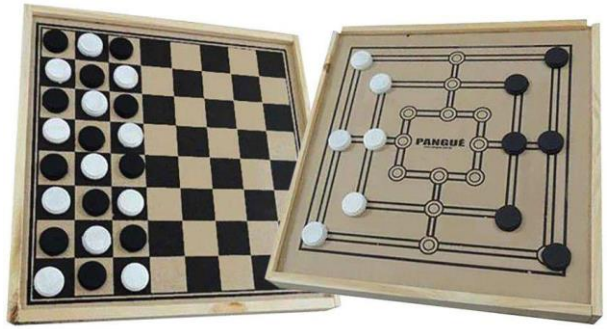
- Sequência Gerada:

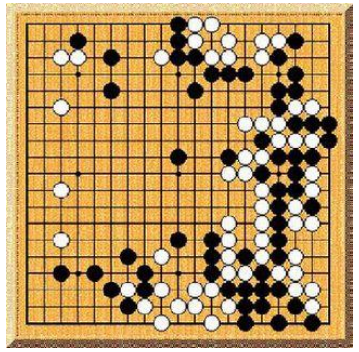
ABCDHIOEFKLGQR

Buscas para Jogos

Jogos Adversariais

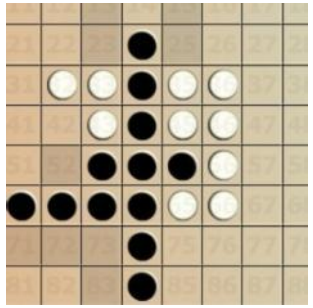
- Presença do Oponente
  - introduz incerteza
  - estado é alterado não somente em função das decisões tomadas - problema da contingência
- Complexidade da Árvore de Busca
  - número de estados a serem testados é muito maior
  - normalmente não há tempo suficiente para calcular **todas** as possíveis consequências de um movimento durante um jogo
- Questões
  - Como encontrar o suposto melhor movimento ?
  - Como agir quando o tempo de decisão é limitado ?




Histórico de Jogos em IA

- XADREZ: Venceu Kasparov em jogos com rodadas de 5 a 25 minutos; em campeonatos fica entre os 100 primeiros.
- 1970: programas complicados com truques de cortes (poda da árvore).
- Belle: primeiro computador dedicado ao jogo de xadrez.
  - Circuitos integrados implementam a geração de movimentos
- Hitech: gera 10^4 posições por jogada
  - função mais precisa de avaliação até hoje
  - Primeiro a vencer um campeão de xadrez (1987).
- Chinook: programa que joga damas. Se tornou campeão do mundo em 1994
  - Sistemas anteriores empregavam aprendizagem para descobrir a melhor função de avaliação
  - Emprega corte alfa-beta + um Banco de Dados com soluções perfeitas
- Othello: muito melhor que humanos. Variação do xadrez com espaço de busca bem menor: 5 a 15 movimentos possíveis.
- 2007: Jogos de Damas totalmente mapeados.



Algoritmo MIN-MAX

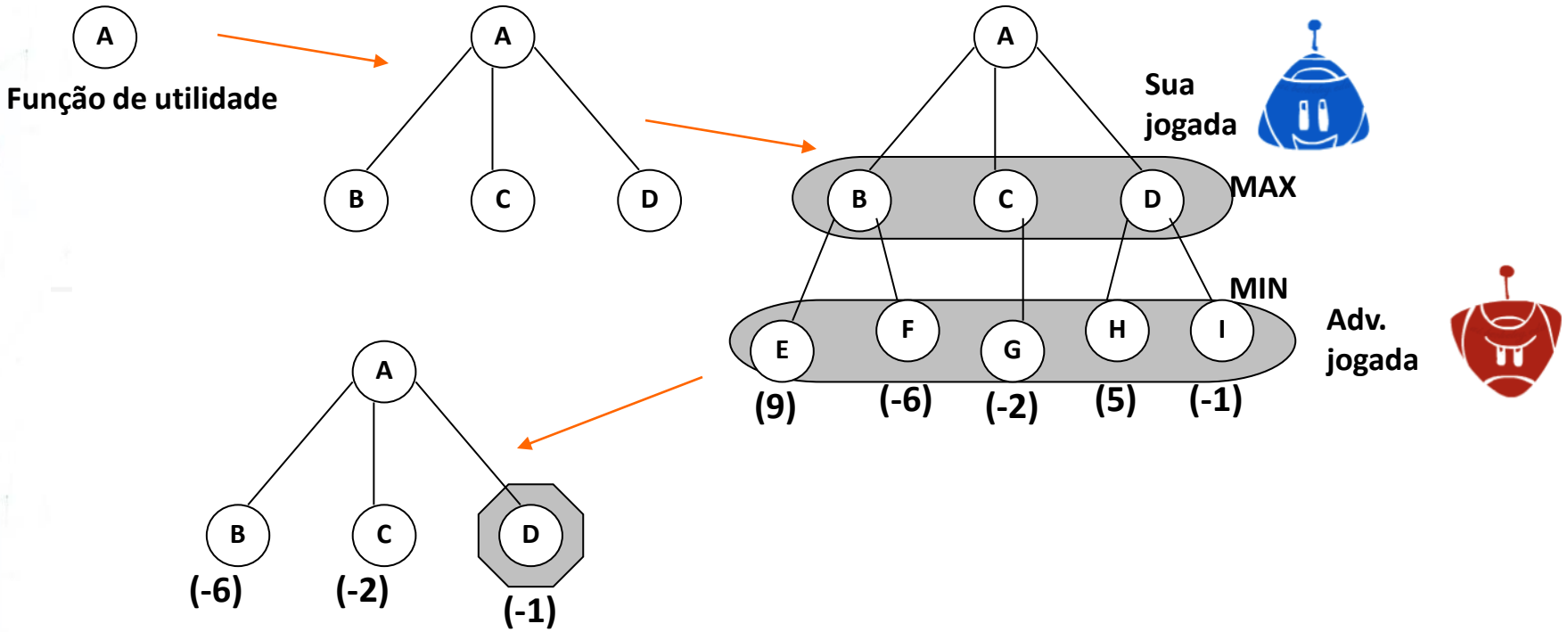
- **MAXimizar sua jogada**
- **MINimizar a jogada do adversário**

- **Algoritmo MINMAX (características):**
  - Necessita de uma função de utilidade;
  - Escolher um passo que maximize sua chance e diminua a chance do adversário;
  - **Considere que o adversário é muito inteligente** → irá escolher a melhor opção para ele e, consequentemente, pior para você;
  - **O adversário não comete erros** por distração e não é afetado psicologicamente (superestima o adversário);

Algoritmo MIN-MAX

Na jogada do seu adversário (MIN) ele irá escolher a pior situação para você. Na sua jogada (MAX) você escolhe a melhor.

Exemplo:



Robot pictures by CS 188- Univ of California - Berkeley

Tic-Tac-Toe Game Tree

MAX (X)

MIN (O)

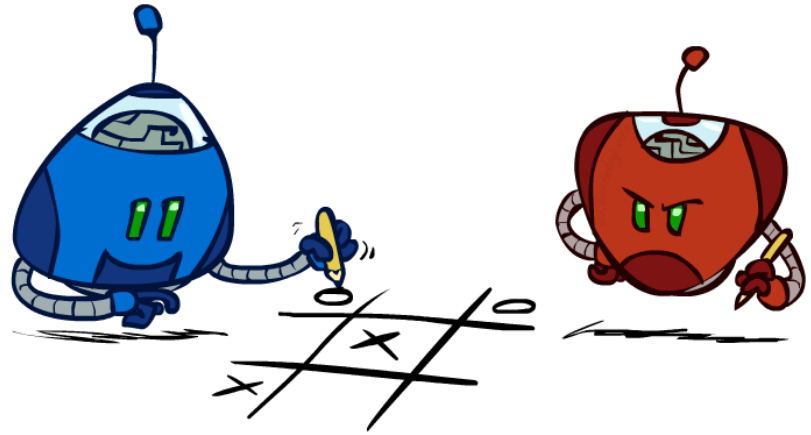
MAX (X)

MIN (O)

TERMINAL

Utility

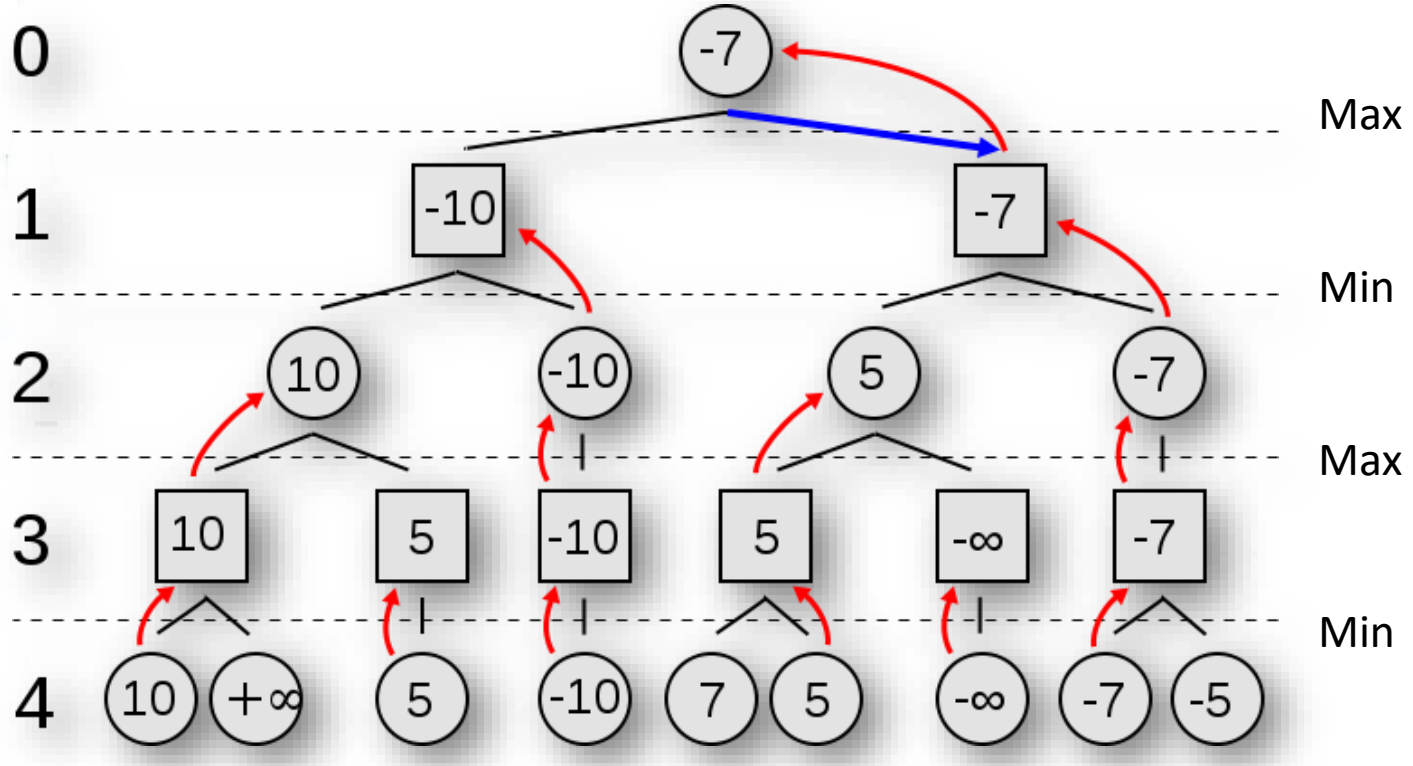
−1 0 +1



Robot pictures by CS 188- Univ of California - Berkeley

Esquema Geral do Min-Max

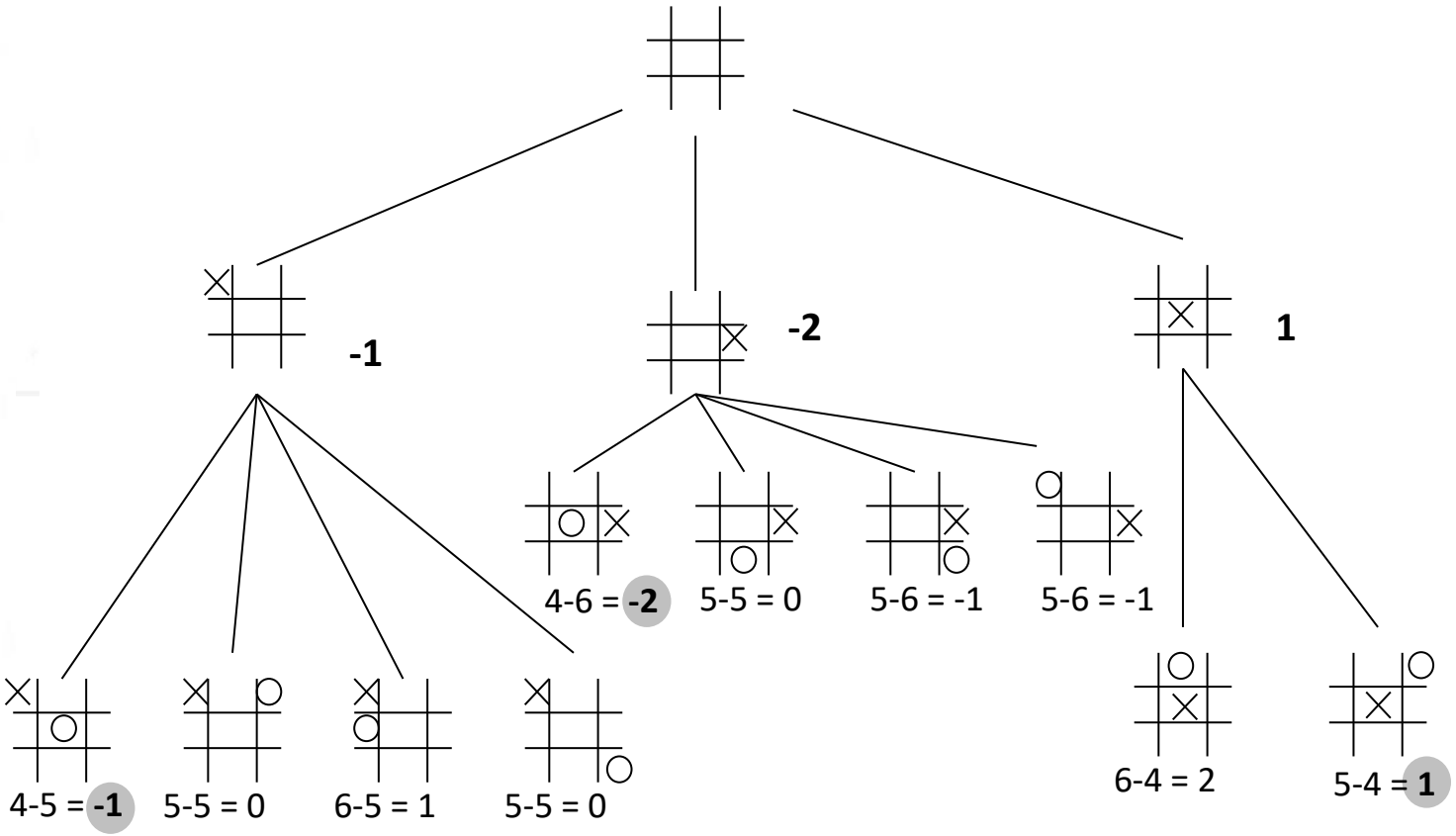
