

Conceitos Básicos

Projeto e Análise de Algoritmos

Daniel de Oliveira Capanema

UIT

2022

Algoritmos

- Os algoritmos fazem parte do dia-a-dia das pessoas. Exemplos de algoritmos:
 - Instruções para o uso de medicamentos;
 - Indicações de como montar um aparelho;
 - Uma receita de culinária.
- Sequência de ações executáveis para a obtenção de uma solução para um determinado tipo de problema.

Algoritmos

- Segundo Dijkstra, um algoritmo corresponde a uma descrição de um padrão de comportamento, expresso em termos de um conjunto finito de ações.
 - Executando a operação $a + b$ percebemos um padrão de comportamento, mesmo que a operação seja realizada para valores diferentes de a e b .
- Conhecido por:
 - Algoritmo de Dijkstra
 - Semáforo
 - THE

$$\int_{-\infty}^{\infty} f(x)e^{-2x}$$

Edsger Wybe Dijkstra
Ciência da computação



Nacionalidade	 Neerlandês
Nascimento	11 de Maio de 1930
Local	Roterdã, Países Baixos
Falecimento	6 de Agosto de 2002 (72 anos)
Local	Nuenen, Países Baixos
Actividade	
Campo(s)	Ciência da computação
Orientador(es)	Adriaan van Wijngaarden
Orientado(s)	Nico Habermann, Jan L. A. van de Snepscheut
Conhecido(a) por	Algoritmo de Dijkstra Semáforo THE

Algoritmos: Seu Papel em Computação

- **Definição:** um algoritmo é um conjunto finito de instruções precisas para executar uma computação.
 - Um algoritmo pode ser visto como uma ferramenta para resolver um problema computacional bem especificado.
- Um algoritmo pode receber como entrada um conjunto de valores e pode produzir como saída um outro conjunto de valores.
 - Um algoritmo descreve uma sequência de passos computacionais que transforma a entrada numa saída, ou seja, uma relação entrada/saída.
- O vocábulo algoritmo origina do nome al-Khowarizmi.

Algoritmos: Origem do Vocábulo

- Abu Ja'Far Mohammed Ibn Musa al-Khowarizmi (780-850), astrônomo e matemático árabe.
 - Membro da “Casa da Sabedoria” (academia de cientistas em Bagdá)
 - O nome al-Khowarizmi significa da cidade de Khowarizmi, que agora é chamada Khiva e é parte do Uzbequistão.
 - Ele escreveu livros de matemática, astronomia e geografia.
 - Introduzida na Europa ocidental a álgebra.
- A palavra álgebra vem do árabe al-jabr, parte do título de seu livro Kitab al-jabr w'al muquabala.
 - Esse livro foi traduzido para o latim e foi usado extensivamente.
 - Seu livro sobre o uso dos numerais hindu descreve procedimentos para operações aritméticas usando esses numerais.
 - Autores europeus usaram uma adaptação latina de seu nome, até finalmente chegar na palavra algoritmo para descrever a área da aritmética com numerais hindu.



Algoritmos: Origem do Vocábulo



Algoritmos: Etimologia¹

- Do antropônimo² árabe *al-Khuwarizmi* (matemático árabe do século IX) formou-se o árabe *al-Khuwarizmi* ‘numeração decimal em arábianos’ que passou ao latim medieval *algorismus* com influência do grego *arithmós* ‘número’; forma histórica 1871 *algorithmo*.
- 1. Estudo da origem e da evolução das palavras.
- 2. Nome próprio de pessoa ou de ser personificado
- Referência: Dicionário Houaiss da Língua Portuguesa, 2001, 1a edição.

Algoritmos: Definições

- Dicionário Houaiss da Língua Portuguesa, 2001, 1ª edição:
 - Conjunto das regras e procedimentos lógicos perfeitamente definidos que levam à solução de um problema em um número de etapas.
- Dicionário Webster da Língua Inglesa:
 - An Algorithm is a procedure for solving a mathematical problem in a finite number of steps that frequently involves a repetition of an operation

Algoritmos: Observações

- Regras são precisas
- Conjunto de regras é finito
- Tempo finito de execução
- Regras são executadas por um computador

