

Contents

1	Introduction	3
1.1	What is <i>competitive programming</i> ?	3
1.2	What is the prerequisite for <i>competitive programming</i> ?	4
1.3	Where can I practice it?	4
1.4	How I design this book?	4
2	Counting	7
2.1	Recurrence relations	7
2.2	Matrix exponentiation	7
2.3	Binomial coefficient	7
2.4	Pigeonhole principle	7
2.5	Inclusion-exclusion principle	7
2.6	Subsets and Permutations	7
2.7	Indistinguishable and distinguishable objects	7
2.8	Lattice problems	7
2.9	Counting on graphs	7
2.10	Polya enumeration theorem	7
3	Moduli Arithmetic	9
4	Probability and Expectation value	11
5	Number Theory	13

Chapter 1

Introduction

Contents

1.1	What is <i>competitive programming</i> ?	3
1.2	What is the prerequisite for <i>competitive programming</i> ?	4
1.3	Where can I practice it?	4
1.4	How I design this book?	4

1.1 What is *competitive programming*?

Competitive programming is basically a type of programming that you need to solve the algorithmic problems in a competitive manner, e.g. fast and efficient, in order to compete with others. Usually, two things needed to consider for programming competitively are (1) **time constraint** and (2) **memory constraint**, and most of the time the deciding factor is (1) time constraint mainly because, for instance, if you have to make an algorithm that takes an input of order $O(n^2)$ but you need to compute an output in $O(n \log n)$, this would be impossible as the bottleneck of your program is $O(n^2)$ from taking in an input.

For the sake of people who still have no idea what *competitive programming* is, I would like to give an example of problems, which are so-called competitive programming tasks:

Problem: Let X be the array of size N of integers with elements x_i . Suppose you want to find the minimum number of swaps with consecutive elements in the list do you need to do in order to make the array sorted, either in ascending order or descending order.

Input constraint: $1 \leq N \leq 10^5$, $1 \leq x_i \leq 10^{18}$.

Time limit: 1 second

In this problem, you have to find how to swap elements in order to make the array sorted and why the method you found is optimal. If you have some backgrounds on this kind of problems, first you would realize that you have to solve this problem twice, one for the ascending order case and another one for the descending order case. Then, you would realize that you just need to find that if you can swap any consecutive elements in order to make the list sorted, then you will swap these two elements and count how many times you do swap. However, the method we just said is in order of $O(N^2)$ and would be impossible to run in the time limit. One approach you will learn in the future is to use either *Merge sort* or *Fenwick Tree/Segment Tree* to count the number of times you do swap, which is in the order of $O(n \log n)$ and faster enough to finish in the time constraint.

1.2 What is the prerequisite for *competitive programming*?

These are very formulaic. I will list everything I know that is important to *competitive programming*:

1. Mathematics

- Combinatorics (Counting /

- Number Theory (Inverse modular, Euclid)

- Geometry (Convex Hull, Trivial Geometry)

- Graph Theory (BFS/DFS/Tree/Shortest Path/Euclidean/Hamiltonian/Network Flow/Matching)

2. Algorithms

- Numerical Algorithms

- Solving paradigm Algorithms (DP/Greedy/Brute force/D&Q)

- Graph Algorithms

- Computational Geometry Algorithms

3. Data Structures

- Basic (Queue/Stack/Linked list)

- Union-Disjoint Set

- Fenwick Tree

- Segment Tree

4. STL

1.3 Where can I practice it?

Again, there are so many websites you can practice at:

1. Topcoder

2. Codeforces

3. SPOJ

4. Google Code Jam

5. Facebook Hacker Cup

6. UVA Online Judge

That's more than enough that you can do.

1.4 How I design this book?

First, I will delve deeper into mathematics you needed to understand and will go to algorithms & data structures.

Mathematics

Chapter 2

Counting

Contents

2.1	Recurrence relations	7
2.2	Matrix exponentiation	7
2.3	Binomial coefficient	7
2.4	Pigeonhole principle	7
2.5	Inclusion-exclusion principle	7
2.6	Subsets and Permutations	7
2.7	Indistinguishable and distinguishable objects	7
2.8	Lattice problems	7
2.9	Counting on graphs	7
2.10	Polya enumeration theorem	7

2.1	Recurrence relations
2.2	Matrix exponentiation
2.3	Binomial coefficient
2.4	Pigeonhole principle
2.5	Inclusion-exclusion principle
2.6	Subsets and Permutations
2.7	Indistinguishable and distinguishable objects
2.8	Lattice problems
2.9	Counting on graphs
2.10	Polya enumeration theorem

Chapter 3

Moduli Arithmetic

Chapter 4

Probability and Expectation value

Chapter 5

Number Theory

Mathematics