SGU 320-359 Report

AsZ yuzhou627

July 5, 2015

Contents

1 概述

这是 USTC-ACM 2015 暑假训练题目的题解总结, 题目选自 SGU320-359.SGU 的题目是毛子题, 往往考验智商, 大部分都不 (划掉) 简单. 有些题目值得想上几天. 这也是我大学最后一个为 acm, 为 $Final$ 奋斗的暑假, 准备把这件事好好做好, 也可以留下些东西给后辈. 我校一直缺少传承. 我就稍微做个不合格的范例吧. 不知道后来人有没有能做得再好些的. 加油吧各位.

2 SGU321

N 个点的有根树，边有黑白两色，将最少的黑边改成白边，使得从根到任意点的路径上，白边的数量不少于黑边。($N \leq 200000$)

2.1 Solution

贪心是显然的, 尽量取靠近根的边变色.

1. 线段树,dfs 走到一条可以变色的边, 询问子树里面有没有不合法的, 有就变色, 然后更新子树.$O(N \log N)$
2. deque 直接搞,dfs 走完所以顶点, 用一个全局的 deque 记录从根到目前节点, 所有可以被染色的边, 然后节点不合法就变色, 并把 deque 维护一下.$O(N)$ by ftiasch.

2.2 Code

```
1 #include <bits/stdc++.h>
2
```

```cpp
#define REP(i, a) REPP(i, 0, (a) - 1)
#define REPP(i, a, b) for (int i = int(a); i <= int(b); i++)
#define MST(a, b) memset(a, b, sizeof(a))

#define L (x << 1)
#define R (x << 1 | 1)
#define MID ((l + r) >> 1)
#define LC L, l, MID
#define RC R, MID + 1, r
#define end en

using namespace std;

const int N = 2e5 + 5;
int dep[N], start[N], end[N], head[N], cnt[N], edge = 1, now = 1;
int mi[N << 2], flag[N << 2];

struct Edge{
    int y, next, type;
}e[N << 1];

void add(int x, int y, int z) {
    e[++edge].next = head[x], head[x] = edge, e[edge].y = y, e[edge].type = z;
}

void gao(int x, int p = 0, int tot = 0) {
    start[x] = now++, dep[x] = dep[p] + 1, cnt[now - 1] = tot - (dep[x] >> 1);
    for (int go = head[x]; go; go = e[go].next) {
        int y = e[go].y;
        if (y != p) {
            gao(y, x, tot + e[go].type);
        }
    }
    end[x] = now - 1;
}

int ql, qr, qd;

void build(int x, int l, int r) {
    if (l == r) mi[x] = cnt[l], flag[x] = 0;
    else {
        build(LC), build(RC);
        mi[x] = min(mi[L], mi[R]);
        flag[x] = 0;
    }
}

```

```
void push(int x, int l, int r) {
    flag[L] += flag[x], flag[R] += flag[x];
    mi[L] += flag[x], mi[R] += flag[x];
    flag[x] = 0;
}

void update(int x, int l, int r) {
    if (ql <= l && qr >= r) {
        flag[x] += 1;
        mi[x] += 1;
    }
    else {
        if (flag[x]) push(x, l, r);
        if (ql <= MID) update(LC);
        if (qr > MID) update(RC);
        mi[x] = min(mi[L], mi[R]);
    }
}

int query(int x, int l, int r) {
    if (ql <= l && qr >= r) {
        return mi[x];
    }
    else {
        if (flag[x]) push(x, l, r);
        int ans = INT_MAX;
        if (ql <= MID) ans = min(ans, query(LC));
        if (qr > MID) ans = min(ans, query(RC));
        return ans;
    }
}

vector<int> ans;

void dfs(int x, int p = 0) {
    for (int go = head[x]; go; go = e[go].next) {
        int y = e[go].y;
        if (y != p) {
            ql = start[y], qr = end[y];
            int tmp = query(1, 1, now);
            if (tmp < 0) {
                if (e[go].type == 0) {
                    ans.push_back(go >> 1);
                    update(1, 1, now);
                }
                dfs(y, x);
            }
        }
    }
}
```

```

int main() {
    ios :: sync_with_stdio(0);
    //freopen("321.in", "r", stdin);
    int n;
    cin >> n;
    REP(i, n - 1) {
        int x, y, z;
        string tmp;
        cin >> x >> y >> tmp;
        if (tmp[0] == 'a') {
            z = 0;
            cin >> tmp;
        }
        else {
            z = 1;
        }
        add(x, y, z), add(y, x, z);
    }
    gao(1);
    now--;
    //REPP(i, 1, n) {
    //  cout << i << ' ' << start[i] << ' ' << end[i] << ' ' << dep[i] << endl;
    //}
    build(1, 1, now);
    dfs(1);
    cout << ans.size() << endl;
    REP(i, ans.size()) {
        cout << ans[i] << " \n"[i == int(ans.size() - 1)];
    }

    return 0;
}
```

Listing 1: 321v1.cc

```
// SGU 321 -- The Spy Network
#include <cstdio>
#include <cstring>
#include <deque>
#include <vector>
#include <utility>
#include <algorithm>

const int N = 200000;

int n;

int first_edge[N], to[N], next_edge[N], type[N];
```

```
14
15 int current;
16 std::deque <int> edges;
17
18 std::vector <int> choice;
19
20 void dfs(int u) {
21     if (current < 0) {
22         current += 2;
23         type[edges.front()] *= -1;
24         choice.push_back(edges.front());
25         edges.pop_front();
26     }
27     for (int iter = first_edge[u]; iter != -1; iter = next_edge
       [iter]) {
28         current += type[iter];
29         if (type[iter] == -1) {
30             edges.push_back(iter);
31         }
32         dfs(to[iter]);
33         if (type[iter] == -1) {
34             edges.pop_back();
35         }
36         current -= type[iter];
37     }
38 }
39
40 int main() {
41     scanf("%d", &n);
42     memset(first_edge, -1, sizeof(first_edge));
43     for (int i = 0, a, b; i < n - 1; ++ i) {
44         char buffer[10];
45         scanf("%d%d%s", &a, &b, buffer);
46         a --;
47         b --;
48         to[i] = a;
49         type[i] = *buffer == 'p' ? 1 : -1;
50         next_edge[i] = first_edge[b];
51         first_edge[b] = i;
52         if (*buffer == 'a') {
53             scanf("%s", buffer);
54         }
55     }
56     dfs(0);
57     printf("%d\n", (int)choice.size());
58     for (int i = 0; i < (int)choice.size(); ++ i) {
59         printf("%d%c", choice[i] + 1, i == (int)choice.size() -
        1 ? '\n' : ' ');
60     }
61     return 0;
```

```
}
```

Listing 2: 321v2.cc

3 SGU322

给两棵树, 你可以对任意对树进行做加边删边操作, 问最小多少步, 可以使两棵树长的完全一样.($N \leq 2000$)

3.1 Solution

首先, 动两棵树是没有意义的. 因为 $|A \to T| + |B \to T| >= |A \to B|$. 操作是可逆的, 既然可以把 B 变成 T, 那么就能把 A 变成 T 再变成 B. 代价不会更差. 所以直接把 A 变成 B.dfs 找环即可.$O(N^2)$.