Algoritmos: Consequências

- Deve-se definir um repertório finito de regras
 - Linguagem de programação
- A maior parte dos algoritmos utiliza métodos de organização de dados envolvidos na computação
 - Estruturas de dados
- Tempo finito não é uma eternidade
 - A maior parte das pessoas não está interessada em algoritmos que levam anos, décadas, séculos, milênios para executarem
- Existem diferentes “tipos de computadores”
 - § Existem diferentes modelos computacionais

Algoritmos: Aspectos

- Estático:
 - Texto contendo instruções que devem ser executadas em uma ordem definida, independente do aspecto temporal
- Dinâmico:
 - Execução de instruções a partir de um conjunto de valores iniciais, que evolui no tempo
- Dificuldade:
 - Relacionamento entre o aspecto estático e dinâmico

Algumas Perguntas Importantes

- Apresente as justificativas:
 1. Um programa pode ser visto como um algoritmo codificado em uma linguagem de programação que pode ser executado por um computador. Qualquer computador pode executar qualquer programa?
 2. Todos os problemas ligados às ciências exatas possuem algoritmos?
 3. Todos os problemas computacionais têm a mesma dificuldade de resolução?
 4. Como algoritmos diferentes para um mesmo problema podem ser comparados/avaliados?

Natureza da Ciência da Computação

- Referência: Juris Hartmanis. On Computational Complexity and the Nature of Computer Science. Communications of the ACM, 37(10):37-43, October 1994
 - *The computer scientist has to create many levels of abstractions to deal with these problems. One has to create intellectual tools to conceive, design, control, program, and reason about the most complicated of human creations. Furthermore, this has to be done with unprecedented precision. The underlying hardware that executes the computations are universal machines and therefore they are chaotic systems: the slightest change in their instructions or data can result in arbitrarily large differences in the results."*

Natureza da Ciência da Computação

- Donald Knuth (Personal communication between Juris Hartmanis and Donald Knuth, March 10, 1992) disse que:
 - *Computer Science and Engineering is a field that attracts a different kind of thinker. I believe that one who is a natural computer scientist thinks algorithmically. Such people are especially good at dealing with situations where different rules apply in different cases; they are individuals who can rapidly change levels of abstraction, simultaneously seeing things “in the large” and “in the small.”*

Problema dos Dois Exércitos

Na Grécia antiga, lugares maravilhosos como este ...



Vale perto de Almfiklia. Grécia

...podiam se transformar em cenários de guerra.



- É quando algum filósofo propõe o “problema dos dois exércitos”.

Problema dos Dois Exércitos: Cenário Ideal




- Exército Alfa está em maior número que o exército Gama mas está dividido em duas metades, cada uma numa lateral do vale.
- Cada metade do exército Alfa está em menor número que o exército Gama.
- Objetivo do exército Alfa: coordenar um ataque ao exército Gama para ganhar a guerra.

Problema dos Dois Exércitos: Coordenação



1. General do exército **Alfa**, do lado esquerdo do vale, chama o seu melhor soldado para levar uma mensagem para o general do exército **Alfa** do lado direito:

 Vamos atacar conjuntamente o exército **Gama** amanhã às 6:00h?

Observações:

- A **única** possibilidade de comunicação entre os dois generais é através de um mensageiro.
- Os dois generais têm um “relógio perfeitamente sincronizado”, ou seja, eles sabem pela posição do sol quando é 6:00h.

