Respostas do capítulo 2 do livro: *Introduction to Algorithms*, de Cormen, Thomas H. et al. (3ª ed., 2009)

Abrantes Araújo Silva Filho

2018-03

Sumário

1 O que é este documento? 1

2 Exercícios 2
2.1 Grupo 2.1: . . . 2
2.2 Grupo 2.2: . . . 3
2.3 Grupo 2.3: . . . 3

3 Problemas 4
3.1 Problema 2.1 . . . 4
3.2 Problema 2.2 . . . 4
3.3 Problema 2.3 . . . 4

1 O que é este documento?

Este documento contém as minhas respostas aos exercícios e problemas do capítulo 2 do livro *Introduction to Algorithms*, de Cormen, Thomas H. et al. (3ª ed., de 2009), que utilizei na disciplina de Algoritmos durante minha gradução em Ciência da Computação.

ATENÇÃO: não garanto que tudo aqui está correto, pelo contrário, algumas respostas expressam minha visão particular e podem estar em desacordo com a "resposta padrão" dos autores do livro ou do professor da disciplina de Algoritmos. Também não garanto que todos os exercícios e problemas do capítulo estarão resolvidos aqui.

De qualquer modo, se você quiser utilizar este documento como base para seu próprio estudo, tenha em mente o seguinte:

ESTE DOCUMENTO É FORNECIDO "NO ESTADO EM QUE SE ENCONTRA", SEM GARANTIAS DE QUALQUER NATUREZA, EXPRESSAS OU IMPLÍCITAS. EM NENHUMA HIPÓTESE O AUTOR PODERÁ SER RESPONSABILIZADO POR QUALQUER RECLAMAÇÃO, DANOS OU OUTROS PROBLEMAS DECORRENTES DO USO DESTE CONTEÚDO.

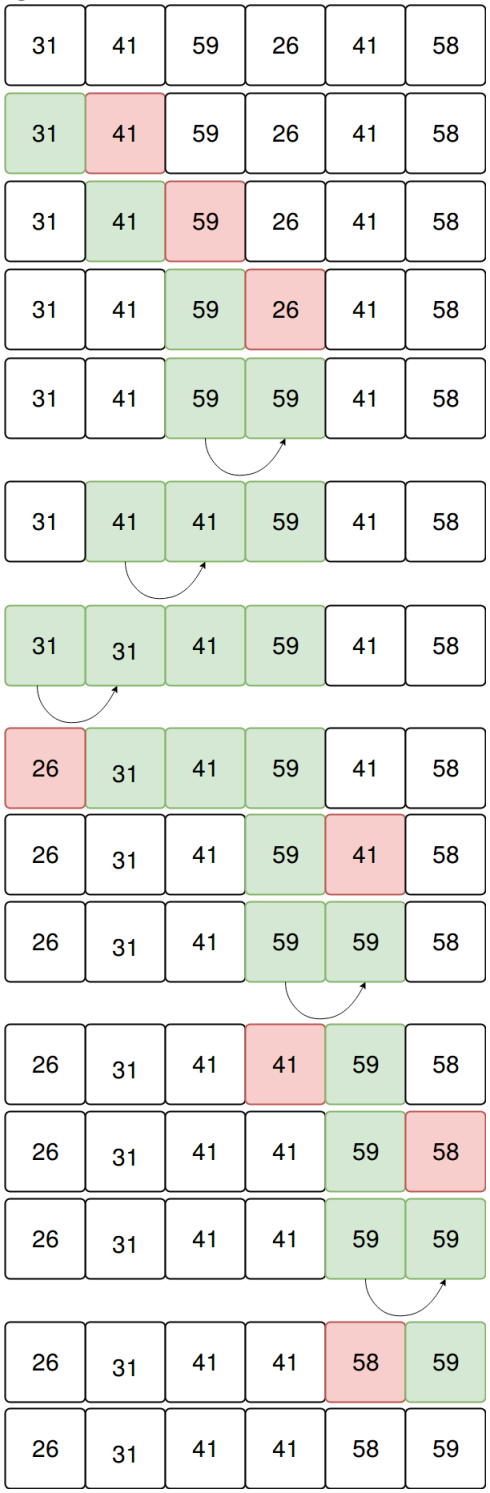
Este documento (em formato PDF), o original em LaTeX, e códigos dos exercícios estão disponíveis no seguinte repositório GitHub: https://github.com/abrantesasf/algoritmos/tree/master/introduction_to_algorithms/cap-02

2 Exercícios

2.1 Grupo 2.1:

Exercício 2.1-1 A ilustração do Insertion Sort para o array [31, 41, 59, 26, 41, 58] é:

Figura 1: Ilustração do Insertion Sort



Exercício 2.1-2 Insertion Sort em ordem decrescente (código completo no repositório GitHub):

```
def insertionSort(l):
    for j in range(1, len(l)):
        atual = l[j]
        i = j - 1

        while i >= 0 and l[i] < atual:
            l[i + 1] = l[i]
            i = i - 1

        l[i + 1] = atual

    return print(l)
```

Exercício 2.1-3 Busca linear pelo índice onde um número se encontra em uma lista (código completo no repositório GitHub):

```
def buscaNumero(l, n):
    for i in range(0, len(l)):
        if l[i] == n:
            return print(i)
    return None
```

Pelo conceito de *loop invariant*, sabemos que a cada iteração de um loop for, o subarray considerado consiste em todos os elementos originalmente no subarray, mas ordenados. Assim, no exemplo acima, temos que checar 3 situações:

- INICIALIZAÇÃO: o *loop invariant* é verdadeiro ANTES da primeira iteração loop? Sim, logo após a alocação inicial do primeiro índice do contador (0), e antes da checagem do primeiro teste do loop, o *loop invariant* é verdadeiro.
- MANUTENÇÃO: o *loop invariant* é verdadeiro a CADA ITERAÇÃO do loop? Sim, a cada iteração o subarray l[i] consiste do mesmo elemento l[i], trivialmente ordenado.
- TÉRMINO: o *loop invariant* é verdadeiro APÓS o término do loop? Sim, pois após o término do loop temos todos os elementos originais.

Portanto, podemos afirmar que o algoritmo acima para a busca linear do índice da localização de um número está correto.

Exercício 2.1-4 ???

2.2 Grupo 2.2:

Exercício 2.2-1

2.3 Grupo 2.3:

Exercício 2.3-1

3 Problemas

3.1 Problema 2.1

3.2 Problema 2.2

3.3 Problema 2.3