Competitive Programming Reference

Enya Quetzalli Gómez Rodríguez

eithnegomez@hotmail.com
github.com/equetzal

Contents

Data Structures

STL

STL Queue

```cpp
#include <queue>

queue<int> q,p;
q.push(int(111)); //Receives an object copy
q.emplace(111); //Uses the constructor of the object
int val = q.front(); //Acces head element
int val = q.back(); //Access tail element
int sz = q.size();
q.pop();
q.empty();
q.swap(p); //Swap queue contents in O(1)
```

STL Stack

```cpp
#include <stack>

stack<int> s,p;
s.push(int(111)); //Receives an object copy
s.emplace(111); //Uses the constructor of the object
int val = s.top();
int sz = s.size();
s.pop();
s.empty();
s.swap(p); //Swap stack contents in O(1)
```

Math

Number Theory

Greatest Common Divisor

```cpp
lli gcd(lli a, lli b){
    lli r;
    while(b != 0) r = a % b, a = b, b = r;
    return a;
}
lli gcd(vector<lli> & nums){
    lli ans = 0;
    for(lli & num : nums) ans = gcd(ans, num);
    return ans;
}
lli extendedGcd(lli a, lli b, lli & s, lli & t){
    lli q, r0 = a, r1 = b, ri, s0 = 1, s1 = 0, si, t0 = 0, t1 = 1, ti;
    while(r1){
        q = r0 / r1;
        ri = r0 % r1, r0 = r1, r1 = ri;
        si = s0 - s1 * q, s0 = s1, s1 = si;
        ti = t0 - t1 * q, t0 = t1, t1 = ti;
    }
    s = s0, t = t0;
    return r0;
}
```

Lowest Common Multiple

```cpp
lli lcm(lli a, lli b){
    return b * (a / gcd(a, b));
}
lli lcm(vector<lli> & nums){
    lli ans = 1;
    for(lli & num : nums) ans = lcm(ans, num);
    return ans;
}
```

Modular Inverse

```
lli modularInverse(lli a, lli m){
    lli r0 = a, r1 = m, ri, s0 = 1, s1 = 0, si;
    while(r1){
        si = s0 - s1 * (r0 / r1), s0 = s1, s1 = si;
        ri = r0 % r1, r0 = r1, r1 = ri;
    }
    if(r0 < 0) s0 *= -1;
    if(s0 < 0) s0 += m;
    return s0;
}
```

Multiplication

```
lli multMod(lli a, lli b, lli n){
    lli ans = 0;
    a %= n, b %= n;
    if(abs(b) > abs(a)) swap(a, b);
    if(b < 0){
        a *= -1, b *= -1;
    }
    while(b){
        if(b & 1) ans = (ans + a) % n;
        b >>= 1;
        a = (a + a) % n;
    }
    return ans;
}
```

Number Base

```
string decimalToBaseB(lli n, lli b){
    string ans = "";
    lli d;
    do{
        d = n % b;
        if(0 <= d && d <= 9) ans = (char)(48 + d) + ans;
        else if(10 <= d && d <= 35) ans = (char)(55 + d) + ans;
        n /= b;
    }while(n != 0);
    return ans;
}
```

```
lli baseBtoDecimal(const string & n, lli b){
    lli ans = 0;
    for(const char & d : n){
        if(48 <= d && d <= 57) ans = ans * b + (d - 48);
        else if(65 <= d && d <= 90) ans = ans * b + (d - 55);
        else if(97 <= d && d <= 122) ans = ans * b + (d - 87);
    }
    return ans;
}
```

Power

```
lli power(lli b, lli e){
    lli ans = 1;
    while(e){
        if(e & 1) ans *= b;
        e >>= 1;
        b *= b;
    }
    return ans;
}

lli powerMod(lli b, lli e, lli m){
    lli ans = 1;
    b %= m;
    if(e < 0){
        b = modularInverse(b, m);
        e *= -1;
    }
    while(e){
        if(e & 1) ans = (ans * b) % m;
        e >>= 1;
        b = (b * b) % m;
    }
    return ans;
}
```

Random Number

```
mt19937_64 rng(chrono::steady_clock::now().time_since_epoch().count());
lli aleatorio(lli a, lli b){
    std::uniform_int_distribution<lli> dist(a, b);
    return dist(rng);
}
```