Problema dos Dois Exércitos: Coordenação



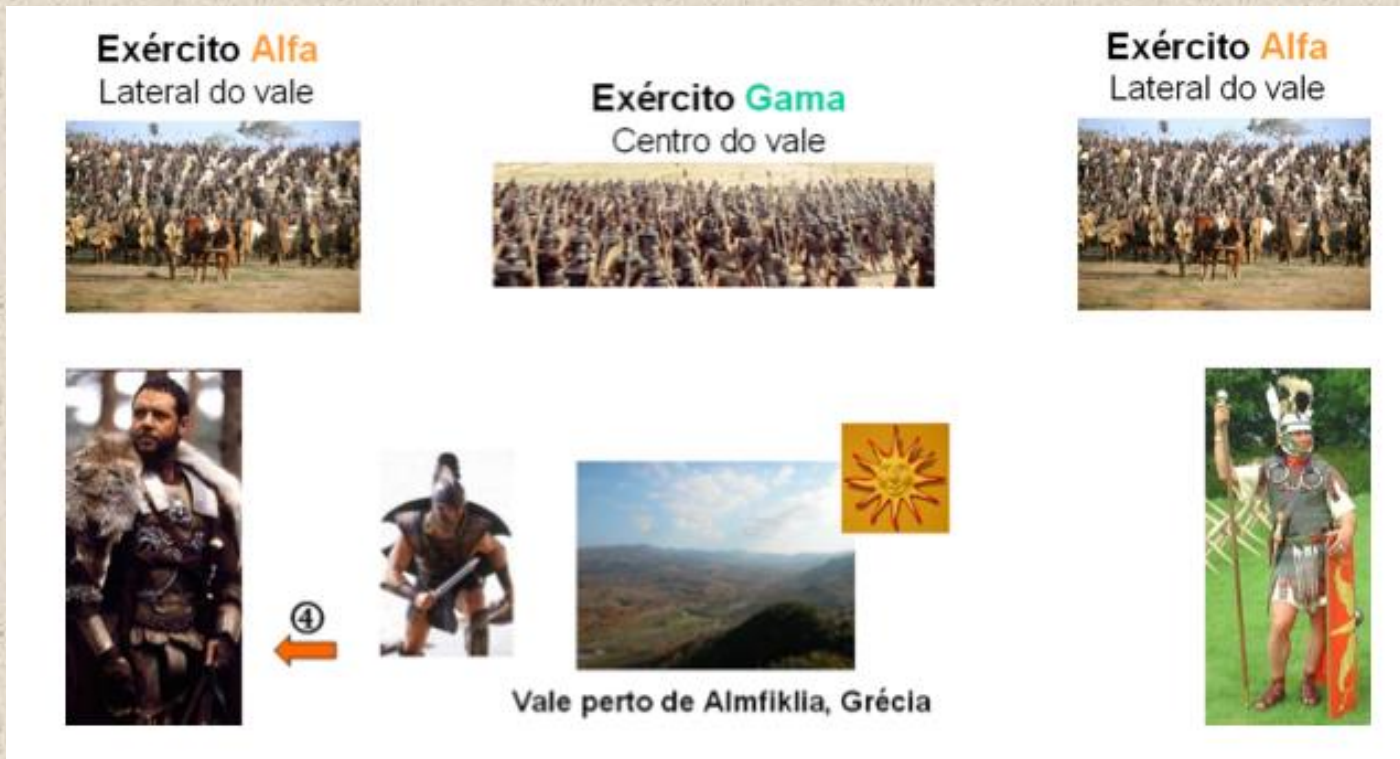
2. O soldado do exército **Alfa** atravessa as linhas inimigas e leva a mensagem até o general do outro lado.

Problema dos Dois Exércitos: Coordenação



3. O general do exército **Alfa** do lado direito concorda em atacar o exército **Gama** no dia seguinte às 6:00h.

Problema dos Dois Exércitos: Coordenação



- O soldado do exército **Alfa** atravessa novamente as linhas inimigas e confirma com seu general o ataque para o dia seguinte.

Problema dos Dois Exércitos: Coordenação



→ Após esses quatro passos terem sido realizados com sucesso, vai haver ataque amanhã às 6:00h?

Algoritmo Correto X Incorreto

- Um algoritmo é **correto** se, para cada instância de entrada, ele para com a saída correta
- Um algoritmo **incorreto** pode não parar em algumas instâncias de entrada, ou então pode parar com outra resposta que não a desejada

Algoritmo Eficiente X Ineficiente

- Algoritmos **eficientes** são os que executam em tempo polinomial
- Algoritmos que necessitam de tempo superpolinomial são chamados de **ineficientes**

Problema Tratável X Intratável

- Problemas que podem ser resolvidos por algoritmo de tempo polinomial são chamados de **tratáveis**
- Problemas que exigem tempo superpolinomial são chamados de **intratáveis**

Tratabilidade

Problema Decidível X Indecidível

- Um problema é **decidível** se existe algoritmo para resolvê-lo
- Um problema é **indecidível** se não existe algoritmo para resolvê-lo

Decidibilidade

Análise de Algoritmos

- Analisar a complexidade computacional de um algoritmo significa prever os recursos de que o mesmo necessitará:
 - Memória
 - Largura de banda de comunicação
 - Hardware
 - Tempo de execução
- Geralmente existe mais de um algoritmo para resolver um problema
- A análise de complexidade computacional é fundamental no processo de definição de algoritmos mais eficientes para a sua solução
- Em geral, o tempo de execução cresce com o tamanho da entrada

Porque Estudar Análise de Algoritmos?

