Team Notebook

ntTas

April 17, 2019

Contents

1 Data Structure

2 Geometry

3 Graph

4 Math

4.1 Number Theory

4.1.1 InverseModulo

```
/**
* Description : find x such that ax = 1 mod m
*/

/* case 1 : when(gcd(a,m) = 1) */
/* use extended euclid : find x such that ax + my = 1 */

/* store x, y, and d as global variables */
void extendedEuclid(int a, int b) {
  if (b == 0) { x = 1; y = 0; d = a; return; }
  /* base case */
  extendedEuclid(b, a % b);
  /* similar as the original gcd */
  int x1 = y;
  int y1 = x - (a / b) * y;
  x = x1;
  y = y1;
}

/* compute the first case inverse modulo*/
int firstInverseModulo(int a, int m){
  /* produces x and y, such that ax + my = 1 */
  extendedEuclid(a, m);
  return (x + m)%m;
}

/* case 2 : m is prime */
/* a^(m-1) = 1 mod m */
/* a^(m-2) = a^-1 mod m */

int power(int a,int b){
  int res = 1;
  while (b > 0){
    if (b%2 == 1)
      res *= a;
    b /= 2;
    a *= a;
  }
  return res;
}

int secondInverseModulo(int a,int m){
  return power(a, m-2);
}
```

4.1.2 PrimeFactor

```
/**
* Description : some function that have relation with prime factor
*/

/* find prime factor */
vector<long long> primefactor(long long N){
    vector<long long> factors;
    long long idx = 0;
    long long PF = primes[idx];
    while (PF <= (long long)sqrt(N)){
      while (N%PF == 0){
          N /= PF;
          factors.push_back(PF);
      }
      PF = primes[++idx];
    }
    if (N != 1) factors.push_back(N);
    return factors;
}

/* number of divisor */
long long numDiv(long long N){
    long long ans = 1;
    long long idx = 0;
    long long PF = primes[idx];
    while (PF <= (long long)sqrt(N)){
        long long power = 0;
        while (N%PF == 0){
            power++;
            N /= PF;
        }
        ans *= (power + 1);
        PF = primes[++idx];
    }
    if (N != 1) ans *= 2;
    return ans;
}

/* sum of divisor */
long long sumDiv(long long N){
    long long ans = 1;
    long long idx = 0;
    long long PF = primes[idx];
    while (PF <= (long long)sqrt(N)){
        long long power = 0;
        while (N%PF == 0){
            power++;
            N /= PF;
        }
        /* 1 + PF + PF^2 + PF^3 + ... + PF^pow = (a.r^n - 1) / (r-1) */
        ans *= ((long long)pow((double)PF, power + 1.0) - 1) / (PF - 1);
        PF = primes[++idx];
    }
    if (N != 1) ans *= ((long long)pow((double)N, 2.0) - 1) / (N - 1);
    return ans;
}

/* Euler Phi */
long long eulerPhi(long long N){
    long long idx = 0;
    long long PF = primes[idx];
    long long ans = N;
    while (PF <= (long long)sqrt(N)){
        if (N%PF == 0) ans -= ans / PF;
        while (N%PF == 0) N /= PF;
        PF = primes[++idx];
    }
    if (N != 1) ans -= ans / N;
    return ans;
}
```

4.1.3 Sieve

```
/**
* Description :Test Primality up to n in O(nlog(logn))
*/

const int SZ = 1e7;
bitset<SZ> bs;
vector<long long> primes;

void sieve(){
    bs.set();
    bs[0] = false; bs[1] = false;
    for (long long i = 2; i <= SZ; i++){
        if (bs[i]){
            primes.push_back(i);
            for (long long j = i * i; j <= SZ; j+=i)
                bs[j] = false;
        }
    }
}
```

4.1.4 extendedEuclid

```
/**
* Description : find x and y such that ax + by = 1
*/

/* store x, y, and d as global variables */
void extendedEuclid(int a, int b) {
  if (b == 0) { x = 1; y = 0; d = a; return; }
  /* base case */
  extendedEuclid(b, a % b);
  /* similar as the original gcd */
  int x1 = y;
  int y1 = x - (a / b) * y;
  x = x1;
  y = y1;
}
```

5 Setup

5.1 C++Template

```
#pragma GCC optimize ("O3")
#pragma GCC target ("sse4")

#include <bits/stdc++.h>
using namespace std;

#define fi first
#define se second
#define pb push_back

typedef long long LL;
typedef vector<int> vi;
typedef pair<int,int> ii;

const int MOD = 1e9 + 7;
const LL INF = 1e18;

void fastscan(int &number) {
    //variable to indicate sign of input number
    bool negative = false;
    register int c;

    number = 0;

    // extract current character from buffer
    c = getchar();
    if (c=='-')
    {
        // number is negative
        negative = true;

        // extract the next character from the buffer
        c = getchar();
    }

    // Keep on extracting characters if they are integers
    // i.e ASCII Value lies from '0'(48) to '9' (57)
    for (; (c>47 && c<58); c=getchar())
        number = number *10 + c - 48;

    // if scanned input has a negative sign, negate the
    // value of the input number
    if (negative)
        number *= -1;
}

int main(){
    //cin / cout user
    //ios_base::sync_with_stdio(0); cin.tie(0); cout.tie(0)


    return 0;
}
```

5.2 FastScanner

```
class FastScanner {
    private InputStream stream;
    private byte[] buf = new byte[1024];
    private int curChar;
    private int numChars;

public FastScanner(InputStream stream) {
    this.stream = stream;
}

int read() {
    if (numChars == -1)
        throw new InputMismatchException();
    if (curChar >= numChars) {
        curChar = 0;
        try {
            numChars = stream.read(buf);
        } catch (IOException e) {
            throw new InputMismatchException();
        }
        if (numChars <= 0) return -1;
    }
    return buf[curChar++];
}

boolean isSpaceChar(int c) {
    return c ==       || c ==   \ n  || c ==   \ r  || c
        ==   \ t  || c == -1;
}

public int nextInt() {
    return Integer.parseInt(next());
}

public long nextLong() {
    return Long.parseLong(next());
}

public double nextDouble() {
    return Double.parseDouble(next());
}

public String next() {
    int c = read();
    while (isSpaceChar(c)) c = read();
    StringBuilder res = new StringBuilder();
    do {
        res.appendCodePoint(c);
        c = read();
    } while (!isSpaceChar(c));
    return res.toString();
}

public String nextLine() {
    int c = read();
    while (isEndline(c))
    c = read();
StringBuilder res = new StringBuilder();
do {
    res.appendCodePoint(c);
    c = read();
```

```
        } while (!isEndline(c));
        return res.toString();
    }
}
```

6 String