# DSA Mini Textbook

Theo Park

# Contents

# Preface

## Chapter 1

# Runtime Analysis

*Algorithms* are any well-defined computational procedures that take some value(s) as input and produce more value(s) as output. They are **effective**, **precise**, and **finite**. There are several ways to analyze the runtime of an algorithm.

## 1.1 Power Law

1. For the algorithm, get a table for the input size $n$ and the runtime $T(n)$.

| $n$ | $T(n)$ |
|---|---|
| 250 | 0.0 |
| 500 | 0.012 |
| 1000 | 0.0954 |
| 2000 | 0.7727 |
| 4000 | 6.1664 |

2. Make sure that the data plots:

   - **have enough data plots.** For instance, if there are only two data plots, you should not make the power law conjecture.
   - **fits the power law.** You can verify this by finding the ratio between data plots.

| $n$ | $T(n)$ | ratio |
|---|---|---|
| 250 | 0.0 | – |
| 500 | 0.012 | – |
| 1000 | 0.0954 | 0.0954 / 0.012 = 7.95 |
| 2000 | 0.7727 | 0.7727 / 0.0954 = 8.10 |
| 4000 | 6.1664 | 6.1664 / 0.7727 = 7.98 |

For the ratios we found,

## 1.2 Runtime Expressions

## 1.3 Asymptotic Runtime Analysis

## 1.4 Recursive Relationship

# Chapter 2

# Intro to Data Structures

*Data structures* are collections of data values, the relationships among them, and the functions or operations that can be applied to the data. All three characteristics need to be present.

## 2.1 Array

*Array* is a linear container of items.

| | | | | | | |
|---|---|---|---|---|---|---|
| Array length 6 | 250 | 251 | 252 | 253 | 254 | 255 |
| | 0 | 1 | 2 | 3 | 4 | 5 |

- Access time: $\Theta(1)$
- Inserting $n$ items in the *tail* for array size $n$: $\Theta(1)$ per item, $n \times \Theta(1) \in \Theta(1)$
- Inserting $n$ items in the *tail* for array size *unknown*: $\Theta(n)$ per item, $n \times \Theta(n) \in \Theta(n)$

Lesson? **Keep track of the tail!**

## 2.2 Linked List

## 2.3 Stack

## 2.4 Queue

## 2.5 Binary Heap

## 2.6 Tree

# Chapter 3

# Sorting Algorithms

## 3.1 Bubble Sort

## 3.2 Selection Sort

## 3.3 Insertion Sort

## 3.4 Shell Sort

## 3.5 Heap Sort

### 3.5.1 Building a Heap – Top-down v.s. Bottom-up

### 3.5.2 Sort Down Algorithm

## 3.6 Merge Sort

### 3.6.1 Merge Algorithm

## 3.7 Quick Sort

### 3.7.1 Pivot and Partition

## 3.8 Decision Tree and $\Omega(n \log n)$ Limit for Comparison Sorting Algorithms

## 3.9 Counting Sort

## 3.10 Radix Sort

# Chapter 4

# Hash Tables

## 4.1 Division Method

## 4.2 Multiplication Method

## 4.3 Collision

### 4.3.1 Chaining

### 4.3.2 Open Addressing

# Chapter 5

# Search Tree

## 5.1 Binary Search Tree and Its Limit

## 5.2 2-3 Tree

## 5.3 Red-Black Tree

## 5.4 Left-Leaning Red-Black Tree

### 5.4.1 Deletion in LLRBT

# Chapter 6

# Graph Traversal

## 6.1 Adjacency Matrix and List

## 6.2 DFS

## 6.3 BFS

# Chapter 7

# Directed Graphs

## 7.1 Strong Connectivity

### 7.1.1 Brute-force Strong Connectivity Algorithm

### 7.1.2 Brute-force using Stack

### 7.1.3 Strongly Connected Components and Kosaraju's Algorithm

## 7.2 Directed Acyclic Graphs

### 7.2.1 Topological Sort

# Chapter 8

# Weighted Graphs

## 8.1 Shortest Path

### 8.1.1 Dijkstra's Algorithm

### 8.1.2 Bellman-Ford Algorithm

## 8.2 Articulation Points

## 8.3 Minimum Spanning Tree

### 8.3.1 Cycle and Cut Properties

### 8.3.2 Prim's Algorithm

## 8.4 Union-Find

### 8.4.1 Kruskal MST Algorithm

# Chapter 9

# Strings

## 9.1 Brute-force String Pattern Matching

## 9.2 KMP Algorithm

## 9.3 Trie

## 9.4 PATRICIA

## 9.5 Huffman Coding