INTELIGÊNCIA COMPUTACIONAL

PROF. JOSENALDE OLIVEIRA

josenalde.oliveira@ufrn.br
https://github.com/josenalde/ic

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS - UFRN

CLASSIFICAÇÃO DE MÉTODOS

- **Métodos fracos** não aparentam necessariamente ser inteligentes. Também são usados em aplicações tradicionais.
  - Exemplo: Lógica, métodos de busca, métodos de planejamento
- **Métodos fortes** lidam diretamente com conhecimento, dependem do conhecimento para funcionar. Desenvolvidos no escopo da IA/IC
  - Exemplo: Sistemas especialistas, Técnicas de aprendizado de máquina, Algoritmos Genéticos, Sistemas multiagentes (BDI), Sistemas *fuzzy*

CLASSIFICAÇÃO DE MÉTODOS

- **Lógica: cálculo sentencial e dedução:** é a verificação da veracidade de sentenças a partir da veracidade ou falsidade dos átomos (tabelas verdade, tautologias, formas de dedução)

a	b	a ∧ b	a ∨ b	a → b	a ↔ b	¬a
V	V	V	V	V	V	F
V	F	F	V	F	F	F
F	V	F	V	V	F	V
F	F	F	F	V	V	V

Tautologias

1. a → a	*Identidade*
2. a ↔ a	*Equivalência*
3. (a ∧ a) ↔ a	*Idempotência*
4. (a ∨ a) ↔ a	*Idempotência*
5. ¬ (¬a) ↔ a	*Dupla Negação*
6. (a ∨ (¬a))	*Terceiro Excluído*
7. ¬ (a ∧ (¬a))	*Não Contradição*
8. ¬a → (a → b)	*Negação do Antecedente*
9. (a → b) ↔ (¬b → ¬a)	*Contraposição*
10. ¬ (a ∧ b) ↔ ((¬a) ∨ (¬b))	*De Morgan*
11. ¬ (a ∨ b) ↔ ((¬a) ∧ (¬b))	*De Morgan*
12. (a ∨ b) ↔ ¬ ((¬a) ∧ (¬b))	
13. (a ∧ b) ↔ ¬ ((¬a) ∨ (¬b))	
14. ¬ (a → b) ↔ (a ∧ (¬b))	
15. (a ∨ b) ↔ ((¬a) → b)	

Formas base de dedução

1. a, a → b ⇒ b	Modus Ponens
2. a → b, ¬b ⇒ ¬a	Modus Tollens
3. a → b, b → c ⇒ a → c	Silogismo Hipotético
4. ¬a, a ∨ b ⇒ b	Silogismo Disjuntivo
5. a ∨ c, (a → b) ∧ (c → d) ⇒ b ∨ d	Dilema Construtivo
6. ((¬b) ∨ (¬d)), (a → b) ∧ (c → d) ⇒ (¬a) ∨ (¬c)	Dilema Dedutivo
7. a ∧ b ⇒ a	Simplificação
8. a, b ⇒ a ∧ b	Conjunção
9. a ⇒ a ∨ b	Adição
10. ¬a, a ⇒ b	Dedução por Absurdo

CLASSIFICAÇÃO DE MÉTODOS

- **Lógica: cálculo de predicados:** O cálculo de predicados também estuda a legitimidade ou não de sentenças, mas agora incluindo os efeitos dos quantificadores (∀ e ∃), bem como os predicados (verbos e objetos).

(*TODAS* as crianças estudam)
(Alice é uma criança)
(Alice estuda)

{(∀x [Criança(x) → Estuda(x)]) (Criança(Alice) = V)} ⇒ Estuda (Alice) = V.

(*EXISTEM* crianças que estudam)
(Alice é uma criança)
(Alice estuda)

{(∃x [Criança(x) ∧ Estuda(x)]) ∧ (Criança(Alice) = V)} ⇒ Estuda (Alice) = V

Dedução FALSA

i. ∀x[Musculoso(x) → Forte(x)]	; *premissa.*
ii. ∀y[Jóquei(y) → (¬Forte(y))]	; *premissa.*
iii. ∃z[Jóquei(z) ∧ Alto(z)]	; *premissa.*
iv. ∀w[¬Alto(w) ∨ Musculoso(w)]	; *premissa obtida por negação da tese.*
v. (¬Musculoso(x)) ∨ Forte(x)	; *cláusula derivada de i .*
vi. (¬Jóquei(y)) ∨ (¬Forte(y))	; *cláusula derivada de ii.*
vii. Jóquei(z') ∧ Alto(z')	; *cláusula derivada de iii.*
viii. Jóquei(z')	; *simplificação de vii.*
ix. Alto(z')	; *simplificação de vii.*
x. ¬Alto(w) ∨ Musculoso(w)	; *cláusula derivada de iv.*
xi. Musculoso(z')	; *resolução ix, x.*
xii. Forte(z')	; *resolução v,xi.*
xiii. ¬Jóquei(z')	; *resolução vi, xii.*
xiv. ∃x[Alto(x) ∧ (¬Musculoso(x))]	; *por absurdo, viii, xiii*

A avaliação da legitimidade pode ser feita por refutação, negando-se a tese e verificando que isso leva a contradição.

CLASSIFICAÇÃO DE MÉTODOS

- A **lógica** é um dos tipos de representação de conhecimento em IA, porém dada sua complexidade e não necessariamente clareza, compromete o uso, mas há comunidade.
- Outros tipos de representação inclui **por regras de produção** (S. Esp, Fuzzy,

i. Se (produz_leite ∧ tem_pelos) Então (mamífero)
ii. Se (mamífero ∧ come_carne) Então (carnívoro)
iii. Se (mamífero ∧ possui_presas ∧ possui_garras) Então (carnívoro)
iv. Se (mamífero ∧ possui_casco) Então (ungulado)
v. Se (carnívoro ∧ pardo ∧ pintado) Então (onça)
vi. Se (carnívoro ∧ pardo ∧ listrado) Então (tigre)
vii. Se (ungulado ∧ pardo ∧ pintado) Então (girafa)
viii. Se (ungulado ∧ branco ∧ listrado) Então (zebra)

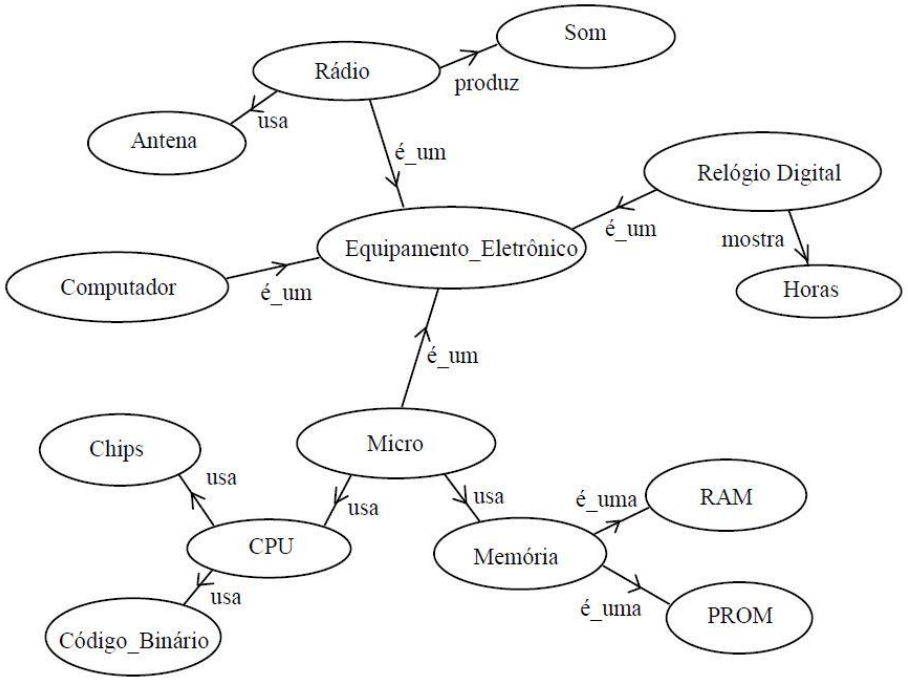
Lisp 1958-

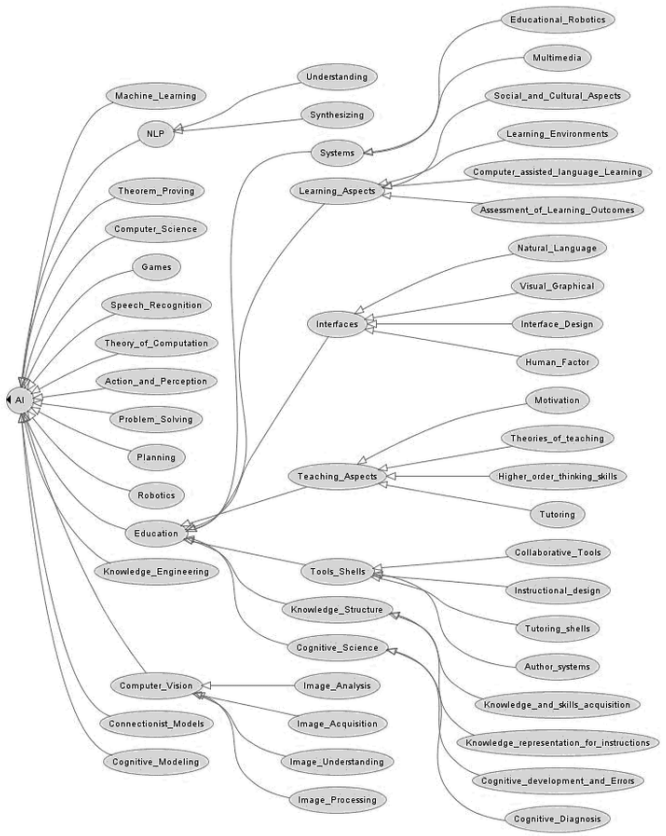
- **Rede semântica:** A rede semântica é uma coleção de nós conectados por arcos. Os nós representam objetos, conceitos ou eventos e os arcos representam as relações. Usualmente os nós e os arcos são etiquetados para indicar o que representam:

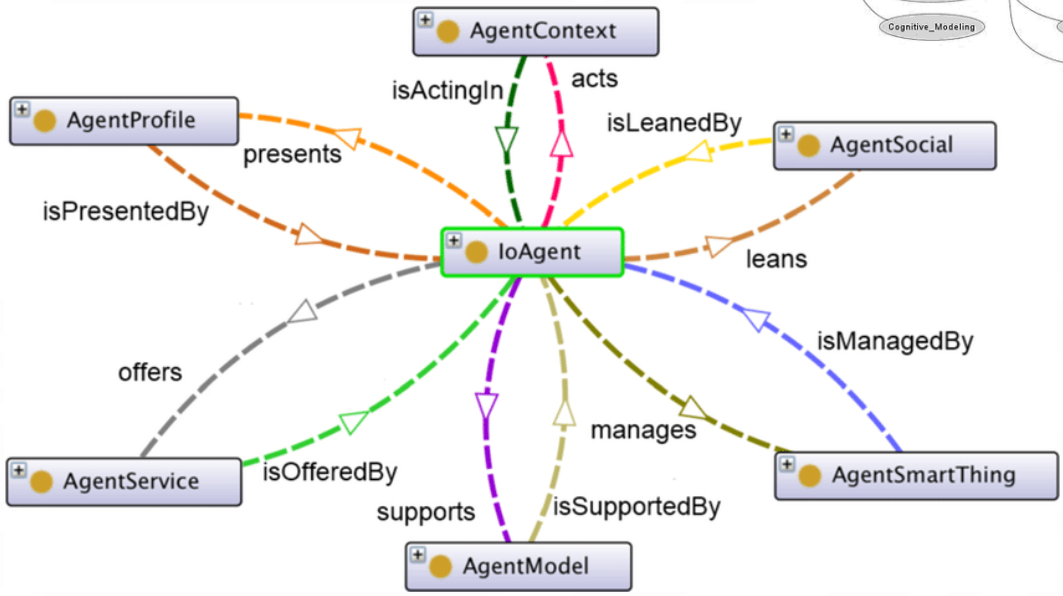
https://codedocs.org/what-is/lisp-programming-language

CLASSIFICAÇÃO DE MÉTODOS

- Rede semântica (Exemplo: Ontology Web Language)







CLASSIFICAÇÃO DE MÉTODOS

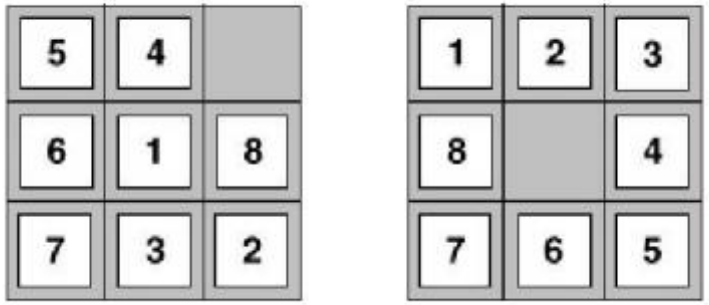
- Rede semântica (Exemplo: Ontology Web Language)

Ontologias são utilizadas para **capturar conhecimento** sobre um domínio de interesse. Uma ontologia descreve os conceitos de um domínio e também as relações que existem entre esses conceitos. As diversas linguagens para construção de ontologias fornecem diferentes funcionalidades. O padrão mais recente de linguagens para ontologias é o OWL, desenvolvido no âmbito do W3C-*World Wide Web Consortium*. O *Protege-OWL* possui um conjunto de operadores (por exemplo, o **AND**, o OR e o NOT) e é baseado em um **modelo lógico** que torna possível definir conceitos da forma como são descritos. Conceitos complexos podem ser constituídos a partir de definições de conceitos simples. Além disso, o modelo lógico permite a utilização de um MI, o qual pode verificar se as declarações e as definições da ontologia são mutuamente consistentes entre si e reconhecer se conceitos são adequados a definições. O MI pode, portanto, ajudar a manter a hierarquia, o que é útil quando existem casos em que uma classe tem mais de um pai.

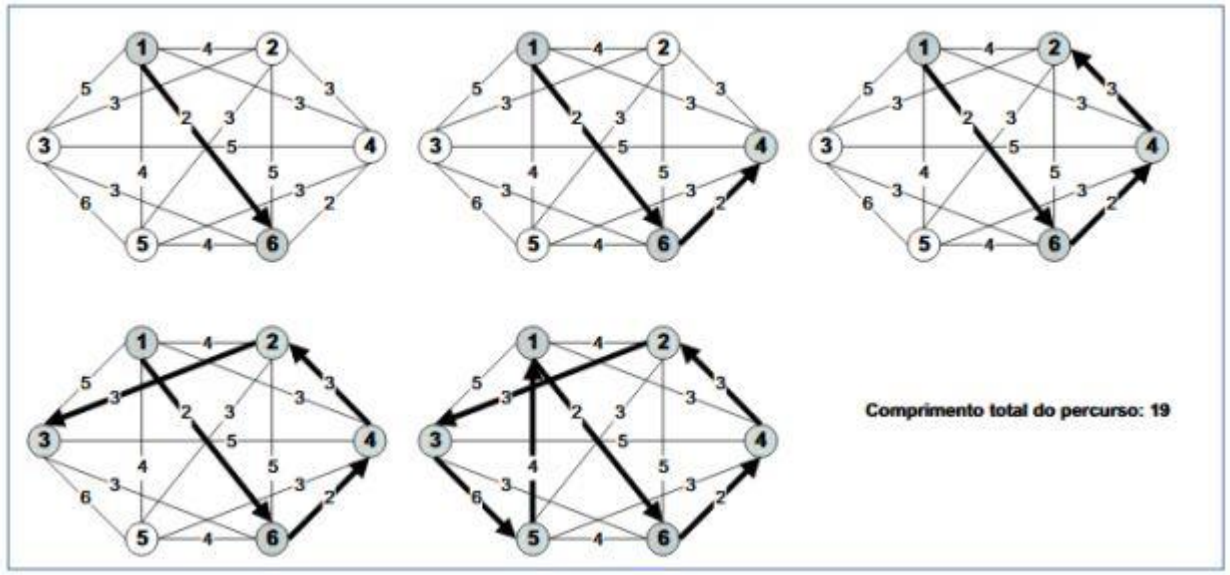
Exemplo de aplicação: https://ainfo.cnptia.embrapa.br/digital/bitstream/item/210473/1/Usando-software-SBIAgro-2007.pdf

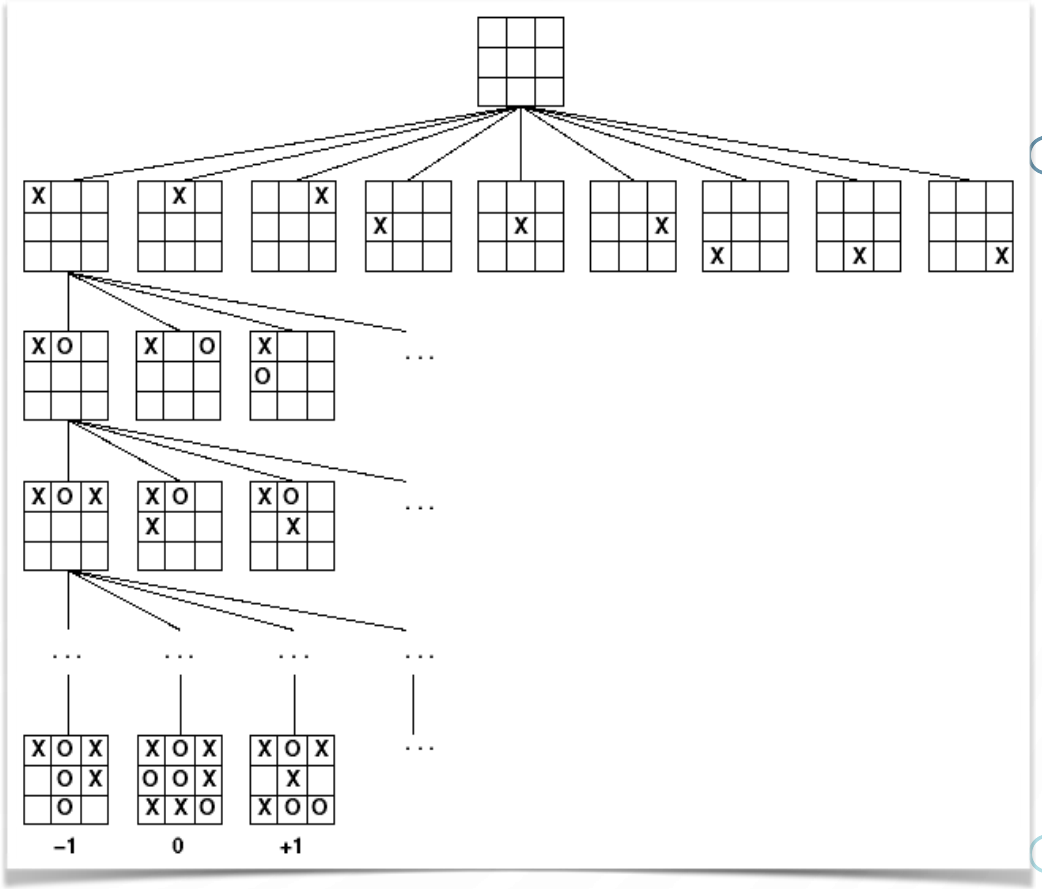
Busca: Muitos problemas de IA necessitam de métodos de busca (*search*), durante o processo de solução.

a. **The 8-puzzle problem:** The 8-puzzle is a small board game for a single player; it consists of 8 square tiles numbered 1 through 8 and one blank space on a 3 x 3 board, as depicted in left board in Figure 2. Moves of the puzzle are made by sliding an adjacent tile into the position occupied by the blank space, which has the effect of exchanging the positions of the tile and blank space. Only tiles that are horizontally or vertically adjacent (not diagonally adjacent) may be moved into the blank space. The object is to reach the configuration in the right board of Figure 2.



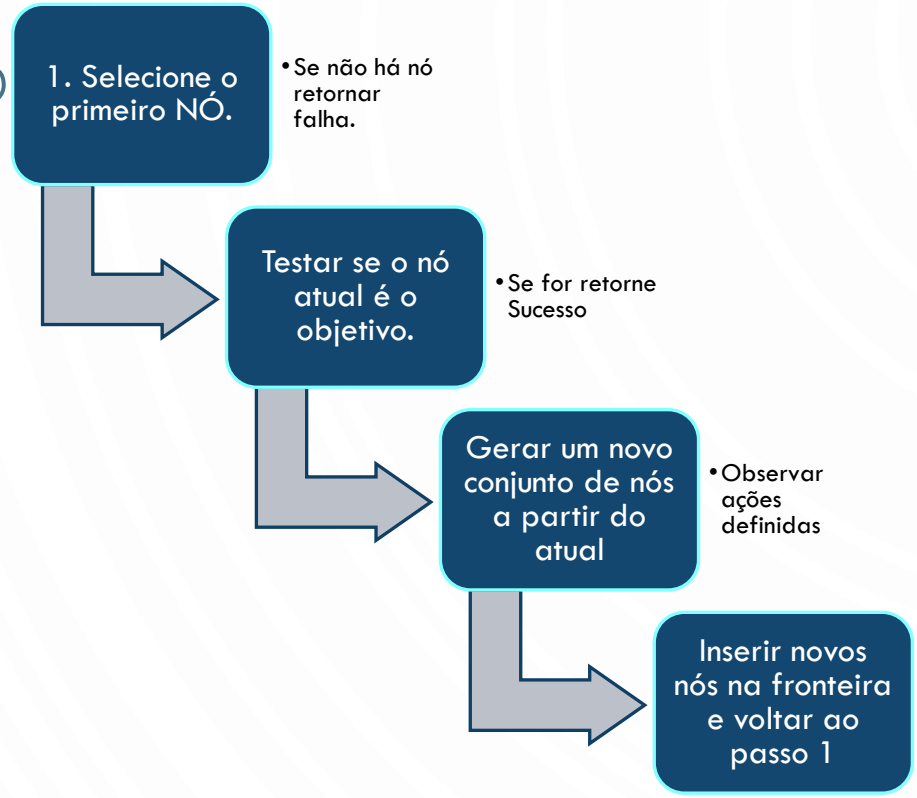
TSP

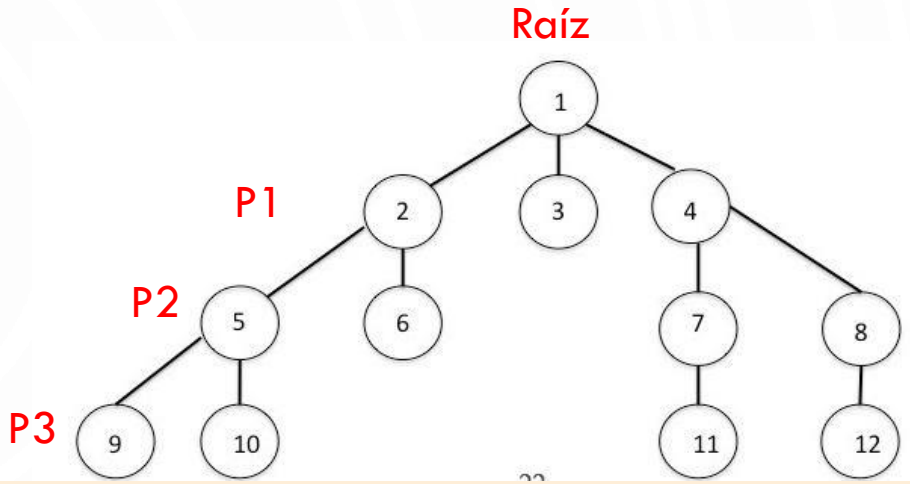




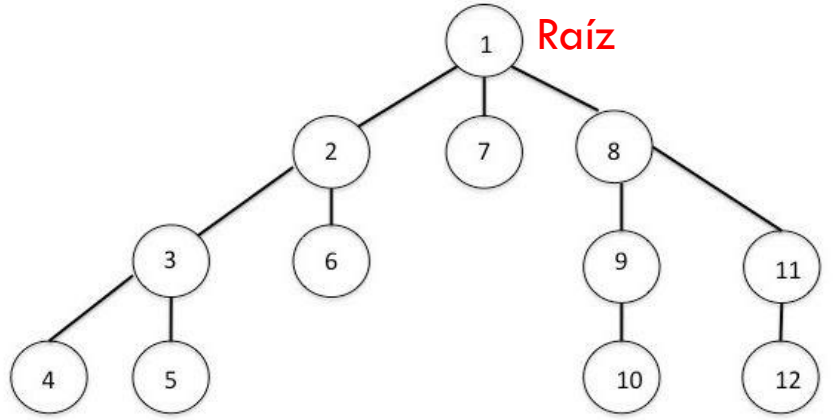
Exemplo de busca exaustiva ou cega, onde não sabe qual nó de cada nível a expandir, não calcula o custo deste nó até o final.

Busca: Em geral seguem este fluxograma para busca genérica



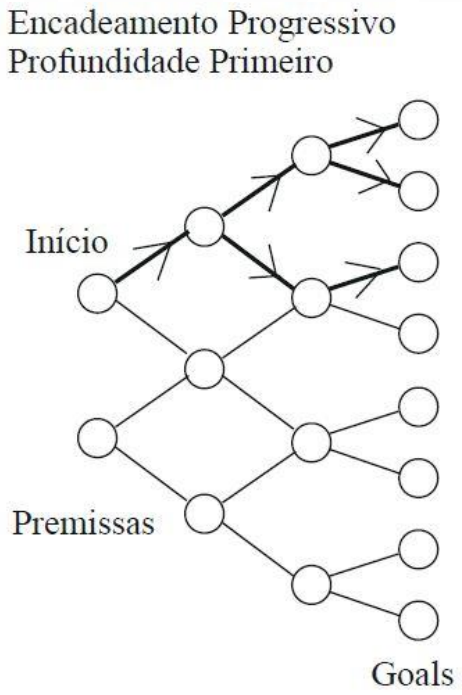


Busca em **Largura**: visita todos nós de Px antes de avançar para Px+1
Se implantado com FILA (FIFO), nós de menor profundidade expande antes
Algoritmo COMPLETO, pois encontra solução caso exista uma
Contudo não é ÓTIMO, pois a solução é a mais rasa! Só se o custo do caminho crescer com a profundidade
Tem custo de tempo e de memória exponencial em função da ramificação

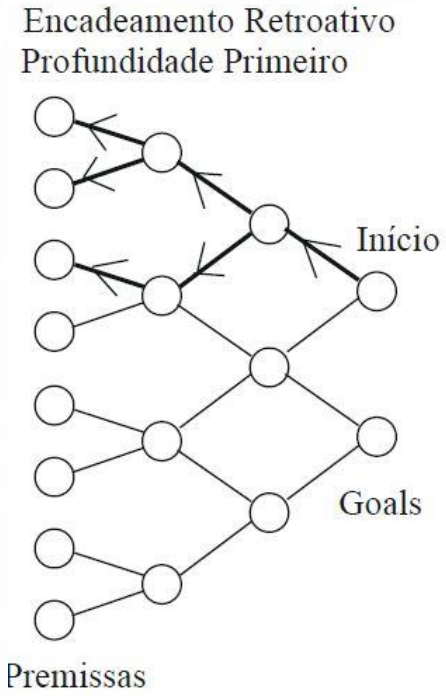


Busca em **Profundidade**: visita primeiro de cada profundidade etc.
Visitará todos os nós de um ramo antes de ir para outro ramo.
Pode ser implementado com LIFO (pilha), recursividade etc.
Custo de memória menor, custo de tempo equivalente

Busca: Em geral seguem este fluxograma para busca genérica



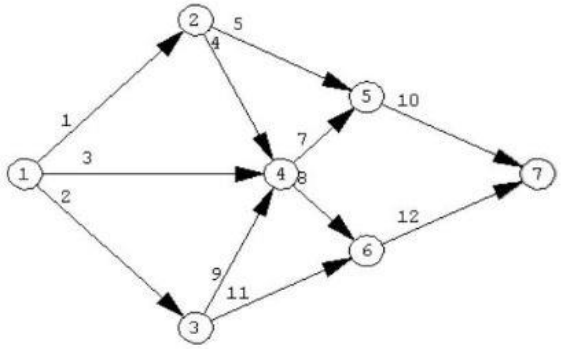
Encadeamento PROGRESSIVO (baseado em dados). Parte do estado inicial e usa ações permitidas para ir em frente até que um objetivo seja atingido. Mais comum.



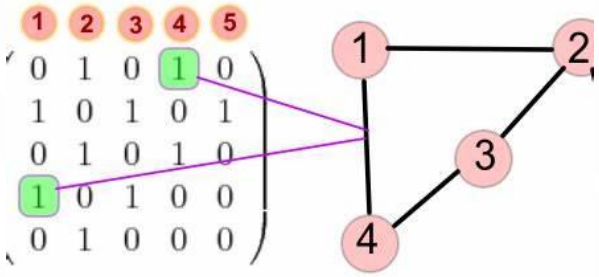
Encadeamento RETROATIVO (baseado em objetivos). Parte do objetivo e volta ao nó. Usado quando o objetivo é bem especificado, como em prova de teoremas matemáticos, achar saída de labirintos etc.

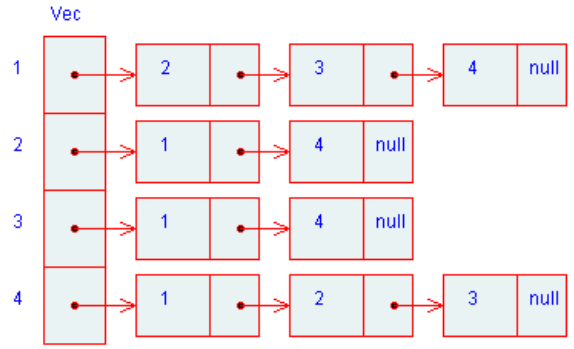
A busca cega no máximo calcula o custo do nó atual para o nó inicial, mas não em relação ao objetivo. Necessário função heurística para estimar distância ao objetivo

Busca: formas de representação em grafos (matrizes de adjacências, listas encadeadas, dicionários, etc}



$$\begin{pmatrix} 0 & 1 & 2 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 5 & 0 & 0 \\ 0 & 0 & 0 & 9 & 0 & 11 & 0 \\ 0 & 0 & 0 & 0 & 7 & 8 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 10 \\ 0 & 0 & 0 & 0 & 0 & 0 & 12 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$



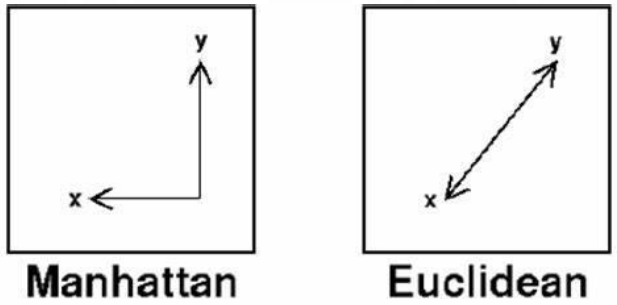


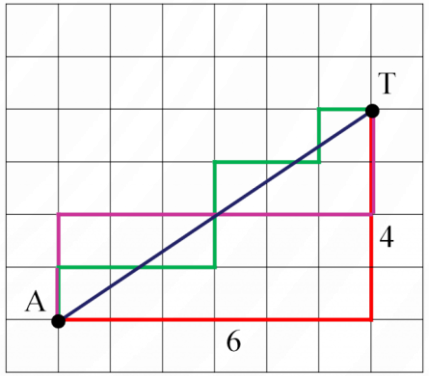
Busca HEURÍSTICA

- Utilizam **conhecimento específico do problema na escolha do próximo nó a ser expandido**
  - Exemplo: Em rotas, poderia utilizar a distância direta entre duas cidades (menor distância entre dois pontos é uma reta)
- A busca heurística **aplica uma função de avaliação a cada nó da fronteira do espaço de estados**
- Essa função estima o custo de caminho do nó atual até o nó objetivo (h)
  - **A função de avaliação heurística depende de cada problema**
- Heurística é uma **estratégia** aplicada na resolução de um problema que utiliza conhecimento

específico do problema (muitas vezes, conhecimento **empírico**) – IDEIA: O nó de menor custo aparente na fronteira do espaço de estados até o nó objetivo é testado e expandido primeiro