- O tempo de computação e o espaço na memória são recursos limitados
 - Os computadores podem ser rápidos, mas não são infinitamente rápidos
 - A memória pode ser de baixo custo, mas é finita e não é gratuita
- Os recursos devem ser usados de forma sensata, e algoritmos eficientes em termos de tempo e espaço devem ser projetados
- Com o aumento da velocidade dos computadores, torna-se cada vez mais importante desenvolver algoritmos mais eficientes, devido ao aumento constante do tamanho dos problemas a serem resolvidos

Qual dos dois algoritmos é o melhor?

Algoritmo 1

```
if (a > b) {  
    if (a > c)  
        System.out.println("A é o maior")  
    else  
        System.out.println("C é o maior")  
} else if (b > c)  
    System.out.println("B é o maior")  
else  
    System.out.println("C é o maior")
```

Algoritmo 2

```
if (a > b) && (a > c)  
    System.out.println("A é o maior")  
if (b > a) && (b > c)  
    System.out.println("B é o maior")  
if (c > a) && (c > b)  
    System.out.println("C é o maior")
```


Qual dos dois algoritmos é o melhor?

Algoritmo 1

```
int fatorial (int n) {  
  
    if (n==0)  
        return 1;  
    else  
        return n*fatorial(n-1);  
}
```

Algoritmo 2

```
int fatorial (int n) {  
  
    int total = 1;  
  
    for (; n > 0; n--) {  
        total = total * n;  
    }  
  
    return total;  
}
```

Análise de Algoritmos

- Que fatores devem ser considerados para escolher qual algoritmo é o melhor?0
 - - Número de Linhas (✖)
 - - Número de Testes (✖)
 - - Número de Laços (✖)
 - - Número de Variáveis (✖)
 - - Tempo de Execução (✓)

Tempo de Execução

- Um algoritmo pode levar tempos diferentes para resolver determinado problema em diferentes execuções
- Depende de
 - - Características da Máquina
 - - Tamanho de Entrada

Análise de Algoritmos

- Funções matemáticas passam a representar o tempo de execução dos algoritmos!

```
int fatorial (int n) {
```

<code>int total = 1;</code>	1
-----------------------------	---

<code>for (; n > 0; n--) {</code>	2
--------------------------------------	---

<code>total = total * n;</code>	1	n
---------------------------------	---	---

<code>}</code>	
----------------	--

<code>return total;</code>	1
----------------------------	---

```
}
```

Principais Classes dos Algoritmos

- Constance - c
- Logarítmica - $\log n$
- Linear - n
- Polinomiais - n^2 , n^3 ...
- Exponencial - 2^n

Porque Estudar Análise de Algoritmos?

- Suponha que para resolver um determinado problema você tem disponível um **algoritmo exponencial** (2^n) e um computador capaz de executar 10^4 operações por segundo

		2^n na máquina 10^4
	tempo (s)	tamanho
	0,10	10
	1	13
1 minuto	60	19
1 hora	3.600	25
1 dia	86.400	30
1 ano	31.536.000	38

Porque Estudar Análise de Algoritmos?

- Compra de um novo computador capaz de executar 10^9 operações por segundo

	tempo (s)	2^n na máquina 10^4	2^n na máquina 10^9
		tamanho	tamanho
	0,10	10	27
	1	13	30
1 minuto	60	19	36
1 hora	3.600	25	42
1 dia	86.400	30	46
1 ano	31.536.000	38	55

Aumento na velocidade computacional tem pouco efeito no tamanho das instâncias resolvidas por algoritmos ineficientes

Porque Estudar Análise de Algoritmos?

