Compilando Conocimiento

Refence

Competitive Programming

Rosas Hernandez Oscar Andrés

July 2018

Contents

Part I

Number Theory

Chapter 1

Primes

1.1 Sieve of Eratosthenes

1.1.1 Get the Boolean Version

```cpp
// *******   ERATOSTHENES SIEVE / IS PRIME IN O(1)   *******
std::vector<bool> EratosthenesSieveIsPrime(ull n) {             //To check if i is prime: Vector[i]
    std::vector<bool> isPrime(n + 1, true);                     //Ok, first, allocate space
    isPrime[0] = isPrime[1] = false;                            //Now, 0 & 1(maybe) are not prime

    for (ull i = 4; i <= n; i += 2) isPrime[i] = false;         //Eliminate all the evens numbers

    for (ull i = 3, limit = std::sqrt(n); i <= limit; i += 2)   //For every odd number < n
        if (isPrime[i])                                         //If we found a prime :0
            for (ull j = i * i; j <= n; j += 2 * i)             //ForEach multiple we have'nt check
                isPrime[j] = false;                             //Each multiple is not prime

    return isPrime;                                             //Return the complete sieve
}
```

1.1.2 Get the Vector of Primes

```cpp
// *******   ERATOSTHENES SIEVE / VECTOR OF PRIMES   *******
std::vector<ull> EratosthenesSievePrimes(ull n) {               //Return a vector of only primes
    std::vector<bool> isPrime(n + 1, true);                     //Create the origianl Sieve
    std::vector<ull> Primes(1, 2);                              //2 is a prime, dahhhhh!

    isPrime[0] = isPrime[1] = false;                            //Now, 0 & 1(maybe) are not prime

    for (ull i = 4; i <= n; i += 2) isPrime[i] = false;         //Eliminate all the evens numbers

    for (ull i = 3, limit = std::sqrt(n); i <= n; i += 2) {     //Check for every odd number
        if (isPrime[i]) {                                       //If we still believe it's a prime
            Primes.push_back(i);                                //Add it to the vector, it's a prime

            if (i <= limit)                                     //It make sense to delete multiples?
                for (ull j = i * i; j <= n; j += 2 * i)         //ForEach multiple we have'nt check
                    isPrime[j] = false;                         //Each multiple is not prime
        }
    }

    return Primes;                                              //Return the vector of only primes
}
```