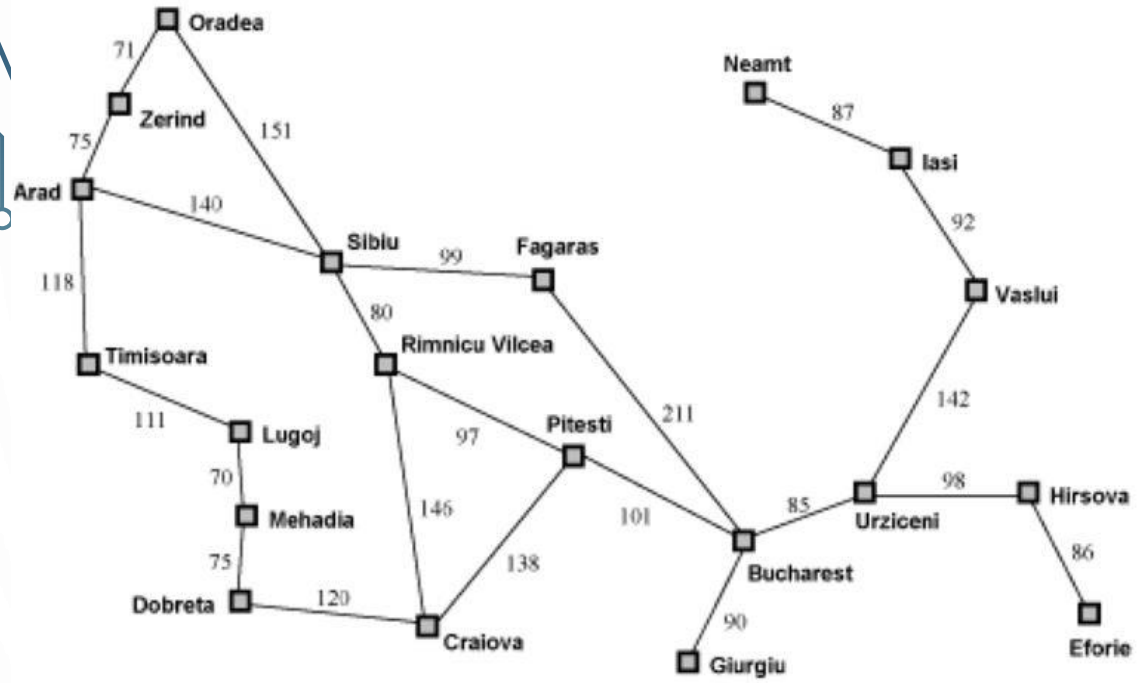
Exemplos busca gulosa e busca A*





Manhattan: 4 + 6 = 10
Euclidiana: 7,21

Busca HEURÍSTICA GULOSA

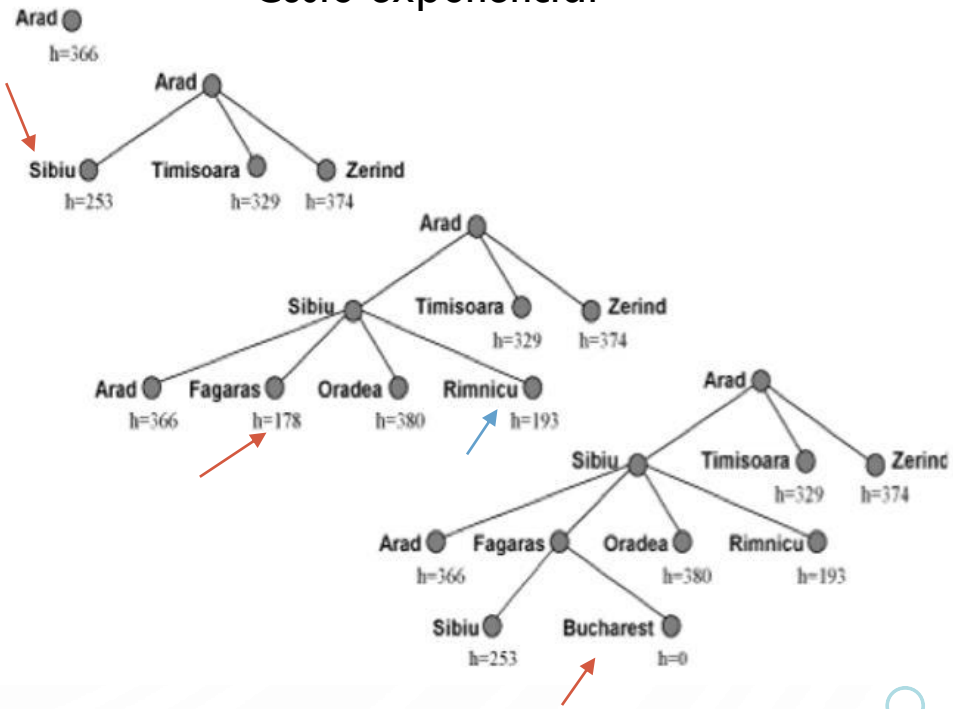


Exemplo rota Arad - Bucareste

Função h - heurística

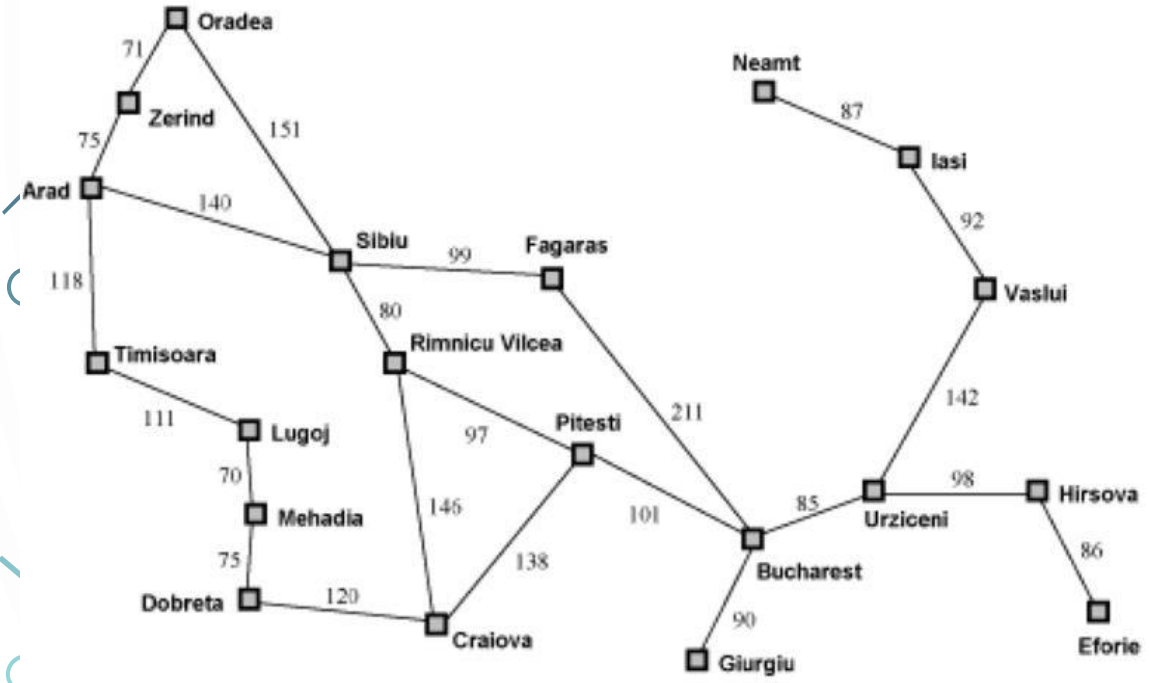
Straight line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Solução não é ótima!
Custo exponencial



Por Fagaras: 450
Por Rimnicu: 418

Busca HEURÍSTICA A*



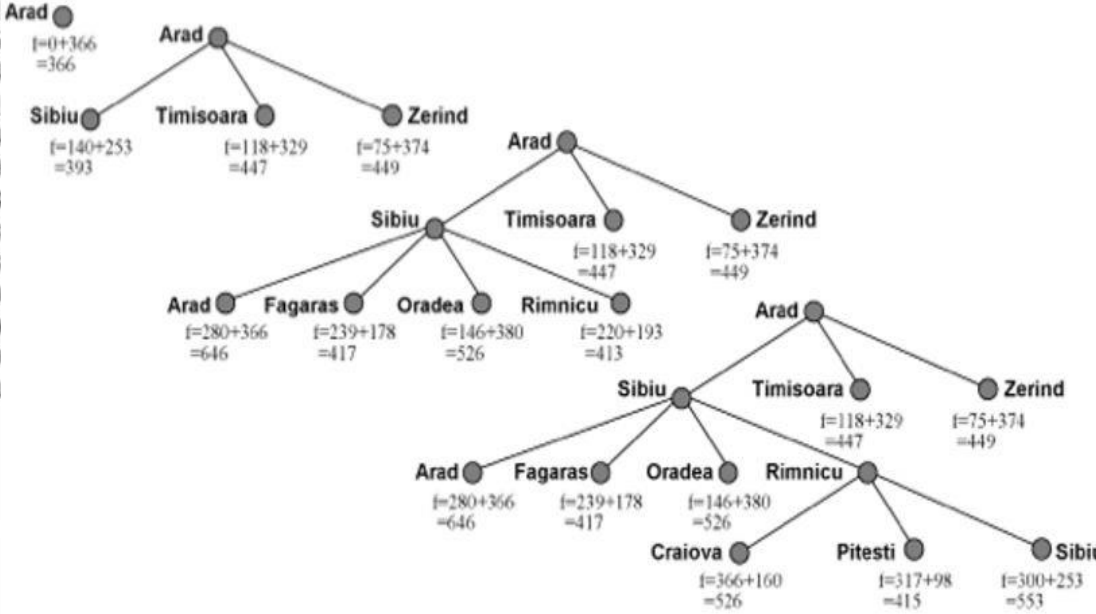
Exemplo rota Arad - Bucareste

Função h - heurística

Straight line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Solução ótima e completa!

