

# Introdução à Análise de Algoritmos

## Maratona de Programação FEI

Prof. Charles Ferreira  
cferreira@fei.edu.br

# Agenda

**Algoritmos**

**Algoritmos Eficientes**

*Introdução à Análise  
de Algoritmos*

Algoritmos

O que são algoritmos



# Algoritmo

**Qualquer procedimento computacional bem definido que:**

- tenha um valor ou um conjunto de valores de entrada ...
- e produz algum valor ou conjunto de valores de saída

**Objetivo:**

- Resolução de problemas computacionais

Para que estudar técnicas de desenvolvimento de  
algoritmos



# Motivação

**Criar algoritmos para solucionar problemas é MUITO fácil:**

# Motivação

**Criar algoritmos para solucionar problemas é MUITO fácil:**

- Exemplo → Programa que testa todas as possibilidades



# Motivação

**Criar algoritmos para solucionar problemas é *MUITO* fácil:**

- Exemplo → Programa que testa todas as possibilidades
- O problema é que essa solução pode não ter um tempo aceitável ...



# Motivação

**Então o desafio é criar algoritmos eficientes**

Que consigam solucionar problemas em tempo aceitável

## Motivação

O que fazer se seu programa está demorando para rodar?



# Possível Solução



Troca por um  
computador  
mais rápido ?



# Motivação

**Trocar o computador pode amenizar momentaneamente**

- Mas não resolve o problema

# Motivação

**Trocar o computador pode amenizar momentaneamente**

- Mas não resolve o problema

Você precisa melhorar o **algoritmo!**

# Motivação

## Trocar o computador pode amenizar momentaneamente

- Mas não resolve o problema

Você precisa melhorar o **algoritmo!**

- Torná-lo eficiente.
- Caso contrário, trocar de hardware só irá adiar o problema.

Como fazer um algoritmo eficiente





# Metodologia

Análise e Técnica

# Metodologia

Análise e Técnica

- **Analizando** a complexidade do problema

# Metodologia

Análise e Técnica

- **Analizando** a complexidade do problema
- **Analizando** a complexidade algoritmo proposto

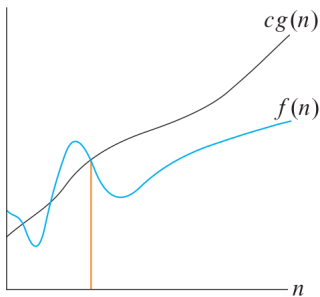
# Metodologia

## Análise e Técnica

- **Analizando** a complexidade do problema
- **Analizando** a complexidade algoritmo proposto
- Aplicando **técnicas** de programação.

# Metodologia

- **O que seria essa análise?**
  - Analisar o comportamento do algoritmo ...
  - mediante um número  $n$  de entrada.



# Metodologia

- **O que seriam essas tais técnicas de programação?**
  - Estrutura de dados avançadas.
  - Métodos de solução e divisão de problemas.
  - Programação dinâmica.
  - Algoritmos gulosos.
  - Outras.

# Exemplo

# Exemplo

## Achando o maior número em um vetor

- Dado um vetor de tamanho  $n$ :

8	20	12	13	1	28	10	11	5	25	12	21	3
---	----	----	----	---	----	----	----	---	----	----	----	---

- Como achar o maior número armazenado no vetor?



# Pseudocódigo

```
1 | achar_maior_valor(v, n)
2 |   maior = v[0]
3 |   para i <- 0 ate n
4 |     se v[i] > maior
5 |       maior <- v[i]
6 |   retorna maior
```

## Análise: contar o número de instruções primitivas

achar_maior_valor(v, n)	custo $\rightarrow$ número de execuções
maior = v[0]	$c_1 \rightarrow 1$
para i <- 0 ate n	$c_2 \rightarrow n + 1$
se v[i] > maior	$c_3 \rightarrow n$
maior <- v[i]	$c_4 \rightarrow n$
retorna maior	

## Exercício 1

**Dado um vetor com tamanho  $n$  e um valor  $k$  a ser buscado:**

## Exercício 1

**Dado um vetor com tamanho  $n$  e um valor  $k$  a ser buscado:**

- Faça um **pseudocódigo** que determine se o valor  $k$  existe
- Seu programa deve retornar **verdadeiro** ou **falso**

## Exercício 1

**Dado um vetor com tamanho  $n$  e um valor  $k$  a ser buscado:**

- Faça um **pseudocódigo** que determine se o valor  $k$  existe
- Seu programa deve retornar **verdadeiro** ou **falso**
- Faça a análise de complexidade do seu código

## Exercício 2

**Faça um pseudocódigo para uma função que:**

- Receba como entrada uma matriz  $n \times n$ .
- Coloque zero em todas as posições da matriz.
- Faça a análise de complexidade do seu código.

