Advanced Number Theory

- Srivaths P

Goal

To learn:

- Factorization using sieve
- Modular Inverse
- Euclidean algorithm

Prime Factorization – Trial Division

Trial division is the most basic method of prime factorization.

We will use the fact that the smallest divisor of any number N is prime, and it will be less than $\sqrt{N}$.

N can be represented as $p_0^{q_0} * p_1^{q_1} * p_2^{q_2} * \cdots * p_k^{q_k}$ where p_i is prime. Let us assume that the prime array is sorted.

We will iterate through the range $\left[2, \sqrt{N}\right]$ to find p_0 first, then p_1, etc.

Prime Factorization – Trial Division Code

```
vector<int> factor(int n) {
    vector<int> facts;
    for (long long d = 2; d * d <= n; d++) {
        while (n % d == 0) {
            facts.push_back(d);
            n /= d;
        }
    }

    if (n > 1)
        facts.push_back(n);

    return facts;
}
```

Sieve of Eratosthenes

Sieve of Eratosthenes can find all the prime numbers upto a given limit in $O(N \ \log\log N)$ time complexity.

Any composite number C must have a prime factor P such that $P < C$.

We will repeatedly find the first number that does not have any prime factors, as that number must be prime. Then we will mark the multiples of the number as composite.

Sieve of Eratosthenes – Code

```
void sieve(int n) {
    bool primes[n+1];
    fill(primes, primes+n+1, true);

    primes[0] = primes[1] = false;
    for (int i = 2; i*i <= n; i++) {
        if (primes[i])
            for (int j = i*i; j <= n; j += i)
                primes[j] = false;
    }
}
```

Sieve of Eratosthenes – Prime Factorization

To use sieve for prime factorization, we make the sieve array store the small prime factor of a number.

To retrieve the values, we can repeatedly divide the current number by the smallest prime factor.

Time complexity: O(log N) per query

Modular Inverse

Modular inverse refers to the reciprocal of a number with some modulo.

If we have the inverse $(A^{-1} \% M)$, then $((A^{-1} \% M) \times A) \% M = 1$.

NOTE: M must be prime, or the result will not be unique.

It can be computed with quite a few methods such as:
- Naïve method
- Extended Euclidian Algorithm
- Fermat's Little Theorem (does not work for non-prime M)

Modular Inverse – Fermat’s Little Theorem

Using Fermat’s Little Theorem, we can take the modular inverse of a value in $O(log_2 M)$ when M is prime.

$$A^{-1} \% M = A^{M-2} \% M$$

```
int inverse(int a, int m) {
    return binary_expo(a, m-2, m);
}
```

Thanks for watching!

https://forms.gle/kXvEfzTkr3eacHhw6