TypeScript ALGORITHMS

Version 0.0.1

Contents

Chapter 1

Introduction

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

Chapter 2

Algorithm Analysis

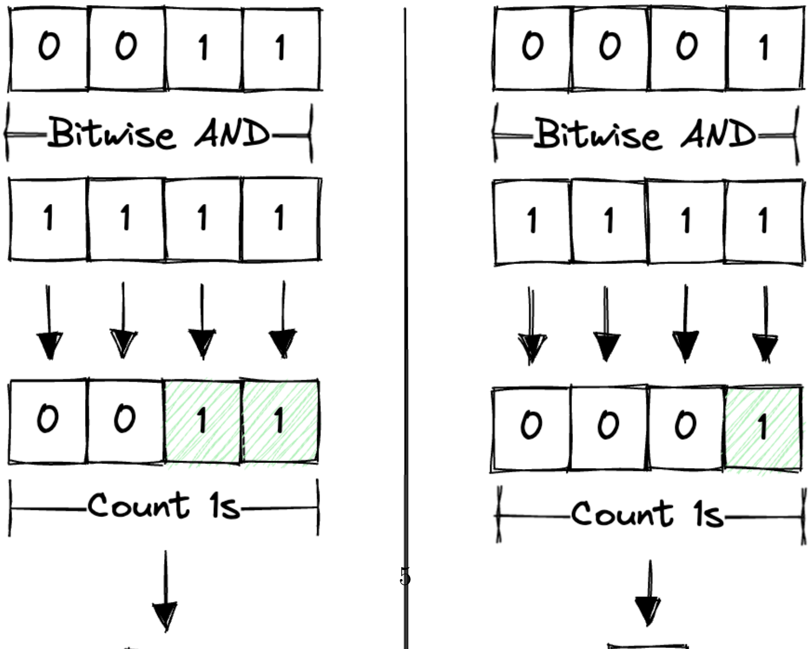
Chapter 3

Data Structures and Algorithms

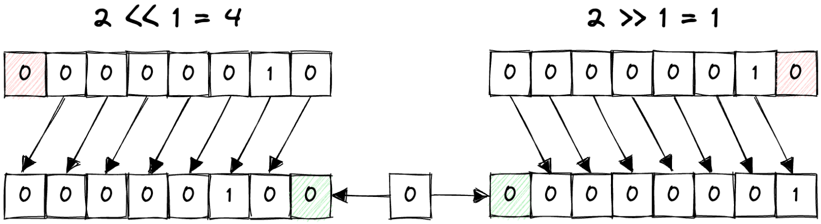
3.1 Bits

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 Recursion

4.1.1 Overview

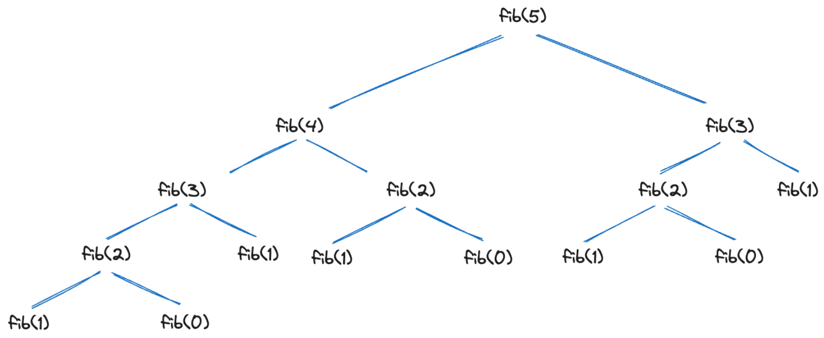
The power of recursion evidently lies in the possibility of defining an infinite set of objects by a finite statement. In the same manner, an infinite number of computations can be described by a finite recursive program, even if this program contains no explicit repetitions.

— Niklaus Wirth, Algorithms + Data Structures = Programs, 1976[1]

4.1.2 Fibonacci Sequence

$F_0 = 0$
$F_1 = 1$
$F_n = F_{n-1} + F_{n-2} \qquad for\ n > 1$

[1]https://archive.org/details/algorithmsdatast00wirt/page/126]



$$
\begin{aligned}
&F_n = F_{n-1} + F_{n-2} \\
&F_5 = F_4 + F_3 \\
&F_5 = (F_3 + F_2) + (F_2 + F_1) \\
&F_5 = ((F_2 + F_1) + (F_1 + F_0)) + ((F_1 + F_0) + F_1) \\
&F_5 = (((F_1 + F_0) + F_1) + (F_1 + F_0)) + ((F_1 + F_0) + F_1) \\
&F_5 = (((1 + 0) + 1) + (1 + 0)) + ((1 + 0) + 1) \\
&F_5 = 5
\end{aligned}
$$

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

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