## Exercício 3

Determine a complexidade dos seguintes algoritmos:

```
1 | x = x + 1
2 | para i de 1 ate n
3 |     m = m + 2
4 | para i de 1 ate n
5 |     para j de 1 ate n
6 |         k = k + 1
```

```
1 | x = x + 1
2 | para i de 1 ate n
3 |     para j de 1 ate n
4 |         imprima("Hello")
5 |         break
```

*Introdução à Análise  
de Algoritmos*

Algoritmos Eficientes



O que é ser eficiente



# Algoritmos Eficientes

## Eficiência

- Em computação, ser eficiente é ter um algoritmo **polinomial**.
- Ou seja, que o tempo  $T$  seja proporcional a algo como  $n^x$ 
  - em que  $n$  é o tamanho da entrada de dados

Então o que seria ineficiência



# Algoritmos Ineficientes

## Ineficiência

- Algoritmos exponenciais ( $2^n, 3^n, \dots n^n$ )
- Estes algoritmos geralmente utilizam Força Bruta
- Ou seja, testam todas as possibilidades para encontrar a melhor solução

Todos os problemas tem solução polinomial

?

# Tipos de problemas

## Há problemas ditos **Intratáveis**

- São problemas que não possuem soluções polinomiais

# Tipos de problemas

Há problemas ditos **Intratáveis**

- São problemas que não possuem soluções polinomiais

**Quer dizer que estes problemas não tem solução?**

# Tipos de problemas

## Há problemas ditos **Intratáveis**

- São problemas que não possuem soluções polinomiais

## **Quer dizer que estes problemas não tem solução?**

- Eles têm solução.
- Porém, não é possível obter soluções ótimas para estes problemas de forma eficiente.



# Problemas Intratáveis

## Como solucionar estes problemas **Intratáveis**?

- Se  $n$  for pequeno, use força bruta
- Se  $n$  for grande, teremos que usar técnicas para encontrar soluções sub-ótimas ou aproximadas (Inteligência Artificial)

# Problema Intratável (exemplo)

- Descobrir uma senha é um exemplo de problema **intratável**
  - Problemas intratáveis só possuem soluções exponenciais
  - Nesta caso ...ainda bem. Veja tabela ao lado

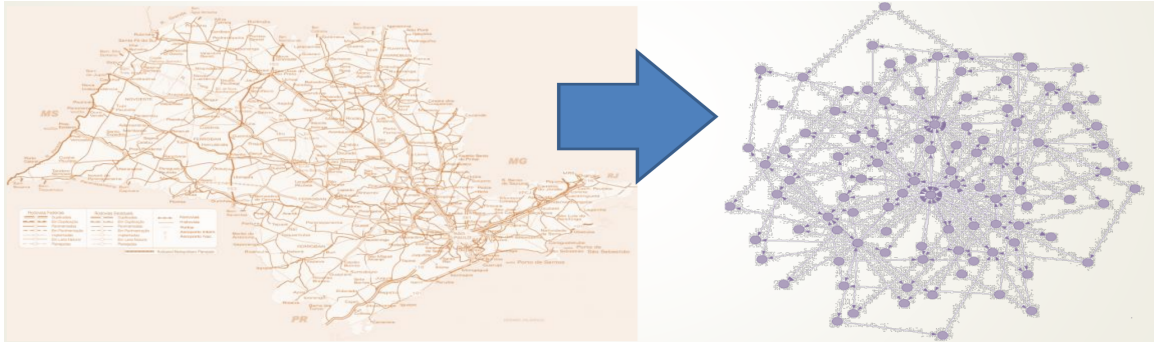
**Mas muitos problemas são tratáveis**

- E podem ter algoritmos eficientes para solucioná-los

Tamanho da senha (só letras minúsculas)	Ataque com teste de 10 senhas por se- gundo
1	2 segundos
2	1 minuto
3	30 min
4	12 horas
5	14 dias
6	1 ano
7	10 anos
8	19 anos
9	26 anos
10	37 anos
11	46 anos
12	55 anos
13	64 anos
14	73 anos
15	82 anos
16	91 anos
17	100 anos