- Calcula a função utilidade do último nível de análise e depois vai “subindo” os Valores MIN ou MAX (depende de quem é a jogada)



Algoritmo MIN-MAX : Jogo da Velha

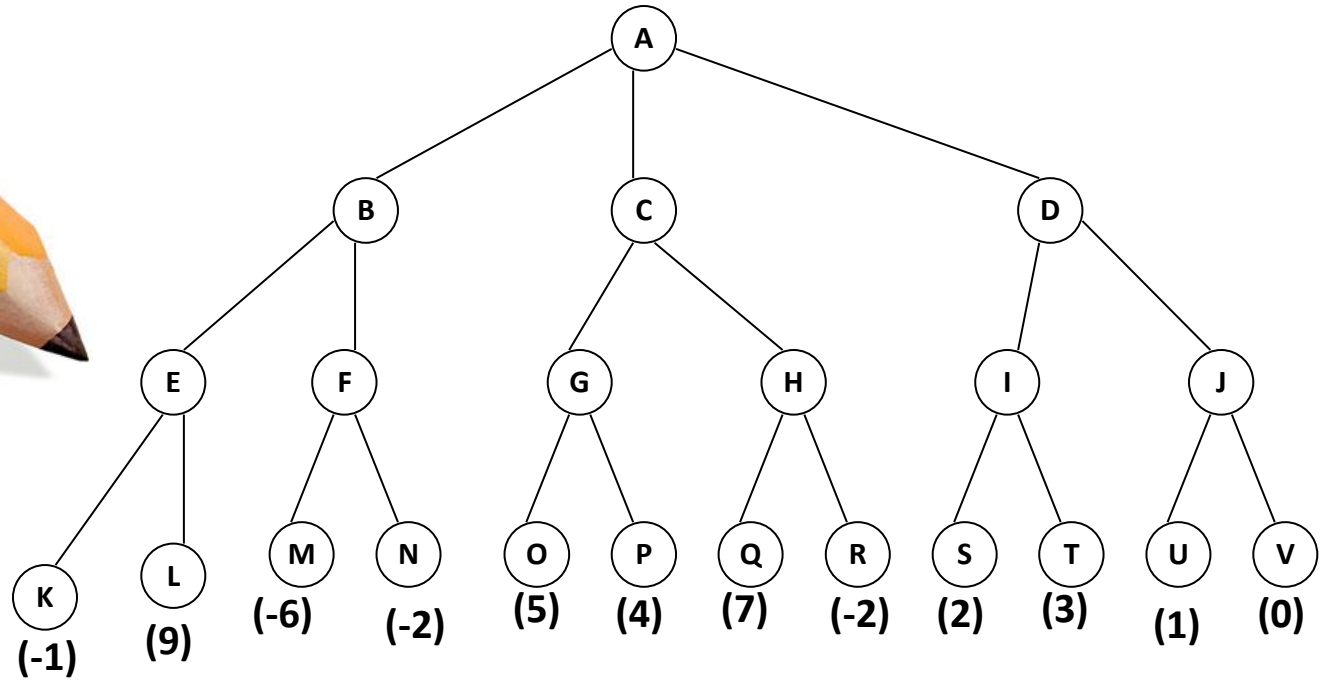
Função de utilidade (F):

