Competitive Programming Reference

Enya Quetzalli Gómez Rodríguez

eithnegomez@hotmail.com
github.com/equetzal

Contents

Data Structures

STL

STL Map

```cpp
#include <map>

map<int,string> m,p;
map<pair<int,int>,string> mpii;
m[111] = "val"; //Insert, or replace element in map
auto val = m[111]; //Return element with key 111, if no key create key with default val
m.erase(111); //Erase element with key 111
m.merge(p); //Insert or replace p elements into s in O(|S|log(|P|+|S|))
int val = s.count(111); //Return number of elements in map with key 111
set<map>::iterator it = s.find(111); //Return m.end() if not found, or map iterator if found
auto it = m.lower_bound(111); //Return first not-less element than key in O(logN)
auto it = m.upper_bound(111); //Return first element greater than key in O(logN)
s.swap(p); //Swap map contents in O(1)
```

STL Multi Map

```cpp
#include <multimap>

multimap<string,string> m;
m.emplace("key","val"); //Insert element in multimap
m.erase(it); //Erase element at iterator = it
m.erase(itBegin, itEnd); //Erase elements in range from itBegin to itEnd
m.erase("key"); //Erase all elements with key
m.clear(); //Erase all elements in multimap
int val = m.count("key"); //Return number of elements with key

//To get elements with same key
auto range = m.equal_range("key"); //Return a pair of begin-end iterators which holds elements equal to key
cout << "for key[key] -> ";
for(auto it=range.first; it!=range.end; it++)
    cout << (*it).second << " ";
cout << endl;
```

STL Multi Set

```cpp
#include <multiset>

//multiset is the same as set, but s.count() can be more than 1
set<int> s,p;
set<pair<int,int>> spii;
s.insert(int(111)); //Insert value in vector
spii.insert(make_pair(11,22));
s.emplace(111); //Insert but with constructor
spii.emplace(11,22);
s.erase(111); //Erase element with value 111
s.merge(p); //Insert p elements into s in O(|S|log(|P|+|S|))
int val = s.count(111); //Return number insertions-erations in set with value 111
set<int>::iterator it = s.find(111); //Return s.end() if not found, or set iterator if found
auto it = s.lower_bound(111); //Return first not-less element than arg in O(logN)
auto it = s.upper_bound(111); //Return first element greater than arg in O(logN)
s.swap(p); //Swap set contents in O(1)
```

STL Queue

```cpp
#include <queue>

queue<int> q,p;
queue<pair<int,int>> qpii;
q.push(int(111)); //Receives an object copy
qpii.push(make_pair(11,22);
q.emplace(111); //Uses the constructor of the object
qpii.emplace(11,22);
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)
q.clear(); //Erase elements of queue
```

STL Set

```cpp
#include <set>

set<int> s,p;
set<pair<int,int>> spii;
s.insert(int(111)); //Insert value in vector
spii.insert(make_pair(11,22));
s.emplace(111); //Insert but with constructor
spii.emplace(11,22);
s.erase(111); //Erase element with value 111
s.merge(p); //Insert p elements into s in O(|S|log(|P|+|S|))
int val = s.count(111); //Return number of elements in set with value
↪ 111
set<int>::iterator it = s.find(111); //Return s.end() if not found, or
↪ set iterator if found
auto it = s.lower_bound(111); //Return first not-less element than arg
↪ in O(logN)
auto it = s.upper_bound(111); //Return first element greater than arg
↪ in O(logN)
s.swap(p); //Swap set contents in O(1)
```

STL Stack

```cpp
#include <stack>

stack<int> s,p;
stack<pair<int,int>> spii;
s.push(int(111)); //Receives an object copy
spii.push(make_pair(11,22));
s.emplace(111); //Uses the constructor of the object
spii.emplace(11,22);
int val = s.top(); //Return value of element at tio
int val = s.size(); //Return number of elements in stack
s.pop(); //Erase the element at the top of stack
bool val = s.empty(); //Return true if stack is empty
s.swap(p); //Swap stack contents in O(1)
s.clear() //Erase elements of stack
```

STL Vector

```cpp
#include <vector>

vector<int> v,p;
vector<pair<int,int>> vpii;
v.push_back(int(111)); //Insert value at end of vector
v.pop_back(); //Erase the element at end of vector
int val = v.back(); //Return value at end of vector
int val = v.front(); //Return value at begin of vector
v.insert(v.begin()+pos, int(111); //Insert element before pos in O(pos)
vpii.insert(v.begin()+pos, make_pair(11,22));
v.emplace(v.begin()+pos, 111); //Insert, but uses constructor after
↪ pos arg
vpii.emplace(v.begin()+pos, 11,22);
v.erase(v.begin()+pos); //Erase element at pos in O(n)
v.clear(); //Clear the contents of a vector O(n)
int val = v.size(); //Return size of vector -> cast size_t to int
bool val = v.emtpy(); //Return true if vector is empty
v.resize(5); //If arg is smaller than size, it cuts the vector, if
↪ not, expand array with default init
v.resize(5,0); //If arg si larger than size, init values with 2nd arg
v.swap(p); //Swap vector contents in O(1)
```

Unordered Map Key=Pair

```
struct pairhash { template <class T1, class T2>
    size_t operator()(const pair<T1, T2> &p) const {
        return hash<T1>{}(p.first) ^ (hash<T2>{}(p.second) << 32);
    }
};

unordered_map<pair<int, int>, int, hash_pair> umap;
```

Math

Number Theory

Greatest Common Divisor

```
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

```
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        2.7182818284590452536
    // log2(e)
    #define M_LOG2E         1.4426950408889634073 6
    // log10(e)
    #define M_LOG10E    0.43429448190325182765 1
    // ln(2)
    #define M_LN2        0.69314718055994530941 7
    // 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        0.78539816339744830961 6
    // 1/pi
    #define M_1_PI         0.31830988618379067153 8
    // 2/pi
    #define M_2_PI        0.63661977236758134307 6
    // 2/sqrt(pi)
    #define M_2_SQRTPI    1.12837916709551257390
    // sqrt(2)
    #define M_SQRT2        1.41421356237309504880 6
    // 1/sqrt(2)
    #define M_SQRT1_2    0.70710678118654752440 1
#endif
```

Output Hacks

```
double pi = 3.14159265359;
cout << pi << endl;
// "3.14159"
cout << setprecision(7) << pi << endl;
// "3.141593"
cout << fixed << setprecision(7) << pi << endl;
// "3.1415927"

int v = 220;
cout << v << endl;
// "220"
cout << setw(6) << v << endl;
// "   220"
cout << setw(6) << setfill('0') << v << endl;
// "000220"
cout << setw(6) << setfill('0') << left << v << endl;
// "220000"

cout << hex << v << endl;
// "dc" (hex lowercase print)
cout << uppercase << v << endl;
// "DC" (HEX uppercase print)
cout << oct << v << endl;
// "334" (Octal print)
cout << bitset<8>(v) << endl;
// "11011100" (N-bit Binary Unsigned Int print)
```

Read Data From Files

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

Template

```cpp
#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: June 15, 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: