In this week, we will combine Chapters 3 and 5, but we will use sorting as examples to do analysis.

We will mainly talk about three algorithms: bubble sort, selection and merge sort. We will show how algorithms will really matter by having experimental results. Also, we will learn how could we count the number of steps that are needed to solve a problem, which is the essential step to do algorithm analysis.

1 Algorithm, Algorithm Analysis, Recursion

First, let’s start with what is an algorithm. An algorithm is a finite set of instructions having the following characteristics:

1. Precision: The steps are precisely stated.
2. Uniqueness: The intermediate and final results of each step of execution are uniquely defined and depend only on the inputs and the results of the proceeding steps.
3. Finiteness: The algorithm should stop after finite steps of execution of instructions.
4. Input: It receives input.
5. Output: It produces output.
6. General: It may be applied to a set of input(not for a special case).

After all these characteristics, we need to talk about the efficiency of a program. There are two criteria about efficiency: time and space. Of these two, time efficiency are of more interest to us.

We will look at algorithm analysis for best-case, worst-case and average-case.

In algorithm analysis, we will use order of the function, which can be captured by Big-$O$, Big-$\Theta$, Big-$\Omega$ notations.

We say that $f(n)$ is of order at most $g(n)$, written as

$$f(n) = O(g(n))$$
if there exists a positive constant $C_1$ such that

$$|f(n)| \leq C_1|g(n)|$$

We say that $f(n)$ is of order at most $g(n)$, written as

$$f(n) = \Omega(g(n))$$

if there exists a positive constant $C_1$ such that

$$|f(n)| \geq C_1|g(n)|$$

We say that $f(n)$ is of order $g(n)$, written as

$$f(n) = \Theta(g(n))$$

if there exists positive constant $C_1, C_2$ such that

$$C_1|g(n)| \leq |f(n)| \leq C_2|g(n)|$$

Examples:

1.1 Counting

The basic method for algorithm analysis is to do counting: how many steps it takes to complete an algorithm for a given input of size $n$.

1.2 Recursion

We will review what’s recursion.

1.3 Examples

1. calculate $2^n$

2. Calculate $GCD(m, n)$ by Euclidean algorithm.
2 Sorting Algorithm Examples

Now, let’s start look at three sorting examples:

2.1 Bubble Sort

The bubble sort works by comparing each item in the list with the item next to it, and swapping them if required. The algorithm repeats this process until it makes a pass all the way through the list without swapping any items (in other words, all items are in the correct order). This causes larger values to “bubble” to the end of the list while smaller values “sink” towards the beginning of the list.

2.2 Selection Sort

It repeated looks for the largest one, and move it to the end of the list.
2.3 Merge Sort

The merge sort splits the list to be sorted into two equal halves, and places them in separate arrays. Each array is recursively sorted, and then merged back together to form the final sorted list.