1 Prime Numbers

A number is prime if its only positive factors are 1 and itself.

1.1 Factorization

A number can be factorized as:

$$n = p_1^{\alpha_1} p_2^{\alpha_2} p_3^{\alpha_3} \cdots p_k^{\alpha_k}$$

where α_i are non-negative integers (not necessarily distinct) and p_i are distinct prime numbers.

1.2 Useful Formulas

- Number of factors of n:
$$N = \prod_{i=1}^{k} (1 + \alpha_i)$$
- Number of even factors: If $p_1 = 2$ with exponent α_1, the number of even factors is $\alpha_1 \cdot \prod_{i=2}^{k}(1+\alpha_i)$. If no factor is 2, there are no even factors.
- Number of odd factors: Total factors minus even factors:
$$\text{Odd factors} = N - \text{Even factors}$$
- Number of factors divisible by a prime p_k: Fix p_k's exponent and compute:
$$(1 + \alpha_k) \cdot \prod_{i \neq k} (1 + \alpha_i)$$
- Sum of factors of n:
$$S_n = \prod_{i=1}^{k} \frac{p_i^{\alpha_i+1} - 1}{p_i - 1}$$
- Product of factors of n:
$$P_n = n^{\frac{N}{2}}$$
- Check if a number is perfect:
$$n = S_n - n$$
- Density of prime numbers between 1 and n:
$$\pi(n) \approx \frac{n}{\ln(n)}$$
- Euler Totient function (number of integers coprime to n):
$$\varphi(n) = \prod_{i=1}^{k} p_i^{\alpha_i - 1}(p_i - 1)$$
If n is prime:
$$\varphi(n) = n - 1$$

1.3 Popular Conjectures

- Goldbach's Conjecture: Every even integer $n > 2$ can be expressed as $n = a + b$, where both a and b are primes.
- Twin Prime Conjecture: There are infinitely many pairs of primes $\{p, p + 2\}$.
- Legendre's Conjecture: For any positive integer n, there is at least one prime between n^2 and $(n + 1)^2$.

1.4 Optimized Algorithms

- Primality testing and prime factorization in $O(\sqrt{n})$ time, improving on the naive $O(n)$.
- Sieve of Eratosthenes: Preprocesses an array to check if a number between 2 and n is prime or find one prime factor. For an array sieve:
  - sieve[k] $= 0$ indicates k is prime.
  - sieve[k] $\neq 0$ indicates k is not prime, with sieve[k] as a prime factor.

  Running time: $O(n \log \log n)$, nearly linear.
- Euclid's Algorithm: Computes LCM and GCD:

$$L_{a,b} = \frac{ab}{G_{a,b}}$$

  Runs in $O(\log n)$ time:

$$\gcd(a, b) = \begin{cases} a & \text{if } b = 0 \\ \gcd(b, a \mod b) & \text{if } b \neq 0 \end{cases}$$

  Worst case: a and b are consecutive Fibonacci numbers.

1.5 Additional Theory and Formulas

- Prime Number Theorem: Refines the prime counting function:

$$\pi(n) \approx \int_2^n \frac{dt}{\ln(t)}$$

  More accurate for large n, useful for estimating prime density in range queries.
- Miller-Rabin Primality Test: Probabilistic primality test with $O(k \log^3 n)$ time, where k is the number of iterations. Essential for testing large numbers ($n \leq 10^{18}$) in competitive programming.
- Pollard's Rho Algorithm: Randomized factorization with expected $O(n^{1/4})$ time for finding small prime factors, ideal for large composite numbers.
- Linear Sieve: Optimized Sieve of Eratosthenes with $O(n)$ time complexity. For each number i, only mark multiples $i \cdot p_j$ where p_j is the smallest prime factor, reducing redundant operations.
- Segmented Sieve: Generates primes in a range $[L, R]$ in $O((R - L) \log \log R + \sqrt{R} \log \log R)$ time, critical for large ranges ($R \leq 10^{12}$).
- Multiplicative Function Properties: For a multiplicative function f, if $n = p_1^{\alpha_1} \cdots p_k^{\alpha_k}$, then $f(n) = f(p_1^{\alpha_1}) \cdots f(p_k^{\alpha_k})$. Used for computing functions like $\varphi(n)$ or $\sigma(n)$ efficiently.

2 Modular Arithmetic

Each number x is represented by $x \mod m$, the remainder after dividing x by m. For example, if $m = 17$, then $75 \mod 17 = 7$.