3.2 Code

```cpp
#include <bits/stdc++.h>

#define LL long long
#define REP(i, a) REPP(i, 0, (a) - 1)
#define REPP(i, a, b) for (int i = int(a); i <= int(b); i++)
#define MST(a, b) memset(a, b, sizeof(a))

using namespace std;

const int N = 2005;
int mp[N][N], g[N][N], a[N], b[N];
vector<int> edge[N], path;
bool vis[N];

int dfs(int x, int p = 0) {
    vis[x] = 1;
    REP(i, edge[x].size()) {
        int y = edge[x][i];
        if (y != p && !vis[y]) {
            if (dfs(y, x)) {
                path.push_back(x);
                return 1;
            }
        }
        else if (y != p) {
            path.push_back(x);
            return 1;
        }
    }
    return 0;
```

```
}

int main() {
    ios :: sync_with_stdio(0);
    int n;
    cin >> n;
    if (n <= 2) {
        cout << 0 << endl;
        return 0;
    }
    REPP(i, 1, n - 1) {
        int x, y;
        cin >> x >> y;
        mp[x][y] = mp[y][x] = 1;
        edge[x].push_back(y);
        edge[y].push_back(x);
    }
    vector<pair<int, int> > ans;
    REPP(i, 1, n - 1) {
        cin >> a[i] >> b[i];
        g[a[i]][b[i]] = g[b[i]][a[i]] = 1;
    }
    REPP(i, 1, n - 1) {
        if (mp[a[i]][b[i]]) continue;
        ans.push_back(make_pair(a[i], b[i]));
        edge[a[i]].push_back(b[i]);
        edge[b[i]].push_back(a[i]);
        path.clear();
        REPP(j, 1, n) vis[j] = 0;
        dfs(a[i]);
        if (!g[path[0]][path.back()]) {
            ans.push_back(make_pair(path[0], path.back()));
            edge[path[0]].erase(find(edge[path[0]].begin(),
    edge[path[0]].end(), path.back()));
            edge[path.back()].erase(find(edge[path.back()].
    begin(), edge[path.back()].end(), path[0]));
            continue;
        }
        else {
            for (int i = 0; i < int(path.size() - 1); i++) {
                int x = path[i], y = path[i + 1];
                if (!g[x][y]) {
                    ans.push_back(make_pair(x, y));
                    edge[x].erase(find(edge[x].begin(), edge[x
    ].end(), y));
                    edge[y].erase(find(edge[y].begin(), edge[y
    ].end(), x));
                    break;
                }
            }
```

```
        }
    }
    cout << ans.size() / 2 << endl;
    for (int i = 0; i < int(ans.size()); i += 2) {
        cout << 1 << ' ' << ans[i].first << ' ' << ans[i].
    second << ' ' << ans[i + 1].first << ' ' << ans[i + 1].
    second << endl;
    }

    return 0;
}
```

Listing 3: 322.cc

4 SGU324

按要求模拟.

4.1 Solution

学会使用 string 类, 使用 istringstream 等, 会让代码变得简单. 另外锻炼思维严密性. 感觉这种题, 我喜欢先把串变得很规整 (单词以一个空格间隔, 或者把所有单词放到 vector 里), 然后慢慢处理. 感觉写的并不是很好. 注意以下用例:

```
7
"fsdfs——dsfd"
"       a        "
"_ _ _ a _ _ _"
" __ aa __ aa _ aa _ a"
"   "
"__"
"  _ _ "
```

Listing 4: 324in

4.2 Code

```
#include <bits/stdc++.h>

#define LL long long
#define REP(i, a) REPP(i, 0, (a) - 1)
#define REPP(i, a, b) for (int i = int(a); i <= int(b); i++)
#define MST(a, b) memset(a, b, sizeof(a))

using namespace std;

vector<string> word;

```

```
bool test(char s) {
    return s == '-';
}

bool test(string s) {
    return s.size() == 1 && s[0] == '-';
}

void modify(string &s) {
    size_t tmp = s.find('-');
    while (tmp != string :: npos) {
        string now = s.substr(0, tmp);
        if (now.size()) {
            word.push_back(now);
        }
        word.push_back("-");
        s = s.substr(tmp + 1);
        tmp = s.find('-');
    }
    if (s.size()) word.push_back(s);
}

void getw(istringstream &sin) {
    string ans;
    while (sin >> ans) {
        modify(ans);
    }
}

int main() {
    ios :: sync_with_stdio(0);
    //freopen("324.in", "r", stdin);
    int t;
    string s;
    cin >> t;
    getline(cin, s);
    while (t--) {
        getline(cin, s);
        s = s.substr(1, s.size() - 2);
        istringstream sin(s);
        word.clear();
        getw(sin);
        string ans;
        bool bad = 0;
        REP(i, word.size()) {
            int j = i;
            while (j < int(word.size()) && test(word[j])) j++;
            if (j == i) {
                ans += " " + word[i];
            }
```

```
            else if (j == i + 1) {
                if (i == 0 || j == int(word.size())) {
                    bad = 1;
                    break;
                }
                else {
                    ans += '-';
                    ans += word[j];
                    i = j;
                }
            }
            else {
                int cnt = j - i;
                while (cnt > 3) {
                    ans += " --";
                    cnt -= 2;
                }
                string tmp;
                REP(i, cnt) tmp += '-';
                ans += " " + tmp;
                i = j - 1;
            }
        }
        if (bad) {
            cout << "error" << endl;
        }
        else {
            if (!ans.size()) {
                cout << "\"\"" << endl;
                continue;
            }
            ans = ans.substr(1);
            if (test(ans[0])) ans = " " + ans;
            if (test(ans[ans.size() - 1])) ans += " ";
            ans = "\"" + ans + "\"";
            cout << ans << endl;
        }
    }
    return 0;
}
```

Listing 5: 324.cc

5 SGU325

求把一个字符串 $swap$ 成回文的做最小步数.($N \leq 1000000$)

5.1 Solution

解法有两个.

