

Contents

1	basic	2
1.1	random	2
1.2	hello	2
1.3	debug	2
2	data structure	2
2.1	sparse table	2
2.2	range set	2
2.3	pbds	2
2.4	treap	2
2.5	bitset	3
3	graph	3
3.1	boruvka	3
3.2	cut bcc	3
3.3	spfa	4
3.4	tarjan	4
3.5	cut	4
3.6	2SAT	4
3.7	bridge	5
3.8	prim	5

3.9	euler-cycle	5
3.10	bridge bcc	5
math		5
4.1	primes er	5
4.2	factorize	5
4.3	fraction	6
4.4	primes linear	6
4.5	big number	6
4.6	milller rabin	7
4.7	math block	7
4.8	count prime	7
other		7
5.1	Geometry	7
5.2	subset	8
5.3	digit dp	8
5.4	mcmf	8
5.5	dinic	8
string		9
6.1	manacher	9
6.2	string trie	9
6.3	kmp	9
6.4	hash	9
6.5	xor trie	10
6.6	z	10
tree		10
7.1	hld	10

1 basic

1.1 random

```
1 mt19937 seed(chrono::steady_clock::now().time_since_epoch().
  count());
2 inline int rnd(int l, int r) {
3     return uniform_int_distribution<int>(l, r)(seed);
4 }
5 inline double drnd(double l, double r) {
6     return uniform_real_distribution<double>(l, r)(seed);
7 }
```

1.2 hello

```
1 #pragma GCC optimize("Ofast,inline,unroll-loops")
2 #pragma GCC target("bmi,bmi2,lzcnt,popcnt,avx2")
3 #include <stdio.h>
4 #define windows_system
5 #ifdef windows_system
6     /* For Windows */
7     inline int getchar_unlocked() { return getchar(); }
8     inline void putchar_unlocked(char _c) { putchar(_c); }
9 #endif
10 inline int in() {
11     int re = 0;
12     char c = getchar_unlocked();
13     while (c == ' ' || c == '\n') c = getchar_unlocked();
14     while (c >= '0' && c <= '9') re = (re << 3) + (re << 1) + c -
        '0', c = getchar_unlocked();
15     return re;
16 }
17 inline void out(int x) {
18     char str[20];
19     int pos = 0;
20     do {
21         str[pos++] = x % 10 + '0';
22         x /= 10;
23     } while (x);
24     for (int i = pos - 1; i >= 0; i--) putchar_unlocked(str[i]);
25     putchar_unlocked('\n');
26 }
```

1.3 debug

```
1 void debug() { cout << endl; }
2 template <typename T, typename ...U>
3 void debug(T i, U ...j) { cout << i << ' ', debug(j...); }
4 #define test(x...) debug("[ " + string(x) + "]", x)
```

2 data structure

2.1 sparse table

```
1 const int N = 100005;
2 int a[N];
3 struct Sparse_table {
4     int n;
5     vector<vector<int>> st;
6     Sparse_table(int _n) : n(_n), st(_n + 1, vector<int>(__lg(
        _n) + 1)) {
7         for (int i = 1; i <= n; i++) st[i][0] = a[i];
8         for (int lg = 1; lg <= __lg(n); lg++) {
9             int len = 1 << (lg - 1);
10            for (int i = 1; i + len <= n; i++)
11                st[i][lg] = max(st[i][lg - 1], st[i + len][lg - 1]);
12        }
13    }
14    int query(int l, int r) {
15        int lg = __lg(r - l + 1);
16        int len = 1 << lg;
17        return max(st[l][lg], st[r - len + 1][lg]);
18    }
19 };
```

2.2 range set

```
1 struct RangeSet { // [L, r)
2     set<pii> st;
3     void cut(int x) {
4         auto it = st.lower_bound({x + 1, -1});
5         if (it == st.begin()) return;
6         auto [l, r] = *prev(it);
7         if (l >= x || x >= r) return;
8         st.erase(prev(it));
9         st.insert({l, x});
10        st.insert({x, r});
11    }
12    vector<pii> split(int l, int r) {
13        // remove and return ranges in [L, r)
14        cut(l), cut(r);
15        vector<pii> res;
16        while (true) {
17            auto it = st.lower_bound({l, -1});
18            if (it == st.end() || r <= it->first) break;
19            res.push_back(*it), st.erase(it);
20        }
21        return res;
22    }
23    void insert(int l, int r) {
24        // add a range [L, r), [L, r) not in st
25        auto it = st.lower_bound({l, r});
26        if (it != st.begin() && prev(it)->second == l)
27            l = prev(it)->first, st.erase(prev(it));
28        if (it != st.end() && r == it->first)
29            r = it->second, st.erase(it);
30        st.insert({l, r});
31    }
32    bool count(int x) {
33        auto it = st.lower_bound({x + 1, -1});
34        return it != st.begin() && prev(it)->first <= x && x < prev
            (it)->second;
35    }
36 };
```

