Contents

1 Contest Setup

1.1 vimrc

```
set number      " Show line numbers
set mouse=a     " Enable inaction via mouse
set showmatch       " Highlight matching brace
set cursorline      " Show underline
set cursorcolumn    " highlight vertical column

filetype on "enable file detection
syntax on   "syntax highlight

set autoindent      " Auto-indent new lines
set shiftwidth=4    " Number of auto-indent spaces
set smartindent     " Enable smart-indent
set smarttab        " Enable smart-tabs
set softtabstop=4   " Number of spaces per Tab

" ---------Optional----------

set undolevels=10000    " Number of undo levels
set scrolloff=5     " Auto scroll

set hlsearch    " Highlight all search results
set smartcase   " Enable smart-case search
set ignorecase  " Always case-insensitive
set incsearch   " Searches for strings incrementally

highlight Comment ctermfg=cyan
set showmode

set encoding=utf-8
set fileencoding=utf-8
scriptencoding=utf-8
```

1.2 bashrc

```
alias g++="g++ -Wall -Wextra -std=c++11 -O2"
```

1.3 C++ template

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

using namespace std;

#define x first
#define y second

typedef long long int ll;
typedef pair<int, int> ii;

int main()
{
    return 0;
}
```

1.4 Java template

```java
import java.io.*;
import java.util.*;

public class Main
{
    public static void main(String[] args)
    {
        MyScanner sc = new MyScanner();
        out = new PrintWriter(new BufferedOutputStream(System.out));
        // Start writing your solution here.

        // Stop writing your solution here.
        out.close();
    }

    public static PrintWriter out;

    public static class MyScanner
    {
        BufferedReader br;
        StringTokenizer st;

        public MyScanner()
        {
            br = new BufferedReader(new InputStreamReader(System.in));
        }

        boolean hasNext()
        {
            while (st == null || !st.hasMoreElements()) {
                try {
                    st = new StringTokenizer(br.readLine());
                } catch (Exception e) {
                    return false;
                }
            }
            return true;
        }

        String next()
        {
            if (hasNext())
                return st.nextToken();
            return null;
        }

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

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

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

        String nextLine()
        {
            String str = "";
            try {
                str = br.readLine();
            } catch (IOException e) {
                e.printStackTrace();
            }
            return str;
        }
    }
}
```

2 Reminder

1. Read the problem statements carefully. Input and output specifications and constraints are crucial!
2. Estimate the **time complexity** and **memory complexity** carefully.
3. Time penalty is 20 minutes per WA, **don't rush**!
4. Sample test cases must all be tested and passed before every submission!
5. Test the corner cases, such as 0, 1, -1. Test all edge cases of the input specification.

3 Useful code

3.1 Leap year

```
year % 400 == 0 || (year % 4 == 0 && year % 100 != 0)
```

3.2 Fast Exponentiation $O(log(exp))$

```cpp
ll fast_pow(ll base, ll exp, ll mod)
{
    if (exp == 0)
        return 1LL;
    ll res = 1;
    while (exp > 0) {
        if (exp & 1) {
            res = ((res % mod) * (base % mod)) % mod;
        }
        exp >>= 1;
        base = (base * base) % mod;
    }
    return res;
}
```

3.3 GCD $O(log(a+b))$

注意負數的 case!

```
ll gcd(ll a, ll b)
{
    return b == 0 ? a : gcd(b, a % b);
}
```

3.4 Extended Euclidean Algorithm

Bezout identity $ax + by = gcd(a, b)$, where $gcd(a, b)$ is the smallest positive integer that can be written as ax + by, and every integer of the form ax + by is a multiple of $gcd(a, b)$.