F = número de chances para X ganhar - número de chances para O



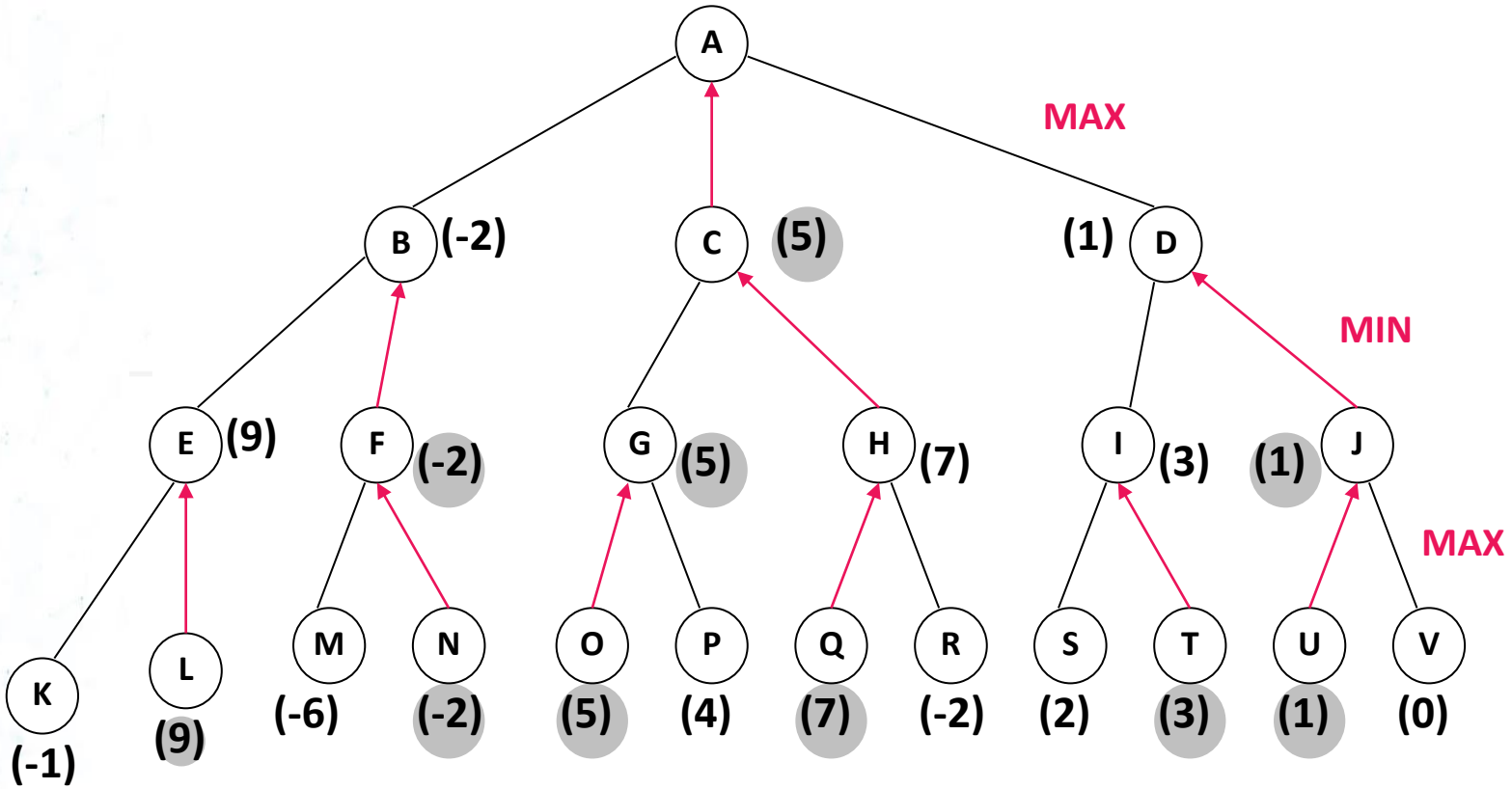
Exercício MINMAX

Qual a melhor jogada, B, C ou D ?



Exercício MINMAX

Solução:

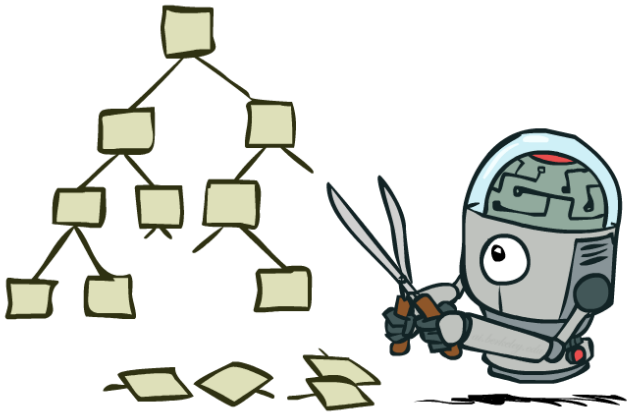


Melhorando o MINMAX

- **Cortes Alfa-Beta**
  - **Diminuir o espaço de busca**
  - **Corte Alfa → Baseia-se no maior valor para a maximização** α não pode diminuir (não pode ser menor que um ancestral)
  - **Corte Beta → Baseia-se no menor valor de minimização** β não pode aumentar (não pode ser maior que um ancestral)

- ***Não precisamos percorrer toda a árvore para saber qual é a melhor opção para nossa jogada***

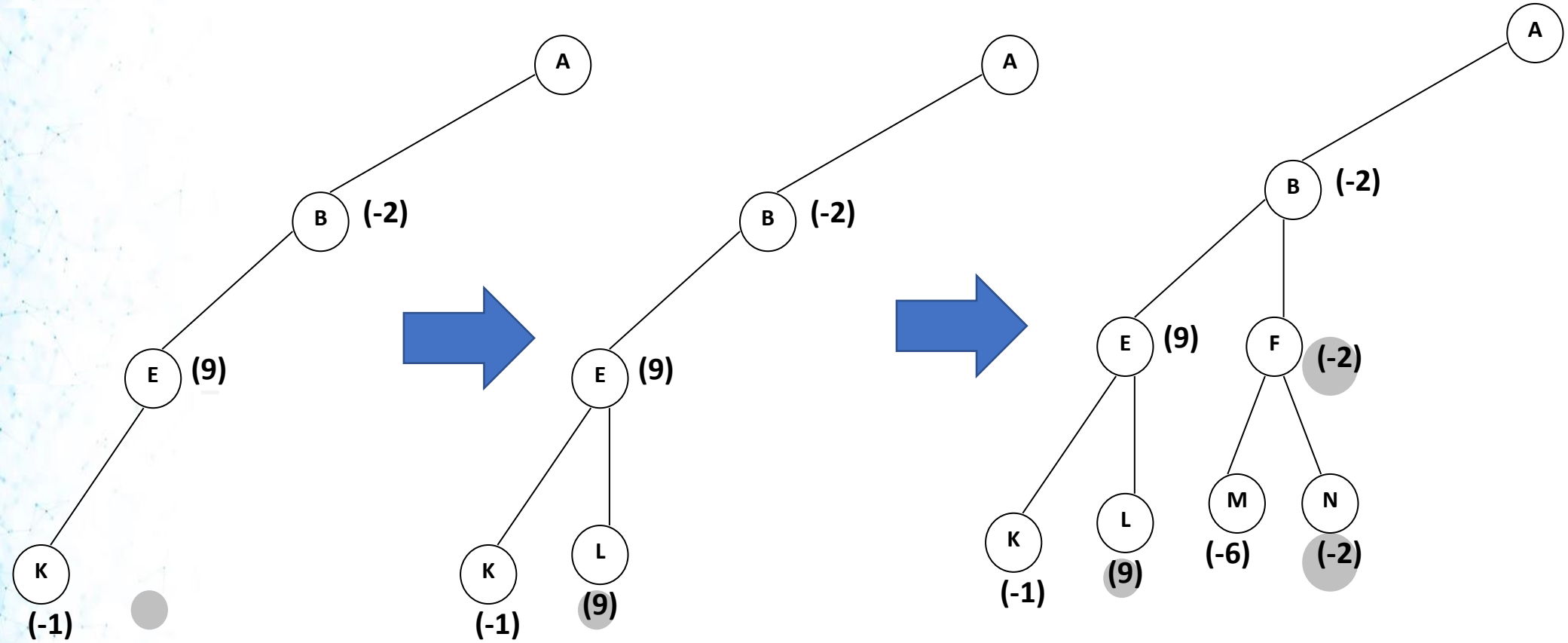
- Idéia: não vale a pena piorar, se já achou algo melhor



Robot pictures by CS 188- Univ of California - Berkeley

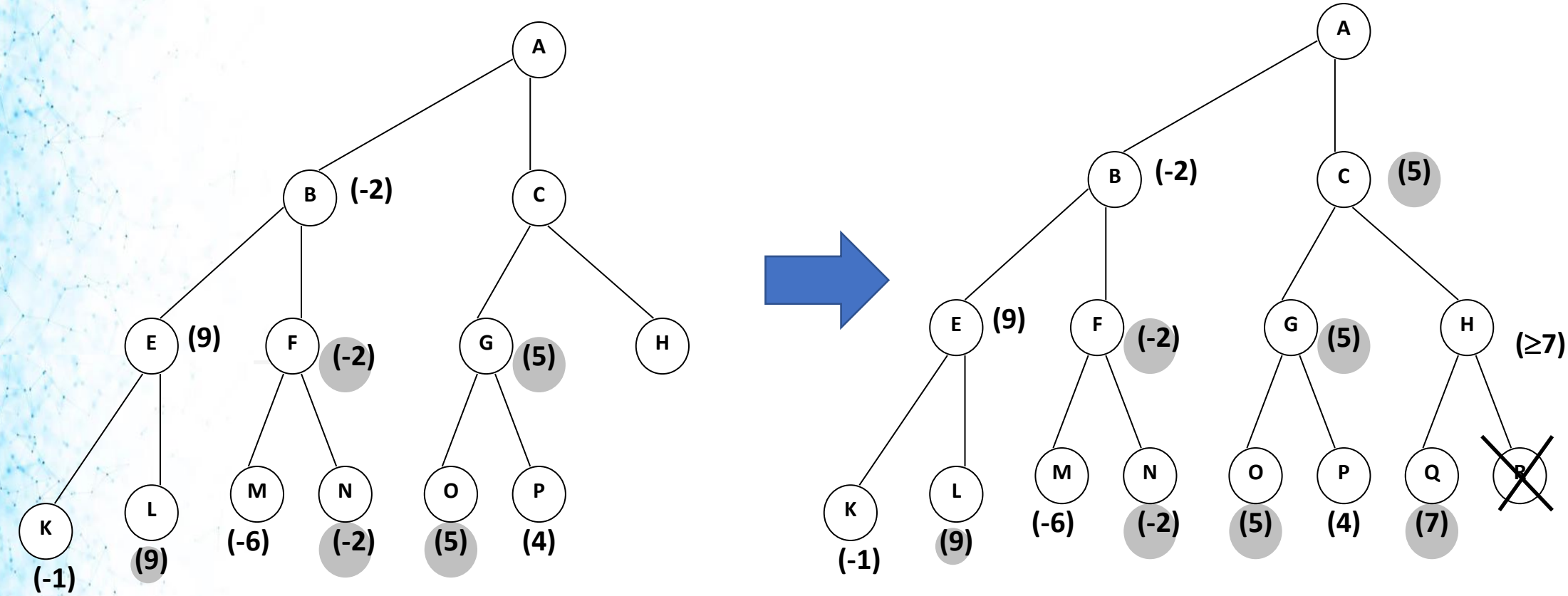
Cortes Alfa-Beta

Corte Beta



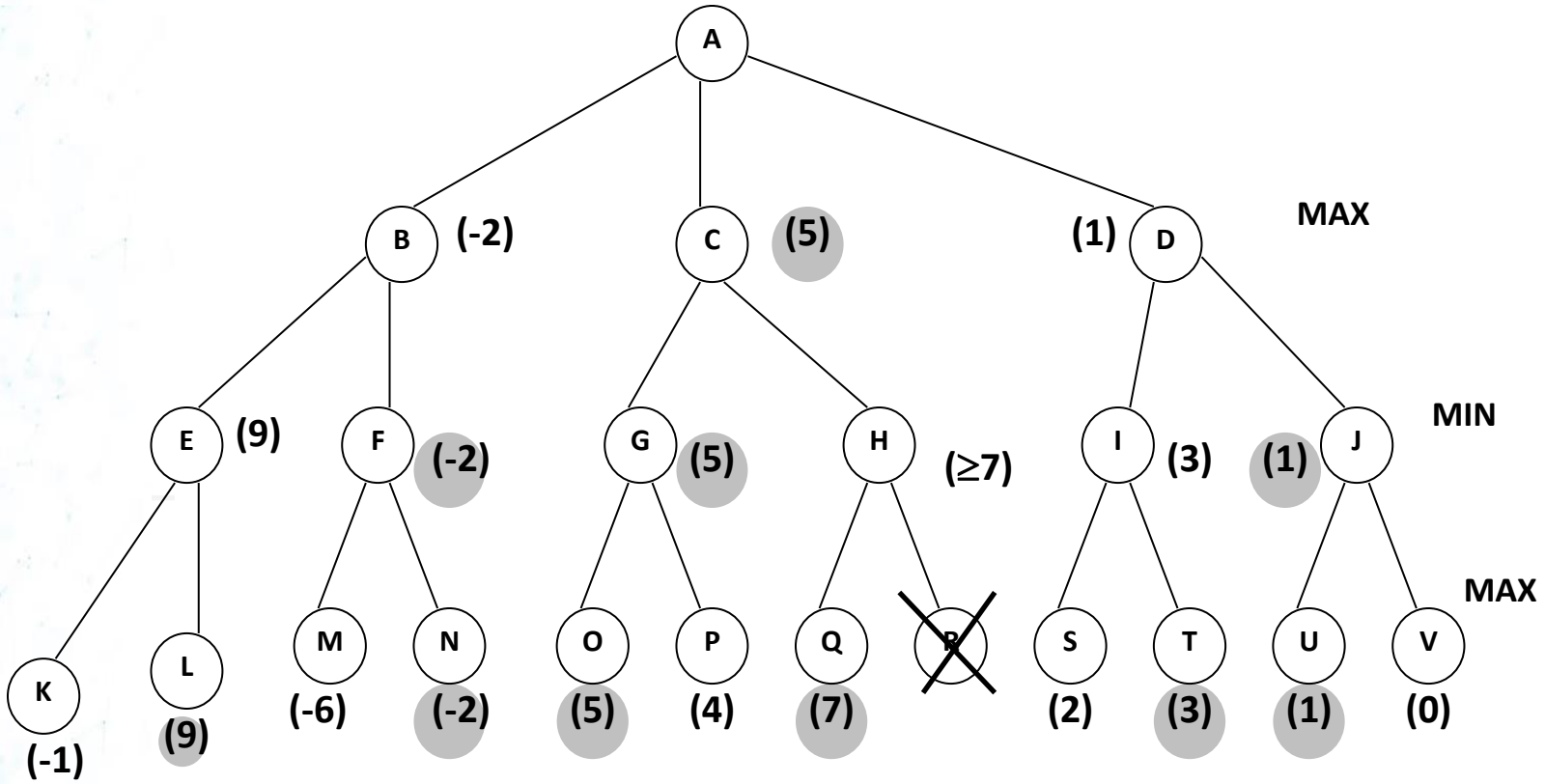
Cortes Alfa-Beta

Corte Beta



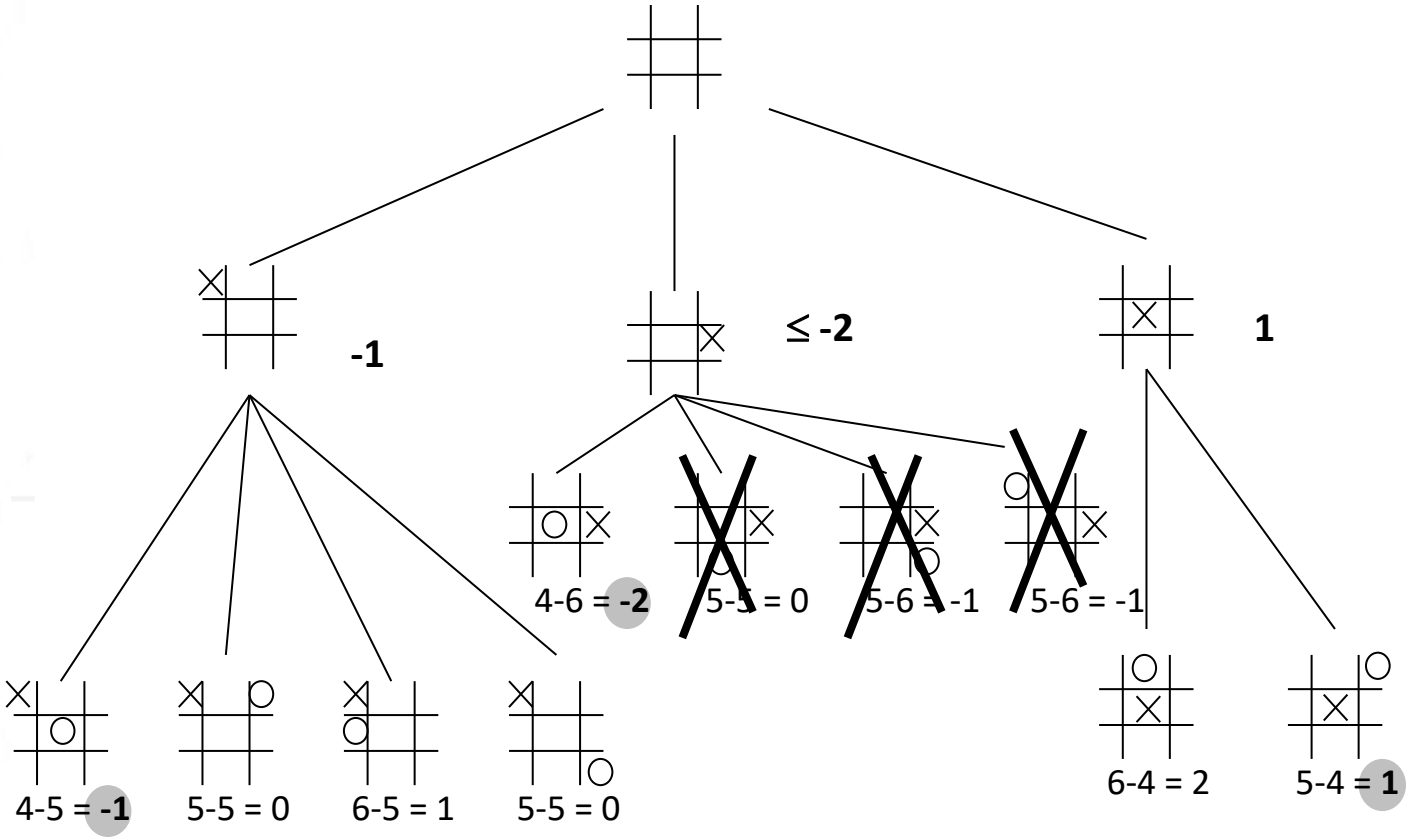
Cortes Alfa-Beta

Corte Beta



Cortes Alfa-Beta

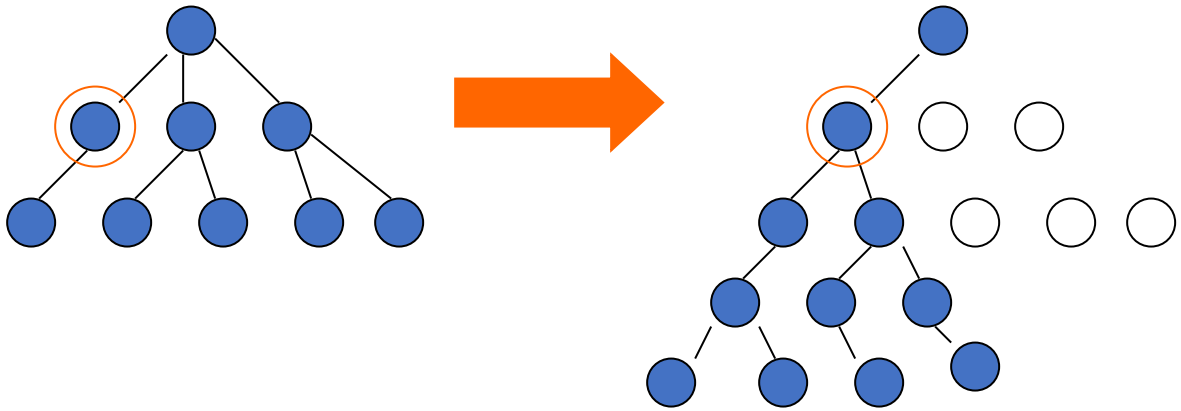
Corte Alfa



MINMAX com Busca Secundária

Podemos melhorar ainda mais o MinMax:

- Qual o grande problema do MinMax ?
  - A profundidade de busca (d) causa uma complexidade $O(b^d)$
- Como resolver o problema ?
  - Fazer uma análise com uma profundidade pequena...
  - ... E depois de escolher a jogada, fazer uma análise com uma profundidade maior apenas do ramo escolhido para verificar se a escolha é realmente boa...



Jogando Xadrez com o Min-Max

Jogo de Xadrez → b ≈ 35 (fator de ramificação)

- **Profundidade de uma solução: 50 para cada jogador (100 p/ dois)**