2.3 pbds

```
1 #include <bits/stdc++.h>
2 #include <ext/pb_ds/assoc_container.hpp>
3 #include <ext/pb_ds/tree_policy.hpp>
4 using namespace __gnu_pbds;
5 #define int long long
6 using namespace std;
7 void solve() {
8     tree<int, null_type, greater<int>, rb_tree_tag,
9         tree_order_statistics_node_update> bst;
10    bst.insert(1);
11    int a = *bst.find_by_order(2); // 第 k + 1 大的數
12    int b = bst.order_of_key(2); // 有幾個數 > k
13    // -----
14    tree<int, null_type, less<int>, rb_tree_tag,
15        tree_order_statistics_node_update> bst;
16    bst.insert(1);
17    int a = *bst.find_by_order(2); // 第 k + 1 小的數
18    int b = bst.order_of_key(2); // 有幾個數 < k
19 }
```

2.4 treap

```
1 struct node {
2     int data, sz;
3     node *l, *r;
4     node(int k) : data(k), sz(1), l(0), r(0) {}
5     void up() {
6         sz = 1;
7         if (l) sz += l->sz;
8         if (r) sz += r->sz;
9     }
10    void down() {}
11 };
12 int sz(node *a) { return a ? a->sz : 0; }
13 node *merge(node *a, node *b) {
14     if (!a || !b) return a ? a : b;
15     if (rand() % (sz(a) + sz(b)) < sz(a))
16         return a->down(), a->r = merge(a->r, b), a->up(), a;
17     return b->down(), b->l = merge(a, b->l), b->up(), b;
18 }
19 void split1(node *o, node *&a, node *&b, int k) {
20     if (!o) return a = b = 0, void();
21     o->down();
22     if (o->data <= k)
23         a = o, split1(o->r, a->r, b, k), a->up();
24     else
25         b = o, split1(o->l, a, b->l, k), b->up();
26 }
27 void split2(node *o, node *&a, node *&b, int k) {
28     if (sz(o) <= k) return a = o, b = 0, void();
```

```

29 o->down();
30 if (sz(o->l) + 1 <= k)
31     a = o, split2(o->r, a->r, b, k - sz(o->l) - 1);
32 else
33     b = o, split2(o->l, a, b->l, k);
34 o->up();
35 }
36 node *kth(node *o, int k) {
37     if (k <= sz(o->l)) return kth(o->l, k);
38     if (k == sz(o->l) + 1) return o;
39     return kth(o->r, k - sz(o->l) - 1);
40 }
41 int Rank(node *o, int key) {
42     if (!o) return 0;
43     if (o->data < key)
44         return sz(o->l) + 1 + Rank(o->r, key);
45     else
46         return Rank(o->l, key);
47 }
48 bool erase(node *&o, int k) {
49     if (!o) return 0;
50     if (o->data == k) {
51         node *t = o;
52         o->down(), o = merge(o->l, o->r);
53         delete t;
54         return 1;
55     }
56     node *&t = k < o->data ? o->l : o->r;
57     return erase(t, k) ? o->up(), 1 : 0;
58 }
59 void insert(node *&o, int k) {
60     node *a, *b;
61     split(o, a, b, k),
62     o = merge(a, merge(new node(k, b)));
63 }
64 void interval(node *&o, int l, int r) {
65     node *a, *b, *c;
66     split2(o, a, b, l - 1), split2(b, b, c, r);
67     // operate
68     o = merge(a, merge(b, c));
69 }

```

2.5 bitset

```

1 #pragma GCC target("popcnt")
2 /// 15 = 1111
3 bitset<10> bit(15); /// 宣告 長度 = 10 的 bitset
4 cout << bit.count() << "\n"; /// 回傳 幾個 1
5 cout << bit.none() << "\n"; /// 回傳 是否全為 0
6 cout << bit.any() << "\n"; /// 回傳 是否有 1
7 cout << bit._Find_first() << "\n"; /// 回傳 第一個 bit = 1 的位
8   置
9 int x = 2;
10 cout << bit._Find_next(x) << "\n"; /// 回傳 x 後 第一個 bit = 1
11   的位置

```

3 graph

3.1 boruvka

```

1 #include<bits/stdc++.h>
2 #define int long long
3
4 using namespace std;
5
6 using Graph = vector<vector<int>>>;
7
8 struct DSU {
9     int cc;
10    vector<int> par, sz;
11    vector<set<int>> S;
12
13    DSU(int n = 0) : cc(n), par(n), sz(n, 1), S(n) {
14        for (int i = 0; i < n; i++) {
15            par[i] = i;
16        }
17    }
18    int find(int x) {
19        if (par[x] == x) return x;
20        return par[x] = find(par[x]);
21    }
22    bool merge(int u, int v) {
23        u = find(u), v = find(v);
24        if (u == v) return false;
25        if (sz[u] < sz[v]) swap(u, v);
26        par[v] = u;
27        sz[u] += sz[v];
28        for (int x : S[v]) {

```

```

29         S[u].insert(x);
30     }
31     S[v].clear();
32     cc--;
33     return true;
34 }
35 };
36
37 struct Edge {
38     int u, v;
39     int cost;
40 };
41
42 bool operator<(const Edge &a, const Edge &b) {
43     return a.cost < b.cost;
44 }
45
46 const int INF = 2e18;
47
48 // 對於目前選到的每個集合，選他周圍的最小邊
49 int MST(int n, vector<Edge> edges) {
50     int m = edges.size();
51
52     DSU dsu(n);
53     vector<Edge> nei(n);
54
55     int mst_ans = 0;
56
57     int conti = true;
58     while (conti) {
59         conti = false;
60         fill(nei.begin(), nei.end(), Edge{-1, -1, INF});
61
62         for (auto [u, v, cost] : edges) {
63             int fu = dsu.find(u), fv = dsu.find(v);
64             if (fu == fv) continue;
65
66             nei[fu] = min(nei[fu], {u, v, cost});
67             nei[fv] = min(nei[fv], {v, u, cost});
68         }
69
70         for (int i = 0; i < n; i++) {
71             auto e = nei[i];
72             if (e.u == -1) continue;
73             if (dsu.find(e.u) != dsu.find(e.v)) {
74                 dsu.merge(e.u, e.v);
75                 mst_ans += e.cost;
76                 conti = true;
77             }
78         }
79     }
80     return mst_ans;
81 }
82
83 signed main() {
84     int n, m;
85     cin >> n >> m;
86     vector<Edge> edges;
87
88     int u, v, w;
89     for (int i = 0; i < m; i++) {
90         cin >> u >> v >> w;
91         u--, v--;
92         edges.push_back({u, v, w});
93     }
94
95     cout << MST(n, edges) << "\n";
96 }

```

3.2 cut bcc

```

1 const int N = 200005;
2 vector<int> G[N];
3 int low[N], depth[N];
4 bool vis[N];
5 vector<vector<int>>> bcc;
6 stack<int> stk;
7 void dfs(int v, int p) {
8     stk.push(v);
9     vis[v] = true;
10    low[v] = depth[v] = (p == -1 ? 1 : depth[p] + 1);
11    for (int u : G[v]) {
12        if (u == p) continue;
13        if (!vis[u]) {
14            /// (v, u) 是樹邊
15            dfs(u, v);
16            low[v] = min(low[v], low[u]);
17            /// u 無法在不經過父邊的情況走到 v 的祖先
18            if (low[u] >= depth[v]) {
19                bcc.emplace_back();
20                while (stk.top() != u) {
21                    bcc.back().push_back(stk.top());

```

```

22     stk.pop();
23 }
24 bcc.back().push_back(stk.top());
25 stk.pop();
26 bcc.back().push_back(v);
27 }
28 } else {
29     /// (v, u) 是回邊
30     low[v] = min(low[v], depth[u]);
31 }
32 }
33 }

```

3.3 spfa

```

1 bool SPFA(int s) {
2     vector<int> dis(n, INF);
3     vector<bool> inq(n);
4     vector<int> cnt(n);
5
6     queue<int> q;
7     q.push(s);
8     dis[s] = 0;
9     inq[s] = true;
10
11     while (q.size()) {
12         int u = q.front();
13         q.pop();
14         cnt[u]++;
15
16         if (cnt[u] == n) {
17             /// negative cycle
18             return true;
19         }
20
21         inq[u] = false;
22
23         for (auto [v, w] : G[u]) {
24             if (dis[u] + w < dis[v]) {
25                 dis[v] = dis[u] + w;
26
27                 if (!inq[v]) {
28                     inq[v] = true;
29                     q.push(v);
30                 }
31             }
32         }
33     }
34
35     return false;
36 }

```

3.4 tarjan

```

1 const int N = 5e5 + 5;
2 int n, m, stamp;
3 vector<int> G[N];
4 int dfn[N], low[N];
5 vector<vector<int>> bcc;
6 stack<int> stk;
7
8 void dfs (int u, int par) {
9     dfn[u] = low[u] = ++stamp;
10    stk.push(u);
11    int cnt = 0; /// 兒子個數
12    for (auto v : G[u]) {
13        if (v == par) continue;
14        if (!dfn[v]) {
15            dfs(v, u);
16            low[u] = min(low[u], low[v]);
17            cnt++;
18            if (low[v] >= dfn[u]) { /// 若 u 為割點
19                int now = 0;
20                bcc.push_back({});
21                do {
22                    now = stk.top();
23                    stk.pop();
24                    bcc.back().push_back(now);
25                } while (now != v);
26                bcc.back().push_back(u);
27            }
28        } else {
29            low[u] = min(low[u], dfn[v]);
30        }
31    }
32    /// 特判孤立點
33    if (par == 0 && cnt == 0) {
34        bcc.push_back({u});
35        return;
36    }
37 }