Custo exponencial, guarda todos nós expandidos



Ainda a mais usada

Função de avaliação:

f (n) = g (n) + h (n)

g (n) = distância de n ao nó inicial

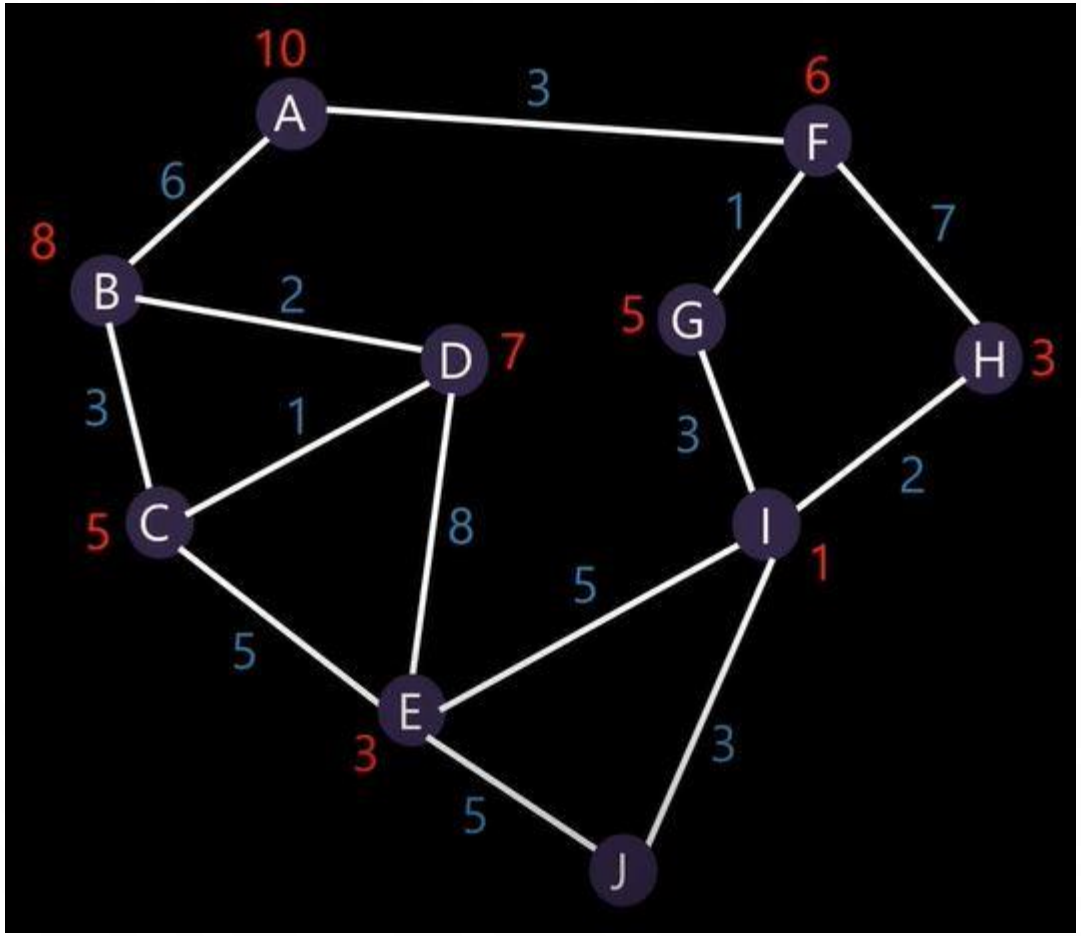
h (n) = distância estimada de n ao nó final

A* expande o nó de menor valor de f na fronteira do espaço de estado

Busca HEURÍSTICA A*

Passo 0. Criar um grafo de busca G, inicialmente contendo somente o nó de partida **S**.
Passo 1. Inicializar uma lista chamada OPEN com o nó de partida **S** e uma lista chamada
CLOSED com ∅. (OPEN é a lista dos nós ainda não selecionados para expansão, enquanto CLOSED
é a lista dos nós já expandidos)
Passo 2. LOOP: Se OPEN = ∅ Então FIM e reportar Falha.
Passo 3. Tomar um nó de OPEN com o menor f'.
Chamar este nó de **N**.
Retirá-lo de OPEN e colocá-lo em CLOSED.
Passo 4. Se **N** é o nó buscado (*goal*) Então FIM e fornecer a solução percorrendo os ponteiros de **N** a **S**.
Passo 5. Expandir **N**, gerando o conjunto de nós SUCESSORES e então adicioná-los no grafo.
Passo 6. Para cada membro **M** de SUCESSORES:
Se **M** ∉ {OPEN, CLOSED}
Então adicionar **M** em OPEN;
Se **M** ∈ OPEN Então chamar a cópia de **M** em OPEN de **M**old; verificar se g(**M**) < g(**M**old): em caso afirmativo, tem-se que é melhor chegar a **M** através do caminho atual do que a indicada pelo ponteiro de **M**old. Portanto, descarta-se **M**old e faz-se o **M** apontar para **N**.
em caso negativo, basta que se remova **M**.
Se **M** ∈ CLOSED Então chamar a cópia de **M** em CLOSED de **M**old; de modo similar ao caso (**M** ∈ OPEN), verificar qual o melhor caminho para **M** e atualizar os valores de g e f".
Passo 7. Reordenar OPEN com base no f".
Passo 8. Return LOOP;

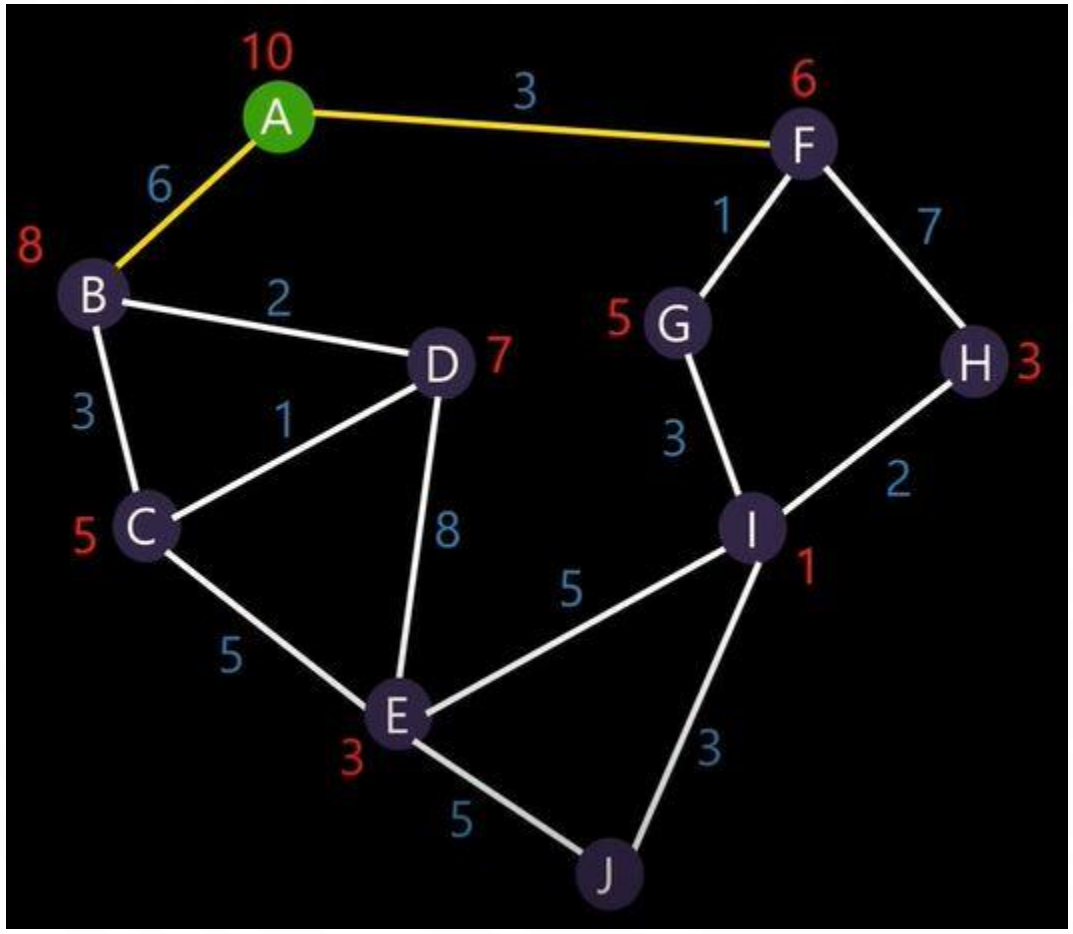
Busca HEURÍSTICA A* - Exemplo:



Encontrar caminho mais curto entre A e J

Azul: distância real
Vermelho: heurística até o destino

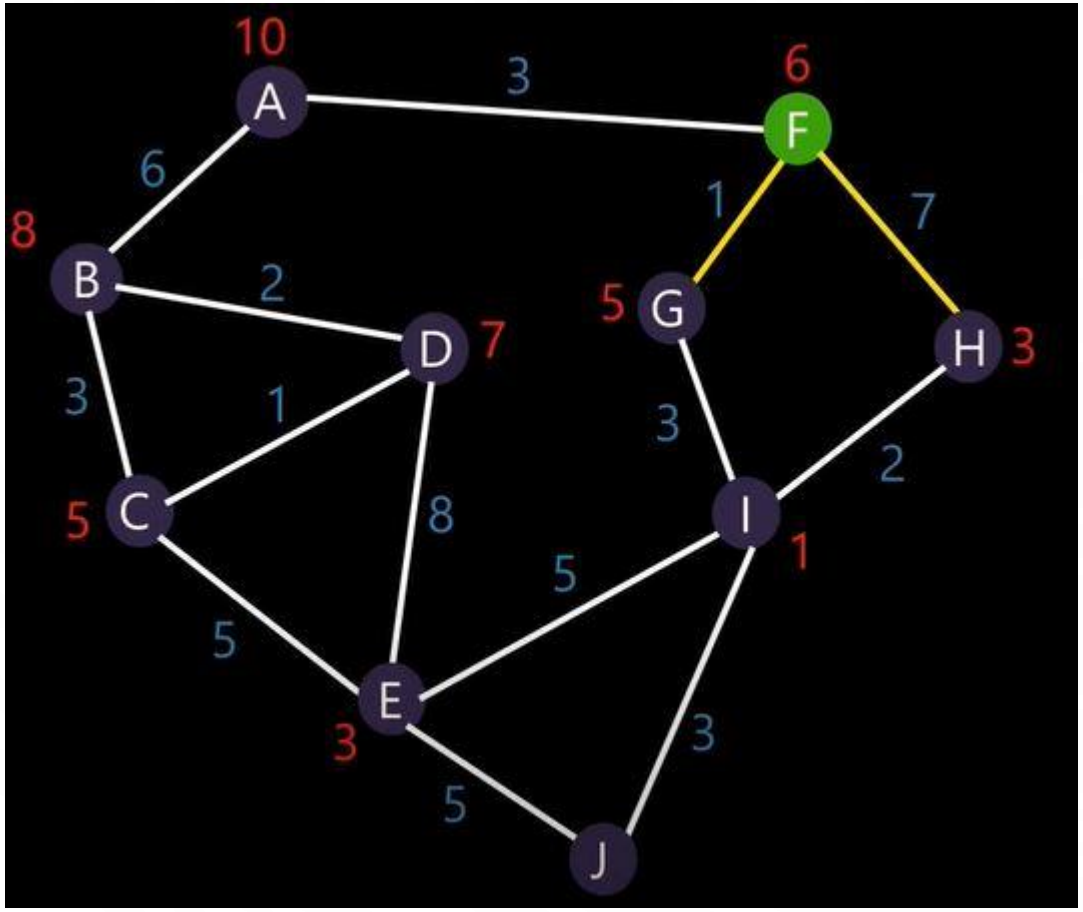
Busca HEURÍSTICA A* - Exemplo:



Encontrar caminho mais curto entre A e J

1) Expandindo A:
    1) F(B)=6+8=14
    2) F(F)=3+10=13
    Seleciona F

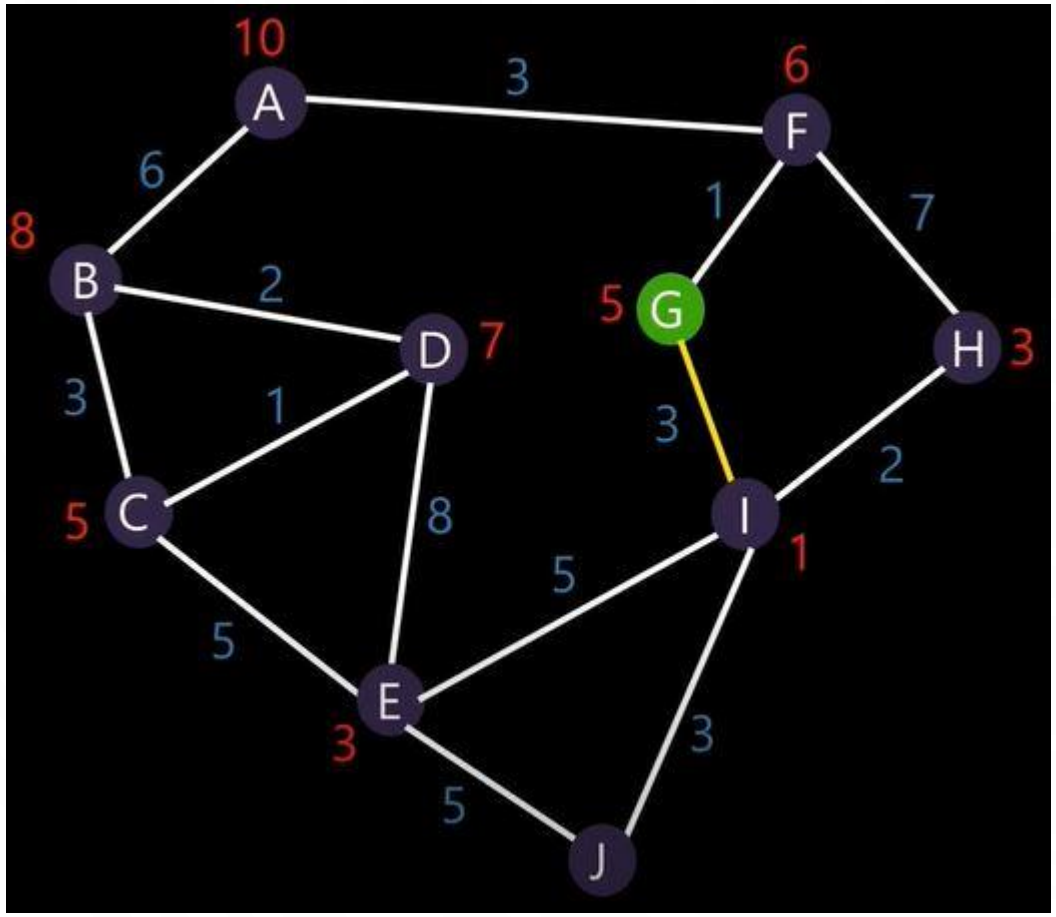
Busca HEURÍSTICA A* - Exemplo:



Encontrar caminho mais curto entre A e J

1) Expandindo A:
    1) F(B)=6+8=14
    2) F(F)=3+10=13
    Seleciona F
2) Expandindo F:
    1) F(G)=(3+1)+5=9
    2) F(H)=(3+7)+3=13
    Seleciona G
3) Expandindo G:

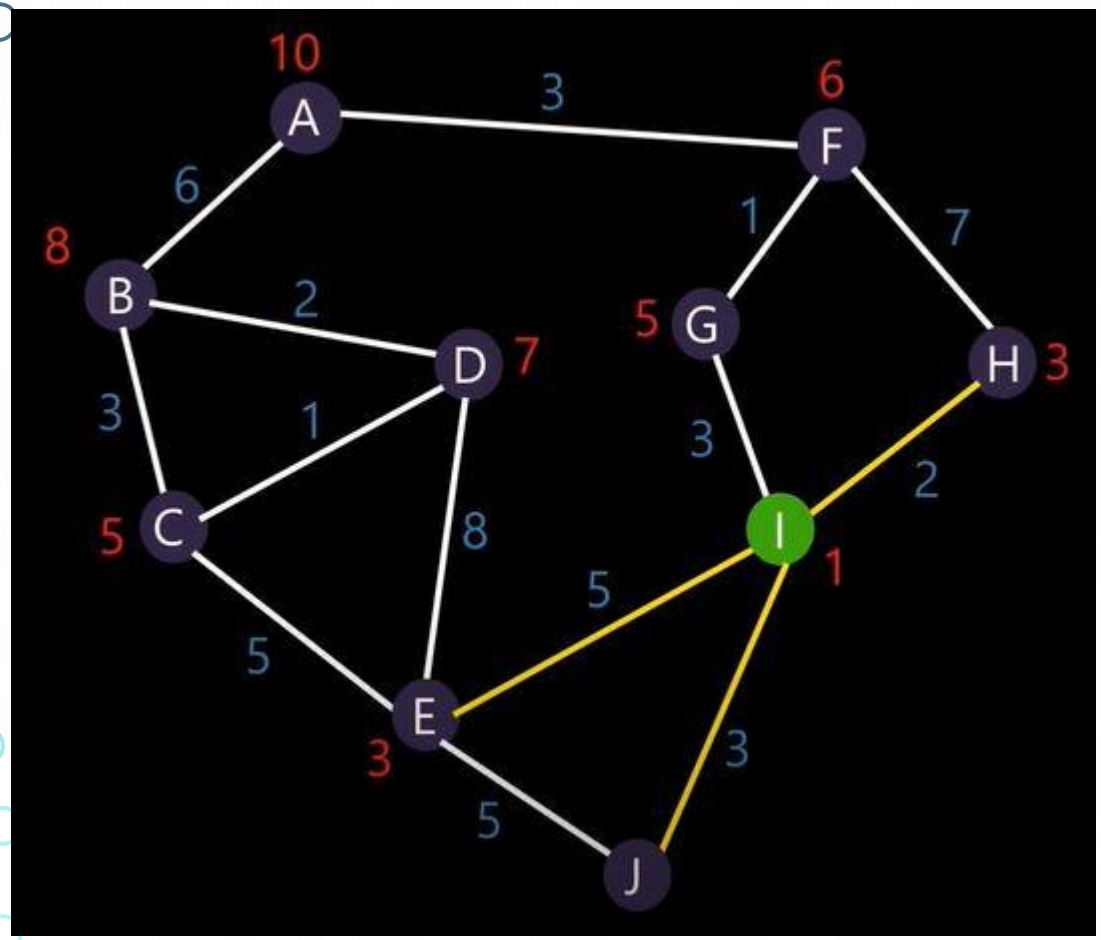
Busca HEURÍSTICA A* - Exemplo:



Encontrar caminho mais curto entre A e J

1) Expandindo A:
    1) F(B)=6+8=14
    2) F(F)=3+10=13
    Seleciona F
2) Expandindo F:
    1) F(G)=(3+1)+5=9
    2) F(H)=(3+7)+3=13
    Seleciona G
3) Expandindo G:
    1) F(I)=(3+1+3)+1=8
4) Expandindo I:

Busca HEURÍSTICA A* - Exemplo:



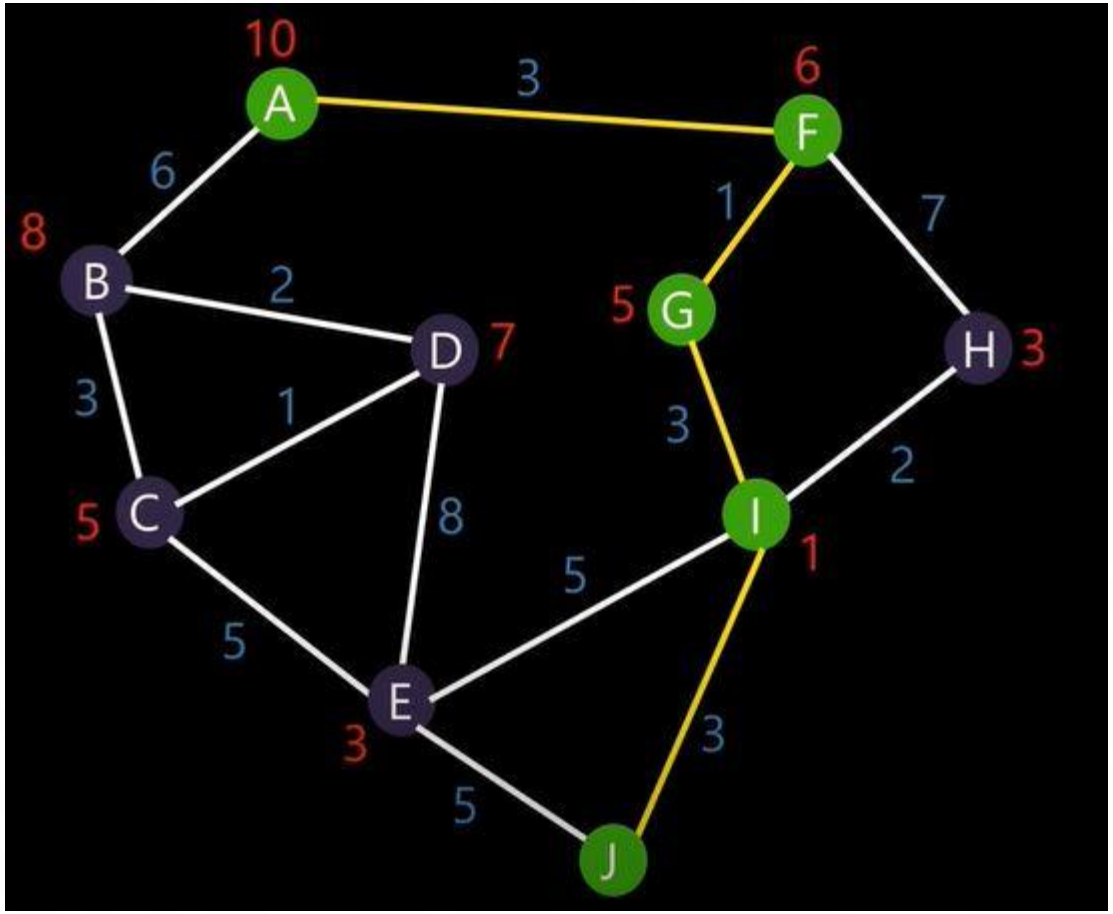
Encontrar caminho mais curto entre A e J

