

TypeScript

ALGORITHMS

Version 0.0.1

Contents

1	Introduction	3
2	Algorithm Analysis	4
2.1	Amortized Analysis	4
3	Data Structures and Algorithms	5
3.1	Bits	6
3.1.1	Overview	6
3.1.2	Bit Parity	6
3.1.3	Bit Shift Operator	7
3.2	Stacks and Queues	7
3.2.1	Overview	7
3.2.2	Fixed Stack	7
4	Problem Solving Methods	8
4.1	Greedy Algorithms	8
4.1.1	Overview	8
4.1.2	Dijkstra's Shortest Path Algorithm	8
4.2	Recursion	8
4.2.1	Overview	8
4.2.2	Fibonacci Sequence	9
5	Domain Specific	10
5.1	Language	10
5.1.1	This	10
5.1.2	Event Loop	10
5.1.3	Asynchronous Programming	10
5.1.4	Runtime Environments	10

6 Appendix 11
6.1 Resources 11

Chapter 1

Introduction

In-progress book about algorithms and data structures in TypeScript.

Chapter 2

Algorithm Analysis

2.1 Amortized Analysis

The motivation for amortized analysis is that looking at the worst-case run time can be too pessimistic. Instead, amortized analysis averages the running times of operations in a sequence over that sequence. Amortized analysis is a useful tool that complements other techniques such as worst-case and average-case analysis.

— https://en.wikipedia.org/wiki/Amortized_analysis

Imagine a dynamic list backed by a fixed size array that doubles once capacity is reached and require n inserts into the new fixed size array.

n	# of inserts
1	1
2	1
3	1
...	1
n	n
...	n + 1

$$\frac{\# \text{ of inserts}}{n} \rightarrow \frac{n+n+1}{n} \rightarrow \frac{2n+1}{n} \rightarrow \frac{2n}{n} \rightarrow 2$$

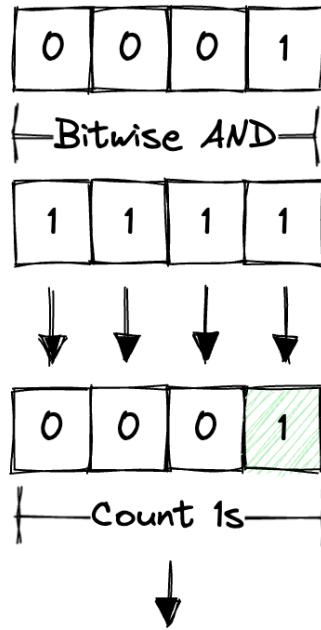
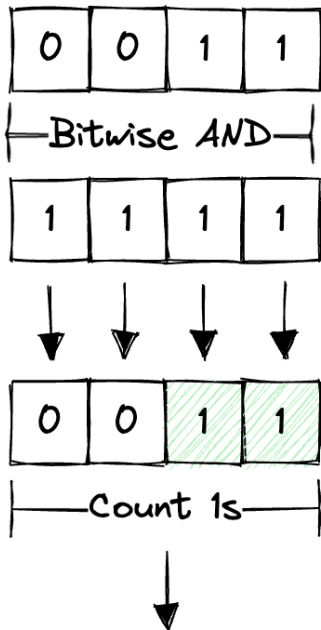
Chapter 3

Data Structures and Algorithms

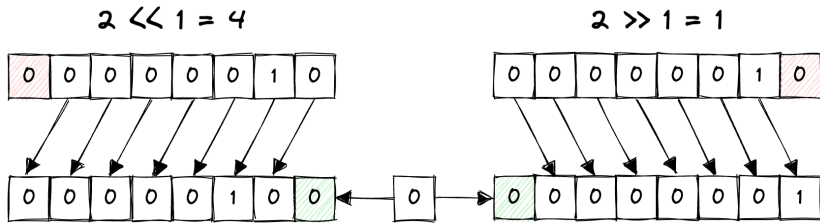
3.1 Bits

3.1.1 Overview

3.1.2 Bit Parity



3.1.3 Bit Shift Operator



3.2 Stacks and Queues

3.2.1 Overview

3.2.2 Fixed Stack

Chapter 4

Problem Solving Methods

4.1 Greedy Algorithms

4.1.1 Overview

A greedy algorithm is any algorithm that follows the problem-solving heuristic of making the locally optimal choice at each stage. In many problems, a greedy strategy does not produce an optimal solution, but a greedy heuristic can yield locally optimal solutions that approximate a globally optimal solution in a reasonable amount of time.

— https://en.wikipedia.org/wiki/Greedy_algorithm

4.1.2 Dijkstra's Shortest Path Algorithm

Dijkstra's shortest path algorithm is a search algorithm that finds the shortest path between a vertex and other vertices in a weighted graph.

4.2 Recursion

4.2.1 Overview

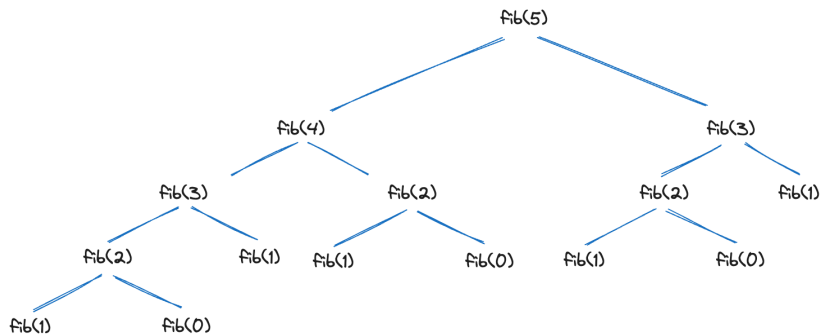
The power of recursion evidently lies in the possibility of defining an infinite set of objects by a finite statement. In the same manner, an infinite number of computations can be described by a finite recursive program, even if this program contains no explicit repetitions.

4.2.2 Fibonacci Sequence

$$F_0 = 0$$

$$F_1 = 1$$

$$F_n = F_{n-1} + F_{n-2} \quad \text{for } n > 1$$



$$F_n = F_{n-1} + F_{n-2}$$

$$F_5 = F_4 + F_3$$

$$F_5 = (F_3 + F_2) + (F_2 + F_1)$$

$$F_5 = ((F_2 + F_1) + (F_1 + F_0)) + ((F_1 + F_0) + F_1)$$

$$F_5 = (((F_1 + F_0) + F_1) + (F_1 + F_0)) + ((F_1 + F_0) + F_1)$$

$$F_5 = (((1 + 0) + 1) + (1 + 0)) + ((1 + 0) + 1)$$

$$F_5 = 5$$

```
export function fib(n: number): number {  
  if (n == 0 || n == 1) {  
    return n  
  }  
  return fib(n - 1) + fib(n - 2)  
}
```

¹<https://archive.org/details/algorithmsdatast00wirth/page/126>

Chapter 5

Domain Specific

5.1 Language

5.1.1 This

5.1.2 Event Loop

5.1.3 Asynchronous Programming

5.1.3.1 Promises

5.1.3.2 Async/Await

5.1.4 Runtime Environments

5.1.4.1 Browser

5.1.4.2 Server

Chapter 6

Appendix

6.1 Resources

- LeetCode
- Project Euler
- The Algorithm Design Manual
- Elements of Programming Interviews