```

3.5 cut

```

1 const int N = 200005;
2 vector<int> G[N];
3 int low[N], depth[N];
4 bool vis[N], cut[N];
5 void dfs(int v, int p) {
6     vis[v] = true;
7     int child = 0;
8     low[v] = depth[v] = (p == -1 ? 1 : depth[p] + 1);
9     for (int u : G[v]) {
10        if (u == p) continue;
11        if (!vis[u]) {
12            /// (v, u) 是樹邊
13            dfs(u, v);
14            child++;
15            low[v] = min(low[v], low[u]);
16            /// u 無法在不經過父邊的情況走到 v 的祖先
17            if (low[u] >= depth[v] && p != -1)
18                cut[v] = true;
19        } else {
20            /// (v, u) 是回邊
21            low[v] = min(low[v], depth[u]);
22        }
23    }
24    /// 根節點有超過 2 個子節點
25    if (p == -1 && child >= 2)
26        cut[v] = true;
27 }

```

3.6 2SAT

```

1 struct TWO_SAT {
2     int n, N;
3     vector<vector<int>> G, rev_G;
4     deque<bool> used;
5     vector<int> order, comp;
6     deque<bool> assignment;
7     void init(int _n) {
8         n = _n;
9         N = _n * 2;
10        G.resize(N + 5);
11        rev_G.resize(N + 5);
12    }
13    void dfs1(int v) {
14        used[v] = true;
15        for (int u : G[v]) {
16            if (!used[u])
17                dfs1(u);
18        }
19        order.push_back(v);
20    }
21    void dfs2(int v, int c1) {
22        comp[v] = c1;
23        for (int u : rev_G[v]) {
24            if (comp[u] == -1)
25                dfs2(u, c1);
26        }
27    }
28    bool solve() {
29        order.clear();
30        used.assign(N, false);
31        for (int i = 0; i < N; ++i) {
32            if (!used[i])
33                dfs1(i);
34        }
35        comp.assign(N, -1);
36        for (int i = 0, j = 0; i < N; ++i) {
37            int v = order[N - i - 1];
38            if (comp[v] == -1)
39                dfs2(v, j++);
40        }
41        assignment.assign(n, false);
42        for (int i = 0; i < N; i += 2) {
43            if (comp[i] == comp[i + 1])
44                return false;
45            assignment[i / 2] = (comp[i] > comp[i + 1]);
46        }
47        return true;
48    }
49    void add_disjunction(int a, bool na, int b, bool nb) { /// A or B
50        /// na means whether a is negative or not
51        /// nb means whether b is negative or not
52        a = 2 * a ^ na;
53        b = 2 * b ^ nb;
54        int neg_a = a ^ 1;
55        int neg_b = b ^ 1;
56        G[neg_a].push_back(b);
57        G[neg_b].push_back(a);
58    }
59 }

```

```

58     rev_G[b].push_back(neg_a);
59     rev_G[a].push_back(neg_b);
60     return;
61 }
62 void get_result(vector<int>& res) {
63     res.clear();
64     for (int i = 0; i < n; i++)
65         res.push_back(assignment[i]);
66 }
67 };
68 /* CSES Giant Pizza
69 3 5
70 + 1 + 2
71 - 1 + 3
72 + 4 - 2
73 - + + + -
74 */
75 int main() {
76     int n, m;
77     cin >> n >> m;
78     TWO_SAT E;
79     E.init(m);
80     char c1, c2;
81     int inp1, inp2;
82     for (int i = 0; i < n; i++) {
83         cin >> c1 >> inp1;
84         cin >> c2 >> inp2;
85         E.add_disjunction(inp1 - 1, c1 == '-' ? inp2 - 1 : c2 == '-')
86             ;
87     }
88     bool able = E.solve();
89     if (able) {
90         vector<int> ans;
91         E.get_result(ans);
92         for (int i : ans)
93             cout << (i == true ? '+' : '-') << ' ';
94         cout << '\n';
95     } else {
96         cout << "IMPOSSIBLE\n";
97     }
98     return 0;
99 }

```

3.7 bridge

```

1  const int N = 200005;
2  vector<int> G[N];
3  int low[N], depth[N];
4  bool vis[N];
5  vector<pair<int, int>> bridge;
6  void dfs(int v, int p) {
7      vis[v] = true;
8      low[v] = depth[v] = (p == -1 ? 1 : depth[p] + 1);
9      for (int u : G[v]) {
10         if (u == p) continue;
11         if (!vis[u]) {
12             /// (v, u) 是樹邊
13             dfs(u, v);
14             low[v] = min(low[v], low[u]);
15         } else {
16             /// (v, u) 是回邊
17             low[v] = min(low[v], depth[u]);
18         }
19     }
20     /// v 在不依靠父邊的情況下永遠沒辦法走到它的祖先
21     /// (root, -1) 不算
22     if (low[v] == depth[v] && p != -1)
23         bridge.push_back({v, p});
24 }

```

3.8 prim

```

1  const int N = 2005;
2  int dis[N][N]; // input
3  int distfromtree[N];
4
5  for (int i = 1; i <= n; i++)
6      distfromtree[i] = dis[1][i];
7  int res = 0;
8  for (int i = 2; i <= n; i++) {
9      int mindist = INT_MAX, next = -1;
10     for (int j = 1; j <= n; j++) {
11         if (distfromtree[j] != 0 && distfromtree[j] < mindist) {
12             mindist = distfromtree[j];
13             next = j;
14         }
15     }
16     res += mindist;
17     for (int j = 1; j <= n; j++)
18         distfromtree[j] = min(distfromtree[j], dis[next][j]);
19 }

```

3.9 euler-cycle

```

1  void dfs(int u) {
2      while(G[u].size()) {
3          auto [v, eid] = G[u].back();
4          G[u].pop_back();
5
6          if (vis[eid]) continue;
7
8          vis[eid] = 1;
9          dfs(v);
10         ans.pb(id);
11     }
12 }

```

3.10 bridge bcc

```

1  const int N = 200005;
2  vector<int> G[N];
3  int low[N], depth[N];
4  bool vis[N];
5  vector<vector<int>> bcc;
6  stack<int> stk;
7  void dfs(int v, int p) {
8      stk.push(v);
9      vis[v] = true;
10     low[v] = depth[v] = (p == -1 ? 1 : depth[p] + 1);
11     for (int u : G[v]) {
12         if (u == p) continue;
13         if (!vis[u]) {
14             /// (v, u) 是樹邊
15             dfs(u, v);
16             low[v] = min(low[v], low[u]);
17         } else {
18             /// (v, u) 是回邊
19             low[v] = min(low[v], depth[u]);
20         }
21     }
22     /// v 在不依靠父邊的情況下永遠沒辦法走到它的祖先
23     if (low[v] == depth[v]) {
24         bcc.emplace_back();
25         while (stk.top() != v) {
26             bcc.back().push_back(stk.top());
27             stk.pop();
28         }
29         bcc.back().push_back(stk.top());
30         stk.pop();
31     }
32 }

```

4 math

4.1 primes er

```

1  // Time Complexity : O(nloglogn)
2  const int N = 200005;
3  bool is_prime[N];
4  vector<int> primes;
5  void build() {
6      fill(is_prime, is_prime + N, true);
7      is_prime[0] = is_prime[1] = false;
8      for (int i = 2; i <= 200000; i++) {
9          if (is_prime[i]) {
10             primes.push_back(i);
11             if (1ll * i * i <= 200000)
12                 for (int j = i * i; j <= 200000; j += i)
13                     is_prime[j] = false;
14         }
15     }
16 }

```

4.2 factorize

```

1  using i64 = long long;
2  using i128 = __int128_t;
3  struct Factorize { // 質因數分解
4      i64 fmul(i64 a, i64 b, i64 p) {
5          return (i128)a * b % p;
6      }
7      i64 fpow(i64 a, i64 b, i64 p) {
8          i64 res = 1;
9          for (; b >= 1; a = fmul(a, a, p))

```

```

10     if (b & 1) res = fmul(res, a, p);
11     return res;
12 }
13 bool check(i64 a, i64 u, i64 n, int t) {
14     a = fpow(a, u, n);
15     if (a == 0 or a == 1 or a == n - 1)
16         return true;
17     for (int i = 0; i < t; i++) {
18         a = fmul(a, a, n);
19         if (a == 1) return false;
20         if (a == n - 1) return true;
21     }
22     return false;
23 };
24 bool isPrime(i64 n) {
25     constexpr array<i64, 7> magic{2, 235, 9375, 28178, 450775,
26         9780504, 1795265022};
27     // for int: {2, 7, 61}
28     if (n < 2) return false;
29     if (n % 2 == 0) return n == 2;
30     i64 u = n - 1;
31     int t = 0;
32     while (u % 2 == 0) u >>= 1, t++;
33     for (auto v : magic) if (!check(v, u, n, t)) return false;
34     return true;
35 }
36 i64 PollardRho(i64 n) { // return non-trivial factor of n
37     if (n % 2 == 0) return 2;
38     i64 x = 2, y = 2, d = 1, p = 1;
39     auto f = [](i64 x, i64 n, i64 p) -> i64 {
40         return ((i128)x * x % n + p) % n;
41     };
42     while (true) {
43         x = f(x, n, p);
44         y = f(f(y, n, p), n, p);
45         d = gcd(abs(x - y), n);
46         if (d != n and d != 1) return d;
47         if (d == n) ++p;
48     }
49 }
50 i64 primeFactor(i64 n) {
51     return isPrime(n) ? n : primeFactor(PollardRho(n));
52 }
53 ftr;
54 void solve() {
55     i64 n;
56     cin >> n;
57     vector<i64> ans;
58     while (n > 1) {
59         i64 p = ftr.primeFactor(n);
60         do {
61             ans.push_back(p);
62             n /= p;
63         } while (n % p == 0);
64     }
65     sort(ans.begin(), ans.end());
66     cout << ans.size() << ' ';
67     for (int i : ans) cout << i << ' ';
68     cout << '\n';
69 }