1) Expandindo A:
    1) F(B)=6+8=14
    2) F(F)=3+10=13

    Seleciona F
2) Expandindo F:
    1) F(G)=(3+1)+5=9
    2) F(H)=(3+7)+3=13

    Seleciona G
3) Expandindo G:
    1) F(I)=(3+1+3)+1=8
4) Expandindo I:
    1) F(E)=(3+1+3+5)+3=15
    2) F(H)=(3+1+3+2)+3=12
    3) F(J)=(3+1+3+3)+0=10

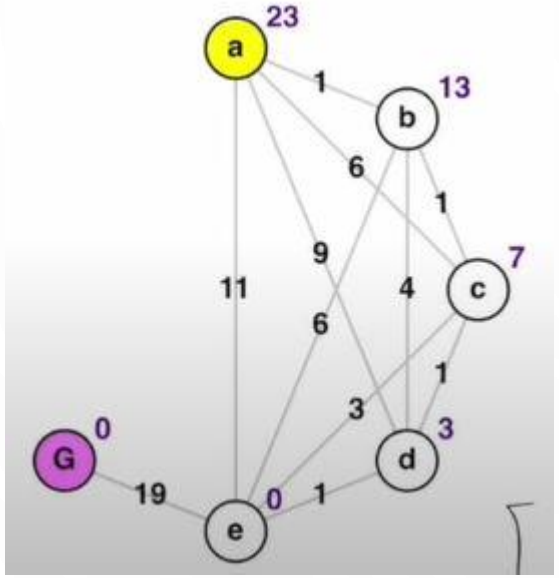
Busca HEURÍSTICA A* - Exemplo:



Encontrar caminho mais curto entre A e J

Solução: A-F-G-I-J (menor caminho)

Busca HEURÍSTICA A* - Exemplo:



Encontrar caminho mais curto entre a e G

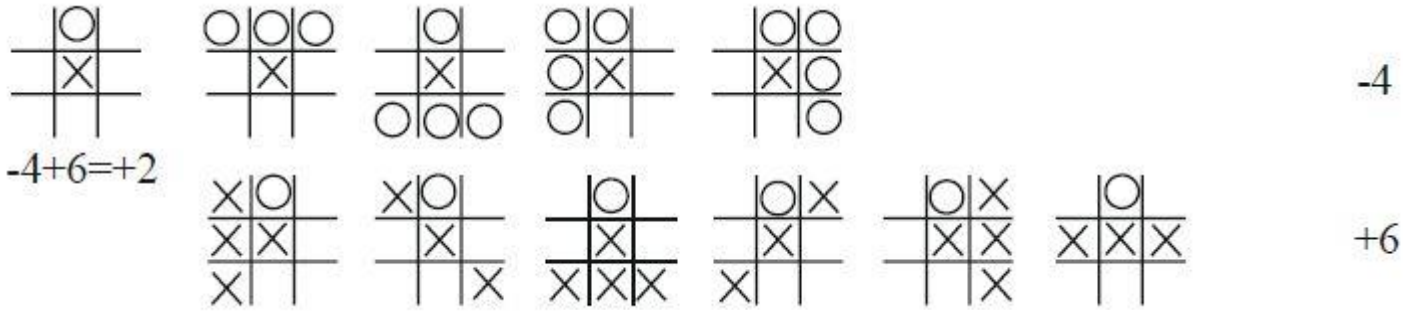
Solução: ?

Para compreender bem o algoritmo e como implementá-lo, fazer passo a passo

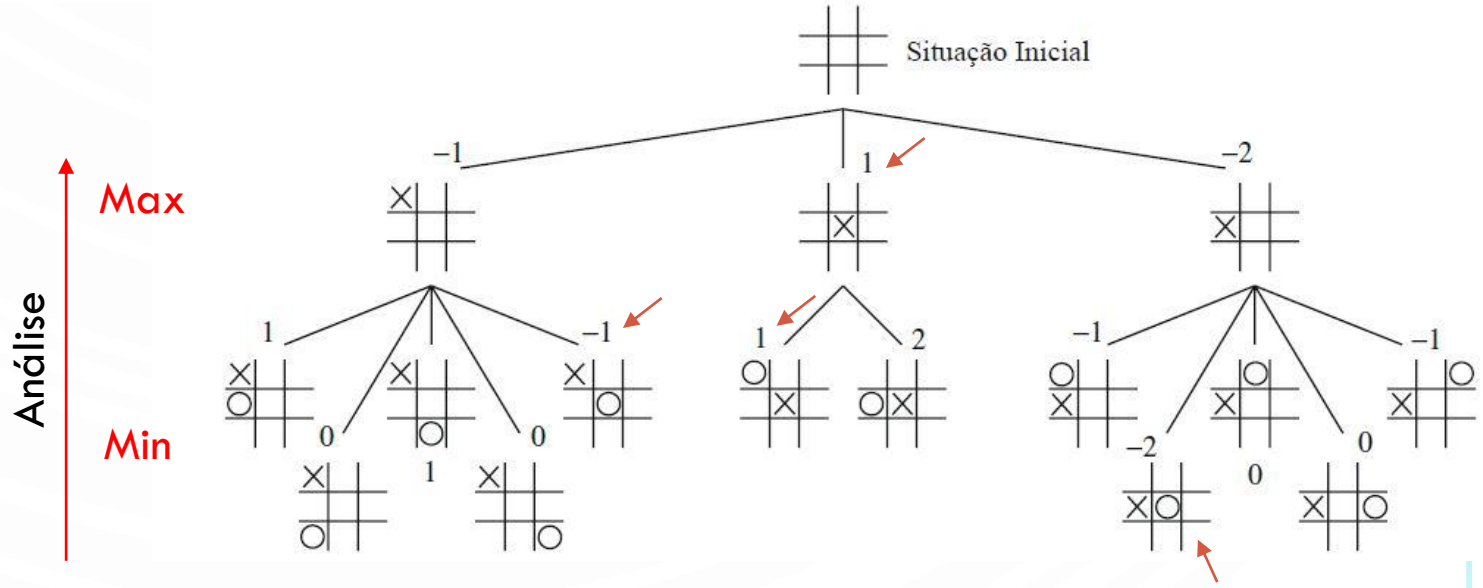
Busca HEURÍSTICA MIN MAX: exemplo jogo da velha

- Uma heurística possível é contar, em cada situação de jogo, a diferença entre o número de alinhamentos possíveis para cada jogador.

Jogador O em (1,2) e X em (2,2)

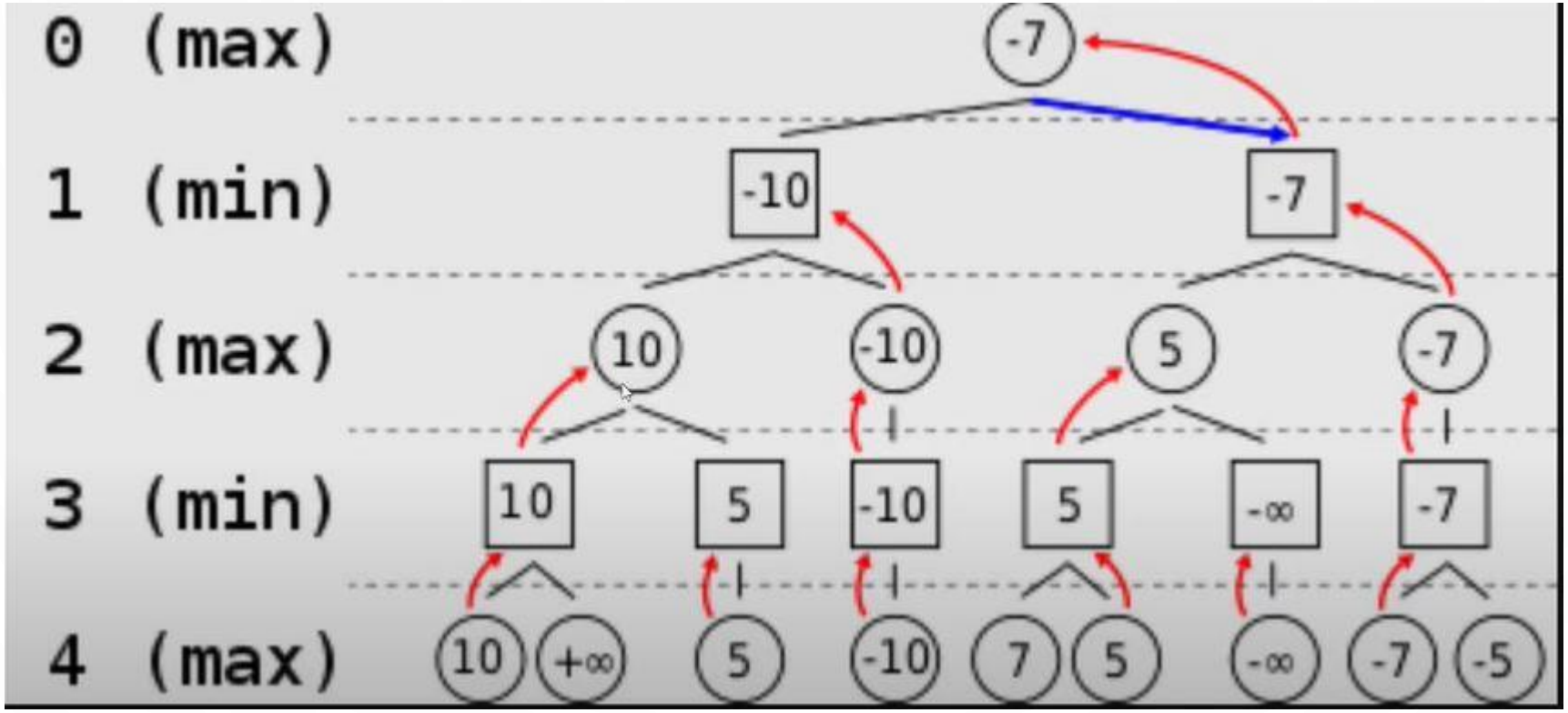


A primeira linha apresenta os alinhamentos possíveis (4) para o jogador (O) e a segunda os possíveis (6) para o jogador (×). A diferença -4+6=+2 é o índice para a configuração no canto superior esquerdo.



