Introdução à Análise de Algoritmos

Márcio Moretto Ribeiro

24 de agosto de 2021

Conteúdo

Capítulo 1

Introdução

1.1 Apresentação

Esta apostila são notas de aula do curso ministrado por mim no segundo semestre de 2021 para as duas turmas noturnas de Sistemas de Informação. Seu conteúdo está baseado nos livros que servem de bibliografia para o curso:

[1] T.H. Cormen. *Algoritmos: teoria e prática*. Campus, 2012. ISBN: 9788535236996.

[3] R. Sedgewick. *Algorithms in C*. Algorithms in C. Addison Wesley Professional, 2001. ISBN: 9780201756081.

[4] R. Sedgewick e K. Wayne. *Algorithms: Algorithms_4*. Pearson Education, 2011. ISBN: 9780132762564.

Além dos livros serviu de material para elaboração dessas notas outros textos citados ao longo do curso bem como as gravações dos cursos do professor Robert Sedgeweick para o Coursera e do professor Ronald Rivest para o MIT.

1.2 Algoritmos

Um *problema computacional* é a especificação de uma relação desejada entre um certo *valor de entrada* escolhido em um conjunto de valores válidos e o *valor de saída* esperado.

Exemplo 1.2.1:

Problema da busca

Entrada: Uma sequência de $n \in \mathbb{N}$ valores $\langle a_1, \ldots, a_n \rangle$ em que $a_i \in \mathbb{Z}$ para $1 \leq i \leq n$ e $b \in \mathbb{Z}$.

Saída: $i \in \mathbb{N}$ tal que $a_i = b$ se existir ou $\perp$ caso contrário.

Exemplo 1.2.2:

Problema da 3-soma

Entrada: Três sequência de $n \in \mathbb{N}$ valores cada $\langle a_1, \ldots, a_n \rangle$, $\langle b_1, \ldots, b_n \rangle$ e $\langle c_1, \ldots, c_n \rangle$ em que $a_i, b_i, c_i \in \mathbb{Z}$ para $1 \leq i \leq n$.

Saída: O número $m \in \mathbb{N}$ de $i, j, k \in \mathbb{N}$ tal que $a_i + b_j + c_j = 0$.

Exemplo 1.2.3:

Problema da ordenação

Entrada: Uma sequência de $n \in \mathbb{N}$ valores $\langle a_1, \ldots, a_n \rangle$ em que $a_i \in \mathbb{Z}$ para $1 \leq i \leq n$.

Saída: Uma permutação da sequência de entrada $\langle a'_1, \ldots, a'_n \rangle$ tal que $a_i \leq a_j$ para todo $i \leq j$.

Uma *instância do problema* consiste de uma entrada válida para a qual se pretende produzir uma saída esperada. Por exemplo, a sequência $\langle 3, 42, 17, 2, -1 \rangle$ é uma instância do problema da ordenação cuja saída esperada é $\langle -1, 2, 3, 17, 42 \rangle$.

A disciplina de Introdução à Teoria da Computação (ITC) tem como objeto de estudo problemas computacionais, como eles se classificam entre os que tem solução ou não e entre os que tem solução eficiente ou não. A solução de um problema computacional é um algoritmo.

Os objetos de estudo desta disciplina são os algoritmos. Mas afinal, o que são algoritmos?

Um *algoritmo* parte de uma entrada escolhida em um conjunto potencialmente infinito de possibilidades (*princípio da massividade*) para produzir um valor de saída. O algoritmo processa a entrada por meio de uma sequência de passos (*princípio da discretude*) que produzem valores intermediários. Cada passo segue uma instrução simples (*princípio da elementaridade*) que só depende dos valores anteriores, não admitindo ambiguidades (*princípio da exatidão*) [2].

Um *prorgrama* é a realização de um algoritmo em certa *linguagem de programação*. Assim, um algoritmo é, de um lado, a solução de um problema de computação e, de outro, uma abstração de um conjunto de programas, ele é a idéia por trás do programa.

Um algoritmo é *correto* se para toda instância do problema ele produz a saída esperada depois de uma sequência finita de passos. Nesse caso dizemos que o algoritmo *resolve* o problema.

Há uma controversa se devemos ou não considerar uma sequência infinita de instruções como um algoritmo. Essa questão, complicada, está no coração do nascimento da ciência da computação e será tratada em ITC. Neste curso focaremos nos algoritmos corretos e, assim, escaparemos da questão.

Para enfatizar o fato de que um algoritmo é uma abstração de um programa, eles serão apresentados neste curso em uma linguagem informal conhecida como *pseudo-código*.

Exemplo 1.2.4:

Considere a seguinte solução para o problema da busca.

```
BuscaSimples(A, b)
1  ▷ Recebe ⟨a_1, ..., a_n⟩ ∈ ℤ^× e b ∈ ℤ
2  ▷ Devolve i tal que a_i = b se existir e ⊥ caso contrário
3  for i ← 1 até n
4        do if a_i = b
5              then return i
6  return ⊥
```

Esta disciplina estuda algoritmos. Como podemos garantir que certo algoritmo resolve um problema, ou seja, que ele é correto? Como podemos comparar duas soluções distintas para um mesmo problema? Ou seja, se

conhecemos dois um mais algoritmos corretos para um mesmo problema, como avaliamos qual é melhor?

Avaliaremos os algoritmos corretos a partir da quantidade de recursos que ele consome. Estudaremos particularmente dois recursos: espaço de memória e, principalmente, o tempo de execução.

Nos capítulos seguintes veremos uma série de algoritmos para resolver alguns problemas centrais da computação como o problema da busca e da ordenação. Avaliaremos os algoritmos apresentado quanto sua corretude e sua eficiência em consumo de tempo e espaço de memória.

No Capítulo X apresentaremos o estudo dos algoritmos a partir do método empírico. Relembraremos o método e veremos um exemplo comparando o tempo de execução de duas soluções para o problema da busca em sequências ordenadas. Então exploraremos técnicas para arrsicar modelos matemáticos adequados para avaliar o consumo de tempo dos algoritmos. E finalmente veremos ferramentas matemáticas uteis para comparar funções quanto ao seu crescimento, a chamada notação assintótica. No Capitulo X estudaremos algoritmos de ordenação como estudo de caso da teoria apresentada anteriormente. Veremos uma série de algoritmos que resolvem o mesmo problema e utilizaremos as técnicas apresentadas para construir e testar modelos do consumo de tempo deles. Estudaremos também um limite teórico da eficiência do problema da ordenação e veremos dois algoritmos que superam esse limite utilizando mais informações do que as assumidas no enunciado do teorema. Concluiremos a apostila no Capítulo X apresentando algoritmos e técnicas um pouco mais avançãdos como programação dinâmica e análise amortizada.

Bibliografia

[1] T.H. Cormen. *Algoritmos: teoria e prática*. Campus, 2012. ISBN: 9788535236996.

[2] A.I. Mal'cev. *Algorithms and Recursive Functions*. Wolters-Noorhoff, 1970. URL: `https://books.google.com.br/books?id=x9BqxgEACAAJ`.

[3] R. Sedgewick. *Algorithms in C*. Algorithms in C. Addison Wesley Professional, 2001. ISBN: 9780201756081.

[4] R. Sedgewick e K. Wayne. *Algorithms: Algorithms_4*. Pearson Education, 2011. ISBN: 9780132762564.