# Exemplo de Eficiência

Encontrar a menor distância de uma cidade a todas as outras

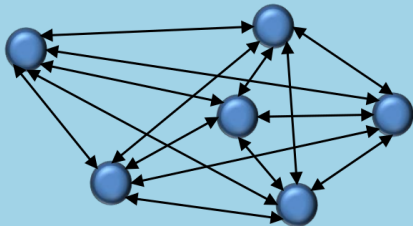


Considere um grafo denso

## Grafo denso

Cada Vértice  $V$  está conectado a todos os demais vértices por arestas ( $E$ )

$$E = V(V-1)$$



## GRAFO DENSO



# Exemplo de eficiência

## Usando força bruta

- Testa-se todas as possibilidades para  $V$  cidades
- Exponencial (exp)  $\rightarrow$  proporcional a  $2^v$

## Usando o algoritmo de Dijkstra

- Polinomial  $\rightarrow$  proporcional a  $V^2$

## Exemplo de eficiência: Menor distância

Estado de São Paulo  
possui aproximadamente  
650 cidades

Processador	Velocidade	Algoritmo	Cidades	Tempo	Complexidade
Core i7 Extreme					

## Exemplo de eficiência: Menor distância

Estado de São Paulo  
possui aproximadamente  
650 cidades

Processador	Velocidade	Algoritmo	Cidades	Tempo	Complexidade
Core i7 Extreme	100 milhões de instr. / seg.				

## Exemplo de eficiência: Menor distância

Estado de São Paulo  
possui aproximadamente  
650 cidades

Processador	Velocidade	Algoritmo	Cidades	Tempo	Complexidade
Core i7 Extreme	100 milhões de instr. / seg.	Força Bruta			



## Exemplo de eficiência: Menor distância

Estado de São Paulo  
possui aproximadamente  
650 cidades

Processador	Velocidade	Algoritmo	Cidades	Tempo	Complexidade
Core i7 Extreme	100 milhões de instr. / seg.	Força Bruta	70		

# Exemplo de eficiência: Menor distância

Estado de São Paulo  
possui aproximadamente  
650 cidades

Processador	Velocidade	Algoritmo	Cidades	Tempo	Complexidade
Core i7 Extreme	100 milhões de instr. / seg.	Força Bruta	70	187 anos	Exp

# Exemplo de eficiência: Menor distância

Estado de São Paulo  
possui aproximadamente  
650 cidades

Processador	Velocidade	Algoritmo	Cidades	Tempo	Complexidade
Core i7 Extreme	100 milhões de instr. / seg.	Força Bruta	70	187 anos	Exp

Super Computador

## Exemplo de eficiência: Menor distância

Estado de São Paulo  
possui aproximadamente  
650 cidades

Processador	Velocidade	Algoritmo	Cidades	Tempo	Complexidade
Core i7 Extreme	100 milhões de instr. / seg.	Força Bruta	70	187 anos	Exp
Super Computador	100 trilhões de instr. / seg.				

## Exemplo de eficiência: Menor distância

Estado de São Paulo  
possui aproximadamente  
650 cidades

Processador	Velocidade	Algoritmo	Cidades	Tempo	Complexidade
Core i7 Extreme	100 milhões de instr. / seg.	Força Bruta	70	187 anos	Exp
Super Computador	100 trilhões de instr. / seg.	Força Bruta			

## Exemplo de eficiência: Menor distância

Estado de São Paulo  
possui aproximadamente  
650 cidades

Processador	Velocidade	Algoritmo	Cidades	Tempo	Complexidade
Core i7 Extreme	100 milhões de instr. / seg.	Força Bruta	70	187 anos	Exp
Super Computador	100 trilhões de instr. / seg.	Força Bruta	70		

# Exemplo de eficiência: Menor distância

Estado de São Paulo  
possui aproximadamente  
650 cidades

Processador	Velocidade	Algoritmo	Cidades	Tempo	Complexidade
Core i7 Extreme	100 milhões de instr. / seg.	Força Bruta	70	187 anos	Exp
Super Computador	100 trilhões de instr. / seg.	Força Bruta	70	3 horas	Exp

# Exemplo de eficiência: Menor distância

Estado de São Paulo  
possui aproximadamente  
650 cidades

Processador	Velocidade	Algoritmo	Cidades	Tempo	Complexidade
Core i7 Extreme	100 milhões de instr. / seg.	Força Bruta	70	187 anos	Exp
Super Computador	100 trilhões de instr. / seg.	Força Bruta	70	3 horas	Exp
Super Computador					



## Exemplo de eficiência: Menor distância

Estado de São Paulo  
possui aproximadamente  
650 cidades

Processador	Velocidade	Algoritmo	Cidades	Tempo	Complexidade
Core i7 Extreme	100 milhões de instr. / seg.	Força Bruta	70	187 anos	Exp
Super Computador	100 trilhões de instr. / seg.	Força Bruta	70	3 horas	Exp
Super Computador	100 trilhões de instr. / seg.				