Busca HEURÍSTICA MIN MAX:

- Para jogos com 2 jogadores, melhor de um pior para o outro!



Busca HEURÍSTICA MIN MAX: exemplo jogo da velha

o mecanismo que o jogador (×) utiliza para posicionar a sua ficha. Caso o jogador (×) posicione a ficha na posição (1,1), o jogador (O) poderá obter valor -1 através da ocupação de (2,2), que é a situação mais favorável para (O), dada a jogada do (×). Se o jogador (O) ocupar a posição (2,1), por exemplo, obtém apenas valor +1, ou seja pior para (O). Note-se que o melhor caso para (O) será também o pior caso para (×). Assim, considerando-se as possibilidades para a primeira jogada de (×), os valores associados são:

(×) em (1,1) -> valor -1,

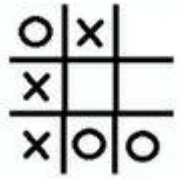
(×) em (2,2) -> valor +1,

(×) em (2,1) -> valor -2.

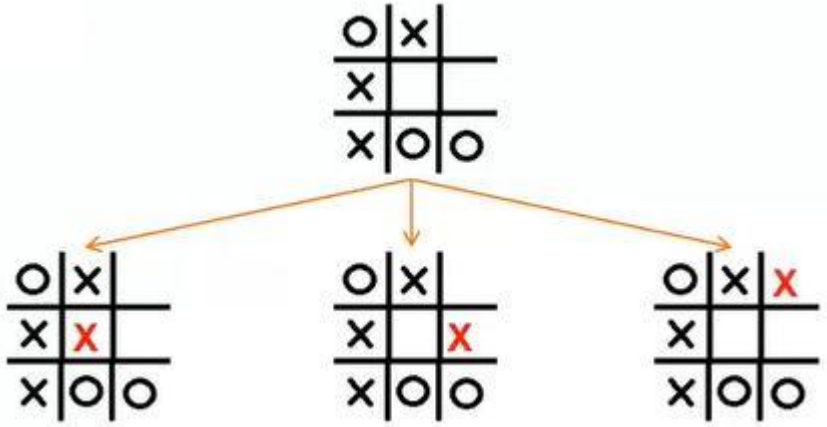
Portanto, considerando o melhor dos piores casos, o jogador (×) deve optar por colocar a sua ficha em (2,2), que corresponde ao máximo entre -1, +1 e -2. Por outro lado, o valor -1 corresponde ao mínimo entre 1, 0, 1, 0 e -1 da sub-árvore à esquerda da Figura. Analogamente, o valor -2 corresponde ao mínimo entre -1, -2, 0, 0, e -1, da sub-árvore à direita da Figura. Esta estratégia de decisão é chamada de ‘max-min’, ou seja, ‘maximizar o ganho considerando os piores casos’.

Busca HEURÍSTICA MIN MAX: exemplo jogo da velha 2

Seja jogada de MAX = X
Árvore com profundidade máxima
Função de avaliação
+1 vitória Max
-1 vitória Min
0 empate

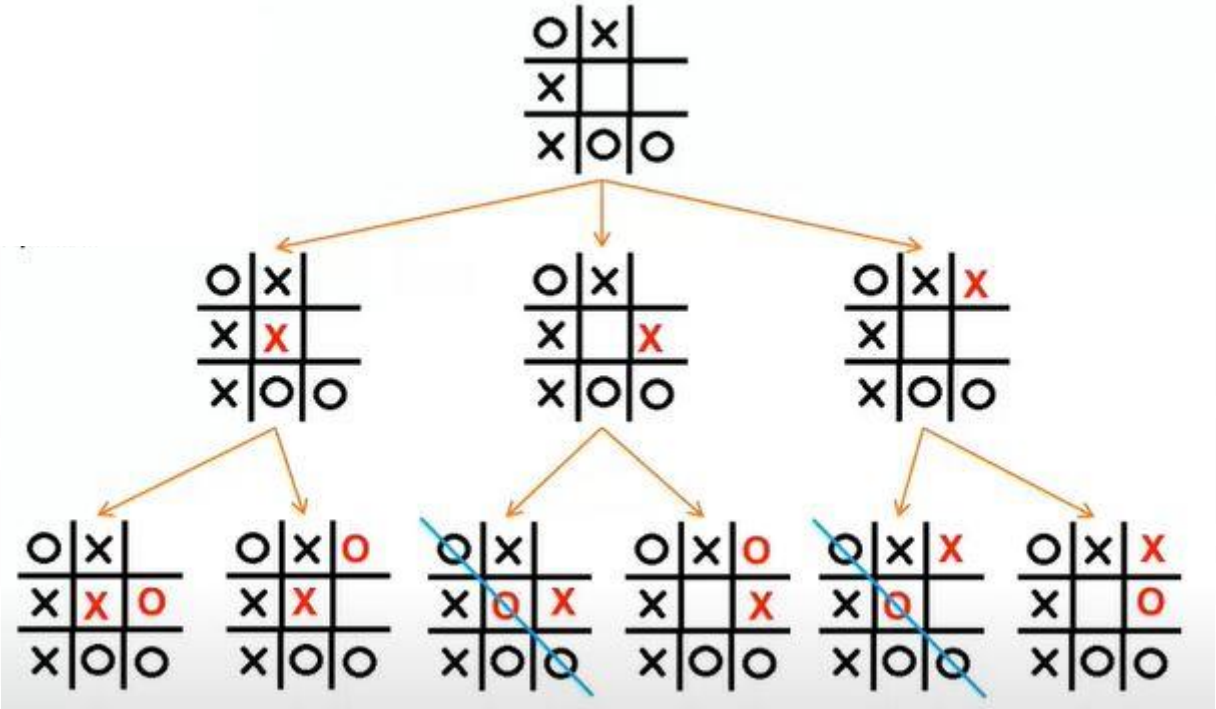


Supor estado inicial



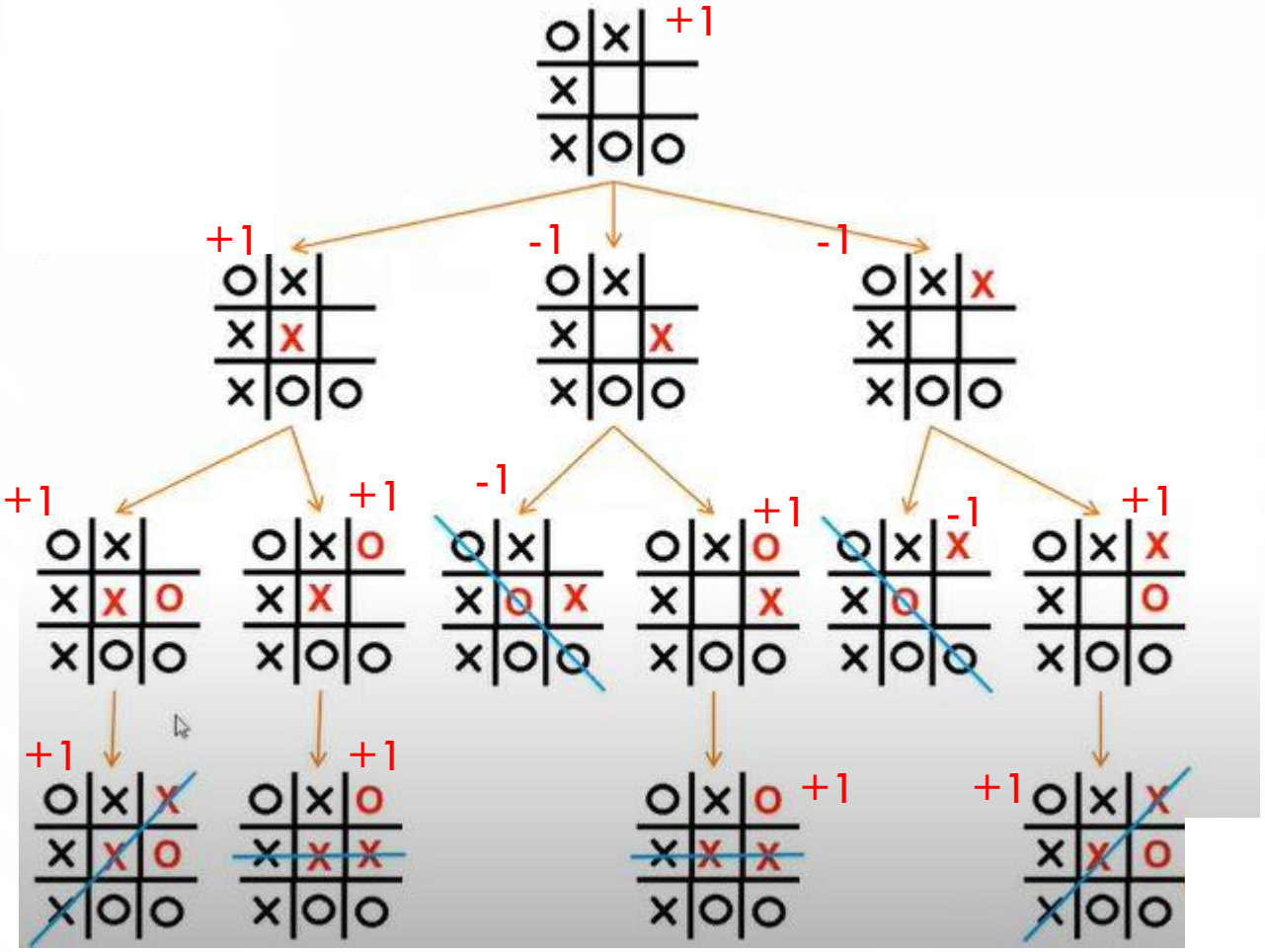
Busca HEURÍSTICA MIN MAX: exemplo jogo da velha 2

Seja jogada de MAX = X
Árvore com profundidade máxima
Função de avaliação
+1 vitória Max
-1 vitória Min
0 empate



Busca HEURÍSTICA MIN MAX: exemplo jogo da velha 2

Seja jogada de MAX = X
Árvore com profundidade máxima
Função de avaliação
+1 vitória Max
-1 vitória Min
0 empate



Busca HEURÍSTICA MIN MAX: exemplo jogo da velha 3

Seja jogada de MAX = X
Árvore com profundidade máxima
Função de avaliação
+1 vitória Max
-1 vitória Min
0 empate