- **Investir em algoritmo:**
 - Você encontrou um algoritmo quadrático (n^2) para resolver o problema

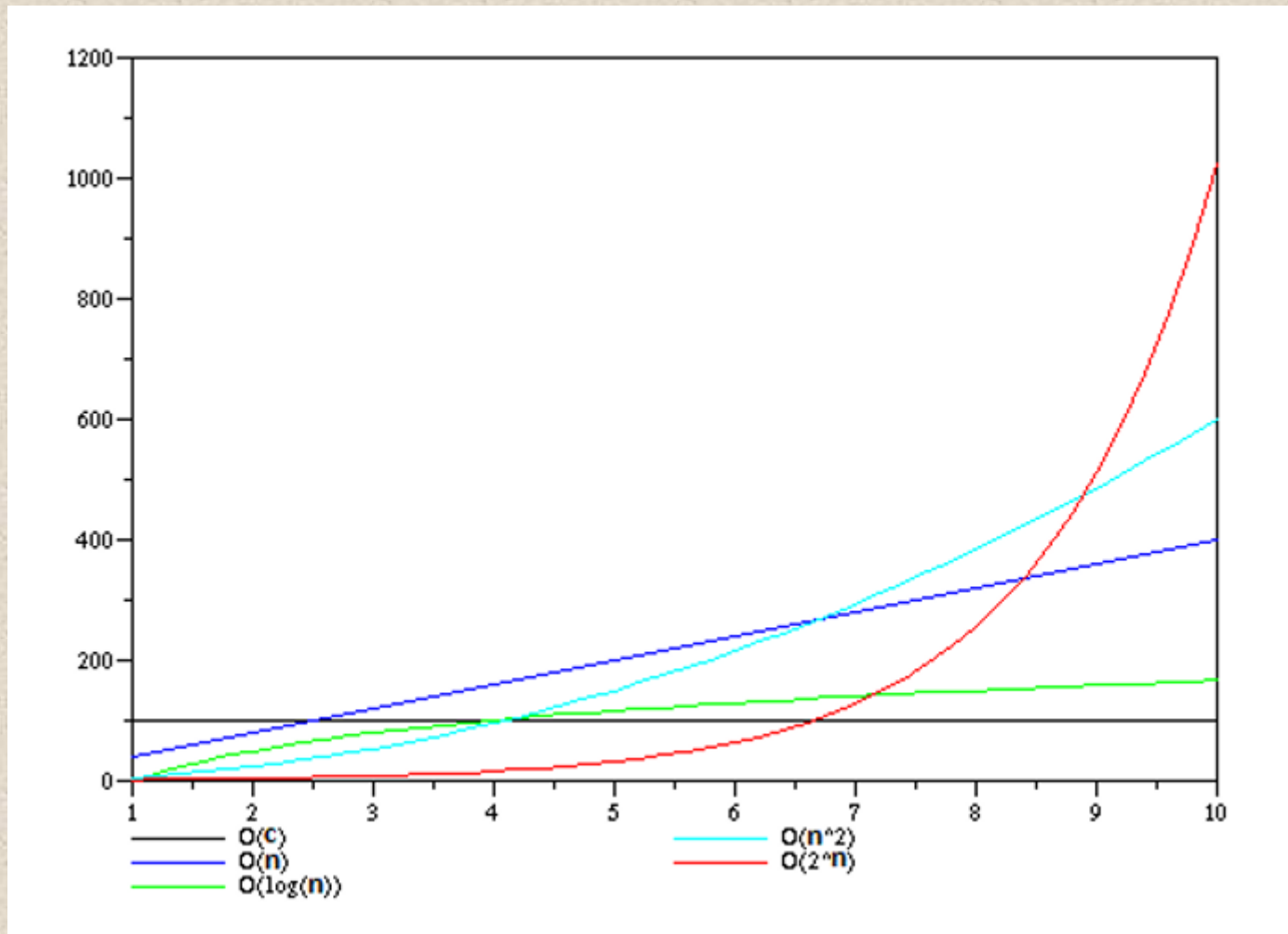
		2 ⁿ na máquina 10 ⁴	2 ⁿ na máquina 10 ⁹	n ² na máquina 10 ⁴	n ² na máquina 10 ⁹
		tempo (s)	tamanho	tamanho	tamanho
1 minuto		0,10	10	27	32
		1	13	30	100
		60	19	36	775
		3.600	25	42	6.000
		86.400	30	46	29.394
1 hora		31.536.000	38	55	561.569
1 dia					177.583.783
1 ano					

*Novo algoritmo oferece uma melhoria maior
que a compra da nova máquina*

Comparação

- Como não faz sentido analisar um algoritmo para apenas um determinado conjunto de entradas e é impossível fazer esta análise para todas as entradas possíveis, normalmente, considera-se apenas dois cenários
 - - Melhor caso
 - Entrada tendendo a zero (entrada pequena)
 - - Pior caso
 - Entrada tendendo ao infinito (entrada grande)

Classes de Algoritmos (Gráficos)



Ordem de Crescimento

- $1 < \log n < n < n \log n < n^2 < n^3 < 2^n$

Porque Estudar Projeto de Algoritmos?

