

The Author  
Mike Chyson (Li Mingming)

The Big Book of  
**ALGORITHMS**  
Theory and practice of algorithms

*First created: Dec, 10, 2020*  
*Last modified: Tuesday 15<sup>th</sup> December, 2020*



# Dedication



# Contents

<b>Contents</b>	<b>1</b>
<b>1 Theory</b>	<b>3</b>
1.1 What is algorithm? . . . . .	3
1.2 Instance of a problem . . . . .	3
1.3 Correct algorithm . . . . .	3
1.4 Data structure . . . . .	3
1.5 The core technique . . . . .	4
1.6 Algorithm efficiency . . . . .	4
1.7 Algorithms use information . . . . .	4
1.8 Loop invariant . . . . .	4
1.9 Analyzing an algorithm . . . . .	4
1.9.1 Worst-case analysis . . . . .	5
1.10 Resource model . . . . .	5
1.11 Growth of functions . . . . .	5
1.11.1 $\Theta$ -notation . . . . .	5
1.11.2 $O$ -notation . . . . .	5
1.11.3 $\Omega$ -notation . . . . .	5
<b>I Data Structure</b>	<b>7</b>
<b>2 Array</b>	<b>9</b>
2.1 What is an array? . . . . .	9



# Chapter 1

## Theory

### 1.1 What is algorithm?

$$input \longrightarrow algorithm \longrightarrow output$$

An algorithm is a sequence of computational that transform the input into the output, it describe a specific computational procedure for achieving the input/output relationship.

### 1.2 Instance of a problem

An instance of a problem is the input needed to compute a solution to the problem.

### 1.3 Correct algorithm

For every input instance, a correct algorithm halts out the corrent output.

### 1.4 Data structure

Data structure is a way to store and organize data in order to faciliate access and modifications. No single data structure works well for all purpose.

## 1.5 The core technique

Learning an algorithm is to learn the technique of algorithm design and analysis.

## 1.6 Algorithm efficiency

Computers are not infinitely fast and memory is not free, thus the efficiency of a algorithm matters.

## 1.7 Algorithms use information

There is information in the problem, the general is, the more information you use, the more efficiency the algorithm is. Data structure is a way to use this information.

## 1.8 Loop invariant

There can be infinite input cases, so it is hard to say an algorithm is correct. Loop invariant is a way to guarantee that the algorithm is correct.

There are 3 elements in a loop invariant:

**initialization** It is true prior to the first iteration of the loop.

**maintenance** If it is true before an iteration of the loop, it remains true before the next iteration.

**termination** When the loop terminates, the invariant gives us a useful property that helps show that the algorithm is correct.

## 1.9 Analyzing an algorithm

Analyzing an algorithm is to predict the resources consumed. The most important resources is time and space.

In general, the time grows with the size of the input, so it is traditional to describe the running time as the function of the size of its input.



### 1.9.1 Worst-case analysis

Because the behavior of an algorithm may be different for each possible input, we need a means for summarizing that behavior in simple, easily understood formulas. The reason to analyze worst-case running time is as follows:

1. It gives an upper bound on the running time.
2. Worst case occurs fairly often.
3. The “average case” is often roughly as bad as the worst case.

## 1.10 Resource model

Before we can analyze an algorithm, we must have a model of the implementation technology that we will use, including a model for the resources of that technology and their costs. However, the focus is algorithm, not the tedious hardware detail. We shall assume a generic one-processor, random-access machine (RAM) model of computation as our implementation technology and understand that our algorithms will be implemented as computer programs. In the RAM model, instructions are executed one after another, with no concurrent operations.

## 1.11 Growth of functions

Although we can sometimes determine the exact running time of an algorithm, the extra precision is not usually worth the effort of computing it. When we look at input sizes large enough to make only the order of growth of the running time relevant, we are studying the **asymptotic efficiency** of algorithms.

### 1.11.1 $\Theta$ -notation

### 1.11.2 $O$ -notation

### 1.11.3 $\Omega$ -notation



## Part I

# Data Structure



## Chapter 2

# Array

### 2.1 What is an array?

An Array is a collection of items. The items are stored in neighboring (contiguous) memory locations.

For example:

2	1	3	2	1
---	---	---	---	---