1. $O(N)$
2. $O(N \log N)$

5.2 Code

```
#include <bits/stdc++.h>

#define LL long long
#define REP(i, a) REPP(i, 0, (a) - 1)
#define REPP(i, a, b) for (int i = int(a); i <= int(b); i++)
#define MST(a, b) memset(a, b, sizeof(a))
#define PII pair<int, int>
#define LB(x) ((x) & (-(x)))

using namespace std;

const int N = 1e6 + 5;
char s[N];
deque<int> pos[26];
int key;

void modify(int &l, int &r) {
    if (l == key) l++;
    if (r == key) r--;
}

struct BIT{
    int s[N];

    void add(int x) {
        while (x < N) {
            s[x] += 1;
            x += LB(x);
        }
    }

    int query(int x) {
        int ans = 0;
        while (x) {
            ans += s[x];
            x -= LB(x);
        }
        return ans;
    }
}t[2];// 0 to left 1 to right
```

```
41
42 int main() {
43     ios :: sync_with_stdio(0);
44     cin >> s;
45     int n = strlen(s), ans = 0;
46     REP(i, n) s[i] -= 'A';
47     LL res = 0;
48     REP(i, n) ans ^= (1 << s[i]);
49     if (ans & (ans - 1)) {
50         cout << -1 << endl;
51     }
52     else {
53         key = -1;
54         priority_queue<PII> max_index;
55         REP(i, n) {
56             pos[int(s[i])].push_back(i);
57         }
58         if (ans) {
59             int id = __builtin_ctz(ans);
60             key = pos[id][pos[id].size() / 2];
61         }
62         REP(i, 26) if (pos[i].size() > 1) {
63             max_index.push({pos[i].back(), i});
64         }
65         int left = 0, right = n - 1;
66         int head = 0, tail = n - 1;
67         while (left < right) {
68             modify(left, right);
69             int a = max_index.top().second; max_index.pop();
70             int x = pos[a].front(), y = pos[a].back();
71             res += x - head + t[0].query(n) - t[0].query(x) - t
    [1].query(x);
72             res += tail - y + t[1].query(y) - t[0].query(n) + t
    [0].query(y);
73             t[0].add(x + 1);
74             t[1].add(y + 1);
75             pos[a].pop_front(), pos[a].pop_back();
76             if (pos[a].size() > 1) max_index.push({pos[a].back
    (), a});
77             left++, right--, modify(left, right);
78             head++, tail--;
79         }
80         cout << res << endl;
81     }
82     return 0;
83 }
```

Listing 6: 325.cc

6 SGU326

NBA 某小组内有 N 支球队, 小组内以及小组间已经进行了若干场比赛。现在给出 N 支球队目前胜利的场数, 还剩多少场没有比 (包括小组内和小组间) 以及小组内任意两支球队之间还剩多少场没有比, 问能否合理安排剩下的所有比赛, 使得球队 1 最后胜利的场数至少和小组内任何一支其他球队一样.($N \leq 20$)

6.1 Solution

所有和球队 1 相关的比赛全让球队 1 赢, 如果此时仍有某支球队胜利的场数大于球队 1, 则已经不可能满足要求. 按如下方法建图: 所有小组内的比赛 i(不包括与球队 1 相关的比赛) 作为一个点并加边 $(s, i, num[i])$, 每支球队 (不包括球队 1) 作为一个点并加边 $(j, t, wins[1] - wins[i])$, 每场比赛向与其关联的两支球队 u, v 连边 $(i, u, \infty), (i, v, \infty)$. 至于其他球队小组间的比赛, 直接让他们输掉就好, 不用管. 若最大流等于 $\sum num[i]$ 则可以满足要求.

6.2 Code

```
#include <bits/stdc++.h>

#define LL long long
#define REP(i, a) REPP(i, 0, (a) - 1)
#define REPP(i, a, b) for (int i = int(a); i <= int(b); i++)
#define MST(a, b) memset(a, b, sizeof(a))

using namespace std;

const int N = 250;
const int M = 1000;
const int INF = 0x3f3f3f3f;

int w[N], r[N], mp[N][N];

struct Edge{
    int y, next, cap;
};

struct MaxFlow{
    int head[N], edge, source, sink, lvl[N], cur[N];
    Edge e[M << 1];

    void init(int S = 0, int T = 1) {
        edge = 1, MST(head, 0);
        source = S, sink = T;
    }

    void _add(int x, int y, int z) {
```

```
30          e[++edge].next = head[x], head[x] = edge, e[edge].y = y
    , e[edge].cap = z;
31      }
32
33      void add(int x, int y, int z) {
34          _add(x, y, z);
35          _add(y, x, z);
36      }
37
38      bool bfs() {
39          MST(lvl, -1), lvl[source] = 0;
40          queue<int> q;
41          q.push(source);
42          while (q.size()) {
43              int x = q.front(); q.pop();
44              for (int go = head[x]; go; go = e[go].next) if (e[
    go].cap > 0) {
45                  int y = e[go].y;
46                  if (lvl[y] < 0) {
47                      lvl[y] = lvl[x] + 1;
48                      q.push(y);
49                  }
50              }
51          }
52          return lvl[sink] >= 0;
53      }
54
55      int dfs(int x, int flow = INF) {
56          if (flow == 0 || x == sink) return flow;
57          int ans = 0, tmp = 0;
58          for (int &go = cur[x]; go; go = e[go].next) if (e[go].
    cap > 0) {
59              int y = e[go].y;
60              if (lvl[y] == lvl[x] + 1 && (tmp = dfs(y, min(flow,
     e[go].cap))) > 0) {
61                  ans += tmp, flow -= tmp;
62                  e[go].cap -= tmp, e[go ^ 1].cap += tmp;
63                  if (flow == 0) return ans;
64              }
65          }
66          return ans;
67      }
68
69      int dinic() {
70          int ans = 0;
71          while (bfs()) {
72              memcpy(cur, head, sizeof(head));
73              ans += dfs(source);
74          }
75          return ans;
```

```
    }
}flow;

int main() {
    ios :: sync_with_stdio(0);
    int n;
    cin >> n;
    REPP(i, 1, n) cin >> w[i];
    REPP(i, 1, n) cin >> r[i];
    int ans = w[1] + r[1];
    REPP(i, 1, n) {
        REPP(j, 1, n) cin >> mp[i][j];
        if (i > 1) {
            r[i] = 0;
            REPP(j, 2, n) {
                r[i] += mp[i][j];
            }
        }
    }
    REPP(i, 2, n) {
        if (w[i] > ans) {
            cout << "NO" << endl;
            return 0;
        }
    }
    flow.init();
    int node = n, sum = 0;
    REPP(i, 2, n) {
        REPP(j, i + 1, n) {
            flow.add(0, ++node, mp[i][j]);
            flow.add(node, i, mp[i][j]);
            flow.add(node, j, mp[i][j]);
            sum += mp[i][j];
        }
    }
    REPP(i, 2, n) flow.add(i, 1, min(ans - w[i], r[i]));
    if (flow.dinic() == sum) {
        cout << "YES" << endl;
    }
    else {
        cout << "NO" << endl;
    }
    return 0;
}
```

Listing 7: 326.cc

7 SGU328

长为 n 的串, 有些被染成黑白,A,B 轮流染未染色的, 相邻不能相同.($N \leq 100000$)

7.1 Solution

SG 博弈, 直接打表找规律, 这里的问题有三类:

1. 左右都被然过的一段区间的 SG $sg = left\ xor\ right\ xor\ 1$
2. 有一个端是在原串的边界, 没有被染色的区间的 SG $sg = length$
3. 两端都没有被染色的区间的 SG $odd\ 1\ even\ 0$

7.2 Code

```
#include <bits/stdc++.h>

#define LL long long
#define REP(i, a) REPP(i, 0, (a) - 1)
#define REPP(i, a, b) for (int i = int(a); i <= int(b); i++)
#define MST(a, b) memset(a, b, sizeof(a))

using namespace std;

const int N = 105;

int sg[2][2][N], sg1[2][N], sg2[N];
int vis[N * 10000];

void init() {
    int now = 0;
    sg[0][0][0] = sg[1][1][0] = 0, sg[0][0][1] = sg[1][1][1] =
    1;
    sg[0][1][0] = sg[1][0][0] = sg[0][1][1] = sg[1][0][1] = 0;
    REPP(length, 2, 100) {
        REP(left, 2) {
            REP(right, 2) {
                now++;
                REPP(pos, 1, length) {
                    if (pos == 1) {
                        vis[sg[left ^ 1][right][length - 1]] =
    now;
                    }
                    else if (pos == length) {
                        vis[sg[left][right ^ 1][length - 1]] =
    now;
                    }
```

```
                    else {
                        REP(col, 2) {
                            vis[sg[left][col][pos - 1] ^ sg[col][right][length - pos]] = now;
                        }
                    }
                }
                int &ans = sg[left][right][length];
                while (ans <= N * 10000 && vis[ans] == now) ans++;
            }
        }
    }
    sg1[0][0] = sg1[1][0] = 0;
    REPP(length, 1, 100) {
        REP(end, 2) {
            now++;
            REPP(pos, 1, length) {
                if (pos == length) {
                    vis[sg1[end ^ 1][length - 1]] = now;
                }
                else {
                    REP(col, 2) {
                        vis[sg1[col][pos - 1] ^ sg[col][end][length - pos]] = now;
                    }
                }
            }
            int &ans = sg1[end][length];
            while (ans <= N * 10000 && vis[ans] == now) ans++;
        }
    }
    sg2[0] = 0;
    REPP(length, 1, 100) {
        now++;
        REPP(pos, 1, length) {
            REP(col, 2) {
                vis[sg1[col][pos - 1] ^ sg1[col][length - pos]] = now;
            }
        }
        int &ans = sg2[length];
        while (ans <= N * 10000 && vis[ans] == now) ans++;
    }
}

int get(int left, int right, int length) {
    return left ^ right ^ 1;
}

```

```
int get1(int end, int length) {
    return length;
}

int get2(int length) {
    return length & 1;
}

int main() {
    string s;
    init();
    //REPP(i, 0, 100) {
    //  cout << i << ' ';
    //  REP(left, 2) {
    //      REP(right, 2) {
    //          cout << sg[left][right][i] << ' ';
    //      }
    //  }
    //  cout << endl;
    //}
    //REPP(i, 0, 100) {
    //  cout << i << ' ';
    //  REP(end, 2) {
    //      cout << sg1[end][i] << ' ';
    //  }
    //  cout << endl;
    //}
    //REPP(i, 0, 100) {
    //  cout << i << ' ' << sg2[i] << endl;
    //}
    int n;
    cin >> n >> s;
    int ans = 0;
    int left = 0, right = n - 1, L = 0, R = 0;
    while (left < n && s[left] == '0') left++;
    while (right >= 0 && s[right] == '0') right--;
    L = left, R = n - right - 1;
    if (L == n) {
        //ans = sg2[n];
        ans = get2(n);
    }
    else {
        s = s.substr(left, right - left + 1);
        n = s.size();
        //ans = sg1[s[0] - '1'][L] ^ sg1[s.back() - '1'][R];
        ans = get1(s[0] - '1', L) ^ get1(s[s.size() - 1] - '1',
 R);
        REP(i, n) {
            if (s[i] != '0') {
                left = s[i] - '1';
```

```
                continue;
            }
            int j = i;
            while (j < n && s[j] == '0') j++;
            right = s[j] - '1';
            ans ^= get(left, right, j - i);
            i = j - 1;
        }
    }
    if (ans) {
        cout << "FIRST" << endl;
    }
    else {
        cout << "SECOND" << endl;
    }
    return 0;
}
```

Listing 8: 328.cc

8 SGU330

给定 A,B, 每次只能加当前数字的某个真约数 (大于 1), 在 500 步内把 A 变到 B.

8.1 Solution

先随便尝试一下: 因为我们希望尽量快 (500 步的限制), 所以会像下面这样,
$10 \to 15 \to 20 \to 30 \to \ldots 100 \to \ldots 1000 \to \ldots 10000 \to \ldots$
大概就得到一个 log 级别的解. 我们审视一下前面随便写的一个东西, 好像是 10 进制, 然后尽量变大, 那么我们取其他进制也是 log 的, 那么我们取什么好呢? 显然是 2 进制, 因为, 我们希望有解的一定不能漏掉.(自己体会一下).