- Algum dia você poderá encontrar um problema para o qual não seja possível descobrir prontamente um algoritmo publicado
- É necessário estudar técnicas de projeto de algoritmos, de forma que você possa desenvolver algoritmos por conta própria, mostrar que eles fornecem a resposta correta e entender sua eficiência

Porque Estudar Análise de Algoritmos

- Porque se você quer trabalhar nas grandes empresas, os problemas que você vai enfrentar são a nível global.



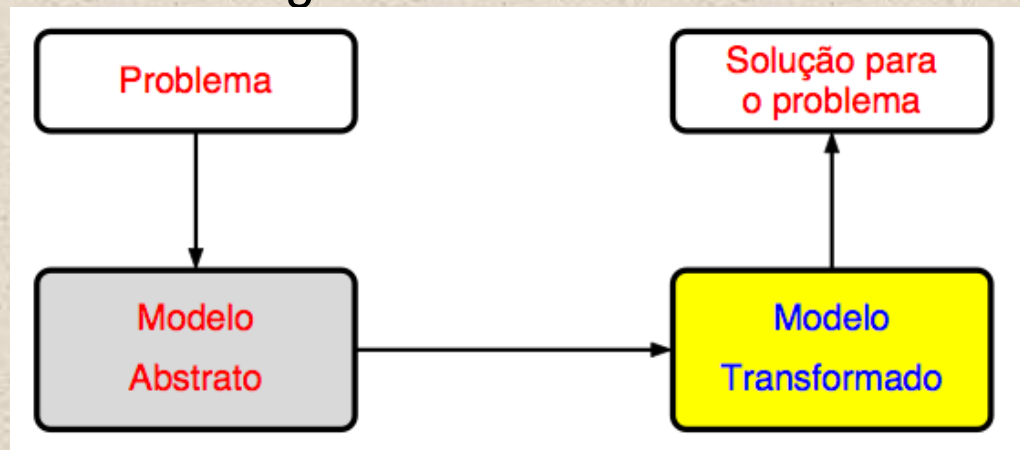
- Imagine trabalhar na Google...
 - Quantos bilhões de acessos o site da Google tem por segundo?
 - Qual é o tamanho da base de dados?

Aquecimento

- Questão (1) - Escreva um programa que receba três números e os imprima em ordem crescente.
- Questão (2) - Escreva um programa que insira 10 números em um vetor, ordene o vetor e no final imprima o vetor ordenado.
- Questão (3) - Escreva um programa que insira 10 números em um vetor, inverta a ordem desses números e imprima o vetor invertido.

Modelagem Matemática

- **Metodologia:** conjunto de conceitos que traz coesão a princípios e técnicas mostrando quando, como e porque usá-los em situações diferentes.
- A metodologia que usa matemática na resolução de problemas é conhecida como modelagem matemática.
- O processo de modelagem:



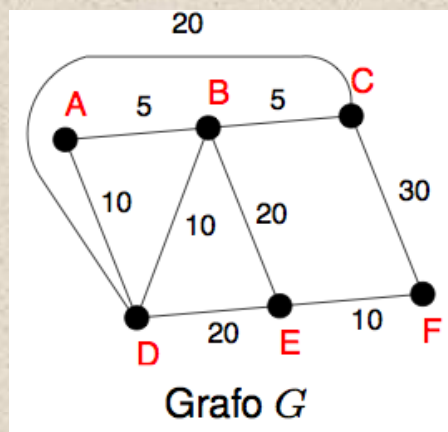
Exemplo de Modelagem: Malha Rodoviária (1)

- Suponha a malha rodoviária entre as seis cidades A, B, C, D, E, e F.
- **Problema:** Achar um subconjunto da malha rodoviária representada pela tabela abaixo que ligue todas as cidades e tenha um comprimento total mínimo.

