



# Análise de Algoritmos

Introdução à Teoria da Computação  
E  
COMPLEXIDADE DE PROBLEMAS



# Visão geral

Teoria da Computação  
COMPLEXIDADE DE PROBLEMAS



# Teoria da Computação

Área dedicada à definições ligadas à perguntas do tipo:

- Quais as capacidades fundamentais dos computadores?
- Quais suas limitações?
- Quais problemas do mundo real podem ser tratados computacionalmente?
- Quais problemas não podem ser tratados eficientemente?
- Como avaliar e provar?



# Teoria da Computação

Exemplos de problemas solucionáveis com a computação

- Cálculo de multiplicação de números inteiros
- Resolução de sistemas de equações lineares
- Combinação de pares de convidados para um jantar de modo a manter satisfação com os seus vizinhos



# Teoria da Computação

Diferentes formas de solucionar os problemas:  
Multiplicação de números inteiros

- Somas sucessivas
  - $A \times B$  é obtido somando A com si mesmo por  $B-1$  vezes
    - Para  $A=577$  e  $B=423$ , são necessárias 422 somas  
 $M = (A+A+A+A+A+.....+A+A)$



# Teoria da Computação

Diferentes formas de solucionar os problemas:  
Multiplicação de números inteiros

- Multiplicações e somas sucessivas
  - $A \times B$  é obtido somando os resultados da multiplicação de cada dígito de B por A
    - Para  $A=577$  e  $B=423$ , são necessárias 3 multiplicações e duas somas

$$\begin{array}{r} & 5 & 7 & 7 \\ \times & 4 & 2 & 3 \\ \hline & 1 & 7 & 3 & 1 \\ & 1 & 1 & 5 & 4 \\ + & 2 & 3 & 0 & 8 \\ \hline & 2 & 4 & 4 & 0 & 7 & 1 \end{array}$$



# Teoria da Computação

Diferentes formas de solucionar os problemas:  
Multiplicação de números inteiros

Avaliação quanto ao número de dígitos

Para um número com “n” dígitos

- Somas sucessivas :  $n10^{(n-1)}$
- Multiplicações e somas sucessivas:  $2n^2$

Resultado: para números grandes (11 dígitos ou mais) um computador simples usando o segundo método será mais rápido que um supercomputador usando o primeiro método



# Teoria da Computação

Outros exemplos:

a) Calcular o determinante de uma matriz  $n \times n$

<b>n</b>	<b>Método de Cramer</b>	<b>Método de Gauss</b>
2	22 $\mu$ s	50 $\mu$ s
3	102 $\mu$ s	159 $\mu$ s
4	456 $\mu$ s	353 $\mu$ s
5	2,35 ms	666 $\mu$ s
10	1,19 min	4,95 ms
20	15 225 séculos	38,63 ms
40	$5 \cdot 10^{33}$ séculos	0,315 s

b) Métodos de ordenação

<http://www.sorting-algorithms.com/>



# Teoria da Computação

Outros exemplos:

c) Resolução de equações lineares

Método de eliminação de Gauss

$O(n^3)$  operações para n equações e n variáveis

Método de Strassen (anos 60)

$O(n^{2,3})$  operações para n equações e n variáveis

[para n=4 Gauss usa 64 e Strassen usa 24 oper.]

[para n=10 Gauss usa 1000 e Strassen usa 199 oper.]

d) Organização de pares para jantar

Usando um algoritmo simples (testar todas as combinações de subconjuntos), torna-se inviável o cálculo para organizar um grupo de 70 pessoas.



# Teoria da Computação

## Resolução de Problemas

- Abordagem utilizada é determinante do resultado
- Existem diversos modelos e possibilidades
- É fundamental avaliar previamente os resultados possíveis
- É necessário comparar abordagens
- Em casos de abordagens ineficientes
  - Complemento com técnicas de apoio (divisão, probabilidade, redução)



# Teoria da Computação

Historicamente:

- Originada na década de 30
- Foco inicial no significado da computação

Atualmente:

- Capacidade de computação aumentando
- Necessidade de abordagens integrando prática e teoria
- Eficiência, tratabilidade, decidibilidade
- Tratamento de problemas difíceis/complexos

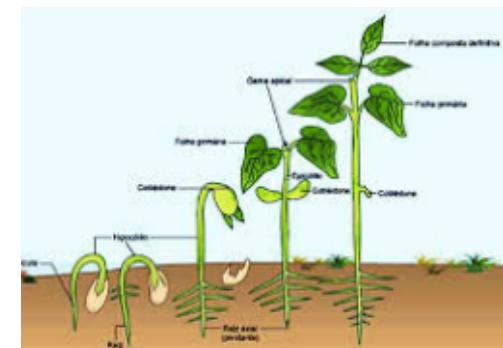
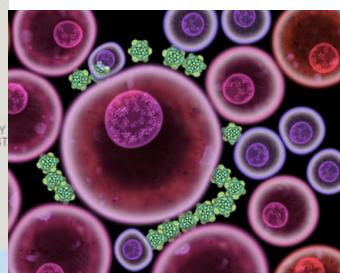


# Teoria da Computação

## A computação e a natureza



Qua-19/07	Qui-20/07	Sex-21/07	Sab-22/07	Dom-23/07
POUCAS NUvens	POUCAS NUvens	CEU CLARO	POUCAS NUvens	PANCADAS
15°C	13°C	16°C	18°C	14°C
24°C	25°C	26°C	27°C	22°C
0mm	0mm	0mm	0mm	14mm
W 22km/h	NW 30km/h	NW 19km/h	NEW 24km/h	E 19km/h
UV moderado	UV moderado	UV moderado	UV moderado	UV moderado
• 7h15m	• 7h15m	• 7h15m	• 7h15m	• 7h15m
• 17h45m	• 17h45m	• 17h45m	• 17h45m	• 17h45m



Fonte: Arora, Sypser



# Teoria da Computação

Áreas centrais de estudo:

- Complexidade
  - Como identificar problemas fáceis/difíceis?
- Computabilidade
  - Como identificar problemas solucionáveis?
- Automatos
  - Como definir e modelar matematicamente o computador?



# Complexidade – classes de problemas

Como identificar problemas fáceis/difíceis?

- Análise de complexidade de algoritmos
  - Tempo, memória
- Complexidade de problemas
  - Classes de complexidade
    - Problemas P, NP, NP-Completos

# Complexidade – classes de problemas



## Problemas P, NP, NP-Completos

P – problemas com solução polinomial (Deterministic Polynomial Time)

**P**



# Complexidade – classes de problemas

## Problemas P, NP, NP-Completos

P – problemas com solução polinomial (Deterministic Polynomial Time)

**Pode ser calculado em tempo polinomial  
(ex:  $x+3$ ;  $2x^3+5$ ;  $3x^2+43n - 56$ )**

Multiplicação  
Ordenação  
...

**P**

# Complexidade – classes de problemas



## Problemas P, NP, NP-Completos

P – problemas com solução polinomial (Deterministic Polynomial Time)

NP – problemas sem solução polinomial conhecida, verificável em tempo polinomial

**P**

**NP**

Rotas,  
criptografia,  
alocação  
...

# Complexidade – classes de problemas



## Problemas P, NP, NP-Completos

P – problemas com solução polinomial (Deterministic Polynomial Time)

NP – problemas sem solução polinomial conhecida, verificável em tempo polinomial

**Pode ser VERIFICADO em tempo  
polinomial**

Rotas,  
criptografia,  
alocação  
...

**P**

**NP**

Pode ser VERIFICADO em tempo polinomial

EXEMPLO:

SUDOKU – PODE DEMORAR MUITO PARA CALCULAR  
- DADA UMA SOLUÇÃO, O TEMPO PARA CONFERIR  
SE A SOLUÇÃO É CORRETA É PEQUENO.

	4	6	2		8		1	
			1	8			7	
	2							
2							8	
		5	7				2	
3	5			4			9	
		1			3			
6	9							
1	4	3	6	9	2			

4	6	7	9	2	1	3	5	8
8	9	5	4	7	3	2	6	1
2	3	1	8	6	5	7	4	9
5	1	3	6	9	8	4	2	7
9	2	8	7		4	6	1	3
7	4	6	1	3	2	9	8	5
3	5	4	2	8	7	1	9	6
1	8	9	3	4	6	5	7	2
6	7	2	5	1	8	3	9	4

