

Contents

1	Contest Setup	2
1.1	vimrc	2
1.2	bashrc	2
1.3	C++ template	2
1.4	Java template	2
1.4.1	Java Issues	3
2	System Testing	3
3	Reminder	3
4	Useful code	3
4.1	Grep Error and Warnings	3
4.2	Leap year	3
4.3	Fast Exponentiation $O(\log(exp))$	3
4.4	GCD $O(\log(a+b))$	3
4.5	Extended Euclidean Algorithm	3
4.6	Mod Inverse	4
4.7	Prime Generator	4
4.8	Binomial Coefficient	4
4.9	STL quick reference	4
4.9.1	Map	4
4.9.2	Set	4
4.9.3	Algorithm	4
4.9.4	String	4
4.9.5	Priority Queue	4
5	Search	5
5.1	Ternary Search	5
5.2	折半完全列舉	5
5.3	Two-pointer 爬行法	5
6	Basic data structure	5
6.1	1D BIT	5
6.2	2D BIT	5
6.3	Union Find	5
6.4	Segment Tree	5
6.5	Sparse Table	5
7	Dynamic Programming	6
8	Tree	6
8.1	LCA	6
9	Graph	6
9.1	Articulation point / edge	6
9.2	CC	6
9.2.1	BCC vertex	6
9.2.2	BCC edge	6
9.2.3	SCC	6
9.3	Shortest Path	6
9.3.1	Dijkstra	6
9.3.2	Dijkstra (next-to-shortest path)	6
9.3.3	SPFA	6
9.3.4	Bellman-Ford	6
9.3.5	Floyd-Warshall	6
9.4	Kruskal MST	6
9.5	Flow	6
9.5.1	Max Flow (Dinic)	6
9.5.2	Min-Cut	6
9.5.3	Min Cost Max Flow	6
9.5.4	Maximum Bipartite Graph	6
10	String	6
10.1	Rolling Hash	6
10.2	KMP	6
10.3	Z Algorithm	7
10.4	Trie	7
10.5	Suffix Array	7
11	Geometry	7
11.1	EPS	7

11.2	Template	7
------	----------	---

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

```

1.4.1 Java Issues

1. Random Shuffle before sorting: `Random rnd = new Random(); rnd.nextInt();`
2. Use `StringBuilder` for large output

2 System Testing

1. Setup `bashrc` and `vimrc`
2. Look for compilation parameter and code it into `bashrc`
3. Test if `c++` and `java` templates work properly on local and judge machine
4. Test "divide by 0" → RE/TLE?
5. Test stack size

3 Reminder

1. 隊友的建議，要認真聽！通常隊友的建議都會突破你盲點
2. Read the problem statements carefully. Input and output specifications and constraints are crucial!
3. Estimate the **time complexity** and **memory complexity** carefully.
4. Time penalty is 20 minutes per WA, **don't rush!**
5. Sample test cases must all be tested and passed before every submission!
6. Test the corner cases, such as 0, 1, -1. Test all edge cases of the input specification.
7. Bus error: the code has `scanf, fgets` but have nothing to read! Check if you have early termination but didn't handle it properly.
8. Binary search? 數學算式移項合併後查詢?
9. Two Pointer ↔ Binary Search
10. Directed graph connectivity → DFS. Undirected graph → Union Find
11. Check connectivity of the graph if the problem statement doesn't say anything
12. `longlong = int * int` won't work!
13. Shifting for `longlongint` should be something like `1LL << 35`
14. For continuous input problems, be sure to read in all input BEFORE terminating and start processing next the input.
15. Don't use anonymous struct

4 Useful code

4.1 Grep Error and Warnings

```
1 || g++ main.cpp 2>&1 | grep -E 'warning|error'
```

4.2 Leap year

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

4.3 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 }

```

4.4 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 }

```

4.5 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 |     return gcd;
15 | }
16 |

```

4.6 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)

4.7 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 | }

```

4.8 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 | }

```

4.9 STL quick reference

4.9.1 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>

```

4.9.2 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>

```

4.9.3 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);

```

4.9.4 String

4.9.5 Priority Queue

```

1 | bool cmp(ii a, ii b)
2 | {
3 |     if(a.first == b.first)
4 |         return a.second > b.second;
5 |     return b.first > a.first;
6 | }

```

```

7 | priority_queue< ii, vector<ii>, function<bool(ii, ii)> > pq(cmp);
8 |

```

5 Search

5.1 Ternary Search

5.2 折半完全列舉

5.3 Two-pointer 爬行法

6 Basic data structure

6.1 1D BIT

```

1 | // BIT is 1-based
2 | const int MAX_N = 20000; //這個記得改!
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 | }

```

6.2 2D BIT

```

1 | // BIT is 1-based
2 | const int MAX_N = 20000, MAX_M = 20000; //這個記得改!
3 | ll bit[MAX_N + 1][MAX_M + 1];
4 |
5 | ll sum(int a, int b) {
6 |     ll s = 0;
7 |     for (int i = a; i > 0; i -= (i & -i))
8 |         for (int j = b; j > 0; j -= (j & -j))
9 |             s += bit[i][j];
10 |    return s;
11 | }
12 |
13 | void add(int a, int b, ll x) {
14 |     // MAX_N, MAX_M 須適時調整!
15 |     for (int i = a; i <= MAX_N; i += (i & -i))
16 |         for (int j = b; j <= MAX_M; j += (j & -j))

```

```

17 |         bit[i][j] += x;
18 |    }

```

6.3 Union Find

```

1 | #define N 20000 // 記得改
2 | struct UFDS {
3 |     int par[N];
4 |
5 |     void init() {
6 |         memset(par, -1, sizeof(par));
7 |     }
8 |
9 |     int root(int x) {
10 |         return par[x] < 0 ? x : par[x] = root(par[x]);
11 |     }
12 |
13 |     void merge(int x, int y) {
14 |         x = root(x);
15 |         y = root(y);
16 |
17 |         if (x != y) {
18 |             if (par[x] > par[y])
19 |                 swap(x, y);
20 |             par[x] += par[y];
21 |             par[y] = x;
22 |         }
23 |     }
24 | }

```

6.4 Segment Tree

6.5 Sparse Table

```

1 | struct {
2 |     int sp[MAX_LOG_N][MAX_N]; // MAX_LOG_N = ceil(lg(MAX_N))
3 |
4 |     void build(int inp[], int n) {
5 |         for (int j = 0; j < n; j++) {
6 |             sp[0][j] = inp[j];
7 |         }
8 |
9 |         for (int i = 1; (1 << i) <= n; i++)
10 |             for (int j = 0; j + (1 << i) <= n; j++)
11 |                 sp[i][j] =
12 |                     min(sp[i - 1][j], sp[i - 1][j + (1 << (i - 1))]);
13 |     }
14 |
15 |     int query(int l, int r) { // [l, r)
16 |         int k = floor(log2(r - l));
17 |
18 |         return min(sp[k][l], sp[k][r - (1 << k)]);
19 |     }
20 | } sptb;

```

7 Dynamic Programming

8 Tree

8.1 LCA

9 Graph

9.1 Articulation point / edge

9.2 CC

9.2.1 BCC vertex

9.2.2 BCC edge

9.2.3 SCC

9.3 Shortest Path

9.3.1 Dijkstra

9.3.2 Dijkstra (next-to-shortest path)

9.3.3 SPFA

9.3.4 Bellman-Ford

9.3.5 Floyd-Warshall

9.4 Kruskal MST

9.5 Flow

9.5.1 Max Flow (Dinic)

9.5.2 Min-Cut

9.5.3 Min Cost Max Flow

9.5.4 Maximum Bipartite Graph

10 String

10.1 Rolling Hash

1. Use two rolling hashes if needed.
2. The prime for pre-calculation can be 137 and 257, for modulo can be $1e9 + 7$ and *0xdefaced*