	B	C	D	E	F
A	5	—	10	—	—
B		5	10	20	—
C			20	—	30
D				20	—
E					10

Exemplo de Modelagem: Malha Rodoviária (1)

- Tabela já é um modelo da situação do mundo real.
- A tabela pode ser transformada numa representação gráfica chamada GRAFO, que será o modelo matemático.



- **Grafo (definição informal):** conjunto de pontos chamados de vértices ou nós, e um conjunto de linhas (normalmente não-vazio) conectando um vértice ao outro.
 - Neste caso, cidades são representadas por vértices e estradas por linhas (arestas).

Exemplo de Modelagem: Malha Rodoviária (1)

- Qual é o próximo passo?
 - Achar uma solução em termos desse modelo.
 - Nesse caso, achar um grafo G' com o mesmo número de vértices e um conjunto mínimo de arestas que conecte todas as cidades e satisfaça a condição do problema.
- **Observação:** o modelo matemático é escolhido, em geral, visando a solução.
- A solução será apresentada na forma de um algoritmo.

Exemplo de Modelagem: Malha Rodoviária (1)

- Veremos várias técnicas de projeto de algoritmos na disciplina.
 - Transformação
 - Divisão e conquista
 - Programação dinâmica
 - Método guloso
 - Enumeração implícita
 - Técnicas de retrocesso e critérios de poda
 - Algoritmos aproximados

Exemplo de Modelagem: Sudoku e Godoku

- **Objetivo:** preencher todos os espaços em branco do quadrado maior, que está dividido em nove grids, com os números de 1 a 9 (letras).
 - Os algarismos não podem se repetir na mesma coluna, linha ou grid.
- **Godoku:** é similar ao Sudoku mas formado apenas por letras.

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

Sudoku

A								
				C				
	B				F	I		D
		A					B	
E				G				
		F				H		
			D		E			
I			B					
							A	

Godoku

Exemplo de Modelagem: Sudoku e Godoku

- O jogo SuperSudoku é similar ao Sudoku e Godoku formado por números e letras. Cada grid tem 16 entradas, sendo nove dos números (0 a 9) e seis letras (A a F).

6	1	B	2			5	3		A	D		8		C	F
		0	8		B	C			7	6	F				5
4	9		D	2			0	8			C	B		7	1
		F			D	8		3	B		1		A	0	E
3	4		5	A		1	7		F	2		C	0	9	
								6				A			B
0		E	B	D	6		9	1	4		A		5		
		6			4		5	9		8	7			E	3
				B				5	3			E	7		4
F		5				6		A		4	8			2	
	8		4				C	F	2			1	3		A
2		7				4		E			6			F	
		C	0	4			2			1	B			5	9
7	E			5		D		4	9		0	3		B	8
5		4		8		7	6		E	F	3		C		
8		1		3	0		B	C		5		7		4	D

Exemplo de Modelagem: Sudoku e Godoku

- O objetivo é projetar um algoritmo para resolver o problema.
 - Veja que o Sudoku e o Deep Blue têm características bem diferentes!
- Esse projeto envolve dois aspectos:
 1. O algoritmo propriamente dito, e
 2. A estrutura de dados a ser usada nesse algoritmo.
- Em geral, a escolha do algoritmo influencia a estrutura de dados e vice-versa.
 - É necessário considerar diferentes fatores para escolher esse par (algoritmo e estrutura de dados).
 - Pontos a serem estudados ao longo do curso, começando pela sequência de disciplinas Algoritmos e Estruturas de Dados.

Exemplo de Modelagem: Sudoku e Godoku

- Um possível algoritmo para resolver o jogo Sudoku é o “Algoritmo de Força Bruta”:
 - Tente todas as possibilidades até encontrar uma solução!
- Nessa estratégia, quantas possibilidades existem para a configuração abaixo?

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

458	248			2579	24579	489	459	
3	3	3	1	4	4	3	3	6
458		145	4589		4589	1489		589
3	7	3	4	3	4	4	2	3
6	124	9	45	2	8	3	7	1345
	3						4	2
2	134	147	345679	579		8	109	1579
	3	3	6	3			3	4
479			479		479			279
3	5	6	3	1	3	3	8	3
3789	138	17		579	5879	169	1579	
	4	3	2	2	3	4	3	4
347	234		79		279		3479	
3	3	8	2	6	3	5	4	1
357		257	578		1257	8		378
	3	9	3	3	4	4	1	6
1	48	457	5789	579			479	789
	2	3	4	3	3	2	3	3

Legenda: X nº de opções para a posição

- Existem $1^1 \times 2^5 \times 3^{32} \times 4^{13} \times 6^1 = 23\,875\,983\,329\,839\,202\,653\,175\,808 \approx 23,8 \times 1024$ possibilidades!

