



TypeScript

ALGORITHMS

Version 0.0.1

Contents

1	Introduction	2
2	Algorithms and Data Structures	3
2.1	Algorithm Analysis	4
2.2	Bits	4
2.2.1	Bit Parity	4
2.2.2	Bit Shift Operator	5
2.3	Recursion	5
2.3.1	Fibonacci Sequence	5
2.4	Stacks	6
3	Domain Specific	7
3.1	Language	7
3.1.1	This	7
3.1.2	Event Loop	7
3.1.3	Asynchronous Programming	7
3.1.4	Runtime Environments	7
4	Appendix	8
4.1	Resources	8

Chapter 1

Introduction

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

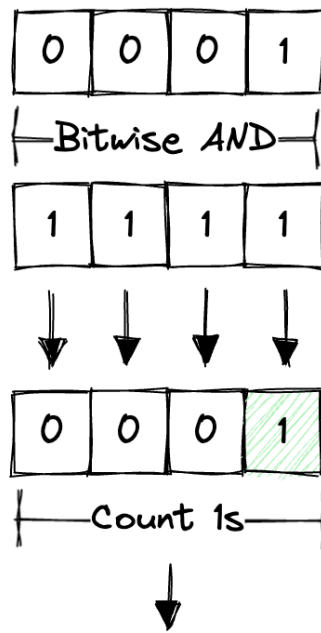
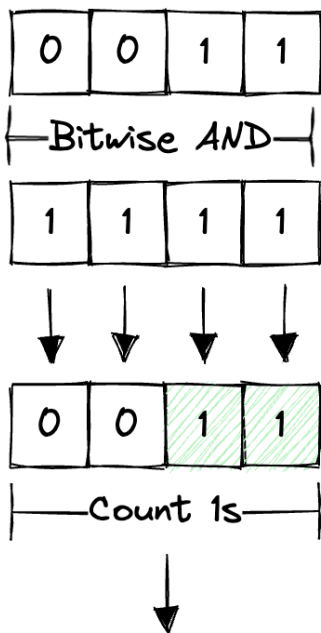
Chapter 2

Algorithms and Data Structures

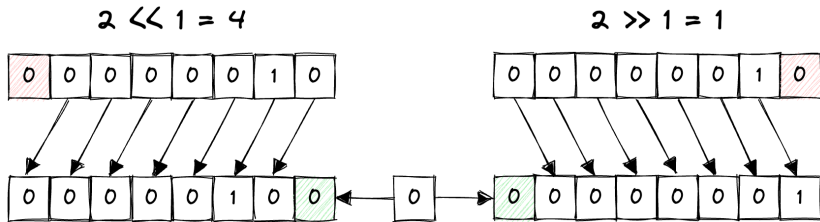
2.1 Algorithm Analysis

2.2 Bits

2.2.1 Bit Parity



2.2.2 Bit Shift Operator



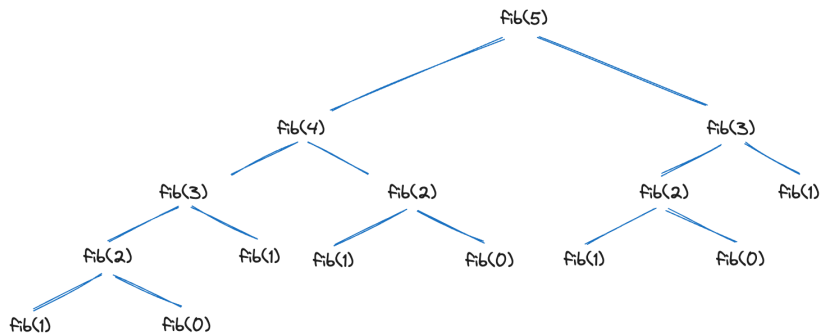
2.3 Recursion

2.3.1 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)  
}
```

2.4 Stacks

Exercises:

- Implement a stack data structure backed by a fixed size array.

Chapter 3

Domain Specific

3.1 Language

3.1.1 This

3.1.2 Event Loop

3.1.3 Asynchronous Programming

3.1.3.1 Promises

3.1.3.2 Async/Await

3.1.4 Runtime Environments

3.1.4.1 Browser

3.1.4.2 Server

Chapter 4

Appendix

4.1 Resources

- [LeetCode](#)
- [Project Euler](#)
- [The Algorithm Design Manual](#)
- [Elements of Programming Interviews](#)