Sieve

usage

- generate all primes from 0 : 1e6
- mostly used in queries problems

main idea

- we initial an array (1e6) and set all elements to true
- we iterate from 2 until the sqrt(n) and if the number has value true so it's prime
- if the number is prime we iterate through the multibles of it starting from the squre (n*n) and set it's value to false
- the remaining elements with true vlaue in the array are the primes

ex

5 is prime -> we mark 5x5 (25) , 5x6 (30) and so on
time complexity --> $n * log(log(n))$

code

```
const int N = 1000005;
bool primes[N];

void sieve_generate_primes() {
    for (int i = 0; i < N; ++i)
        primes[i]=true;
    primes[0] = primes[1] = false;
    for (ll i = 2; i * i < N; i++) {
        if(primes[i]) {
            for (ll j = i * i;  j < N; j += i)
                primes[j] = false;
        }
    }
}
```

linear sieve O(N)

```
int n = 1e6;
    vector<bool> isPrime(n + 1, true);
    vector<int> primes;
    isPrime[0] = isPrime[1] = false;

    for (int i = 2; i <= n; ++i) {
        if (isPrime[i]) {
            primes.push_back(i);
        }
        for (int p : primes) {
            if (i * p > n) break;
            isPrime[i * p] = false;
            if (i % p == 0) break;
        }
    }
```