```
ll ext_gcd(ll a, ll b, ll &x, ll &y)
{
    if (a == 0) {
        x = 0;
        y = 1;
        return b;
    }

    ll x1, y1;
    ll gcd = ext_gcd(b % a, a, x1, y1);

    x = y1 - (b / a) * x1;
    y = x1;

    return gcd;
}
```

3.5 Mod Inverse

Case 1 $gcd(a, m) = 1$: ax + my = gcd(a, m) = 1 (use ext_gcd)

Case 2 $m\ is\ prime$: $a^{m-2} \equiv a^{-1} mod\ m$ (use Fermat's little theorem)

3.6 Prime Generator

```
bool is_prime[N];
vector<ll> primes;
void init()
{
    fill(is_prime, is_prime + N, true);
    for (int i = 2; i < N; i++) {
        if (is_prime[i] == true) {
            primes.push_back(i);
            for (int j = i * i; j < N; j += i)
                is_prime[j] = false;
        }
    }
}
```

3.7 Binomial Coefficient

```
int binomialCoeff(int n, int k)
{
    int res = 1;

    if ( k > n - k ) // Since C(n, k) = C(n, n-k)
        k = n - k;

    for (int i = 0; i < k; ++i) // n...n-k / 1...k
    {
        res *= (n - i);
        res /= (i + 1);
    }

    return res;
}
```

3.8 STL quick reference

3.8.1 Map

```
map<T1, T2> m; // iterable
void clear();
void erase(T1 key);
it find(T1 key); // <key, val>
void insert(pair<T1, T2> P);
T2 &[](T1 key); // if key not in map, new key will be inserted with
    default val
it lower_bound(T1 key); // = m.end() if not found, *it = <key, val>
it upper_bound(T1 key); // = m.end() if not found, *it = <key, val>
```

3.8.2 Set

```
set<T> s; // iterable
void clear();
size_t count(T val); // number of val in set
void erase(T val);
it find(T val); // = s.end() if not found
void insert(T val);
it lower_bound(T val); // = s.end() if not found, *it = <key, val>
it upper_bound(T val); // = s.end() if not found, *it = <key, val>
```

3.8.3 Algorithm

```
// return if i is smaller than j
comp = [&](const T &i, const T &j) -> bool;
vector<T> v;
bool any_of(v.begin(), v.end(), [&](const T &i) -> bool);
bool all_of(v.begin(), v.end(), [&](const T &i) -> bool);
void copy(inp.begin(), in.end(), out.begin());
int count(v.begin(), v.end(), int val); // number of val in v
it unique(v.begin(), v.end());          // it - v.begin() = size
// after calling, v[nth] will be n-th smallest elem in v
void nth_element(v.begin(), nth_it, bin_comp);
void merge(in1.begin(), in1.end(), in2.begin(), in2.end(), out.begin(),
     comp);
// include union, intersection, difference, symmetric_difference(xor)
void set_union(in1.begin(), in1.end(), in2.begin(), in2.end(), out.
    begin(), comp);
bool next_permutation(v.begin(), v.end());
// v1, v2 need sorted already, whether v1 includes v2
bool inclues(v1.begin(), v1.end(), v2.begin(), v2.end());
it find(v.begin(), v.end(), T val); // = v.end() if not found
it search(v1.begin(), v1.end(), v2.begin(), v2.end());
it lower_bound(v.begin(), v.end(), T val);
it upper_bound(v.begin(), v.end(), T val);
bool binary_search(v.begin(), v.end(), T val); // exist in v ?
void sort(v.begin(), v.end(), comp);
void stable_sort(v.begin(), v.end(), comp);
```

3.8.4 String

4 Search

4.1 Binary Search

4.1.1 Find key

4.1.2 Upper / lower Bound

4.2 折半完全列舉

4.3 Two-pointer 爬行法

5 Basic data structure

5.1 1D BIT

5.2 2D BIT

5.3 Union Find

5.4 Segment Tree

6 Dynamic Programming

7 Tree

7.1 LCA

8 Graph

8.1 Articulation point / edge

8.2 CC

8.2.1 BCC vertex

8.2.2 BCC edge

8.2.3 SCC

8.3 Shortest Path

8.3.1 Dijkatra

8.3.2 SPFA

8.3.3 Bellman-Ford

8.3.4 Floyd-Warshall

8.4 Kruskal MST

8.5 Flow

8.5.1 Max Flow (Dinic)

8.5.2 Min-Cut

8.5.3 Min-Cost Max-Fl