5	8	3	7	2	9	4	6	1
4	2	9	8	6	1	7	3	5
1	6	7	3	4	5	8	9	2
7	9	5	6	8	4	2	1	3
8	3	1	9	7	2	6	5	4
2	4	6	1	5	3	9	7	8
9	7	2	5	1	8	4	3	6
6	1	4	2	3	7	5	8	9
3	5	8	4	9	6	1	2	7
5	6	8	2	7	9	3	4	5
1	2	3	6	5	4	8	7	9
4	7	5	9	2	1	8	6	3
3	4	5	8	1	9	7	6	2
2	6	8	4	5	7	3	9	1
1	7	9	2	3	6	5	4	8
8	9	2	1	6	5	3	7	4
5	3	6	7	4	2	1	8	9
4	1	7	9	8	3	5	2	6
6	5	1	3	7	4	2	9	8
7	2	3	6	9	8	4	1	5
9	8	4	5	2	1	6	7	3
3	4	5	8	1	9	7	6	2
2	6	8	4	5	7	3	9	1
1	7	9	2	3	6	5	4	8
8	9	2	1	6	5	3	7	4
5	3	6	7	4	2	1	8	9
4	1	7	9	8	3	5	2	6
6	5	1	3	7	4	2	9	8
7	2	3	6	9	8	4	1	5
9	8	4	5	2	1	6	7	3
3	4	5	8	1	9	7	6	2
2	6	8	4	5	7	3	9	1
1	7	9	2	3	6	5	4	8
8	9	2	1	6	5	3	7	4
5	3	6	7	4	2	1	8	9
4	1	7	9	8	3	5	2	6
6	5	1	3	7	4	2	9	8
7	2	3	6	9	8	4	1	5
9	8	4	5	2	1	6	7	3
3	4	5	8	1	9	7	6	2
2	6	8	4	5	7	3	9	1
1	7	9	2	3	6	5	4	8
8	9	2	1	6	5	3	7	4
5	3	6	7	4	2	1	8	9
4	1	7	9	8	3	5	2	6
6	5	1	3	7	4	2	9	8
7	2	3	6	9	8	4	1	5
9	8	4	5	2	1	6	7	3
3	4	5	8	1	9	7	6	2
2	6	8	4	5	7	3	9	1
1	7	9	2	3	6	5	4	8
8	9	2	1	6	5	3	7	4
5	3	6	7	4	2	1	8	9
4	1	7	9	8	3	5	2	6
6	5	1	3	7	4	2	9	8
7	2	3	6	9	8	4	1	5
9	8	4	5	2	1	6	7	3
3	4	5	8	1	9	7	6	2
2	6	8	4	5	7	3	9	1
1	7	9	2	3	6	5	4	8
8	9	2	1	6	5	3	7	4
5	3	6	7	4	2	1	8	9
4	1	7	9	8	3	5	2	6
6	5	1	3	7	4	2	9	8
7	2	3	6	9	8	4	1	5
9	8	4	5	2	1	6	7	3
3	4	5	8	1	9	7	6	2
2	6	8	4	5	7	3	9	1
1	7	9	2	3	6	5	4	8
8	9	2	1	6	5	3	7	4
5	3	6	7	4	2	1	8	9
4	1	7	9	8	3	5	2	6
6	5	1	3	7	4	2	9	8
7	2	3	6	9	8	4	1	5
9	8	4	5	2	1	6	7	3
3	4	5	8	1	9	7	6	2
2	6	8	4	5	7	3	9	1
1	7	9	2	3	6	5	4	8
8	9	2	1	6	5	3	7	4
5	3	6	7	4	2	1	8	9
4	1	7	9	8	3	5	2	6
6	5	1	3	7	4	2	9	8
7	2	3	6	9	8	4	1	5
9	8	4	5	2	1	6	7	3
3	4	5	8	1	9	7	6	2
2	6	8	4	5	7	3	9	1
1	7	9	2	3	6	5	4	8
8	9	2	1	6	5	3	7	4
5	3	6	7	4	2	1	8	9
4	1	7	9	8	3	5	2	6
6	5	1	3	7	4	2	9	8
7	2	3	6	9	8	4	1	5
9	8	4	5	2	1	6	7	3
3	4	5	8	1	9	7	6	2
2	6	8	4	5	7	3	9	1
1	7	9	2	3	6	5	4	8
8	9	2	1	6	5	3	7	4
5	3	6	7	4	2	1	8	9
4	1	7	9	8	3	5	2	6
6	5	1	3	7	4	2	9	8
7	2	3	6	9	8	4	1	5
9	8	4	5	2	1	6	7	3
3	4	5	8	1	9	7	6	2
2	6	8	4	5	7	3	9	1
1	7	9	2	3	6	5	4	8
8	9	2	1	6	5	3	7	4
5	3	6	7	4	2	1	8	9
4	1	7	9	8	3	5	2	6
6	5	1	3	7	4	2	9	8
7	2	3	6	9	8	4	1	5
9	8	4	5	2	1	6	7	3
3	4	5	8	1	9	7	6	2
2	6	8	4	5	7	3	9	1
1	7	9	2	3	6	5	4	8
8	9	2	1	6	5	3	7	4
5	3	6	7	4	2	1	8	9
4	1	7	9	8	3	5	2	6
6	5	1	3	7	4	2	9	8
7	2	3	6	9	8	4	1	5
9	8	4	5	2	1	6	7	3
3	4	5	8	1	9	7	6	2
2	6	8	4	5	7	3	9	1
1	7	9	2	3	6	5	4	8
8	9	2	1	6	5	3	7	4
5	3	6	7	4	2	1	8	9
4	1	7	9	8	3	5	2	6
6	5	1	3	7	4	2	9	8
7	2	3	6	9	8	4	1	5
9	8	4	5	2	1	6	7	3
3	4	5	8	1	9	7	6	2
2	6	8	4	5	7	3	9	1
1	7	9	2	3	6	5	4	8
8	9	2	1	6	5	3	7	4
5	3	6	7	4	2	1	8	9
4	1	7	9	8	3	5	2	6
6	5	1	3	7	4	2	9	8
7	2	3	6	9	8	4	1	5
9	8	4	5	2	1	6	7	3
3	4	5	8	1	9	7	6	2
2	6	8	4	5	7	3	9	1
1	7	9	2	3	6	5	4	8
8	9	2	1	6	5	3	7	4
5	3	6	7	4	2	1	8	9
4	1	7	9	8	3	5	2	6
6	5	1	3	7	4	2	9	8
7	2	3	6	9	8	4	1	5
9	8	4	5	2	1	6	7	3
3	4	5	8	1	9	7	6	2
2	6	8	4	5	7	3	9	1
1	7	9	2	3	6	5	4	8
8	9	2	1	6	5	3	7	4
5	3	6	7	4	2	1	8	9
4	1	7	9	8	3	5	2	6
6	5	1	3	7	4	2	9	8
7	2	3	6	9	8	4	1	5
9	8	4	5	2	1	6	7	3
3	4	5	8	1	9	7	6	2
2	6	8	4	5	7	3	9	1
1	7	9	2	3	6	5	4	8
8	9	2	1	6	5	3	7	4
5	3	6	7	4	2	1	8	9
4	1	7	9	8	3	5	2	6
6	5	1	3	7	4	2	9	8
7	2	3	6	9	8	4	1	5
9	8	4	5	2	1	6	7	3
3	4	5	8	1	9	7	6	2
2	6	8	4	5	7	3	9	1
1	7	9	2	3	6	5	4	8
8	9	2	1	6	5	3	7	4
5	3	6	7	4	2	1	8	9
4	1	7	9	8	3	5	2	6
6	5	1	3	7	4	2	9	8
7	2	3	6	9	8	4	1	5
9	8	4	5	2	1	6	7	3
3	4	5	8	1	9	7	6	2
2	6	8	4	5	7	3	9	1
1	7	9	2	3	6	5	4	8
8	9	2	1	6	5	3	7	