# Exemplo de eficiência: Menor distância

Estado de São Paulo  
possui aproximadamente  
650 cidades

Processador	Velocidade	Algoritmo	Cidades	Tempo	Complexidade
Core i7 Extreme	100 milhões de instr. / seg.	Força Bruta	70	187 anos	Exp
Super Computador	100 trilhões de instr. / seg.	Força Bruta	70	3 horas	Exp
Super Computador	100 trilhões de instr. / seg.	Força Bruta			

# Exemplo de eficiência: Menor distância

Estado de São Paulo  
possui aproximadamente  
650 cidades

Processador	Velocidade	Algoritmo	Cidades	Tempo	Complexidade
Core i7 Extreme	100 milhões de instr. / seg.	Força Bruta	70	187 anos	Exp
Super Computador	100 trilhões de instr. / seg.	Força Bruta	70	3 horas	Exp
Super Computador	100 trilhões de instr. / seg.	Força Bruta	90		

# Exemplo de eficiência: Menor distância

Estado de São Paulo  
possui aproximadamente  
650 cidades

Processador	Velocidade	Algoritmo	Cidades	Tempo	Complexidade
Core i7 Extreme	100 milhões de instr. / seg.	Força Bruta	70	187 anos	Exp
Super Computador	100 trilhões de instr. / seg.	Força Bruta	70	3 horas	Exp
Super Computador	100 trilhões de instr. / seg.	Força Bruta	90	392 anos	Exp

# Exemplo de eficiência: Menor distância

Estado de São Paulo  
possui aproximadamente  
650 cidades

Processador	Velocidade	Algoritmo	Cidades	Tempo	Complexidade
Core i7 Extreme	100 milhões de instr. / seg.	Força Bruta	70	187 anos	Exp
Super Computador	100 trilhões de instr. / seg.	Força Bruta	70	3 horas	Exp
Super Computador	100 trilhões de instr. / seg.	Força Bruta	90	392 anos	Exp
Core i7 Extreme					

## Exemplo de eficiência: Menor distância

Estado de São Paulo  
possui aproximadamente  
650 cidades

Processador	Velocidade	Algoritmo	Cidades	Tempo	Complexidade
Core i7 Extreme	100 milhões de instr. / seg.	Força Bruta	70	187 anos	Exp
Super Computador	100 trilhões de instr. / seg.	Força Bruta	70	3 horas	Exp
Super Computador	100 trilhões de instr. / seg.	Força Bruta	90	392 anos	Exp
Core i7 Extreme	100 milhões de instr. / seg.				

## Exemplo de eficiência: Menor distância

Estado de São Paulo  
possui aproximadamente  
650 cidades

Processador	Velocidade	Algoritmo	Cidades	Tempo	Complexidade
Core i7 Extreme	100 milhões de instr. / seg.	Força Bruta	70	187 anos	Exp
Super Computador	100 trilhões de instr. / seg.	Força Bruta	70	3 horas	Exp
Super Computador	100 trilhões de instr. / seg.	Força Bruta	90	392 anos	Exp
Core i7 Extreme	100 milhões de instr. / seg.	<b>Dijkstra</b>			

## Exemplo de eficiência: Menor distância

Estado de São Paulo  
possui aproximadamente  
650 cidades

Processador	Velocidade	Algoritmo	Cidades	Tempo	Complexidade
Core i7 Extreme	100 milhões de instr. / seg.	Força Bruta	70	187 anos	Exp
Super Computador	100 trilhões de instr. / seg.	Força Bruta	70	3 horas	Exp
Super Computador	100 trilhões de instr. / seg.	Força Bruta	90	392 anos	Exp
Core i7 Extreme	100 milhões de instr. / seg.	<b>Dijkstra</b>	<b>650</b>		