2.1 Modular Operations

- Addition:

$$(x + y) \mod m = (x \mod m + y \mod m) \mod m$$

- Subtraction:

$$(x - y) \mod m = (x \mod m - y \mod m) \mod m$$

- Multiplication:

$$(x \cdot y) \mod m = (x \mod m \cdot y \mod m) \mod m$$

- Exponentiation:

$$x^n \mod m = (x \mod m)^n \mod m$$

2.2 Fast Modular Exponentiation

Compute $x^n \mod m$ in $O(\log n)$ time:

$$x^n = \begin{cases} 1 & \text{if } n = 0 \\ x^{n/2} \cdot x^{n/2} & \text{if } n \text{ is even} \\ x^{n-1} \cdot x & \text{if } n \text{ is odd} \end{cases}$$

Applications:

1. Cryptography: RSA, Diffie-Hellman, e.g., $c = m^e \mod n$, $m = c^d \mod n$.
2. Number Theory: Fermat's and Euler's theorems, modular inverse, e.g., $a^{-1} \equiv a^{p-2} \mod p$.
3. Competitive Programming: Fast exponentiation for large n, k in $n^k \mod m$.
4. Hashing: Polynomial rolling hash, e.g., $\text{hash}(s) = \sum_{i=0}^{n-1} s_i \cdot p^i \mod m$.
5. Randomized Algorithms: Managing large probabilistic spaces.

2.3 Euler-Fermat Theorem

If x and m are coprime:

$$x^{\varphi(n)} \mod n = 1$$

Useful for computing modular inverses.

2.4 Additional Theory and Formulas

- Chinese Remainder Theorem: For pairwise coprime moduli $m_1, \ldots, m_k$, and $x \equiv a_i \mod m_i$, there exists a unique $x \mod M$, where $M = m_1 \cdots m_k$. Solves systems of congruences in $O(k \log M)$ time with extended Euclidean algorithm.
- Modular Inverse: For coprime a, m, find a^{-1} such that $a \cdot a^{-1} \equiv 1 \mod m$ in $O(\log m)$ time using extended Euclidean algorithm. For prime m, use Fermat's theorem: $a^{-1} \equiv a^{m-2} \mod m$.
- Lucas' Theorem: For prime p, compute $\binom{n}{k} \mod p$:

  $$\binom{n}{k} \equiv \prod_{i=0}^{r} \binom{n_i}{k_i} \mod p$$

  where n_i, k_i are base-p digits of n, k. Handles large binomial coefficients ($n \leq 10^{18}$).
- Fast Fourier Transform (FFT): Computes polynomial multiplication in $O(n \log n)$ time, used for problems like string matching or convolution in modular arithmetic.
- Modular Arithmetic with Large Numbers: To avoid overflow, use iterative multiplication:

  $$(a \cdot b) \mod m = ((a \mod m) \cdot (b \mod m)) \mod m$$

  For very large a, b, split into parts using bit manipulation.
- Precomputation of Inverses: Precompute modular inverses for 1 to n in $O(n)$ time using:

  $$\text{inv}[i] = -\left\lfloor \frac{m}{i} \right\rfloor \cdot \text{inv}[m \mod i] \mod m$$

  Useful for problems requiring frequent division modulo m.

3 Solving Diophantine Equations

A Diophantine equation is:

$$ax + by = c$$

- Solvable using Euclid's algorithm if c is divisible by $\gcd(a, b)$; otherwise, no solution exists.

- If (x, y) is a solution, all solutions are:

$$\left(x + \frac{kb}{\gcd(a,b)}, y - \frac{ka}{\gcd(a,b)}\right)$$

for any integer k.

3.1 Additional Theory and Formulas

- Extended Euclidean Algorithm: Finds x, y such that $ax + by = \gcd(a, b)$ in $O(\log \min(a, b))$ time. Used for Diophantine equations and modular inverses.
- Linear Congruence: Solve $ax \equiv b \mod m$ by finding $x = b \cdot a^{-1} \mod m$, where a^{-1} is the modular inverse. Solvable if $\gcd(a, m) \mid b$.
- Multi-variable Diophantine Equations: For $a_1x_1 + \cdots + a_nx_n = c$, solutions exist if $\gcd(a_1, \ldots, a_n) \mid c$. Parameterize solutions using the GCD.
- Bézout's Identity: For integers a, b, there exist x, y such that $ax + by = \gcd(a, b)$. Extends to solving systems of linear Diophantine equations.