# Complexidade – classes de problemas



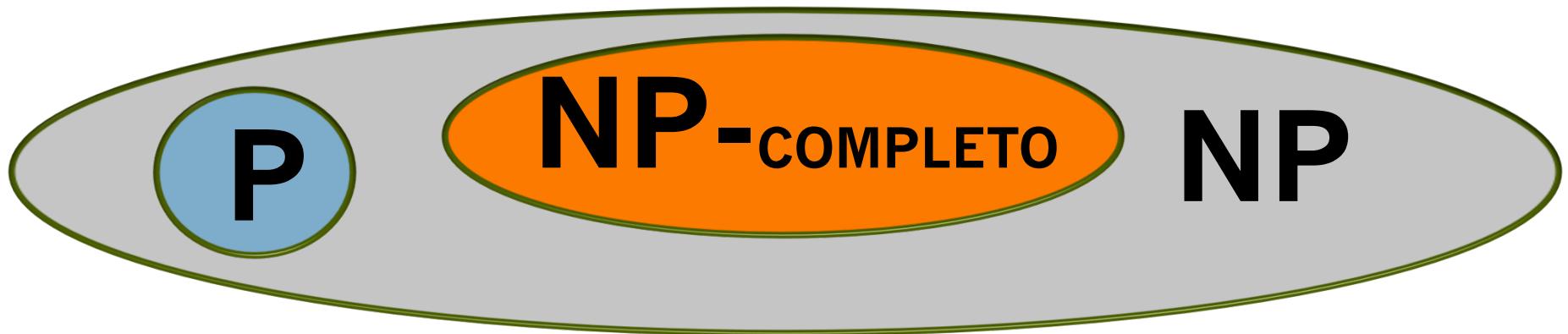
## Problemas P, NP, NP-Completos

P – problemas com solução polinomial (Deterministic Polynomial Time)

NP – problemas sem solução polinomial conhecida, verificável em tempo polinomial

NP-Completo – sem solução polinomial conhecida, sem provas de que NÃO existe solução polinomial.

### Problemas mais difíceis em NP





# Complexidade – classes de problemas

## Problemas P, NP, NP-Completos

P – problemas com solução polinomial

NP – problemas sem solução polinomial conhecida, verificável em tempo polinomial

NP-Completo – sem solução polinomial conhecida, sem provas de que NÃO existe solução polinomial. A prova, se existir, pode ser aplicada a uma classe similar.



# Complexidade – classes de problemas

## Problemas P, NP, NP-Completos

P – problemas com solução polinomial

NP – problemas sem solução polinomial conhecida, verificável em tempo polinomial

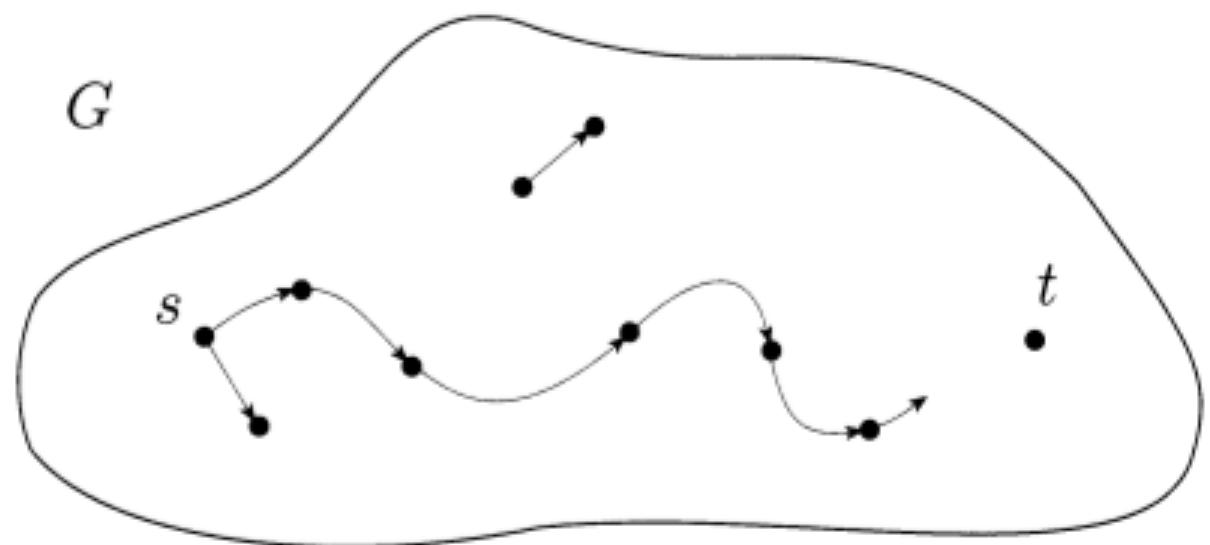
NP-Completo – sem solução polinomial conhecida, sem provas de que NÃO existe solução polinomial. A prova, se existir, pode ser aplicada a uma classe similar.



# Complexidade

Como identificar problemas fáceis/difíceis?

Encontre caminho entre  $s - t$

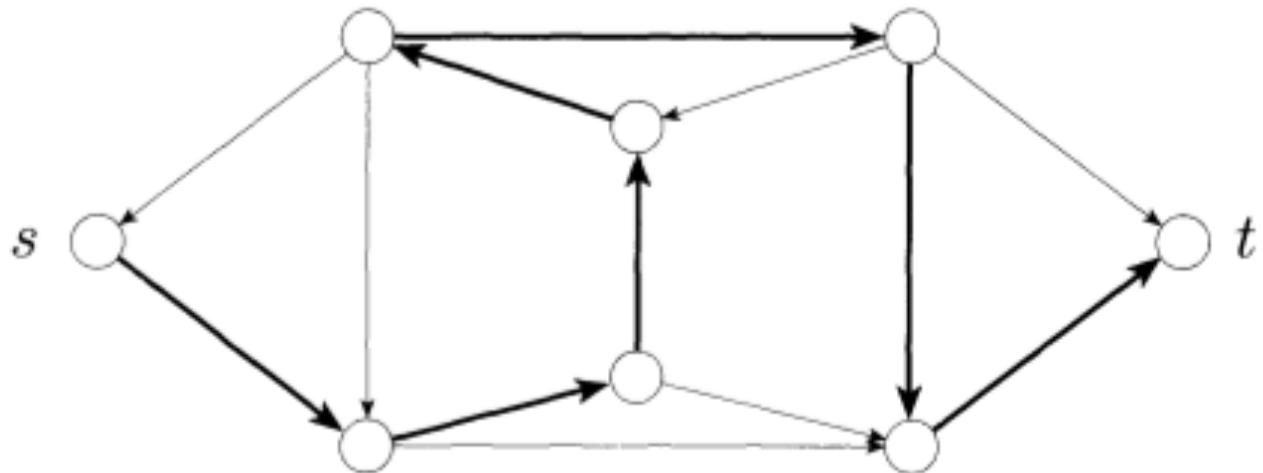




# Complexidade

Como identificar problemas fáceis/difíceis?

Grafo Hamiltoniano – encontrar caminho que percorre todos os nodos e apenas uma vez cada um dos nodos.

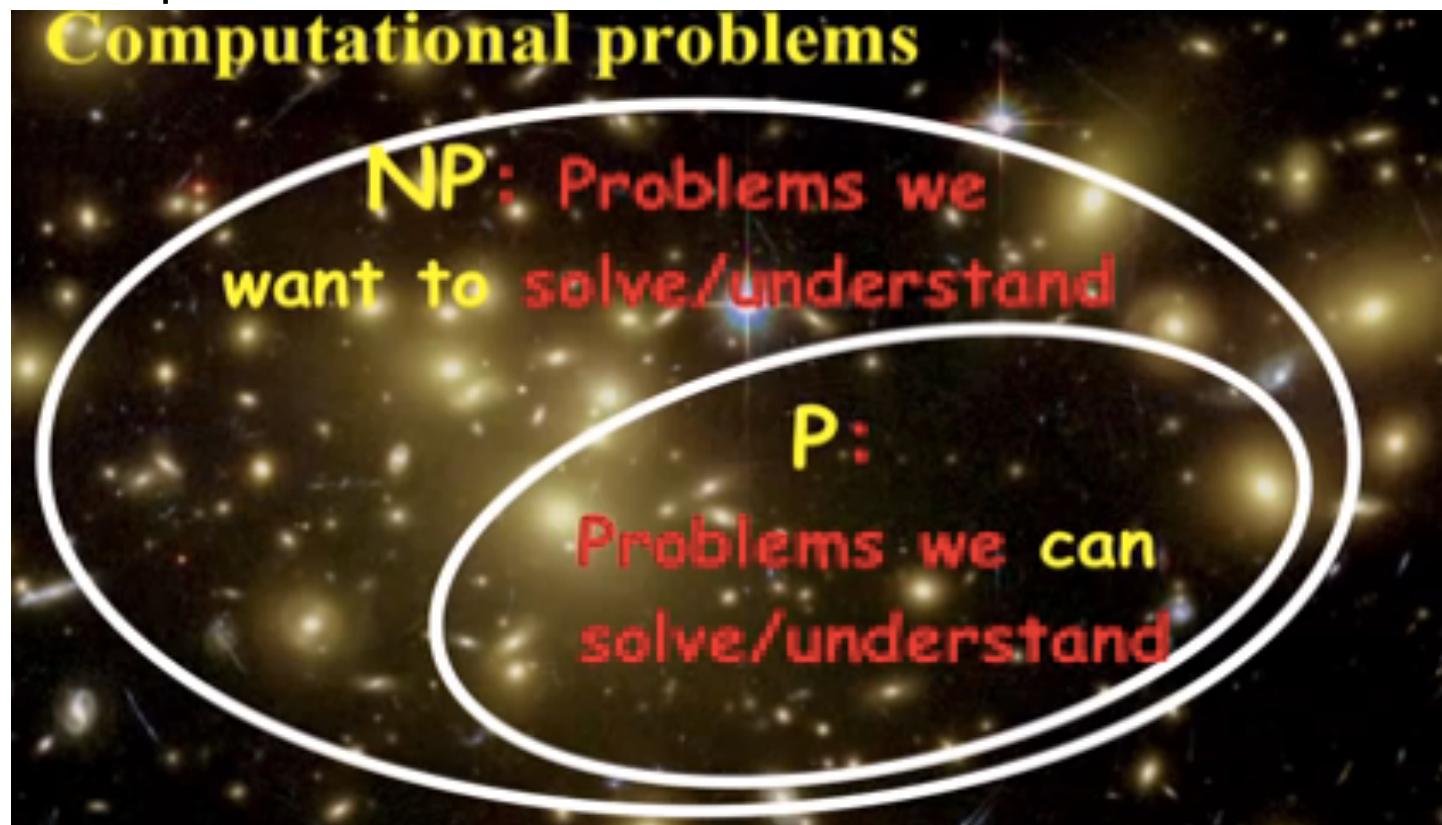




# Complexidade

Como identificar problemas fáceis/difíceis?

Avi Wigderson:



Classe P – problemas resolvidos eficientemente

Classe NP – problemas certificados eficientemente.

Fonte: Arora, Sypser

# Unsolvable

Zürich

# Solvable

## NP - complete

Solving  
Sudoku

Theorem  
Proving

Integer  
Factoring

Shortest  
Path

Pattern  
Matching

Error  
Correction

Multiplication

Addition

P

NP

Classe P – problemas resolvidos eficientemente

Classe NP – problemas certificados eficientemente.

Classe NP-Completo – problemas difíceis e cuja solução impacta diversas outras classes de problemas

Fonte: Arora, Sypser



7 problemas de matemática (prêmio de 1 milhão de dólares).

Um deles é o P versus NP: “É verdade que todo problema cuja solução pode ser verificada em tempo polinomial é um problema que pode ser resolvido em tempo polinomial?”

Os problemas:

P versus NP

Conjectura de Hodge

Conjectura de Poincaré (solução)

Hipótese de Riemann

Existência de Yang-Mills e intervalo de massa

Existência e suavidade de Navier-Stokes

Conjectura de Birch e Swinnerton-Dyer



# Complexidade

## Problemas P, NP, NP-Completos

**Problemas P** – todos aqueles com algoritmos de tempo polinomial, ou seja, para uma entrada de tamanho  $n$  o pior tempo de execução é de  $O(n^k)$

**Problemas NP (Não Determinístico de tempo polinomial)** – todos aqueles com algoritmos VERIFICÁVEIS em tempo polinomial.

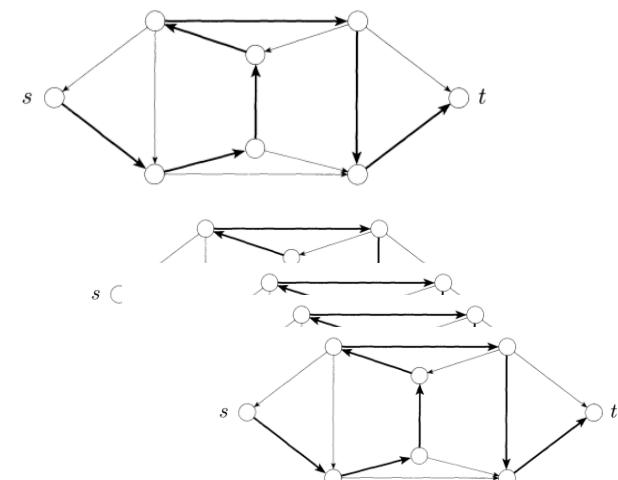
Ex.: grafo com ciclo Hamiltoniano (ciclo onde todos os vértices do grafo são vistos uma e apenas uma vez)

$$G = (V, A)$$

$S = (v_1, v_2, v_3, \dots, v_{|V|})$  com  $|V|$  vértices

Avaliar se:  $(v_i, v_{i+1}) \in A$  para  $i=1, 2, 3, \dots, |V|-1$

E se  $(v_{|V|}, v_1) \in A$



Fonte: Arora, Sypser



# Complexidade

## Problemas P, NP, NP-Completos

**Problemas P** – todos aqueles com algoritmos de tempo polinomial, ou seja, para uma entrada de tamanho  $n$  o pior tempo de execução é de  $O(n^k)$

**Problemas NP (Não Determinístico de tempo polinomial)** – todos aqueles com algoritmos VERIFICÁVEIS em tempo polinomial.

Portanto,  $P \subseteq NP$



# Complexidade

## Problemas P, NP, NP-Completos

**Problemas NPC (NP-Completos)** – problemas que estão no conjunto NP e são tão difíceis como outros problemas NP.

Se NP-Completo pode ser resolvido em tempo polinomial, todos os problemas NP desta classe também possuem algoritmo polinomial.

Assume-se que NP-Completo é intratável, pelo volume de estudos já dedicados ao tema sem uma solução observada.

**Utilidade para o desenvolvedor:** Estabelecer se o problema é NP-Completo, para então adotar abordagem correta: algoritmos de aproximação no lugar de algoritmo exato.



# Complexidade

## Problemas P, NP, NP-Completos

**Problemas NPH (NP-Difíces / NP-Hard)** – problemas que não podem ser verificáveis em tempo polinomial.

Exemplo: melhor movimento de xadrez em uma partida. A verificação pode demandar tempo exponencial.



Fonte: Arora, Sypser



# Complexidade

## Problemas P, NP, NP-Completos

### A questão P x NP

- Eventualmente alguns problemas de NP foram solucionados em tempo razoável.
- Isso gerou a especulação :
  - Todos NP são P ?  
Ou seja,  $P = NP$  ?
  - Existem problemas Np que nunca serão P?  
Ou seja  $P \neq NP$  ?
- Se  $P = NP$  então teremos soluções polinomiais para diversos problemas difíceis ...

# Problemas P, NP, NP-Completos

## A questão P x NP

Lógica básica utilizada:

a) Resultado correto de uma multiplicação pode ser verificado rapidamente.

Então isso indica que P se comporta como NP.

Então  $P = NP$  ?

Multiplicação

**P**

**NP**

# Problemas P, NP, NP-Completos

## A questão P x NP

Lógica básica utilizada:

a) Resultado correto de uma multiplicação pode ser verificado rapidamente.

Então isso indica que P se comporta como NP.

Então  $P = NP$  ?

b) Alguns problemas NP foram solucionados com tempo razoável.

Então  $P = NP$  ?

Multiplicação

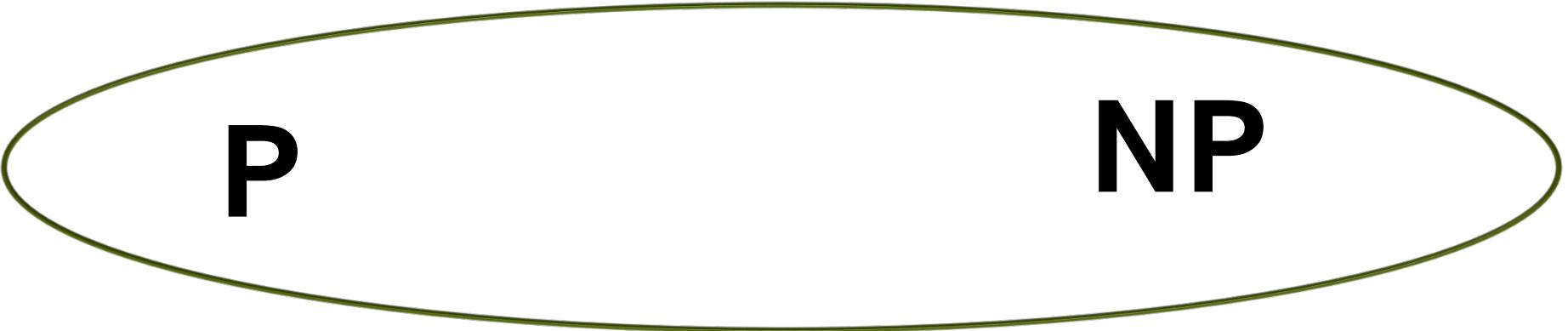
**P**

**NP**

# Problemas P, NP, NP-Completos

## A questão P x NP

- Pesquisa de opinião:
  - $P = NP$ : 9%
  - $P \neq NP$  : 83%
  - Não sabe : 8%