8.2 Code

```
#include <bits/stdc++.h>

#define REP(i, a) REPP(i, 0, (a) - 1)
#define REPP(i, a, b) for (int i = int(a); i <= int(b); i++)
#define MST(a, b) memset(a, b, sizeof(a))
#define LL long long

using namespace std;

vector<LL> ans;

void modify(LL &x, int ty) {
```

```
    if (x & 1) {
        ans.push_back(x);
        int find = 0;
        for (int i = 3; 1LL * i * i <= x; i += 2) {
            if (x % i == 0) {
                find = i;
                break;
            }
        }
        if (find) {
            x += ty * find;
        }
    }
}

bool dfs(LL now, LL target) {
    ans.push_back(now);
    if (now > target || now % 2 || target % 2) {
        return 0;
    }
    else if (now == target) {
        return 1;
    }
    else {
        LL tmp = 1LL << __builtin_ctzll(now);
        if (tmp == now) tmp >>= 1;
        if (target - now < tmp) {
            tmp = 1LL << __builtin_ctzll(target - now);
        }
        return dfs(now + tmp, target);
    }
}

int main() {
    ios :: sync_with_stdio(0);
    LL A, B;
    cin >> A >> B;
    modify(A, 1), modify(B, -1);
    if (dfs(A, B)) {
        sort(ans.begin(), ans.end());
        REP(i, ans.size()) {
            cout << ans[i] << " \n"[i == int(ans.size() - 1)];
        }
    }
    else {
        cout << "Impossible" << endl;
    }

    return 0;
```

```
}
```

Listing 9: 330.cc

9 SGU337

长度为 N 的循环串, 让你选出最长的字典序最小的子串, 使得他两半差异不超过 K.

9.1 Solution

这题告诉我们容错匹配是存在 naive 的 $O(N^2)$ 的算法的. 直接做 dp 就好了. 从后往前倒着推. 记录失配的位置和个数.

9.2 Code

```
#include <bits/stdc++.h>

#define LL long long
#define REP(i, a) REPP(i, 0, (a) - 1)
#define REPP(i, a, b) for (int i = int(a); i <= int(b); i++)
#define MST(a, b) memset(a, b, sizeof(a))

using namespace std;

const int N = 4005;
short dp[N][N];
string s, ans;

int main() {
    ios :: sync_with_stdio(0);
    int n, k;
    cin >> k >> s;
    s = s + s;
    n = s.size();
    queue<int> pos;
    REPP(dif, 1, n - 1) {
        while (pos.size()) pos.pop();
        int end = n, b = n - 1, a = b - dif;
        while (a >= 0) {
            if (s[a] != s[b]) {
                pos.push(b);
            }
            while (int(pos.size()) > k) {
                end = pos.front();
                pos.pop();
            }
            dp[a][b] = end - b;
```

```
            a--, b--;
        }
    }
    int mx = -1;
    REP(i, n) {
        REPP(j, i + 1, n - 1) {
            //wrong j - i not j - i + 1
            if (dp[i][j] >= j - i && (j - i) * 2 <= n / 2) {
                //cout << i << ' ' << j << ' ' << mx << endl;
                if (j - i > mx) {
                    mx = j - i;
                    ans = s.substr(i, 2 * (j - i));
                }
                else if (j - i == mx) {
                    ans = min(ans, s.substr(i, 2 * (j - i)));
                }
            }
        }
    }
    cout << ans << endl;
    return 0;
}
```

Listing 10: 337.cc

10 SGU339

250000 操作, 加区间, 减区间 (不存在就跳过此操作), 每次加区间的时候回答, 现在有多少还存在的区间被包含在要加的区间内部 (边界可以重合). 保证任何时刻存在的区间不会超过 1000.

10.1 Solution

这题我傻逼了, 既然同时最多 1000 个区间, 那么直接暴力 for 好了, 复杂度 $O(250000 * 1000)$, 可以过, 然后我并没有想到, 看了又姐代码之后泪奔了, 我的做法没有考虑到 1000 这个条件也可以做. 直接上树套树, 实际上是树状数组套 treap. 我们把左端点的值种在位置在右端点的 treap 上, 然后删除就是直接删, 询问就是问区间 [l,r] 间的 treap 上有多少点是大于等于 l 的. 嗯, 我是傻逼. 写了 180 行.$O(N \log N \log N)$

10.2 Code

```
#include <bits/stdc++.h>

#define REP(i, a) REPP(i, 0, (a) - 1)
#define REPP(i, a, b) for (int i = int(a); i <= int(b); i++)
#define MST(a, b) memset(a, b, sizeof(a))
```

```cpp
#define LB(x) ((x) & (-(x)))

using namespace std;

const int N = 5e5 + 5;

struct Node{
    Node *l, *r;
    int value;
    short size;

    Node* update() {
        size = l->size + 1 + r->size;
        return this;
    }
}bar[N * 6], *foo, *null, *rt[N];

#define PNN pair<Node*, Node*>

void init(int tot) {
    null = foo = bar;
    null->l = null->r = null;
    REPP(i, 1, tot) rt[i] = null;
    foo++;
}

Node* New_Node(int x) {
    return new (foo++) (Node) {null, null, x, 1};
}

bool gen(int a, int b) {
    return rand() % (a + b) < a;
}

PNN split(Node *u, int s) {
    if (u == null) return {null, null};
    if (u->l->size >= s) {
        PNN res = split(u->l, s);
        u->l = res.second;
        return {res.first, u->update()};
    }
    else {
        PNN res = split(u->r, s - u->l->size - 1);
        u->r = res.first;
        return {u->update(), res.second};
    }
}

Node* merge(Node *a, Node *b) {
    if (a == null) return b;
```

```
    if (b == null) return a;
    if (gen(a->size, b->size)) {
        a->r = merge(a->r, b);
        return a->update();
    }
    else {
        b->l = merge(a, b->l);
        return b->update();
    }
}

pair<int, bool> find(Node *u, int num) {
    if (u == null) return {0, 0};
    else {
        if (u->value > num) {
            return find(u->l, num);
        }
        else {
            pair<int, bool> tmp = find(u->r, num);
            tmp.first += u->l->size + 1;
            tmp.second |= u->value == num;
            return tmp;
        }
    }
}

void insert(Node *&u, Node *a) {
    int tmp = find(u, a->value).first;
    PNN res = split(u, tmp);
    u = a;
    u = merge(res.first, u);
    u = merge(u, res.second);
}

void erase(Node *&u, int num) {
    pair<int, bool> tmp = find(u, num);
    if (tmp.second) {
        PNN res = split(u, tmp.first - 1);
        PNN res2 = split(res.second, 1);
        u = merge(res.first, res2.second);
    }
}

int type[N], L[N], R[N], tot, n;
char s[3];

int query(int pos, int num) {
    int ans = 0;
    while (pos) {
        ans += rt[pos]->size - find(rt[pos], num - 1).first;
```

```
106         pos -= LB(pos);
107     }
108     return ans;
109 }
110 
111 int count(int pos, int num) {
112     int ans = 0;
113     while (pos) {
114         ans += find(rt[pos], num).first - find(rt[pos], num -
    1).first;
115         pos -= LB(pos);
116     }
117     return ans;
118 }
119 
120 void insert(int pos, int num) {
121     while (pos <= tot) {
122         Node* tmp = New_Node(num);
123         insert(rt[pos], tmp);
124         pos += LB(pos);
125     }
126 }
127 
128 void erase(int pos, int num) {
129     int tmp = count(pos, num) - count(pos - 1, num);
130     if (tmp) {
131         while (pos <= tot) {
132             erase(rt[pos], num);
133             pos += LB(pos);
134         }
135     }
136 }
137 
138 int main() {
139     //freopen("tmp.in", "r", stdin);
140     vector<int> num;
141     int x, y;
142     while(scanf("%s%d%d", s, &x, &y) > 0) {
143         n++;
144         num.push_back(x), num.push_back(y);
145         L[n] = x, R[n] = y, type[n] = s[0] == '+';
146     }
147     sort(num.begin(), num.end());
148     num.resize(unique(num.begin(), num.end()) - num.begin());
149     tot = num.size();
150     init(tot);
151     REPP(i, 1, n) {
152         L[i] = lower_bound(num.begin(), num.end(), L[i]) - num.
    begin() + 1;
```

```
153         R[i] = lower_bound(num.begin(), num.end(), R[i]) - num.
    begin() + 1;
154     }
155     REPP(i, 1, n) {
156         if (type[i] == 1) {
157             printf("%d\n", query(R[i], L[i]) - query(L[i] - 1,
    L[i]));
158             insert(R[i], L[i]);
159         }
160         else {
161             erase(R[i], L[i]);
162         }
163     }
164 
165     return 0;
166 }
```