Roman Numbers

```
string decimalToRoman(int n){
    int d, b = 0;
    string ans = "";
    vector<vector<char>> datos = {{'I', 'V'}, {'X', 'L'}, {'C', 'D'},
    ↪  {'M', '\0'}};
    int miles = n / 1000;
    do{
        string tmp = "";
        d = n % 10;
        n /= 10;
        if(b < 3){
            if(0 <= d && d <= 3){
                tmp.append(d, datos[b][0]);
            }else if(d == 4){
                tmp += datos[b][0];
                tmp += datos[b][1];
            }else if(5 <= d && d <= 8){
                tmp += datos[b][1];
                tmp.append(d - 5, datos[b][0]);
            }else if(d == 9){
                tmp += datos[b][0];
                tmp += datos[b + 1][0];
            }
        }else{
            tmp.append(miles, 'M');
            ans = tmp + ans;
            break;
        }
        ans = tmp + ans;
        b++;
    }while(n != 0);
    return ans;
}
```

```
int romanToDecimal(string n){
    int ans = 0;
    char curr, prev;
    bool f = false;
    map<char, int> datos = {{'I', 1}, {'V', 5}, {'X', 10}, {'L', 50},
    ↪  {'C', 100}, {'D', 500}, {'M', 1000}};
    for(int i = n.size() - 1; i >= 0; i--){
        curr = n[i];
        if(i > 0) prev = n[i - 1];
        if(curr == 'V' && prev == 'I') ans += 4, f = true;
        else if(curr == 'X' && prev == 'I') ans += 9, f = true;
        else if(curr == 'L' && prev == 'X') ans += 40, f = true;
        else if(curr == 'C' && prev == 'X') ans += 90, f = true;
        else if(curr == 'D' && prev == 'C') ans += 400, f = true;
        else if(curr == 'M' && prev == 'C') ans += 900, f = true;
        else{
            if(!f) ans += datos[curr];
            f = false;
        }
    }
    return ans;
}
```

Rounding

```
lli piso(lli a, lli b){
    if((a >= 0 && b > 0) || (a < 0 && b < 0)){
        return a / b;
    }else{
        if(a % b == 0) return a / b;
        else return a / b - 1;
    }
}

lli techo(lli a, lli b){
    if((a >= 0 && b > 0) || (a < 0 && b < 0)){
        if(a % b == 0) return a / b;
        else return a / b + 1;
    }else{
        return a / b;
    }
}
```

Extras

Definitions

```
#if defined(_USE_MATH_DEFINES) && !defined(_MATH_DEFINES_DEFINED)
#define _MATH_DEFINES_DEFINED
    // e
    #define M_E        2.71828182845904523536
    // log2(e)
    #define M_LOG2E        1.44269504088896340736
    // log10(e)
    #define M_LOG10E    0.43429448190325182765
    // ln(2)
    #define M_LN2        0.69314718055994530942
    // ln(10)
    #define M_LN10        2.30258509299404568402
    // pi
    #define M_PI        3.14159265358979323846
    // pi/2
    #define M_PI_2        1.57079632679489661923
    // pi/4
    #define M_PI_4        0.78539816339744830962
    // 1/pi
    #define M_1_PI        0.31830988618379067154
    // 2/pi
    #define M_2_PI        0.63661977236758134308
    // 2/sqrt(pi)
    #define M_2_SQRTPI    1.12837916709551257390
    // sqrt(2)
    #define M_SQRT2        1.41421356237309504880
    // 1/sqrt(2)
    #define M_SQRT1_2    0.70710678118654752440
#endif
```

Read Data From Files

```
freopen("input.txt", "r", stdin);
freopen("output.txt", "w", stdout);
```

Template

```
#include <bits/stdc++.h>

#define endl "\n"
#define fast_io ios_base::sync_with_stdio(false);cin.tie(NULL);

using namespace std;

typedef long long int lli;

int main(){


    return 0;
}
```

Competitive Programming Reference

Created by: Enya Quetzalli Gómez Rodríguez
Created with: mkcpr reference
Created on: March 30, 2020
Last Update: May 17, 2020

I met the competitive programming at my university "The Superior School of Computer Sciences of the National Polytechnic Institute", thanks to a club within the school called "algorithmic club", where I met ICPC and loved competitive programming, this group of people at I belong has offered me everything I know now, we always pass all our knowledge to the following generations, and we all contribute to our community to achieve more and more. I will always be grateful to this group of people who changed my life

Special thanks: