Contents

1 Todo

1. Add code and complexity
2. Add brief explanations

2 Contest Setup

2.1 vimrc

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

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 on the bottom 5
     lines

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
```

contest_setup/vimrc

2.2 bashrc

```
alias g++="g++ -Wall -Wextra -O2"
```

contest_setup/bashrc

2.3 C++ template

```cpp
#include <bits/stdc++.h>
//LLONG_MIN LLONG_MAX INT_MIN INT_MAX

#ifdef _WIN32
#define lld "I64d"
#else
#define lld "lld"
#endif

using namespace std;

#define x first
#define y second

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

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

contest_setup/main.cpp

2.4 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;
        }
    }
}
```

contest_setup/Main.java

3 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.

4 Useful code

4.1 Fast Exponentiation

4.2 GCD

小心負數!

4.3 Extended Euclidean Algorithm

4.4 STL quick reference

4.4.1 Map / Set

4.4.2 String

5 Search

5.1 Binary Search

5.1.1 Find key

5.1.2 Upper / lower Bound

5.2 折半完全列舉

5.3 Two-pointer 爬行法

6 Basic data structure

6.1 1D BIT

6.2 2D BIT

6.3 Union Find

6.4 Segment Tree

Hehe

7 Dynamic Programming

8 Tree

8.1 LCA

9 Graph

9.1 Articulation point / edge

9.2 BCC vertex

9.3 BCC edge

9.4 SCC

9.5 Shortest Path

9.5.1 Dijkatra

9.5.2 SPFA

9.5.3 Bellman-Ford

9.6 Flow

9.6.1 Max Flow (Dinic)

9.6.2 Min-Cut

9.6.3 Min Cost Max Flow

9.6.4 Maximum Bipartite Graph

10 String

10.1 KMP

10.2 Z Algorithm

10.3 Trie

10.4 Suffix Array

11 Geometry

11.1 Template

11.1.1 Point / Line

11.1.2 Intersection

11.2 Half-plane intersection

11.3 Convex Hull