

# Contents

<b>1</b>	<b>Contest Setup</b>	<b>2</b>
1.1	vimrc	2
1.2	bashrc	2
1.3	C++ template	2
1.4	Java template	3
1.4.1	Java Issues	3
<b>2</b>	<b>Reminder</b>	<b>3</b>
<b>3</b>	<b>Useful code</b>	<b>3</b>
3.1	Grep Error and Warnings	3
3.2	Leap year	3
3.3	Fast Exponentiation $O(\log(exp))$	3
3.4	GCD $O(\log(a+b))$	3
3.5	Extended Euclidean Algorithm	3
3.6	Mod Inverse	3
3.7	Prime Generator	4
3.8	Binomial Coefficient	4
3.9	STL quick reference	4
3.9.1	Map	4
3.9.2	Set	4
3.9.3	Algorithm	4
3.9.4	String	4
3.9.5	Priority Queue	4
<b>4</b>	<b>Search</b>	<b>5</b>
4.1	Binary Search	5
4.1.1	Find key	5
4.1.2	Upper / lower Bound	5
4.2	Ternary Search	5
4.3	折半完全列舉	5
4.4	Two-pointer 爬行法	5
<b>5</b>	<b>Basic data structure</b>	<b>5</b>
5.1	1D BIT	5
5.2	2D BIT	5
5.3	Union Find	5
5.4	Segment Tree	5
5.5	Sparse Table	5
<b>6</b>	<b>Dynamic Programming</b>	<b>6</b>
<b>7</b>	<b>Tree</b>	<b>6</b>
7.1	LCA	6
<b>8</b>	<b>Graph</b>	<b>6</b>
8.1	Articulation point / edge	6
8.2	CC	6
8.2.1	BCC vertex	6
8.2.2	BCC edge	6
8.2.3	SCC	6
8.3	Shortest Path	6
8.3.1	Dijkstra	6
8.3.2	Dijkstra (next-to-shortest path)	6
8.3.3	SPFA	6
8.3.4	Bellman-Ford	6
8.3.5	Floyd-Warshall	6
8.4	Kruskal MST	6
8.5	Flow	6
8.5.1	Max Flow (Dinic)	6
8.5.2	Min-Cut	6
8.5.3	Min Cost Max Flow	6
8.5.4	Maximum Bipartite Graph	6
<b>9</b>	<b>String</b>	<b>6</b>
9.1	Rolling Hash	6
9.2	KMP	6
9.3	Z Algorithm	7
9.4	Trie	7
9.5	Suffix Array	7
<b>10</b>	<b>Geometry</b>	<b>7</b>

<b>10.1</b>	<b>Template</b>	<b>7</b>
10.1.1	Point / Line	7
10.1.2	Intersection	7
<b>10.2</b>	<b>Half-plane intersection</b>	<b>7</b>
<b>10.3</b>	<b>Convex Hull</b>	<b>7</b>

# 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 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  $\leftrightarrow$  Binary Search
10. Directed graph connectivity  $\rightarrow$  DFS. Undirected graph  $\rightarrow$  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`

## 3 Useful code

### 3.1 Grep Error and Warnings

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

### 3.2 Leap year

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

### 3.3 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 }

```

### 3.4 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 }

```

### 3.5 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
15    return gcd;
16 }

```

### 3.6 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)

### 3.7 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 }

```

### 3.8 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 }

```

### 3.9 STL quick reference

#### 3.9.1 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>

```

#### 3.9.2 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>

```

#### 3.9.3 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 inclues(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);

```

#### 3.9.4 String

#### 3.9.5 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
8 priority_queue< ii, vector<ii>, function<bool(ii, ii)> > pq(cmp);

```

## 4 Search

### 4.1 Binary Search

#### 4.1.1 Find key

#### 4.1.2 Upper / lower Bound

### 4.2 Ternary Search

### 4.3 折半完全列舉

### 4.4 Two-pointer 爬行法

## 5 Basic data structure

### 5.1 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 }
```

### 5.2 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 }
```

### 5.3 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 }
```

### 5.4 Segment Tree

### 5.5 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;
```

## 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 Dijkstra (next-to-shortest path)

#### 8.3.3 SPFA

#### 8.3.4 Bellman-Ford

#### 8.3.5 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 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 }
```

### 9.2 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 }
```

## 9.3 Z Algorithm

## 9.4 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 }
```

## 9.5 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
12 // product double operator!(pt p) { return sqrt(p*p); } // norm
13 double operator^(pt u, pt v) { return u.x*v.y - u.y*v.x; } // cross
14 // product
```

#### 10.1.1 Point / Line

#### 10.1.2 Intersection

### 10.2 Half-plane intersection

### 10.3 Convex Hull