

# Introdução à Inteligência Artificial

## PVP 3B – Procura Cega Algoritmos

José Coelho,  
2023



PVP 3 – Procura Cega de José Coelho é disponibilizado  
sob a Licença [Creative Commons-Atribuição -](#)  
[NãoComercial-Compartilhagual 4.0 Internacional](#)

# Índice

## 1. Algoritmos de procura

- 1. Procura em árvore
- 2. Procura em grafo

## 2. Performance

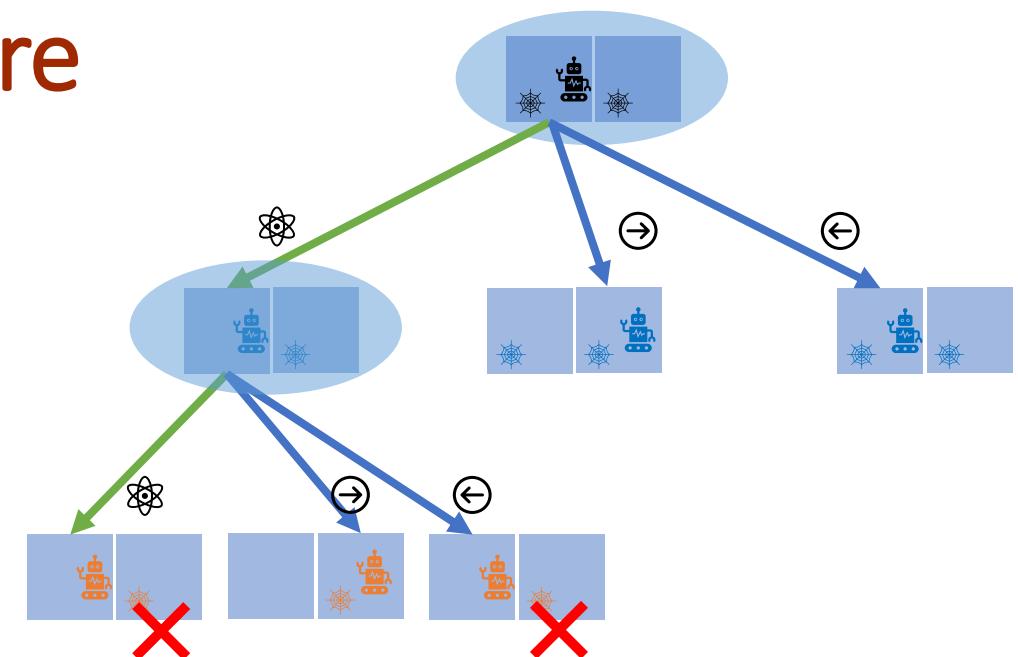
## 3. Procuras

- 1. Largura
- 2. Profundidade
- 3. Bidirecional

## 4. Atividades formativas

# Algoritmos – Procura em árvore

- Árvore de procura
- Ramificação
- Procura em árvore
  - Objetivo?
    - Se sim, retorna sucesso
  - Expandir (ações + execução)
    - [Procura em grafo] Não foi ainda gerado?
    - Adicionar sucessores à fronteira
  - Selecionar estado da fronteira



Estado igual ao pai ou ao avô?  
Estado igual a um ascendente?

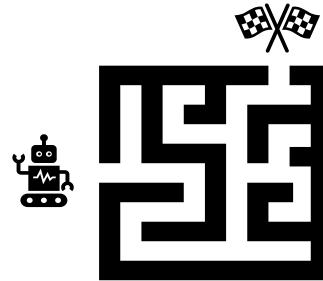
# Algoritmos - Performance

- Performance
  - Completo – consegue encontrar uma solução se esta existir?
  - Ótimo – consegue retornar a melhor solução?
  - Complexidade temporal – quanto tempo leva?
    - Unidade: nós gerados
  - Complexidade espacial – quanto espaço necessita?
    - Unidade: nós gerados em memória
- Indicadores:
  - b – ramificação: número máximo de sucessores num nó
  - d – profundidade solução – número de ações para o objetivo mais próximo
  - m – profundidade máxima – número máximo de ações na árvore de procura

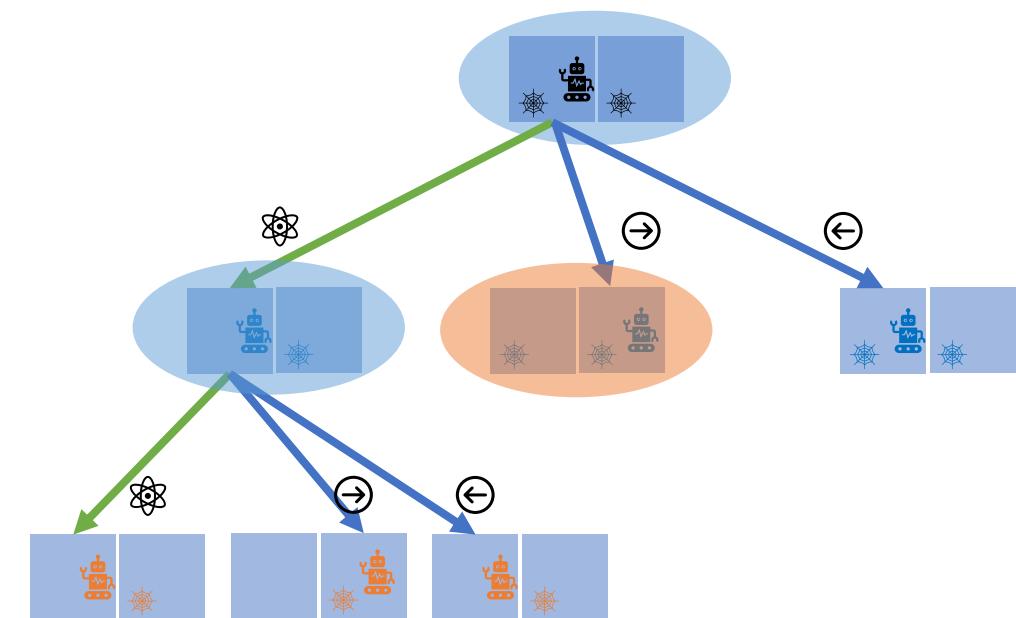
Problema	b	d	m
Aspirador	3	3	-
Puzzle 8	4	2	-
8 Damas	8	8	8
Partição	2	5	5

# Algoritmos Cegos: Largura

- Largura
  - Expandir estado gerado há mais tempo
  - Fronteira: pilha FIFO (first in first out)
  - Teste na geração
- Custo uniforme
  - Expandir estado com menor custo
  - Fronteira: pilha ordenada por custo
  - Teste na expansão



Problema	b	d	m	
Aspirador	3	3	-	✓
Puzzle 8	4	2	-	✓
8 Damas	8	8	8	✗
Partição	2	5	5	✗

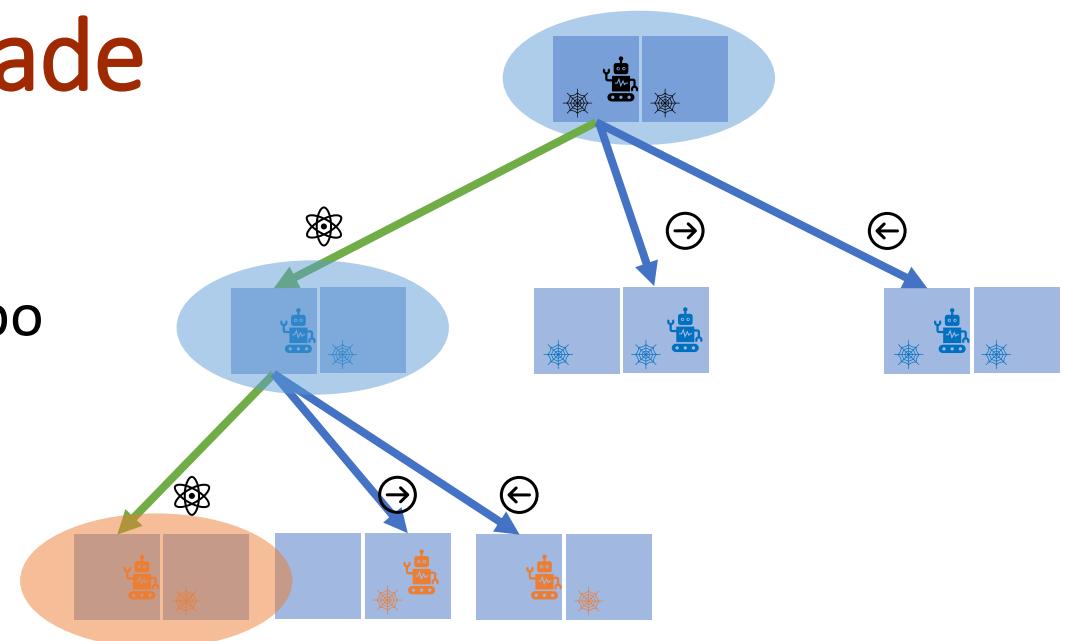


Performance	Largura
Completo	✓
Ótimo	✓ ✗
Complexidade Temporal	$O(b^d)$
Complexidade Espacial	$O(b^d)$

# Algoritmos Cegos: Profundidade

- Profundidade primeiro
  - Expandir estado gerado há menos tempo
  - Fronteira: pilha LIFO (last in first out)
- Profundidade limitada
  - Não gera sucessores após o limite
- Profundidade iterativo
  - Limite=0,1,...,d

Problema	b	d	m	P.	L.	I.
Aspirador	3	3	-	✗	✓	✓
Puzzle 8	4	2	-	✗	✓	✓
8 Damas	8	8	8	✓	✗	✗
Partição	2	5	5	✓	✗	✗



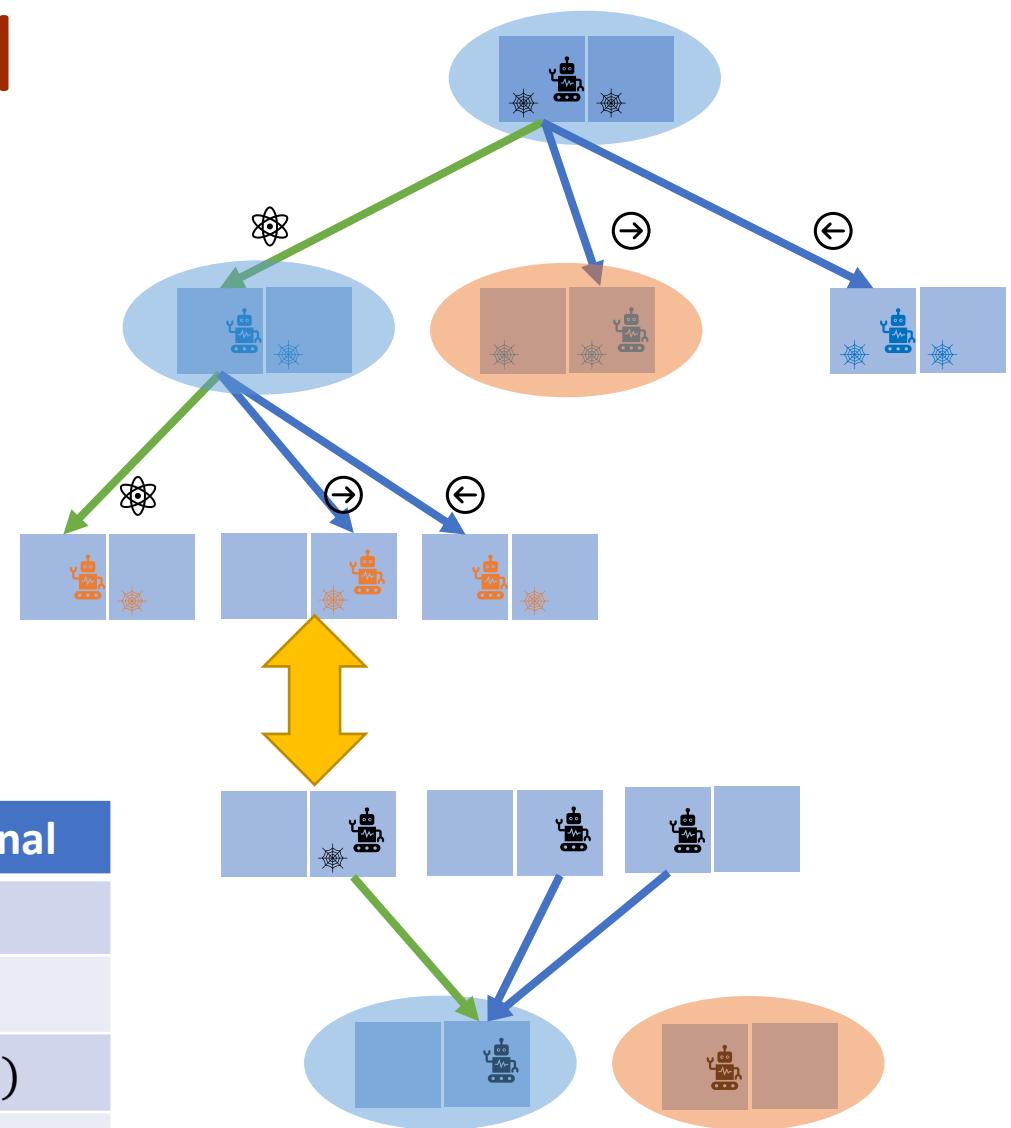
Performance	Profundidade	Limitado	Iterativo
Completo	✗	✗	✓
Ótimo	✗	✗	✓✗
Complexidade Temporal	$O(b^m)$	$O(b^l)$	$O(b^d)$
Complexidade Espacial	$O(bm)$	$O(bl)$	$O(bd)$
Um sucessor de cada vez: $O(m)$ Modifica o estado atual: $O(1)$			

# Algoritmos Cegos: Bidirecional

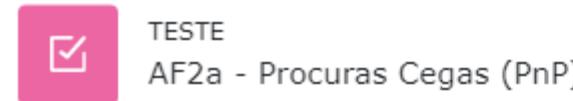
- Bidirecional
  - Duas árvores de procura:
    - Para a frente e para trás
  - Parar assim que as árvores se encontrem.

Problema	b	d	m	
Aspirador	3	3	-	✓
Puzzle 8	4	2	-	✓
8 Damas	8	8	8	✗
Partição	2	5	5	✗

Performance	Bidirecional
Completo	✓
Ótimo	✗
Complexidade Temporal	$O(b^{d/2})$
Complexidade Espacial	$O(b^{d/2})$



# Atividades formativas



 Exercícios de problemas do manual, para resolver com a técnica de papel-e-lápis (PnP)

Construções 20 (3x3)

Tentativas a realizar: 1



TESTE  
AF2c - Procuras Cegas (implementar)

 Exercícios em que se solicita a implementação um problema parecido com os do manual, podendo reutilizar código existente.



 Exercícios para executar código com a implementação de problemas do manual, e observar as diferentes características dos algoritmos na prática.

VPL: compilador online, com código contendo os algoritmos da UC, com código exemplo dos algoritmos e problemas do manual, para experimentação. O código está pronto a correr, mas pode fazer as alterações que considerar necessárias. [ Introdução ao código. | Código (2023). ]

Console: connected (Running: 11 seg)

```

Puzzle8 (TProcuraConstrutiva)
[Configurações] debug 0 | calcularCaminho 0 | limiteNível 10
[Estatísticas] expansões 30 | gerações 85 | avaliações 0
1 2 5
3 7 8
4 6 .

```

File list

```

JogoDoGalo.cpp
JogoDoGalo.h
OitoDamas.cpp
OitoDamas.h
Particao.cpp
Particao.h
ProcuraEngine.cpp
Puzzle8.cpp
Puzzle8.h
TProcuraAdversa.cpp
TProcuraAdversa.h
TProcuraConstrutiva.cpp
TProcuraConstrutiva.h
TProcuraConstrutivaCom...

```

```

1 - LimparEstatísticas | 2 - SoluçãoVazia | 3 - Heurística [Inicialização]
4 - LarguraPrimeiro | 5 - CustoUniforme | 6 - Prof. Primeiro [Procuras Cegas]
7 - MelhorPrimeiro | 8 - AStar [Procuras Informadas]
9 - debug | 10 - calcularCaminho | 11 - limiteNível [Configurações]
12 - Sucessores [Testes]
Opcão:
8 - excecuteJogoDoGalo();
9 - {
10 -
11 -
12 // Cria um objecto que é uma cópia deste
13 TProcuraConstrutiva* CjogoDoGalo::Duplicar(void)

```

# Recursos utilizados

- Microsoft Power Point
- Clipchamp, voz de síntese Duarte
- Vimeo
- Russell, S. J. & Norvig, P. (2010). Artificial intelligence: A modern approach (3rd ed). Prentice Hall.