- **Espaço de busca da ordem de 35^{100}**

Alguns sistemas que jogam xadrez:

Deep Thought 2 (IBM + pesquisadores do CMU)

- **função simples de avaliação e poda alfa-beta**
- **10 processadores com capacidade para examinar 1/2 bilhão de posições por jogada: atinge profundidade de 10 a 11 jogadas (já chegou a profundidade 37 com casos identificados)**

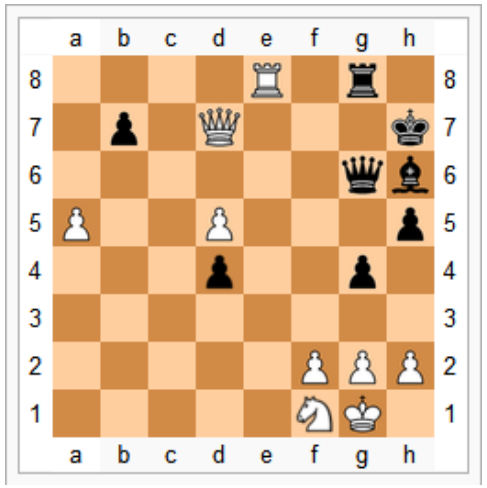
Jogando Xadrez com o Min-Max

DeepBlue ®
IBM ®

- Alguns sistemas que jogam xadrez:
- **Deep Blue**: Ganhou algumas partidas do Kasparov em 1996 e 1997 e perdeu outras.

- X3D Fritz (software alemão) empatou com Kasparov no final de 2003 numa melhor de 4 jogos. Rodou em um computador Intel Pentium 4 Xeon 2.8GHz

Final da segunda rodada entre Kasparov e X3D Fritz com vitória do X3D Fritz



Jogando Xadrez

- **Uma Função de Utilidade para Xadrez**
  - **Atribuir um valor material para cada peça:**

