

Contents

1	Contest Setup	1
1.1	vimrc	1
1.2	bashrc	1
1.3	C++ template	1
1.4	Java template	2
2	Reminder	2
3	Useful code	2
3.1	Leap year	2
3.2	Fast Exponentiation $O(\log(exp))$	2
3.3	GCD $O(\log(a+b))$	3
3.4	Extended Euclidean Algorithm	3
3.5	Mod Inverse	3
3.6	Prime Generator	3
3.7	Binomial Coefficient	3
3.8	STL quick reference	3
3.8.1	Map	3
3.8.2	Set	3
3.8.3	Algorithm	3
3.8.4	String	4
4	Search	4
4.1	Binary Search	4
4.1.1	Find key	4
4.1.2	Upper / lower Bound	4
4.2	折半完全列舉	4
4.3	Two-pointer 爬行法	4
5	Basic data structure	4
5.1	1D BIT	4
5.2	2D BIT	5
5.3	Union Find	5
5.4	Segment Tree	5
6	Dynamic Programming	5
7	Tree	5
7.1	LCA	5
8	Graph	5
8.1	Articulation point / edge	5
8.2	CC	5
8.2.1	BCC vertex	5
8.2.2	BCC edge	5
8.2.3	SCC	5
8.3	Shortest Path	5
8.3.1	Dijkstra	5
8.3.2	SPFA	5
8.3.3	Bellman-Ford	5
8.3.4	Floyd-Warshall	5
8.4	Kruskal MST	5
8.5	Flow	5
8.5.1	Max Flow (Dinic)	5
8.5.2	Min-Cut	5
8.5.3	Min Cost Max Flow	5
8.5.4	Maximum Bipartite Graph	5
9	String	5
9.1	KMP	5
9.2	Z Algorithm	5
9.3	Trie	5
9.4	Suffix Array	5
10	Geometry	5
10.1	Template	6
10.1.1	Point / Line	6
10.1.2	Intersection	6
10.2	Half-plane intersection	6
10.3	Convex Hull	6

1 Contest Setup

1.1 vimrc

```

1 set number      " Show line numbers
2 set mouse=a     " Enable inaction via mouse
3 set showmatch   " Highlight matching brace
4 set cursorline  " Show underline
5 set cursorcolumn " highlight vertical column
6
7 filetype on "enable file detection
8 syntax on   "syntax highlight
9
10 set autoindent " Auto-indent new lines
11 set shiftwidth=4 " Number of auto-indent spaces
12 set smartindent " Enable smart-indent
13 set smarttab    " Enable smart-tabs
14 set softtabstop=4 " Number of spaces per Tab
15
16 " -----Optional-----
17
18 set undolevels=10000 " Number of undo levels
19 set scrolloff=5     " Auto scroll
20
21 set hlsearch " Highlight all search results
22 set smartcase " Enable smart-case search
23 set ignorecase " Always case-insensitive
24 set incsearch " Searches for strings incrementally
25
26 highlight Comment ctermfg=cyan
27 set showmode
28
29 set encoding=utf-8
30 set fileencoding=utf-8
31 scriptencoding=utf-8

```

1.2 bashrc

```

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

```

1.3 C++ template

```

1 #include <bits/stdc++.h>
2
3 using namespace std;
4
5 #define x first
6 #define y second
7
8 typedef long long int ll;
9 typedef pair<int, int> ii;
10
11 int main()
12 {

```

```

13     return 0;
14 }

```

1.4 Java template

```

1 import java.io.*;
2 import java.util.*;
3
4 public class Main
5 {
6     public static void main(String[] args)
7     {
8         MyScanner sc = new MyScanner();
9         out = new PrintWriter(new BufferedOutputStream(System.out));
10        // Start writing your solution here.
11
12        // Stop writing your solution here.
13        out.close();
14    }
15
16    public static PrintWriter out;
17
18    public static class MyScanner
19    {
20        BufferedReader br;
21        StringTokenizer st;
22
23        public MyScanner()
24        {
25            br = new BufferedReader(new InputStreamReader(System.in));
26        }
27
28        boolean hasNext()
29        {
30            while (st == null || !st.hasMoreElements()) {
31                try {
32                    st = new StringTokenizer(br.readLine());
33                } catch (Exception e) {
34                    return false;
35                }
36            }
37            return true;
38        }
39
40        String next()
41        {
42            if (hasNext())
43                return st.nextToken();
44            return null;
45        }
46
47        int nextInt()
48        {
49            return Integer.parseInt(next());
50        }

```

```

51
52        long nextLong()
53        {
54            return Long.parseLong(next());
55        }
56
57        double nextDouble()
58        {
59            return Double.parseDouble(next());
60        }
61
62        String nextLine()
63        {
64            String str = "";
65            try {
66                str = br.readLine();
67            } catch (IOException e) {
68                e.printStackTrace();
69            }
70            return str;
71        }
72    }
73 }

```

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.
6. Bus error: the code has *scanf*, *fgets* but have nothing to read! Check if you have early termination but didn't handle it properly.

3 Useful code

3.1 Leap year

```

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

```

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

```

1 ll fast_pow(ll base, ll exp, ll mod)
2 {
3     if (exp == 0)
4         return 1LL;
5     ll res = 1;
6     while (exp > 0) {
7         if (exp & 1) {
8             res = ((res % mod) * (base % mod)) % mod;
9         }
10        exp >>= 1;
11        base = (base * base) % mod;

```

```

12     }
13     return res;
14 }

```

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

注意負數的 case!

```

1 ll gcd(ll a, ll b)
2 {
3     return b == 0 ? a : gcd(b, a % b);
4 }

```

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 $\gcd(a, b)$.

```

1 ll ext_gcd(ll a, ll b, ll &x, ll &y)
2 {
3     if (a == 0) {
4         x = 0;
5         y = 1;
6         return b;
7     }
8
9     ll x1, y1;
10    ll gcd = ext_gcd(b % a, a, x1, y1);
11
12    x = y1 - (b / a) * x1;
13    y = x1;
14
15    return gcd;
16 }

```

3.5 Mod Inverse

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

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

3.6 Prime Generator

```

1 bool is_prime[N];
2 vector<ll> primes;
3 void init()
4 {
5     fill(is_prime, is_prime + N, true);
6     for (int i = 2; i < N; i++) {
7         if (is_prime[i] == true) {
8             primes.push_back(i);
9             for (int j = i * i; j < N; j += i)
10                is_prime[j] = false;
11        }
12    }
13 }

```

3.7 Binomial Coefficient

```

1 int binomialCoeff(int n, int k)
2 {
3     int res = 1;
4
5     if (k > n - k) // Since C(n, k) = C(n, n-k)
6         k = n - k;
7
8     for (int i = 0; i < k; ++i) // n...n-k / 1...k
9     {
10        res *= (n - i);
11        res /= (i + 1);
12    }
13
14    return res;
15 }

```

3.8 STL quick reference

3.8.1 Map

```

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

```

3.8.2 Set

```

1 set<T> s; // iterable
2 void clear();
3 size_t count(T val); // number of val in set
4 void erase(T val);
5 it find(T val); // = s.end() if not found
6 void insert(T val);
7 it lower_bound(T val); // = s.end() if not found, *it = <key, val>
8 it upper_bound(T val); // = s.end() if not found, *it = <key, val>

```

3.8.3 Algorithm

```

1 // return if i is smaller than j
2 comp = [&](const T &i, const T &j) -> bool;
3 vector<T> v;
4 bool any_of(v.begin(), v.end(), [&](const T &i) -> bool);
5 bool all_of(v.begin(), v.end(), [&](const T &i) -> bool);
6 void copy(inp.begin(), inp.end(), out.begin());
7 int count(v.begin(), v.end(), int val); // number of val in v
8 it unique(v.begin(), v.end()); // it - v.begin() = size

```

```

9 // after calling, v[nth] will be n-th smallest elem in v
10 void nth_element(v.begin(), nth_it, bin_comp);
11 void merge(in1.begin(), in1.end(), in2.begin(), in2.end(), out.begin(),
    comp);
12 // include union, intersection, difference, symmetric_difference(xor)
13 void set_union(in1.begin(), in1.end(), in2.begin(), in2.end(), out.
    begin(), comp);
14 bool next_permutation(v.begin(), v.end());
15 // v1, v2 need sorted already, whether v1 includes v2
16 bool includes(v1.begin(), v1.end(), v2.begin(), v2.end());
17 it find(v.begin(), v.end(), T val); // = v.end() if not found
18 it search(v1.begin(), v1.end(), v2.begin(), v2.end());
19 it lower_bound(v.begin(), v.end(), T val);
20 it upper_bound(v.begin(), v.end(), T val);
21 bool binary_search(v.begin(), v.end(), T val); // exist in v ?
22 void sort(v.begin(), v.end(), comp);
23 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

```

1 const int MAX_N = 20000;
2 int N;
3 ll bit[MAX_N + 1];
4
5 int sum(int i) {
6     int s = 0;
7     while (i > 0) {
8         s += bit[i];
9         i -= (i & -i);
10    }
11    return s;
12 }
13
14 void add(int i, int x) {
15     while (i <= MAX_N) {
16         bit[i] += x;
17         i += (i & -i);
18    }
19 }

```

- 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 Dijkstra
 - 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 Flow
 - 8.5.4 Maximum Bipartite Graph
- 9 String
 - 9.1 KMP
 - 9.2 Z Algorithm
 - 9.3 Trie
 - 9.4 Suffix Array
- 10 Geometry
 - 10.1 Template

```

1 | #define x first
2 | #define y second
3 | typedef pair <double , double > pt;
4 | struct line {
5 |     double a, b, c;
6 |     // coefficients in general form, compare up to constant factor
7 | }
8 | pt operator-(pt u, pt v) { return pt(u.x-v.x, u.y-v.y); }
9 | pt operator+(pt u, pt v) { return pt(u.x+v.x, u.y+v.y); }
10 | pt operator*(pt u, double d) { return pt(u.x*d, u.y*d); }
11 | double operator*(pt u, pt v) { return u.x*v.x + u.y*v.y; } // dot
    product double operator!(pt p) { return sqrt(p*p); } // norm

```

```

12 | double operator^(pt u, pt v) { return u.x*v.y - u.y*v.x; } // cross
    product

```

10.1.1 Point / Line

10.1.2 Intersection

10.2 Half-plane intersection

10.3 Convex Hull