

Contents

1	Todo	2
2	Contest Setup	2
2.1	vimrc	2
2.2	bashrc	2
2.3	c++ template	2
2.4	Java template	2
3	Reminder	2
4	Useful code	2
4.1	Fast Exponentiation	2
4.2	GCD	2
4.3	Extended Euclidean Algorithm	2
4.4	STL quick reference	2
4.4.1	Map / Set	2
4.4.2	String	2
5	Search	2
5.1	Binary Search	2
5.1.1	Find key	2
5.1.2	Upper / lower Bound	2
5.2	折半完全列举	2
5.3	Two-pointer 爬行法	2
6	Basic data structure	2
6.1	1D BIT	2
6.2	2D BIT	2
6.3	Union Find	2
6.4	Segment Tree	2
7	Dynamic Programming	2
8	Tree	2
8.1	LCA	2
9	Graph	2
9.1	Articulation point / edge	2
9.2	BCC vertex	2
9.3	BCC edge	2
9.4	SCC	2
9.5	Shortest Path	2
9.5.1	Dijkstra	2
9.5.2	SPFA	2
9.5.3	Bellman-Ford	2
9.6	Flow	2
9.6.1	Max Flow (Dinic)	2
9.6.2	Min-Cut	2
9.6.3	Min Cost Max Flow	2
9.6.4	Maximum Bipartite Graph	2
10	String	2
10.1	KMP	2
10.2	Z Algorithm	2
10.3	Trie	2
10.4	Suffix Array	2
11	Geometry	2
11.1	Template	2
11.1.1	Point / Line	2
11.1.2	Intersection	2
11.2	Half-plane intersection	2
11.3	Convex Hull	2

1 Todo

1. Add code and complexity
2. Add brief explanations

2 Contest Setup

2.1 vimrc

```

"General
set number " Show line numbers
set linebreak " Break lines at word (requires Wrap lines)
set showbreak=+++ " Wrap-broken line prefix
set textwidth=150 " Line wrap (number of cols)
set showmatch " Highlight matching brace
set visualbell " Use visual bell (no beeping)

set hlsearch " Highlight all search results
set smartcase " Enable smart-case search
set ignorecase " Always case-insensitive
set incsearch " Searches for strings incrementally

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
set cursorline " Show underline
set cursorcolumn
set ruler " Show row and column ruler information

set mouse=a

set undolevels=10000 " Number of undo levels
set backspace=indent,eol,start " Backspace behaviour

set splitbelow "New split starts below
set splitright "New split starts on the right
set scrolloff=5 "auto scroll on the bottom 5 lines

filetype on "enable file detection
syntax on "syntax highlight

highlight Comment ctermfg=cyan
set showmode

set encoding=utf-8
set fileencoding=utf-8
set scriptencoding=utf-8

```

contest_setup/vimrc

2.2 bashrc

```

1 alias clang++="clang++-3.6 -Wall -Wextra -O2 -o main
.o"
2 alias clang-format="clang-format-3.6 -i -style=LLVM"
3 alias astyle="astyle --style=linux"

```

contest_setup/bashrc

2.3 c++ template

2.4 Java template

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

```

1 #include <bits/stdc++.h>
2
3 using namespace std;
4
5 #define LIMIT 500100
6 #define LSB(x) ((x) & (-x))
7
8 long long int bit[LIMIT];
9
10 void add(int i, int val)
11 {
12     while (i <= LIMIT) {
13         bit[i] += val;
14         i += LSB(i);
15     }
16 }
17
18 int sum(int i)
19 {
20     int ans = 0;
21     while (i > 0) {
22         ans += bit[i];
23         i -= LSB(i);
24     }
25
26     return ans;
27 }
28
29 int main()
30 {

```

```

31 int n;
32 while (scanf("%d", &n) == 1 && n) {
33     int inp[n + 1], orig[n + 1];
34     for (int i = 1; i <= n; i++) {
35         scanf("%d", &inp[i]);
36         orig[i] = inp[i];
37     }
38
39     sort(inp + 1, inp + n + 1);
40
41     /*
42     因為數字的範圍很大
43     所以我們把答案離散化
44
45     按照排列之後的順序
46     我們把數字重新給一個編號
47     從1開始編起 (BIT是從1開始編起的)
48     */
49     map<int, int> m;
50     for (int i = 1; i <= n; i++)
51         m[inp[i]] = i; // duplicated key?
52
53     memset(bit, 0, sizeof(bit));
54
55     long long int ans = 0; // Ops
56     for (int cnt = 0; cnt < n; cnt++) {
57         /*
58         BIT所記錄的資料是利用重新編號之後的數列來紀
59         錄
60         所以像說
61         6 2 8 4
62         會被重新編號成
63         3 1 4 2
64
65         在BIT中，對於每一個i (就是sum(i))所要記錄
66         的是目前為止合格的數字有幾個(<= i的數字)
67
68         以剛剛那個數列來說 (6 2 8 4)
69         對於第一個數字 3 而言
70         cnt = 0, sum(6 對應到 3) = 0
71         所以ans += 0 - 0; (代表在3之前沒有大於他
72         的數字)
73         之後，在 3 那一格加 1
74
75         這很重要，因為這麼做就把bit[i >= 3]的所有數
76         值都都加 1 了!
77         (這是BIT的特性，畢竟BIT是拿來算prefix sum
78         的!)
79
80         所以下一次如果我們問到>=3的數字，假設是4好
81         了
82         我們會知道已經有一個<=4的數字已經出現
83         也就可以反推有幾個不合格的數字出現了!
84         */
85         ans += cnt - sum(m[orig[cnt + 1]]);
86         add(m[orig[cnt + 1]], 1);
87     }
88
89     printf("%lld\n", ans);
90 }
91
92 return 0;
93 }

```

old/"UVA 10810 BIT.cpp"

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 Dijkstra

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