Listing 11: 339v1.cc

```
1  // SGU 339 -- Segments
2  #include <cstdio>
3  #include <cstring>
4  #include <vector>
5  #include <utility>
6  #include <algorithm>
7  
8  #define foreach(i, v) for (__typeof((v).begin()) i = (v).begin
       (); i != (v).end(); ++ i)
9  
10 std::vector <std::pair <int, int> > segments;
11 
12 int main() {
13     char operation[2];
14     int l, r;
15     while (scanf("%s%d%d", operation, &l, &r) == 3) {
16         if (*operation == '+') {
17             int total = 0;
18             foreach (iter, segments) {
19                 total += l <= iter->first && iter->second <= r;
20             }
21             segments.push_back(std::make_pair(l, r));
22             printf("%d\n", total);
23         } else {
24             segments.erase(std::find(segments.begin(), segments
    .end(), std::make_pair(l, r)));
25             //叉姐这里写错了,直接erase好像会出问题,因为可能find
    返回end()没找到,然后就RE了,我电脑上是这样,但是SGU上可以AC.
    看到的注意一下.
26         }
27     }
```

```
    return 0;
}
```

Listing 12: 339v2.cc

11 SGU340

模拟题.

11.1 Solution

我会告诉你这题, 变量可以是 vim 这样的? 就是不一定是一个字母表示一个变量. 这题直接用栈.

11.2 Code

```
#include <bits/stdc++.h>

#define LL long long
#define REP(i, a) REPP(i, 0, (a) - 1)
#define REPP(i, a, b) for (int i = int(a); i <= int(b); i++)
#define MST(a, b) memset(a, b, sizeof(a))

using namespace std;

const string oper = "+-*/";
const string flag[7] = {" ", "<sup>", "</sup>", "<sub>", "
    </sub>", "<i>", "</i>"};

bool good(char c) {
    return oper.find(c) != string :: npos;
}

void modify(string &s) {
    string ans, res;
    istringstream sin(s);
    while (sin >> ans) {
        res += ans;
    }
    s = res;
}

string solve(string &s) {
    vector<char> op;
    string ans;
    op.push_back('(');
    int pos = 0, n = s.size();
    while (pos < n) {
```

```
        if (s[pos] == '$' || isspace(s[pos])) {
            ;
        }
        else if (s[pos] == '(' || s[pos] == '{') {
            if (s[pos] == '(') ans += '(';
            op.push_back(s[pos]);
        }
        else if (s[pos] == ')') {
            ans += ')';
            op.pop_back();//'('
        }
        else if (s[pos] == '}') {
            op.pop_back();//'{'
            if (op.back() == '^') ans += flag[2];
            else ans += flag[4];
            op.pop_back();//'^' || '_'
        }
        else if (good(s[pos])) {
            ans += flag[0] + s[pos] + flag[0];
        }
        else if (s[pos] == '^' || s[pos] == '_') {
            op.push_back(s[pos]);
            if (s[pos] == '^') {
                ans += flag[1];
            }
            else {
                ans += flag[3];
            }
        }
        else {
            if (isalpha(s[pos])) {
                string tmp;
                while (pos < n && isalpha(s[pos])) tmp += s[pos
    ++];
                pos--;
                ans += flag[5] + tmp + flag[6];
            }
            else {
                string num;
                while (isdigit(s[pos])) {
                    num += s[pos++];
                }
                ans += num;
                pos--;
            }
            if (op.back() == '^' || op.back() == '_') {
                if (op.back() == '^') {
                    ans += flag[2];
                }
                else {
```

```
                    ans += flag[4];
                }
                op.pop_back();
            }
        }
        pos++;
    }
    return ans;
}

int main() {
    ios :: sync_with_stdio(0);
    //freopen("340.in", "r", stdin);
    string s;
    while (getline(cin, s)) {
        modify(s);
        //cout << s << endl;
        cout << solve(s) << endl;
    }
    return 0;
}
```

Listing 13: 340.cc

12 SGU344

一个各自如果与两个 X 相邻, 就会变成 X. 问最后 X 的个数.

12.1 Solution

bfs, 话说我发现了逗比之星抄袭的题目来源呢.$O(N^2)$. 每个 X, 或者变成 X 的点进队去更新别的格子. 容易知道复杂度是对的, 而且所有应该是 X 的点都会被更新到.

