

# Resolução de Problemas I

## Definição do Problema e Busca Cega

---

Prof. Karan Luciano

10 de fevereiro de 2026

Ciência da Computação

1. Agentes de Resolução de Problemas
2. Algoritmos de Busca
3. A Matemática da Busca (Explicada)
4. Busca em Largura (BFS)
5. Busca em Profundidade (DFS)
6. Outros Conceitos Importantes
7. Resumo do Dia

# **1. Agentes de Resolução de Problemas**

---

## O Objetivo






Um **Agente de Resolução de Problemas** é aquele que planeja antes de agir. Ele "pensa": *"Quais passos eu preciso dar para chegar onde eu quero?"*

### Como ele pensa (4 Passos):

1. **Objetivo:** Definir o destino (Ex: Chegar em Bucareste).
2. **Problema:** Entender o mapa e as regras.
3. **Busca:** Simular caminhos na "cabeça" até achar um que funcione.
4. **Execução:** Seguir o plano traçado no mundo real.

# Definindo o Problema

Para um computador resolver um problema, ele precisa responder a 5 perguntas:

-  **Onde estou?** (Estado Inicial)  
*Ex: Estou na cidade de Paris.*
-  **O que posso fazer?** (Ações)  
*Ex: Ir para Londres, Berlim ou Madri.*
-  **Onde vou parar?** (Modelo de Transição)  
*Ex: Se eu for para Londres, estarei em Londres.*
-  **Cheguei?** (Teste de Objetivo)  
*Ex: Estou em Roma? (Sim/Não).*
-  **Quanto custa?** (Custo do Caminho)  
*Ex: Gastei 340km de combustível.*

## Exemplo Prático: O GPS

Imagine o Waze/Google Maps calculando uma rota:

- **Estado:** Sua localização atual (GPS).
- **Ações:** Virar na próxima rua, seguir reto.
- **Transição:** Se virar à direita, entra na Rua X.
- **Objetivo:** Endereço de destino.
- **Custo:** Tempo (minutos) ou Distância (km).

*Todo GPS é um agente de resolução de problemas!*

## **2. Algoritmos de Busca**

---

A busca é a exploração de possibilidades.

## Nó vs. Estado (Analogia)

- **O Estado (s):** É o lugar físico no mapa (Ex: A cidade de Arad).
- **O Nó (n):** É o "post-it" que o algoritmo cola no mapa mental dizendo:  
*"Cheguei em Arad vindo de Zerind, e já andei 150km."*

Os algoritmos diferem na **ordem** em que colam esses post-its.



### **3. A Matemática da Busca (Explicada)**

---

## Quantos caminhos existem? (O Problema da Senha)

Imagine que você esqueceu a senha de um cadeado numérico.

### Vamos calcular passo-a-passo:

- Se a senha tem **1 dígito** (0-9):  
Você tenta 10 vezes. Simples.
- Se a senha tem **2 dígitos**:  
Para cada um dos 10 primeiros, você tenta 10 segundos.  
 $10 \times 10 = 100$  tentativas.
- Se a senha tem **3 dígitos**:  
 $10 \times 10 \times 10 = 1000$  tentativas.



Podemos criar uma "regra" para saber o trabalho total:

$$N = b^d$$

Vamos "traduzir" essa fórmula:

- **N** é o Número total de tentativas (o trabalho do computador).
- **b** é o "Fator de Ramificação"(quantas escolhas tenho agora?).
- **d** é a "Profundidade"(quantos passos até o fim?).

**Lê-se:** "O trabalho é igual ao número de escolhas vezes ele mesmo,  $d$  vezes."

## Por que computadores travam? (Explosão Combinatória)

Essa fórmula inocente ( $b^d$ ) é perigosa. Veja:

Suponha um jogo de Xadrez onde você tem 35 opções de jogada ( $b = 35$ ) e quer ver 10 jogadas à frente ( $d = 10$ ).

$$35^{10} = 2.758.547.353.515.625$$

**Isso são 2 Quadrilhões de possibilidades!**

### Conclusão

É por isso que não conseguimos calcular o "jogo perfeito" até o fim. O número cresce rápido demais para qualquer computador do universo.

## O Dilema:

- Temos **trilhões** de caminhos possíveis.
- Não temos tempo nem memória para testar todos.

**A Solução:** Precisamos de uma **Ordem Inteligente** de busca.

*"Qual caminho eu devo testar primeiro?  
O mais curto? Ou o que vai mais longe?"*



*Para onde ir?*

## **4. Busca em Largura (BFS)**

---

## A Ideia

Explore tudo o que está perto antes de ir para longe.

### **Analogia: A Pedra no Lago**

Quando você joga uma pedra, a onda se espalha em círculos perfeitos.

- Primeiro atinge quem está a 1 metro.
- Depois quem está a 2 metros...

### **Consequência:**

Se a solução estiver perto, o BFS **garante** encontrá-la rapidamente e pelo caminho mais curto!



*Expansão em Camadas*

## **5. Busca em Profundidade (DFS)**

---



## ↓ A Ideia

Escolha um caminho e vá até o fim. Se der errado, volte um pouco e tente outro.

### Analogia: O Labirinto

*"Sempre vire à esquerda e siga em frente."*

- Você se aprofunda rapidamente.
- Se encontrar um beco sem saída (*dead-end*), você volta (*backtrack*) até a última encruzilhada.



*Fundo primeiro...*

**Risco:** Você pode pegar o caminho mais longo do mundo sem querer!

## **6. Outros Conceitos Importantes**

---

## Busca de Custo Uniforme (O "Pão-Duro")

E se os caminhos tiverem custos diferentes? (Ex: Uma estrada é de terra, a outra é asfaltada).

### Busca de Custo Uniforme (UCS):

- O BFS sempre escolhe o caminho com *menos passos*.
- O UCS sempre escolhe o caminho com *menor custo total* acumulado até agora.

Ele age como um turista econômico: *"Sempre pego o próximo trecho mais barato, não importa quantos trechos sejam."*

# O Perigo dos Loops (Cachorro correndo atrás do rabo)

Em um mapa, você pode andar em círculos:

$$A \rightarrow B \rightarrow C \rightarrow A \rightarrow \dots$$

**O Problema:** Algoritmos simples (como o DFS) podem ficar presos nesse loop **para sempre**.

**A Solução:** Precisamos de uma "Lista de Visitados". Antes de ir para um lugar, o agente verifica: *"Eu já passei por aqui?"*



*Loop Infinito*

## **7. Resumo do Dia**

---

## Qual usar?

	BFS (Largura)	DFS (Profundidade)
<b>Lema</b>	"Devagar e sempre"	"Audacioso e rápido"
<b>Garante o melhor?</b>	Sim (Sempre acha o mais curto)	Não (Pode dar voltas)
<b>Memória</b>	Gasta Muita (Explode)	Gasta Pouca (Eficiente)
<b>Use quando...</b>	A solução está perto	O mapa é muito grande

**Obrigado!**

Dúvidas?