Introduction to Algorithms Notes

Jean philo Gong

September 16, 2023

Contents

1 The Role of Algorithms in Computing Notes **5**

1.1 Algorithms 5

1.1.1 What kinds of problems are solved by algorithms 5

1.1.2 Data structures 5

1.1.3 Technique 5

2 Getting Started **7**

2.1 Insertion sort 7

2.1.1 Insertion Sort Algorithm 7

2.1.2 Code Implementation 7

3 Growth of Functions **9**

4 Divide-and-Conquer **11**

5 Probabilistic Analysis and Randomized Algorithms **13**

6 Heapsort **15**

7 Quicksort **17**

8 Sorting in Linear Time **19**

9 Medians and Order Statistics **21**

10 Elementary Data Structures **23**

11 Hash Tables **25**

12 Binary Search Trees **27**

13 Red-Black Trees **29**

14 Augmenting Data Structures **31**

15 Dynamic Programming **33**

16 Greedy Algorithms 35

17 Amortized Analysis 37

18 B-Trees 39

19 Fibonacci Heaps 41

20 Van Emde Boas Trees 43

21 Data Structures for Disjoint Sets 45

22 Graph Algorithms 47

23 Minimum Spanning Trees 49

24 Single-Source Shortest Paths 51

25 All-Pairs Shortest Paths 53

26 Maximum Flow 55

27 Multithreaded Algorithms 57

28 Matrix Operations 59

29 Linear Programming 61

30 Polynomials and the FFT 63

31 Number-Theoretic Algorithms 65

32 String Matching 67

33 Computational Geometry 69

34 NP-Completeness 71

35 Approximation Algorithms 73

36 Mathematical Background 75

37 Problems, Hints, and Solutions 77

Chapter 1

The Role of Algorithms in Computing Notes

1.1 Algorithms

An example of an algorithm is as follows:

> Input: A sequence of n numbers $(a_1, a_2, \ldots, a_n)$.
> Output: A permutation (reordering) $(a'_1, a'_2, \ldots, a'_n)$
> such that $a'_1 \leq a'_2 \leq \ldots \leq a'_n$.

1.1.1 What kinds of problems are solved by algorithms

1.1.2 Data structures

Definition:

> A data structure is a way to store and organize data in order to facilitate access and modifications.

1.1.3 Technique

Chapter 2

Getting Started

2.1 Insertion sort

Input: A sequence of n numbers $(a_1, a_2, \ldots, a_n)$.
Output: A permutation (reordering) $(a'_1, a'_2, \ldots, a'_n)$
such that $a'_1 \leq a'_2 \leq \ldots \leq a'_n$.

Methods

2.1.1 Insertion Sort Algorithm

The insertion sort algorithm can be broken down into the following steps:

1. Define a function to perform the insertion sort operation.
2. Loop starts from the second element.
3. Store the current element as the key.
4. Initialize j as the element just before i.
5. Move elements that are greater than the key to one position ahead of their current position.
6. Place the key in its correct position.

2.1.2 Code Implementation

The Python code for the insertion sort algorithm is given below:

```
def insertion_sort(arr):
    for i in range(1, len(arr)):
        key = arr[i]
```

```
j = i - 1
while j >= 0 and key < arr[j]:
    arr[j + 1] = arr[j]
    j -= 1
arr[j + 1] = key
```

Chapter 3

Growth of Functions

Chapter 4

Divide-and-Conquer

Chapter 5

Probabilistic Analysis and Randomized Algorithms

Chapter 6

Heapsort

Chapter 7

Quicksort

Chapter 8

Sorting in Linear Time

Chapter 9

Medians and Order Statistics

Chapter 10

Elementary Data Structures

Chapter 11

Hash Tables

Chapter 12

Binary Search Trees

Chapter 13

Red-Black Trees

Chapter 14

Augmenting Data Structures

Chapter 15

Dynamic Programming

Chapter 16

Greedy Algorithms

Chapter 17

Amortized Analysis

Chapter 18

B-Trees

Chapter 19

Fibonacci Heaps

Chapter 20

Van Emde Boas Trees

Chapter 21

Data Structures for Disjoint Sets

Chapter 22

Graph Algorithms

Chapter 23

Minimum Spanning Trees

Chapter 24

Single-Source Shortest Paths

Chapter 25

All-Pairs Shortest Paths

Chapter 26

Maximum Flow

Chapter 27

Multithreaded Algorithms

Chapter 28

Matrix Operations

Chapter 29

Linear Programming

Chapter 30

Polynomials and the FFT

Chapter 31

Number-Theoretic Algorithms

Chapter 32

String Matching

Chapter 33

Computational Geometry

Chapter 34

NP-Completeness

Chapter 35

Approximation Algorithms

Chapter 36

Mathematical Background

Chapter 37

Problems, Hints, and Solutions