2022-2023 Introductory Math

TJHSST SCT Officers

September 2022

1 Introduction

Many recent computing contests have involved more math than before. Having a basic foundation for mathematics is important for tackling many algorithmic problems. In this lecture we'll cover some basic math-based concepts.

2 Combinatorics

2.1 Basic Principles

Many competitive programming problems involve counting possible arrangements or computing probabilities. In these situations, finding the correct algorithm quickly is important. First, let's review the basics of combinatorics principles, and then work through a few simple problems.

1. We can arrange n distinct objects in $n!$ different ways.
2. From n distinct objects, we can pick $\frac{n!}{(n-r)!}$ arrangements of r objects where order matters. Arrangements like these (where the order of the objects is important) are called permutations.
3. We can pick $\frac{n!}{r!(n-r)!}$ arrangements of r objects where order does not matter. These arrangements are called combinations.
4. We can arrange n distinct objects in r distinct subsets in r^n different ways.
5. We can put n identical objects into r labeled boxes in $\binom{n+k-1}{n}$ ways.
6. If there are many incorrect cases in the collection of all possible cases, we can find number of correct cases by subtracting the number of incorrect cases from the total number of cases.

2.2 Independent vs. Dependent

When two events are independent of each other, the outcome of the first event will not affect the outcome of the second. Likewise, as the name suggests, when two events are dependent on one another, the outcome of the first event will affect the outcome of the second event.

2.3 Examples

Recognizing when to apply the correct principles comes with practice. Here are some practice problems that outline basic combinatorics strategies.

1. (UVa 11401 - Triangle Counting): You are given n rods of length $1, 2, ..., n$ where $1 \leq n \leq 10^6$. You have to pick any 3 of them and build a triangle. How many distinct triangles can you make? Note that two triangles will be considered different if they have at least 1 pair of arms with different lengths. (Hint: consider answers for multiple small n)

   Solution: We recognize that the number of triangles that can be formed with longest side n is $f(n) = (n-3) + (n-5) + (n-7)...$ for as long as each term is positive, and $f(n) + f(n-1) = \binom{n-2}{2}$. This observation can be used to find a recurrence relation for the total number of triangles that can be formed.

2. (USACO 2019 Gold - Cow Poetry): Unbeknownst to Farmer John, Bessie is quite the patron of the arts! Most recently, she has begun studying many of the great poets, and now, she wants to try writing some poetry of her own. Bessie knows N words ($1 \leq N \leq 5000$), and she wants to arrange them into poems. Bessie has determined the length, in syllables, of each of her words, and she has also assigned them into "rhyme classes". Every word rhymes only with other words in the same rhyme class. Bessie's poems each include M lines ($1 \leq M \leq 10^5$), and each line must consist of K syllables ($1 \leq K \leq 5000$). Moreover, Bessie's poetry must adhere to a specific rhyme scheme. Bessie would like to know how many different poems she can write that satisfy the given constraints.

   Solution: Let us define the frequency of each rhyme class to be f_c and the number of ways to complete a line with K syllables that ends with a rhyme as w_n. We can then write the number of ways to form lines for that particular rhyme class as

   $$\sum_{n=1}^{num_{rhymes}} w_n^{f_i}$$

   From here, we see that the overall answer is the combined product of this summation for each rhyme class. To find the number of ways to form a line for a particular rhyme, we can use dynamic programming. For the number of syllables in each word w_s, we see that $dp[w_s + i]$ += $dp[i]$. When $w_s + i$ is equal to K, it represents a new way to form a line that ends in the rhyme class of that word.

3 Binomial Theorem

Expanding $(x+y)^n$ can get tedious, especially once n gets larger and drawing out Pascal's triangle takes too much time. To simplify this process, there exists the binomial theorem:

$$(x+y)^n = \sum_{k=0}^{n} \binom{n}{k} x^{n-k}y^k = x^n + \binom{n}{1}x^{n-1}y + ... + \binom{n}{n-1}xy^{n-1} + y^n$$

This theorem becomes very useful when you only need to find one term out of an otherwise expansive list of terms.

4 Number Theory

Modular Arithmetic works with remainders instead of integers. For example,

$$9 \bmod 5 = 4$$

since the remainder of 9 when divided by 5 is 4. In contests, you'll often see modular arithmetic used to avoid dealing with large numbers that overflow.

4.1 Modular Arithmetic Properties

Some common properties in modular arithmetic:

$$(a+b) \bmod m = (a \bmod m + b \bmod m) \bmod m$$

$$(a-b) \bmod m = (a \bmod m - b \bmod m) \bmod m$$

$$(a \cdot b) \pmod m = ((a \bmod m) \cdot (b \bmod m)) \bmod m$$

$$a^b \bmod m = (a \bmod m)^b \bmod m$$

4.2 Tips & Tricks

- If we're taking the modulo of some set of numbers over and over again(i.e powers of numbers), it may be faster to pre-calculate the modulo of each of these numbers beforehand.
- Since the % operator has a much higher constant factor compared to more elementary operators like addition or subtraction, always use addition and subtraction when you have the choice. For example,

$$9 \bmod 5 = 9 - 5.$$

- Take the modulo of each number before performing operations to prevent overflow.
- While debugging, if you come across a negative number it almost always means overflow.

4.3 Modular Inverse

Dividing can be very difficult while in some mod n. For example,

$$(9/3) \bmod 5 \neq ((9 \bmod 5)/(3 \bmod 5)) \bmod 5$$

Luckily, using modular inverses, we can safely divide numbers without worrying about mistakes during division. The modular inverse of a number is equivalent to the reciprocal but in a certain mod.

$$a/b \bmod m = a * i \bmod m$$

where i is the modular inverse of b. The modular inverse of $b \bmod m$, is equivalent to $b^{m-2} \bmod m$ since by Fermat's:

$$b^{m-1} \bmod m \equiv 1 \bmod m$$

if m is prime. Finding b^{m-2} is a much simpler task which can be solved using binary exponentation or pre-calculating the powers of a number as mentioned before.

4.4 Example Problems

Calculate n binomial coefficients modulo $10^9 + 7$. A binomial coefficient $\binom{a}{b}$ can be calculated using the formula $\frac{a!}{b!(a-b)!}$. We assume that a and b are integers and $0 \leq b \leq a$.