Estrutura de Dados

- Estruturas de dados e algoritmos estão intimamente ligados:
 - Não se pode estudar estruturas de dados sem considerar os algoritmos associados a elas;
 - Assim como a escolha dos algoritmos em geral depende da representação e da estrutura dos dados.
- Para resolver um problema é necessário escolher uma abstração da realidade, em geral mediante a definição de um conjunto de dados que representa a situação real.
- A seguir, deve ser escolhida a forma de representar esses dados.

Escolha da Representação dos Dados

- A escolha da representação dos dados é determinada, entre outras, pelas operações a serem realizadas sobre os dados.
- Considere a operação de adição:
 - Para pequenos números, uma boa representação é por meio de barras verticais (caso em que a operação de adição é bastante simples).
 - Já a representação por dígitos decimais requer regras relativamente complicadas, as quais devem ser memorizadas.
- Quando consideramos a adição de grandes números é mais fácil a representação por dígitos decimais (devido ao princípio baseado no peso relativo a posição de cada dígito).

Programas

- Programar é basicamente estruturar dados e construir algoritmos.
- Programas são formulações concretas de algoritmos abstratos, baseados em representações e estruturas específicas de dados.
- Programas representam uma classe especial de algoritmos capazes de serem seguidos por computadores.
- Um computador só é capaz de seguir programas em linguagem de máquina (sequência de instruções obscuras e desconfortáveis).
- É necessário construir linguagens mais adequadas, que facilitem a tarefa de programar um computador.
- Uma linguagem de programação é uma técnica de notação para programar, com a intenção de servir de veículo tanto para a expressão do raciocínio algorítmico quanto para a execução automática de um algoritmo por um computador.

Tipos de Dados

- Caracteriza o conjunto de valores a que uma constante pertence, ou que podem ser assumidos por uma variável ou expressão, ou que podem ser gerados por uma função.
- Tipos simples de dados são grupos de valores indivisíveis (como os tipos básicos integer, boolean, char e real do Pascal).
 - Exemplo: uma variável do tipo boolean pode assumir o valor verdadeiro ou o valor falso, e nenhum outro valor.
- Os tipos estruturados em geral definem uma coleção de valores simples, ou um agregado de valores de tipos diferentes.

Tipos Abstratos de Dados

- Modelo matemático, acompanhado das operações definidas sobre o modelo.
 - Exemplo: o conjunto dos inteiros acompanhado das operações de adição, subtração e multiplicação.
- TADs são utilizados extensivamente como base para o projeto de algoritmos.
- A implementação do algoritmo em uma linguagem de programação específica exige a representação do TAD em termos dos tipos de dados e dos operadores suportados.
- A representação do modelo matemático por trás do tipo abstrato de dados é realizada mediante uma estrutura de dados.
- Podemos considerar TADs como generalizações de tipos primitivos e procedimentos como generalizações de operações primitivas.
- O TAD encapsula tipos de dados:
 - A definição do tipo e todas as operações ficam localizadas numa seção do programa.

Implementação de TADs

- Considere uma aplicação que utilize uma lista de inteiros.
- Poderíamos definir o TAD Lista, com as seguintes operações:
 1. Faça a lista vazia;
 2. Obtenha o primeiro elemento da lista; se a lista estiver vazia, então retorne nulo;
 3. Insira um elemento na lista.
- Há várias opções de estruturas de dados que permitem uma implementação eficiente para listas (por exemplo, o tipo estruturado arranjo).

Implementação de TADs

- Cada operação do tipo abstrato de dados é implementada como um procedimento na linguagem de programação escolhida.
- Qualquer alteração na implementação do TAD fica restrita à parte encapsulada, sem causar impactos em outras partes do código.
- Cada conjunto diferente de operações define um TAD diferente, mesmo atuem sob um mesmo modelo matemático.
- A escolha adequada de uma implementação depende fortemente das operações a serem realizadas sobre o modelo.