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 -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 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 Fast Exponentiation

```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;
}
```

3.2 GCD

注意負數的 case!

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


2

3.3 Leap year

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

3.4 Extended Euclidean Algorithm

3.5 STL quick reference

3.5.1 Map / Set

3.5.2 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

Hehe

6 Dynamic Programming

7 Tree

7.1 LCA

8 Graph

8.1 Articulation point / edge

8.2 BCC vertex

8.3 BCC edge

8.4 SCC

8.5 Shortest Path

8.5.1 Dijkatra

8.5.2 SPFA

8.5.3 Bellman-Ford

8.6 Flow

8.6.1 Max Flow (Dinic)

8.6.2 Min-Cut

8.6.3 Min Cost Max Flow

8.6.4 Maximum Bipartite Graph

9 String

9.1 KMP

9.2 Z Algorithm

9.3 Trie

9.4 Suffix Array

10 Geometry

10.1 Template

10.1.1 Point / Line

10.1.2 Intersection

10.2 Half-plane intersection

10.3 Convex Hull