## Exemplo de eficiência: Menor distância

Estado de São Paulo  
possui aproximadamente  
650 cidades

Processador	Velocidade	Algoritmo	Cidades	Tempo	Complexidade
Core i7 Extreme	100 milhões de instr. / seg.	Força Bruta	70	187 anos	Exp
Super Computador	100 trilhões de instr. / seg.	Força Bruta	70	3 horas	Exp
Super Computador	100 trilhões de instr. / seg.	Força Bruta	90	392 anos	Exp
Core i7 Extreme	100 milhões de instr. / seg.	<b>Dijkstra</b>	<b>650</b>	<b>2 micro segundos</b>	Polinomial

## Como ser eficiente (Exemplo)

**Como seria um algoritmo para buscar um valor em um vetor?**

- Qual seria a complexidade?

## Como ser eficiente (Exemplo)

**Como seria um algoritmo para buscar um valor em um vetor?**

- Qual seria a complexidade?
  - proporcional ao tamanho do vetor

## Como ser eficiente (Exemplo)

**Como seria um algoritmo para buscar um valor em um vetor?**

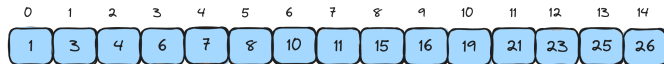
- Qual seria a complexidade?
  - proporcional ao tamanho do vetor

**Agora considere que o vetor já esteja ordenado ...**

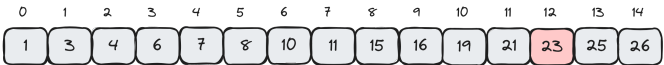
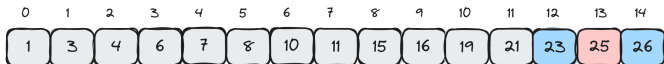
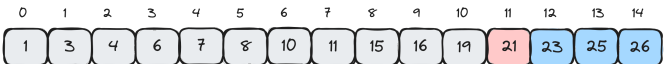
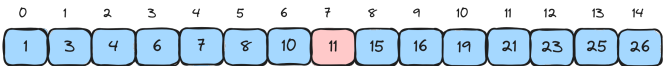
- Faz alguma diferença?

# Como ser eficiente (Exemplo)

## Demonstrando visualmente



23?



## Como ser eficiente (Exemplo)

Vamos colocar alguns números ...

Tamanho do vetor	Proporcional $n$	Proporcional $\log_2 n$
100		

## Como ser eficiente (Exemplo)

Vamos colocar alguns números ...

Tamanho do vetor	Proporcional $n$	Proporcional $\log_2 n$
100	100 iterações	

## Como ser eficiente (Exemplo)

Vamos colocar alguns números ...

Tamanho do vetor	Proporcional $n$	Proporcional $\log_2 n$
100	100 iterações	7 iterações



## Como ser eficiente (Exemplo)

Vamos colocar alguns números ...

Tamanho do vetor	Proporcional $n$	Proporcional $\log_2 n$
100	100 iterações	7 iterações
10.000		

## Como ser eficiente (Exemplo)

Vamos colocar alguns números ...

Tamanho do vetor	Proporcional $n$	Proporcional $\log_2 n$
100	100 iterações	7 iterações
10.000	10.000 iterações	

## Como ser eficiente (Exemplo)

Vamos colocar alguns números ...

Tamanho do vetor	Proporcional $n$	Proporcional $\log_2 n$
100	100 iterações	7 iterações
10.000	10.000 iterações	13 iterações

## Como ser eficiente (Exemplo)

Vamos colocar alguns números ...

Tamanho do vetor	Proporcional $n$	Proporcional $\log_2 n$
100	100 iterações	7 iterações
10.000	10.000 iterações	13 iterações
1.000.000		

## Como ser eficiente (Exemplo)

Vamos colocar alguns números ...

Tamanho do vetor	Proporcional $n$	Proporcional $\log_2 n$
100	100 iterações	7 iterações
10.000	10.000 iterações	13 iterações
1.000.000	1.000.000 iterações	

## Como ser eficiente (Exemplo)

Vamos colocar alguns números ...

Tamanho do vetor	Proporcional $n$	Proporcional $\log_2 n$
100	100 iterações	7 iterações
10.000	10.000 iterações	13 iterações
1.000.000	1.000.000 iterações	20 iterações

Por quê a busca binária precisa apenas de  $\log_2 n$   
iterações

?

Obrigado

[cferreira@fei.edu.br](mailto:cferreira@fei.edu.br)