4 Useful Theorems

- Lagrange's Theorem: Every positive integer is a sum of four squares.
- Zeckendorf's Theorem: Every positive integer has a unique sum of non-consecutive Fibonacci numbers.
- Pythagorean Triples: If (a, b, c) is a Pythagorean triple, then (ka, kb, kc) for $k > 1$ are also triples. Primitive triples are coprime and given by:

$$(n^2 - m^2, 2nm, n^2 + m^2)$$

where $0 < m < n$, n and m are coprime, and at least one is even.

- Wilson's Theorem: n is prime if:

$$(n-1)! \mod n = n - 1$$

4.1 Additional Theory and Formulas

- Fermat's Little Theorem: If p is prime and $a \not\equiv 0 \mod p$:

$$a^{p-1} \equiv 1 \mod p$$

Used in modular arithmetic problems.

- Catalan's Conjecture (Mihăilescu's Theorem): The only solution to $a^x - b^y = 1$ for integers $a, b, x, y > 1$ is $3^2 - 2^3 = 1$.
- Euler's Theorem for Planar Graphs: For a connected planar graph, $V - E + F = 2$, where V is vertices, E is edges, and F is faces. Useful in geometric problems.
- Burnside's Lemma: Counts distinct objects under group actions:

$$\text{Number of orbits} = \frac{1}{|G|} \sum_{g \in G} \text{fix}(g)$$

where $\text{fix}(g)$ is the number of objects fixed by group element g. Common in combinatorial counting problems.

5 Additional Useful Results

- Number of zeros in $n!$:

$$\sum_{k=1}^{\infty} \left\lfloor \frac{n}{5^k} \right\rfloor$$

- Number of diagonals in an n-polygon:

$$\binom{n}{2} - n = \frac{n(n-3)}{2}$$

- Sum of numbers formed by distinct n digits:

$$(111\ldots n) \cdot (n-1)! \cdot \left(\sum \text{digits}\right)$$

- Sum of numbers formed by choosing k distinct integers from n digits:

$$(111\ldots k) \cdot (k-1)! \cdot \left(\sum \text{chosen digits}\right)$$

  for each of the $\binom{n}{k}$ combinations.

- Sum of numbers formed by n digits (including 0):

$$(111\ldots n) \cdot (n-1)! \cdot \left(\sum \text{digits}\right) - (111\ldots(n-1)) \cdot (n-2)! \cdot \left(\sum \text{digits}\right)$$

- Sum of numbers formed by n digits with k repeated:

$$\frac{(111\ldots n) \cdot (n-1)! \cdot (\sum \text{digits})}{k!}$$

5.1 Additional Theory and Formulas

- Stirling's Approximation: For large n:

$$n! \approx \sqrt{2\pi n} \left(\frac{n}{e}\right)^n$$

  Useful for estimating factorials in combinatorial problems with large n ($n \leq 10^6$).

- Partition Function: Number of ways to write n as a sum of positive integers:

$$p(n) \approx \frac{1}{4n\sqrt{3}} \exp\left(\pi\sqrt{\frac{2n}{3}}\right)$$

  Used in partition-related problems.

- Precomputation Technique: Precompute factorials, inverse factorials, and binomial coefficients modulo a prime in $O(n)$ time for $O(1)$ query time. For example:

$$\text{fac}[n] = n \cdot \text{fac}[n-1], \quad \text{invfac}[n] = \text{invfac}[n-1] \cdot \text{inv}[n]$$

- Ternary Search: Finds the maximum/minimum of a unimodal function in $O(\log n)$ time by dividing the search space into three parts. Common in optimization problems.

- Segment Tree with Lazy Propagation: Supports range updates and queries in $O(\log n)$ time, essential for dynamic problems with large test cases ($n \leq 10^6$).

- Fenwick Tree (Binary Indexed Tree): Computes prefix sums and updates in $O(\log n)$ time, more space-efficient than segment trees for cumulative sum problems.

- Matrix Exponentiation: Solves linear recurrence relations in $O(k^3 \log n)$ time, where k is the matrix size. For example, compute the n-th Fibonacci number:

$$\begin{bmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^n$$

- Mo's Algorithm: Optimizes range queries on static arrays with $O(\sqrt{n} \cdot Q + n\sqrt{n})$ time, where Q is the number of queries. Useful for offline processing of massive test cases.