**P**

**NP**



# Complexidade

## Problemas P, NP, NP-Completos

### Problemas NPC (NP-Completos)

**Utilidade para o desenvolvedor:** Estabelecer se o problema é NPC, para então adotar abordagem correta: algoritmos de aproximação no lugar de algoritmo exato.

Visão geral de abordagens

- a) Problemas de otimização x decisão
- b) Redução

# Problemas P, NP, NP-Completos

## Problemas NPC (NP-Completos)

**Utilidade para o desenvolvedor:** Estabelecer se o problema é NPC, para então adotar abordagem correta: algoritmos de aproximação no lugar de algoritmo exato.

EXEMPLO CLÁSSICO: Problema do Vendedor Ambulante

Objetivo:

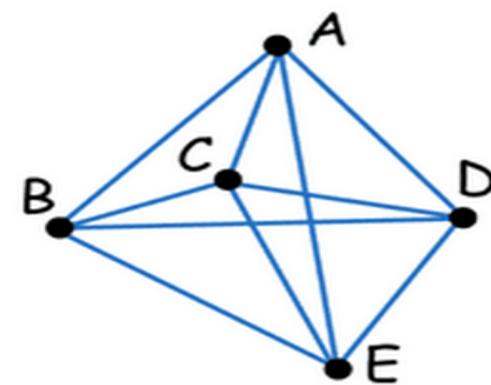
visitar todas as cidades usando o menor caminho.

Qual seria o menor caminho? ABCDE? ADEBC? BCADE

Solução força-bruta:

Testar todas as possibilidades

Resultado: factorial [  $t = n!$  ou melhor  $t=(n-1)!$  ]



# Problemas P, NP, NP-Completos

## Problemas NPC (NP-Completos)

**Utilidade para o desenvolvedor:** Estabelecer se o problema é NPC, para então adotar abordagem correta: algoritmos de aproximação no lugar de algoritmo exato.

**EXEMPLO CLÁSSICO:** Problema do Vendedor Ambulante

Resultado: fatorial [  $t = n!$  ou melhor  $t=(n-1)!$  ]

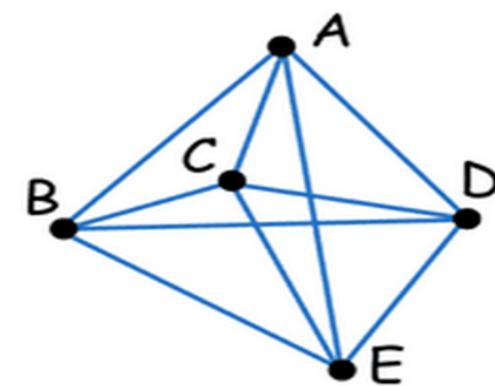
Análise:

Caso visitar 20 locais use 1 segundo de cálculo

⇒ Visitar 21 cidades demoraria 21 segundos de cálculo

⇒ Visitar 22 cidades demoraria 462 segundos (7 minutos)

⇒ Visitar 30 cidades: 3 milhões de anos !



Redução:

⇒ Abordagem com programação dinâmica:  $t=2^n$

⇒ Visitar 30 cidades: 10 minutos

⇒ Visitar 70 cidades: 35000 anos !



## Complexidade

### Problemas P, NP, NP-Completos

#### a) Problemas de otimização x decisão

Adotar limite para avaliar problema de otimização como problema de decisão

Por exemplo:

1 - Algoritmo de menor caminho (problema de otimização)

2 – Reescrita do enunciado:

Dado um grafo direcional  $G$ , vértices  $u$  e  $v$ , um valor inteiro  $k$ , existe um caminho entre  $u$  e  $v$  com no máximo  $k$  arestas?

3 – Utilizar resultado e comparar

# Problemas P, NP, NP-Completos

## b) Redução

Utilizar a simplicidade de um algoritmo para provar a simplicidade de outro algoritmo

Por exemplo:

- 1 – Comparação de dois problemas de decisão, com algoritmos A e B.
- 2 – Identifica uma instância a tratada pelo algoritmo A
- 3 – Transforma a instância a em uma instância b tratada pelo Algoritmo B
- 4 – Utiliza o algoritmo B para tratar a instância b e avalia os resultados