12.2 Code

```
#include <bits/stdc++.h>

#define LL long long
#define REP(i, a) REPP(i, 0, (a) - 1)
#define REPP(i, a, b) for (int i = int(a); i <= int(b); i++)
#define MST(a, b) memset(a, b, sizeof(a))

using namespace std;

const int N = 1005;
char mp[N][N], vis[N][N];
int dx[] = {
```

```
    0, 0, 1, -1
};
int dy[] = {
    1, -1, 0, 0
};

int n, m;
bool good(int x, int y) {
    return x >= 1 && x <= n && y >= 1 && y <= m;
}

int main() {
    ios :: sync_with_stdio(0);
    cin >> n >> m;
    REPP(i, 1, n) {
        cin >> (mp[i] + 1);
    }
    queue<int> q;
    int ans = 0;
    REPP(i, 1, n) {
        REPP(j, 1, m) {
            if (mp[i][j] == 'X' && !vis[i][j]) {
                vis[i][j] = 1;
                q.push(i), q.push(j);
                ans++;
                while (q.size()) {
                    int x = q.front(); q.pop();
                    int y = q.front(); q.pop();
                    REP(dir, 4) {
                        int tx = x + dx[dir];
                        int ty = y + dy[dir];
                        if (!good(tx, ty) || vis[tx][ty])
    continue;
                        REP(r, 4) {
                            int sx = tx + dx[r];
                            int sy = ty + dy[r];
                            //sx = i && sy = j wrong
                            if (!good(sx, sy) || (sx == x && sy
 == y)) continue;
                            if (vis[sx][sy]) {
                                ans++;
                                vis[tx][ty] = 1;
                                q.push(tx), q.push(ty);
                                break;
                            }
                        }
                    }
                }
            }
        }
```

```
    }
    cout << ans << endl;
    return 0;
}
```

Listing 14: 344.cc

13 SGU346

模拟.

13.1 Solution

水题这么多有些不能忍. 就当锻炼手速和耐心, 细心了.

13.2 Code

```
#include <bits/stdc++.h>

#define LL long long
#define REP(i, a) REPP(i, 0, (a) - 1)
#define REPP(i, a, b) for (int i = int(a); i <= int(b); i++)
#define MST(a, b) memset(a, b, sizeof(a))

using namespace std;

int a[7];
string s;

int main() {
    ios :: sync_with_stdio(0);
    REP(i, 7) cin >> a[i];
    cin >> s;
    int ans = 0;
    if (a[0] > 0) {
        while (a[0]) {
            ans += 8;
            a[0]--;
        }
        ans += 27;
        if (s == "RED") {
            ans += 7;
        }
    }
    else {
        if (s == "RED") {
            ans = 34;
        }
```

```
32          else {
33              REP(i, 7) {
34                  ans += (i + 1) * a[i];
35              }
36          }
37      }
38      cout << ans << endl;
39
40      return 0;
41  }
```

Listing 15: 346.cc

14 SGU347

n 个字符串, 求链接之后字典序最小. 应该是最简单的题了.

14.1 Solution

直接按字典序排序是错的, 应该是按 $a + b < b + a$ 这样的比较函数排序.

14.2 Code

```
1  #include <bits/stdc++.h>
2
3  #define REP(i, a) REPP(i, 0, (a) - 1)
4  #define REPP(i, a, b) for (int i = int(a); i <= int(b); i++)
5  #define MST(a, b) memset(a, b, sizeof(a))
6
7  using namespace std;
8
9  const int N = 105;
10
11 string s[N];
12
13 bool cmp(string a, string b) {
14     return a + b < b + a;
15 }
16
17 int main() {
18     ios :: sync_with_stdio(0);
19     int n;
20     cin >> n;
21     REP(i, n) cin >> s[i];
22     sort(s, s + n, cmp);
23     string ans;
24     REP(i, n) ans += s[i];
25     cout << ans << endl;
```

```
26
27      return 0;
28  }
```

Listing 16: 347.cc

15 SGU350

B 由 A 中两个不同元素抑或得到, 现在给定 B, 求一个合法的 A.($M \leq 100$),A 保证不存在两个或以上元素抑或之后是 0.

15.1 Solution

显然我们可以通过只选 B 中的数和另外一个数就构造出 A. 就是让你找一个大小是 $n-1$ 的团. 然后搭配一个 x 即可, 注意这里 x 只能是 0. 因为题目要求了 A 有特殊性. 如果取其他, 你都是可能得到一个能产生 B 但是不满足条件的 A.

15.2 Code

```cpp
#include <bits/stdc++.h>

#define LL long long
#define REP(i, a) REPP(i, 0, (a) - 1)
#define REPP(i, a, b) for (int i = int(a); i <= int(b); i++)
#define MST(a, b) memset(a, b, sizeof(a))

using namespace std;
const int N = 105;
int ans[N], id[N], a[N], b[N];
bool use[N];
vector<int> mp[N][N];
map<int, int> g;

int main() {
    int m, n;
    cin >> m;
    n = int(sqrt(2 * m + 0.5)) + 1;
    assert(n * (n - 1) / 2 == m);
    REP(i, m) cin >> a[i], assert(a[i] > 0);
    REP(i, m) g[a[i]] = i;
    REP(i, m) {
        REPP(j, i + 1, m - 1) {
            int tmp = a[i] ^ a[j];
            if (g.count(tmp)) {
                tmp = g[tmp];
                mp[i][j].push_back(tmp);
                mp[j][i].push_back(tmp);
```

```
            }
        }
    }
    REP(i, m) {
        REPP(j, i + 1, m - 1) if (mp[i][j].size()) {
            REP(k, m) {
                if (k != mp[i][j][0] && mp[i][k].size() && mp[j
][k].size()) {
                    mp[i][j].push_back(k);
                    mp[j][i].push_back(k);
                }
            }
        }
    }
    REP(i, m) {
        REPP(j, i + 1, m - 1) {
            assert(a[i] != a[j]);
        }
    }
    ans[0] = 0, ans[1] = a[0] ^ ans[0];
    int start = -1, cnt = 2;
    REP(i, m) {
        if (mp[0][i].size()) {
            id[cnt] = i;
            ans[cnt++] = a[i] ^ ans[0];
            int id = g[a[0] ^ a[i]];
            use[id] = 1;
            start = i;
            break;
        }
    }
    //REP(i, m) {
    //  REP(j, m) {
    //      cout << mp[i][j].size() << ' ';
    //  }
    //  cout << endl;
    //}
    while (cnt < n) {
        REP(i, mp[0][start].size()) {
            int y = mp[0][start][i];
            if (use[y]) continue;
            int good = 1;
            REP(i, cnt) {
                if (!mp[id[i]][y].size()) {
                    good = 0;
                    break;
                }
            }
            if (!good) continue;
            id[cnt] = y;
```

```
78              ans[cnt++] = ans[0] ^ a[y];
79              use[y] = 1;
80              REP(i, cnt - 1) {
81                  use[g[a[id[i]] ^ a[id[cnt - 1]]]] = 1;
82              }
83              start = y;
84              break;
85          }
86      }
87      vector<int> A, B;
88      REP(i, m) A.push_back(a[i]);
89      REP(i, n) {
90          REPP(j, i + 1, n - 1) {
91              B.push_back(ans[i] ^ ans[j]);
92          }
93      }
94      sort(A.begin(), A.end());
95      sort(B.begin(), B.end());
96      assert(A == B);
97      REP(i, n) {
98          cout << ans[i] << " \n"[i == n - 1];
99      }
100     return 0;
101 }
```

Listing 17: 350.cc

16 SGU353

模拟.