Peão = 1

Cavalo ou bispo = 3

Torre = 5

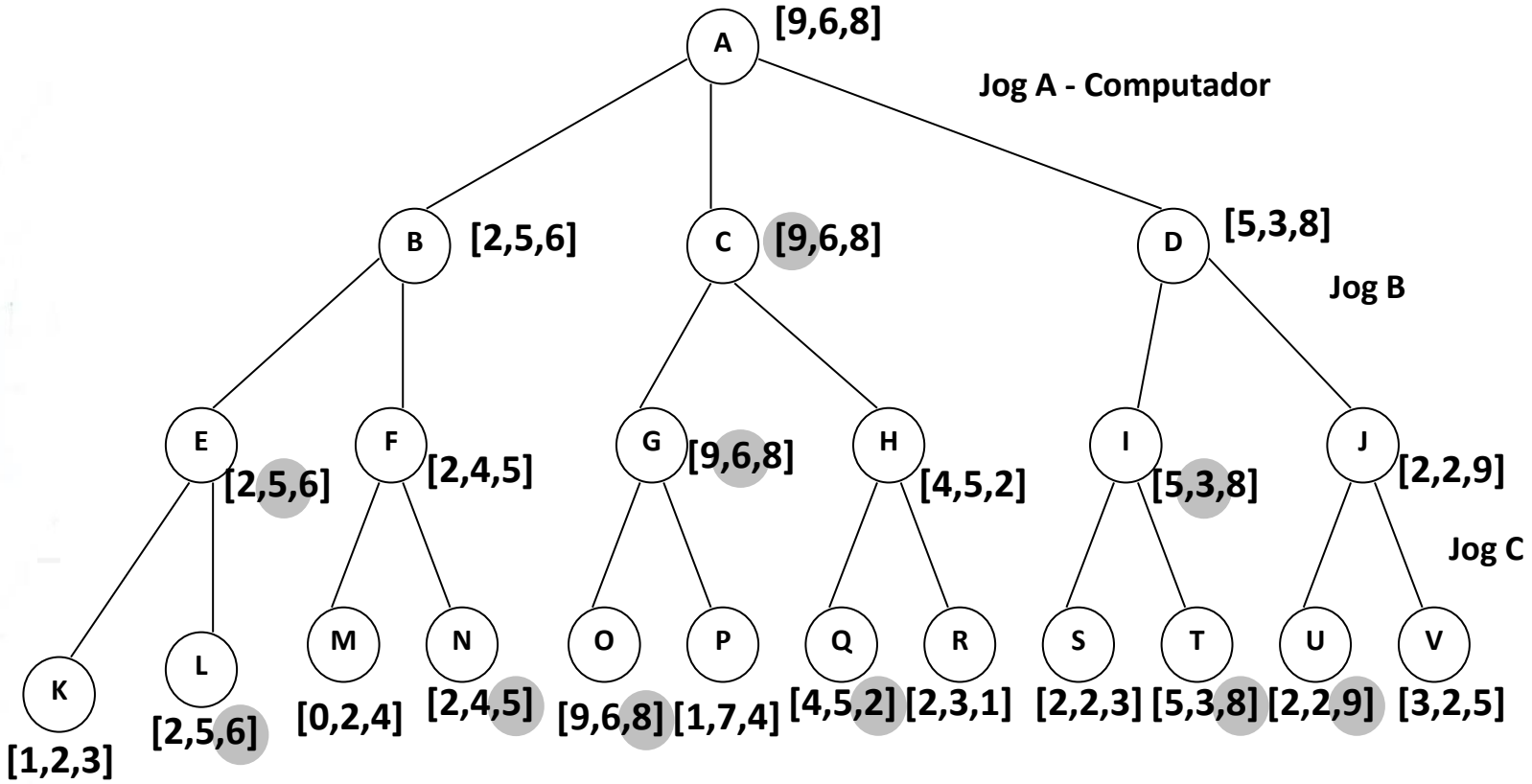
Rainha = 9

- **Jogador possui um valor material de peças.**
- Função Utilidade: F = valor de suas peças – valor peças adversário
- **O lado com uma vantagem segura de valor material de um peão ou mais, provavelmente será o vencedor.**
- Podemos ainda incluir: bispo mais pro final do jogo vale mais, posição da peça em segurança vale 1 ponto, etc...

Jogos com mais de 2 jogadores

- Existem jogos com mais de dois jogadores
- Como adaptar o MINMAX para esses jogos ?
- Características:
  - Ao invés de um valor, teremos um VETOR de valores
  - Um valor para cada jogador
  - Jogadores A, B e C → [Va, Vb,Vc]
  - Cada passo na árvore será de um jogador
  - A raíz continua sendo a jogada do computador
- A função de utilidade deverá retornar o vetor de valores
- Permite implementar alianças ou jogos entre parceiros.
- Lembre-se: funciona apenas para jogos onde se conhece todos os valores de todos os jogadores.

Exemplo jogo com 3 jogadores



Curiosidades Jogos em IA - Xadrez

O que o Kasparov disse após sua derrota para o Deep Blue:

- *"...A máquina se recusou a efetuar um movimento para uma posição que teria uma vantagem decisiva a curto prazo – mostrando um sentido de perigo humano"*

Atualmente

Stockfish

- Usa busca MinMax com cortes Alpha-Beta mais aprimorados
- Opensource
- Vencedor de vários campeonatos de Chess
- Pode usar 512 CPU threads (Sistema multiprocessado)

Foi Brutalmente derrotado em 2018 pelo AlphaZero (DeepMind)

Google DeepMind

100 partidas
AlphaZero: 28 vitórias
Empates: 72
StockFish: 0 vitórias

AlphaGo – Google DeepMind

AlphaGo

MAKING HISTORY

AlphaGo is the first computer program to defeat a professional human Go player, the first to defeat a Go world champion, and is arguably the strongest Go player in history.

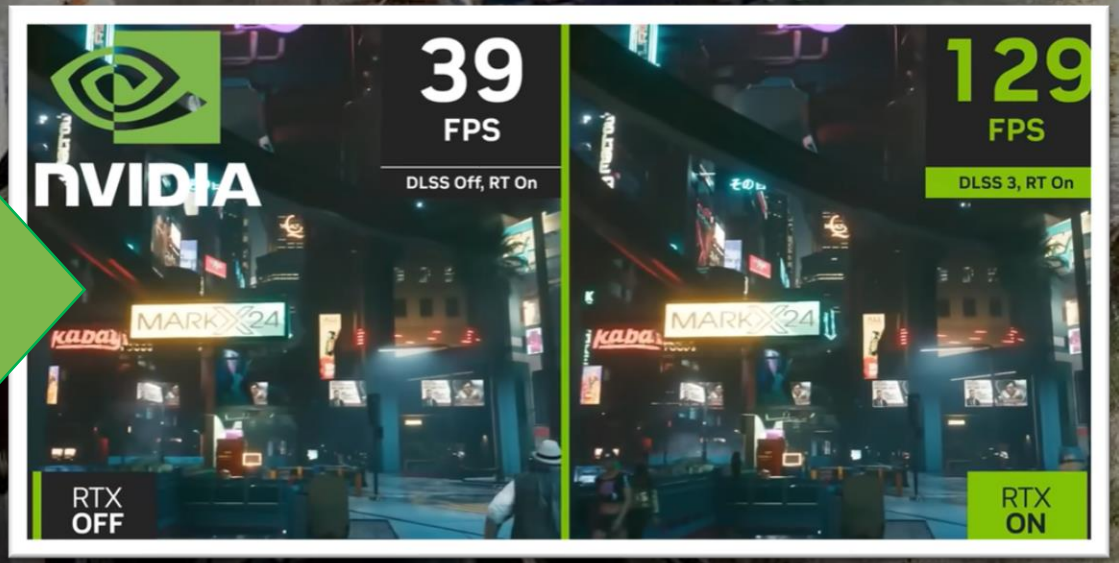
MORE AI in GAMES !!!!!

ALÉM DOS JOGOS DE TABULEIRO

Uma área onde a IA já stá causando impactos significativos é no comportamento dos personagens e nas tomadas de decisão dos NPCs (Non-Playable Character).

Por exemplo, NPC pode ser programado para reagir às ações do jogador, adaptar-se para mudar alguma condição do jogo, e inclusive aprender com suas experiências e das do jogador.

Melhorando a experiência do usuário



Machine Learning aplicada em jogos

Computer Vision aplicada em jogos (captura de tela)

Bibliografia desta Aula

Para aprofundamento nos assuntos desta aula, segue a seguinte referência bibliográfica

- Rich, E. (Inteligência Artificial)
  - Capítulos 2, 3 e 12 (jogos)
- Russel & Norvig (Artificial Intelligence)
  - Capítulos 3,4 e 6

- Alguns slides desta aula foram baseados no slides:
- Anna Reali Costa e Geber Ramalho: “Técnicas de Busca Cega”, Poli-USP e Cin-UFPE.
- Anna Reali Costa: “Busca Informada” em Aula5e6-BuscaInformada.pdf – Poli-USP
- Geber Ramalho: Busca 2 e Busca 3, Cin-UFPE.
- José Augusto Baranauskas: Estratégias de Busca - FFCLRP-USP