```

1 #define N 1000100
2 #define B 137
3 #define M 1000000007
4
5 typedef long long ll;
6
7 char inp[N];
8 int len;
9 ll p[N], h[N];
10
```

```

11 void init()
12 { // build polynomial table and hash value
13     p[0] = 1; // b to the ith power
14     for (int i = 1; i <= len; i++) {
15         h[i] = (h[i - 1] * B % M + inp[i - 1]) % M; // hash value
16         p[i] = p[i - 1] * B % M;
17     }
18 }
19
20 ll get_hash(int l, int r) // [l, r] of the inp string array
21 {
22     return ((h[r + 1] - (h[l] * p[r - l + 1])) % M + M) % M;
23 }
```

10.2 KMP

```

1 void fail()
2 {
3     int len = strlen(pat);
4
5     f[0] = 0;
6     int j = 0;
7     for (int i = 1; i < len; i++) {
8         while (j != 0 && pat[i] != pat[j])
9             j = f[j - 1];
10
11         if (pat[i] == pat[j])
12             j++;
13
14         f[i] = j;
15     }
16 }
17
18 int match()
19 {
20     int res = 0;
21     int j = 0, plen = strlen(pat), tlen = strlen(text);
22
23     for (int i = 0; i < tlen; i++) {
24         while (j != 0 && text[i] != pat[j])
25             j = f[j - 1];
26
27         if (text[i] == pat[j]) {
28             if (j == plen - 1) { // find match
29                 res++;
30                 j = f[j];
31             } else {
32                 j++;
33             }
34         }
35     }
36
37     return res;
38 }
```

10.3 Z Algorithm

10.4 Trie

```

1 #define N 600010
2 struct node {
3     int child[26];
4     bool ending;
5 } trie[N];
6
7 /*
8  root is 0
9  memset(trie, 0, sizeof(trie));
10 freeNode = 1;
11 */
12 int freeNode;
13 void insert(string &str, int pos, int node)
14 {
15     if (pos == (int)str.length()) {
16         trie[node].ending = true;
17     } else { // find which way to go
18         int c = str[pos] - 'a';
19         if (trie[node].child[c] == 0) // give a new node
20             trie[node].child[c] = freeNode++;
21         insert(str, pos + 1, trie[node].child[c]);
22     }
23 }

```

10.5 Suffix Array

11 Geometry

1. Keep things in integers as much as possible!
2. Try not to divide
3. If you have decimals, if they are fixed precision, you can usually just multiply all the input and use integers instead

11.1 EPS

$a > b \rightarrow a - b > 0 \rightarrow a - b > EPS$ (stands for positive)
 $a \geq b \rightarrow a - b \geq 0 \rightarrow a - b > -EPS$ (stands for positive or zero)

11.2 Template

```

1 // if the points are given in doubles form, change the code accordingly
2
3 typedef long long ll;
4
5 typedef pair<ll, ll> pt; // points are stored using long long
6 typedef pair<pt, pt> seg; // segments are a pair of points
7
8 #define x first
9 #define y second
10
11 #define EPS 1e-9

```

```

12 pt operator+(pt a, pt b)
13 {
14     return pt(a.x + b.x, a.y + b.y);
15 }
16
17 pt operator-(pt a, pt b)
18 {
19     return pt(a.x - b.x, a.y - b.y);
20 }
21
22 pt operator*(pt a, int d)
23 {
24     return pt(a.x * d, a.y * d);
25 }
26
27 ll cross(pt a, pt b)
28 {
29     return a.x * b.y - a.y * b.x;
30 }
31
32 int ccw(pt a, pt b, pt c)
33 {
34     ll res = cross(b - a, c - a);
35     if (res > 0) // left turn
36         return 1;
37     else if (res == 0) // straight
38         return 0;
39     else // right turn
40         return -1;
41 }
42
43 double dist(pt a, pt b)
44 {
45     double dx = a.x - b.x;
46     double dy = a.y - b.y;
47     return sqrt(dx * dx + dy * dy);
48 }
49
50 bool zero(double x)
51 {
52     return fabs(x) <= EPS;
53 }
54
55 bool overlap(seg a, seg b)
56 {
57     return ccw(a.x, a.y, b.x) == 0 && ccw(a.x, a.y, b.y) == 0;
58 }
59
60 bool intersect(seg a, seg b)
61 {
62     if (overlap(a, b) == true) { // non-proper intersection
63         double d = 0;
64         d = max(d, dist(a.x, a.y));
65         d = max(d, dist(a.x, b.x));
66         d = max(d, dist(a.x, b.y));
67     }
68 }

```

```

68     d = max(d, dist(a.y, b.x));
69     d = max(d, dist(a.y, b.y));
70     d = max(d, dist(b.x, b.y));
71
72     // d > dist(a.x, a.y) + dist(b.x, b.y)
73     if (d - (dist(a.x, a.y) + dist(b.x, b.y)) > EPS)
74         return false;
75     return true;
76 }
77 // -----|
78 // Equal sign for -----| case
79 // non equal sign => proper intersection
80 if (ccw(a.x, a.y, b.x) * ccw(a.x, a.y, b.y) <= 0 &&
81     ccw(b.x, b.y, a.x) * ccw(b.x, b.y, a.y) <= 0)
82     return true;
83 return false;
84 }
85
86 double area(vector<pt> pts)
87 {
88     double res = 0;
89     int n = pts.size();
90     for (int i = 0; i < n; i++)
91         res += (pts[i].y + pts[(i + 1) % n].y) * (pts[(i + 1) % n].x -
92             pts[i].x);
93     return res / 2.0;
94 }
95
96 vector<pt> halfHull(vector<pt> &points)
97 {
98     vector<pt> res;
99
100     for (int i = 0; i < (int)points.size(); i++) {
101         while ((int)res.size() >= 2 &&
102             ccw(res[res.size() - 2], res[res.size() - 1], points[i])
103             < 0)
104             res.pop_back(); // res.size() - 2 can't be assign before
105             size() >= 2
106             // check, bitch
107
108         res.push_back(points[i]);
109     }
110
111     return res;
112 }
113
114 vector<pt> convexHull(vector<pt> &points)
115 {
116     vector<pt> upper, lower;
117
118     // make upper hull
119     sort(points.begin(), points.end());

```

```

118     upper = halfHull(points);
119     // make lower hull
120     reverse(points.begin(), points.end());
121     lower = halfHull(points);
122
123     // merge hulls
124     if ((int)upper.size() > 0) // yes sir~
125         upper.pop_back();
126     if ((int)lower.size() > 0)
127         lower.pop_back();
128
129     vector<pt> res(upper.begin(), upper.end());
130     res.insert(res.end(), lower.begin(), lower.end());
131
132     return res;
133 }
134
135 bool completelyInside(vector<pt> &outer, vector<pt> &inner)
136 {
137     int even = 0, odd = 0;
138     for (int i = 0; i < (int)inner.size(); i++) {
139         // y = slope * x + offset
140         int cntIntersection = 0;
141         ll slope = rand() % INT_MAX + 1;
142         ll offset = inner[i].y - slope * inner[i].x;
143
144         ll farx = 111111 * (slope >= 0 ? 1 : -1);
145         ll fary = farx * slope + offset;
146         seg a = seg(pt(inner[i].x, inner[i].y), pt(farx, fary));
147         for (int j = 0; j < (int)outer.size(); j++) {
148             seg b = seg(outer[j], outer[(j + 1) % (int)outer.size()]);
149
150             if ((b.x.x * slope + offset == b.x.y) ||
151                 (b.y.x * slope + offset == b.y.y)) { // on-line
152                 i--;
153                 break;
154             }
155
156             if (intersect(a, b) == true)
157                 cntIntersection++;
158         }
159
160         if (cntIntersection % 2 == 0) // outside
161             even++;
162         else
163             odd++;
164     }
165
166     return odd == (int)inner.size();
167 }
168
169 // srand(time(NULL))
170 // rand()

```