16.1 Solution

技巧:$a/b + (a \bmod b > 0) = (a + b - 1)/b$

16.2 Code

```
1  #include <bits/stdc++.h>
2
3  #define LL long long
4  #define REP(i, a) REPP(i, 0, (a) - 1)
5  #define REPP(i, a, b) for (int i = int(a); i <= int(b); i++)
6  #define MST(a, b) memset(a, b, sizeof(a))
7
8  using namespace std;
9
10 int main() {
11     ios :: sync_with_stdio(0);
```

```
    int n, k1, k2, p1, p2, p3;
    cin >> n >> k1 >> k2 >> p1 >> p2 >> p3;
    if (n < p1) {
        cout << 0 << endl;
    }
    else {
        int ans = k1;
        n -= p1;
        int tot = 0;
        while (n > 0 && tot < k2) {
            ans++, tot++;
            n -= p2;
        }
        if (n > 0) ans += (n + p3 - 1) / p3;
        cout << ans << endl;
    }

    return 0;
}
```

Listing 18: 353.cc

17 SGU355

给 $1 \to n$ 染色, 保证成倍数关系的数字不能同色.

17.1 Solution

额, 观察一下, 发现 $2, 4, 8, 16, \ldots$ 两两不能同色. 提示我们直接按因子个数染色. 那么染成相同颜色的数, 一定不能整除. 因为至少有一个质因此不一样. 满足要求. 你会线性筛预处理因此个数么? $O(N)$

17.2 Code

```
#include <bits/stdc++.h>

#define REP(i, a) REPP(i, 0, (a) - 1)
#define REPP(i, a, b) for (int i = int(a); i <= int(b); i++)
#define MST(a, b) memset(a, b, sizeof(a))

using namespace std;

const int N = 1e4 + 5;
int p[N], cnt[N], vis[N], tot;

void prime() {
    for (int i = 2; i < N; i++) {
        if (!vis[i]) p[tot++] = i, cnt[i] = 1;
```

```
        for (int j = 0; j < tot && i * p[j] < N; j++) {
            vis[i * p[j]] = p[j];
            cnt[i * p[j]] = cnt[i] + 1;
            if (i % p[j] == 0) break;
        }
    }
}

int main() {
    ios :: sync_with_stdio(0);
    prime();
    int n;
    cin >> n;
    int ans = 0;
    REPP(i, 1, n) ans = max(ans, cnt[i] + 1);
    cout << ans << endl;
    REPP(i, 1, n) {
        cout << cnt[i] + 1 << " \n"[i == n];
    }

    return 0;
}
```

Listing 19: 355.cc

18 SGU357

模拟, 最短路.

18.1 Solution

$spfa$.

18.2 Code

```
#include <bits/stdc++.h>

#define LL long long
#define REP(i, a) REPP(i, 0, (a) - 1)
#define REPP(i, a, b) for (int i = int(a); i <= int(b); i++)
#define MST(a, b) memset(a, b, sizeof(a))

using namespace std;

const int N = 100;
int dp[N], vis[N], mp[4][4];

bool good(int x) {
```

```
14      if (x == 0) return mp[3][1];
15      x--;
16      return mp[x / 3][x % 3];
17  }
18
19  void spfa(int st) {
20      queue<int> q;
21      q.push(st), dp[st] = 0, vis[st] = 1;
22      while (q.size()) {
23          int x = q.front(); q.pop();
24          int y = (x + 1) % 100;
25          if (mp[0][3] && dp[y] > dp[x] + 1) {
26              dp[y] = dp[x] + 1;
27              if (!vis[y]) q.push(y), vis[y] = 1;
28          }
29          y = (x + 99) % 100;
30          if (mp[1][3] && dp[y] > dp[x] + 1) {
31              dp[y] = dp[x] + 1;
32              if (!vis[y]) q.push(y), vis[y] = 1;
33          }
34          REP(i, 10) {
35              if (good(i) && dp[i] > dp[x] + 1) {
36                  dp[i] = dp[x] + 1;
37                  if (!vis[i]) q.push(i), vis[i] = 1;
38              }
39          }
40          REPP(a, 1, 9) {
41              REPP(b, 0, 9) {
42                  if (mp[3][0] && good(a) && good(b) && dp[10 * a
       + b] > dp[x] + 3) {
43                      dp[10 * a + b] = dp[x] + 3;
44                      if (!vis[10 * a + b]) q.push(10 * a + b),
      vis[10 * a + b] = 1;
45                  }
46              }
47          }
48          vis[x] = 0;
49      }
50  }
51
52  int main() {
53      ios :: sync_with_stdio(0);
54      REP(i, 2) {
55          REP(j, 4) cin >> mp[i][j];
56      }
57      REP(j, 3) cin >> mp[2][j];
58      REP(j, 2) cin >> mp[3][j];
59      int start, end;
60      cin >> start >> end;
61      MST(dp, 0x3f);
```

```
    spfa(start);
    if (dp[end] == 0x3f3f3f3f) dp[end] = -1;
    cout << dp[end] << endl;
    return 0;
}
```

Listing 20: 357.cc

19 SGU358

模拟.

19.1 Solution

最水的题原来是这个.

19.2 Code

```
#include <bits/stdc++.h>

#define LL long long
#define REP(i, a) REPP(i, 0, (a) - 1)
#define REPP(i, a, b) for (int i = int(a); i <= int(b); i++)
#define MST(a, b) memset(a, b, sizeof(a))

using namespace std;

int main() {
    ios :: sync_with_stdio(0);
    vector<int> a[3], b;
    REP(i, 3) {
        REP(j, 3) {
            int x;
            cin >> x;
            a[i].push_back(x);
        }
        sort(a[i].begin(), a[i].end());
        b.push_back(a[i][1]);
    }
    sort(b.begin(), b.end());
    cout << b[1] << endl;

    return 0;
}
```

Listing 21: 358.cc

20 SGU359

20.1 Solution

20.2 Code

21 总结

总结一下目前做到的一些好题:321, 326, 330, 337, 339, 340, 350.