```

4.3 fraction

```

1 struct frac {
2     int a, b;
3     frac(int _a = 0, int _b = 1) : a(_a), b(_b) {
4         int g = __gcd(a, b);
5         a /= g; b /= g;
6         if (b < 0) { a *= -1; b *= -1; }
7     }
8 };
9 frac operator+(frac x, frac y) { return frac(x.a * y.b + y.a *
10     x.b, x.b * y.b); }
11 frac operator-(frac x, frac y) { return frac(x.a * y.b - y.a *
12     x.b, x.b * y.b); }
13 frac operator*(frac x, frac y) { return frac(x.a * y.a, x.b * y
14     .b); }
15 frac operator/(frac x, frac y) { return frac(x.a * y.b, x.b * y
16     .a); }
17 bool operator>(frac x, frac y) { return x.a * y.b > y.a * x.b;
18 }
19 bool operator<(frac x, frac y) { return x.a * y.b < y.a * x.b;
20 }
21 bool operator>=(frac x, frac y) { return x.a * y.b >= y.a * x.b
22 ; }
23 bool operator<=(frac x, frac y) { return x.a * y.b <= y.a * x.b
24 ; }
25 bool operator==(frac x, frac y) { return x.a * y.b == y.a * x.b
26 ; }
27 frac abs(frac x) { return frac(abs(x.a), abs(x.b)); }
28 ostream& operator<<(ostream &os, const frac& x) { os << x.a <<
29     "/" << x.b; return os; }

```

4.4 primes linear

```

1 // Time Complexity : O(n)
2 // if lpf(i) = i, then it means that i is a prime.
3 // else lpf(i) is the smallest prime factor.
4 // The 199999-th of the prime is 2750131
5 bool is_prime[2750135];
6 int lpf[2750135];
7 vector<int> primes;
8 void init() {
9     fill(is_prime, is_prime + 2750135, true);
10    for (int i = 2; i <= 2750131; i++) {
11        if (is_prime[i]) {
12            primes.push_back(i);
13            lpf[i] = i;
14        }
15        for (int p : primes) {
16            if (p * i > 2750131) break;
17            is_prime[p * i] = false;
18            lpf[p * i] = p;
19            if (i % p == 0) break;
20        }
21    }
22 }

```

4.5 big number

```

1 void init(string &a, string &b) {
2     while (a.size() < b.size()) a = '0' + a;
3     while (a.size() > b.size()) b = '0' + b;
4 }
5 int Compare(string &a, string &b) {
6     if (a < b) {
7         swap(a, b);
8         return -1;
9     }
10    return 1;
11 }
12 bool del(string &a) {
13     if (a[0] == '0') {
14         a.erase(0, 1);
15         return true;
16     }
17     return false;
18 }
19 void del_all_zero(string &a) {
20     while (del(a)) ;
21 }
22 string add(string a, string b) {
23     init(a, b);
24     a = '0' + a; b = '0' + b;
25     for (int i = a.size() - 1; i >= 0; i--) {
26         int p = a[i] - '0', q = b[i] - '0';
27         if (p + q >= 10) {
28             a[i - 1] = '0' + (a[i - 1] - '0' + 1);
29             a[i] = '0' + (p + q - 10);
30         } else {
31             a[i] = '0' + (p + q);
32         }
33     }
34     del(a);
35     return a;
36 }
37 string sub(string a, string b) {
38     init(a, b);
39     if (a == b) return "0";
40     int flag = Compare(a, b);
41     for (int i = a.size() - 1; i >= 0; i--) {
42         int p = a[i] - '0', q = b[i] - '0';
43         if (p < q) {
44             a[i - 1] = '0' + (a[i - 1] - '0' - 1);
45             a[i] = '0' + (p - q + 10);
46         } else {
47             a[i] = '0' + (p - q);
48         }
49     }
50     del_all_zero(a);
51     if (flag == -1) a = "-" + a;
52     return a;
53 }
54 string mul(string a, string b) {
55     string res = "0";
56     init(a, b);
57     Compare(a, b);
58     del_all_zero(b);
59     for (int i = b.size() - 1; i >= 0; i--) {
60         int x = b[i] - '0';
61         if (i != b.size() - 1) a = a + '0';
62         for (int i = 0; i < x; i++)
63             res = add(a, res);
64     }
65     return res;
66 }

```

```

67 string div(string a, string b) {
68     init(a, b);
69     if (a < b) return "0";
70     del_all_zero(b);
71     string res = "0", restmp = "1", tmp = b;
72     for (int i = 0; i < (a.size() - b.size()); i++) {
73         restmp += '0';
74         tmp += '0';
75     }
76     init(a, b);
77     while (a >= b) {
78         init(a, tmp);
79         if (a >= tmp) {
80             a = sub(a, tmp);
81             res = add(res, restmp);
82         } else {
83             restmp.erase(restmp.size() - 1);
84             tmp.erase(tmp.size() - 1);
85         }
86         init(a, b);
87     }
88     return res;
89 }
90 signed main() {
91     string x, y, ch;
92     cin >> x >> ch >> y;
93     string ans;
94     if (ch[0] == '+') ans = add(x, y);
95     else if (ch[0] == '-') ans = sub(x, y);
96     else if (ch[0] == '*') ans = mul(x, y);
97     else ans = div(x, y);
98     cout << ans << "\n";
99     return 0;
100 }

```

4.6 miller rabin

```

1 struct Miller_Rabin {
2     static constexpr uint32_t ws32[3] = {2, 7, 61};
3     static constexpr uint64_t ws64[7] = {2, 325, 9375, 28178,
4         450775, 9780504, 1795265022};
5     template <class uint>
6     static constexpr uint fast_pow(uint b, uint p, uint mod) {
7         using Uint = conditional_t<is_same_v<uint, uint32_t>,
8             uint64_t, __uint128_t>;
9         uint ret{1};
10        for (; p; p >>= 1) {
11            if (p & 1) ret = Uint{ret} * b % mod;
12            b = Uint{b} * b % mod;
13        }
14        return ret;
15    }
16    template <class uint, class = enable_if_t<is_unsigned_v<
17        uint>>>
18    static constexpr bool is_prime(uint n) {
19        const auto& witness = []() -> const auto& {
20            if constexpr (is_same_v<uint, uint32_t>) return
21                ws32;
22            else return ws64;
23        }();
24        using Uint = conditional_t<is_same_v<uint, uint32_t>,
25            uint64_t, __uint128_t>;
26        if (n < 3 || !(n & 1)) return n == 2;
27        auto u = n - 1, t = (uint) __builtin_ctzll(u);
28        u >>= t;
29        for (auto x : witness) {
30            auto a = x % n;
31            if (a == 0 || a == 1 || a == n - 1) continue;
32            auto v = fast_pow(a, u, n);
33            uint i = 0;
34            for (; v != 1 && v != n - 1 && i < t; i++) v = Uint
35                {v} * v % n;
36            if (v != n - 1 && i > 0) return false;
37        }
38        return true;
39    }
40    template <class I, class = enable_if_t<is_integral_v<I>>>
41    constexpr bool operator()(I x) const {
42        if constexpr (is_signed_v<I>) if (x < 0) return false;
43        using U = make_unsigned_t<I>;
44        return is_prime((U)x);
45    }
46 } is_prime;

```

4.7 math block

```

1 // the sum of floor(n / i) for i = 1 ~ n
2 int f(int n) {
3     int ans = 0;
4     int l = 1, r = 0;
5     while (l <= n) {

```

```

6         r = n / (n / l);
7         ans += (r - l + 1) * (n / l);
8         l = r + 1;
9     }
10    return ans;
11 }
12 // the sum of ceil(n / i) for i = 1 ~ n
13 int f(int n) {
14     int ans = 0;
15     int l = 1, r = 0;
16     while (l < n) {
17         r = (n - 1) / ((n - 1) / l);
18         ans += (r - l + 1) * ((n + 1 - 1) / l);
19         l = r + 1;
20     }
21     if (l == n) ans += (n + 1 - 1) / l;
22     return ans;
23 }

```

4.8 count prime

```

1 using namespace std;
2 // Count the number of primes not greater than n
3 int primeCount(const int n) {
4     if (n <= 1) { return 0; }
5     if (n == 2) { return 1; }
6     const int v = sqrtl(n);
7     int s = (v + 1) / 2;
8     vector<int> smalls(s), roughs(s), skip(v + 1);
9     vector<int> larges(s);
10    iota(smalls.begin(), smalls.end(), 0);
11    for (int i = 0; i < s; i++) {
12        roughs[i] = 2 * i + 1;
13        larges[i] = (n / roughs[i] - 1) / 2;
14    }
15    const auto half = [](int n) -> int { return (n - 1) >> 1; };
16    int pc = 0;
17    for (int p = 3; p <= v; p += 2) {
18        if (skip[p]) { continue; }
19        int q = p * p;
20        if (1LL * q * q > n) { break; }
21        skip[p] = true;
22        for (int i = q; i <= v; i += 2 * p) skip[i] = true;
23        int ns = 0;
24        for (int k = 0; k < s; k++) {
25            int i = roughs[k];
26            if (skip[i]) { continue; }
27            int d = 1LL * i * p;
28            larges[ns] = larges[k] - (d <= v ? larges[smalls[d / 2] -
29                pc] : smalls[half(n / d)]) + pc;
30            roughs[ns++] = i;
31        }
32        s = ns;
33        for (int i = half(v), j = v / p - 1 | 1; j >= p; j -= 2) {
34            int c = smalls[j / 2] - pc;
35            for (int e = j * p / 2; i >= e; i--) { smalls[i] -= c; }
36        }
37        pc++;
38    }
39    larges[0] += 1LL * (s + 2 * (pc - 1)) * (s - 1) / 2;
40    for (int k = 1; k < s; k++) { larges[0] -= larges[k]; }
41    for (int l = 1; l < s; l++) {
42        int q = roughs[l];
43        int M = n / q;
44        int e = smalls[half(M / q)] - pc;
45        if (e <= 1) { break; }
46        int t = 0;
47        for (int k = 1 + 1; k <= e; k++) { t += smalls[half(M /
48            roughs[k])]; }
49        larges[0] += t - 1LL * (e - 1) * (pc + 1 - 1);
50    }
51    return larges[0] + 1;
52 }

```

5 other

5.1 Geometry

```

1 double eps = 0.000001;
2 pdd operator + (pdd a, pdd b) { return {a.X + b.X, a.Y + b.Y}; }
3 pdd operator - (pdd a, pdd b) { return {a.X - b.X, a.Y - b.Y}; }
4 double dot(pdd a, pdd b) { return a.X * b.X + a.Y * b.Y; }
5 double cross(pdd a, pdd b) { return a.X * b.Y - a.Y * b.X; }
6 int sign(double x) { return (fabs(x) < eps ? 0 : (x > 0 ? 1 :
7     -1)); }
8 int ori(pdd a, pdd b, pdd c) { return sign(cross(b - a, c - a))
9     ; }
10 /// c is between a and b

```



```

9  bool btw(pdd a, pdd b, pdd c) {
10     return (sign(cross(a - c, b - c)) == 0 && sign(dot(a - c, b -
11         c)) <= 0);
12 }
13 /// ab is touch cd
14 bool banana(pdd a, pdd b, pdd c, pdd d) {
15     int a123 = ori(a, b, c), a124 = ori(a, b, d);
16     int a341 = ori(c, d, a), a342 = ori(c, d, b);
17     if (btw(a, b, c) || btw(a, b, d) || btw(c, d, a) || btw(c, d,
18         b))
19         return true;
20     return (a123 * a124 < 0 && a341 * a342 < 0);
21 }
22 vector<pdd> convex_hull(vector<pdd> pts) {
23     sort(pts.begin(), pts.end());
24     pts.resize(unique(pts.begin(), pts.end()) - pts.begin());
25     vector<pdd> hull(1, pts[0]);
26     for (int t = 0; t < 2; t++) {
27         int sz = hull.size();
28         for (int i = 1; i < (int)pts.size(); i++) {
29             while ((int)hull.size() > sz && ori(hull[(int)hull.size()
30                 - 2], hull.back(), pts[i]) < 0)
31                 hull.pop_back();
32             hull.push_back(pts[i]);
33         }
34         reverse(pts.begin(), pts.end());
35     }
36     hull.pop_back();
37     return hull;
38 }
39 double area(vector<pdd> pt) {
40     double cnt = 0.0;
41     for (int i = 0; i < (int)pt.size() - 1; i++)
42         cnt += cross(pt[i], pt[i + 1]);
43     cnt += cross(pt[(int)pt.size() - 1], pt[0]);
44     return fabs(cnt);
45 }
46 void solve() {
47     int n;
48     cin >> n;
49     vector<pdd> pts;
50     pts.resize(n);
51     for (int i = 0; i < n; i++)
52         cin >> pts[i].X >> pts[i].Y;
53     int ans = area(convex_hull(pts));
54     if (ans % 2 == 1)
55         cout << ans / 2 << ".5\n";
56     else
57         cout << ans / 2 << ".0\n";
58 }

```

5.2 subset

```

1  for (int i = 0; i < (1 << n); i++){
2      for (int m = i; m; m = (m - 1) & i){
3          // m 是 i 的非空子集
4      }
5  }

```

5.3 digit dp

```

1  // 原題：計算 [L, r] 中兩兩相鄰數字不同的數字數量
2  // 注意邊界
3  int dp[20][10]; // i 位數，尾數是 j
4  void init() {
5      for (int j = 0; j <= 9; j++) dp[1][j] = 1;
6      for (int i = 2; i <= 18; i++)
7          for (int now = 0; now <= 9; now++) // 這個位數的數字
8              for (int pre = 0; pre <= 9; pre++) // 上個位數的數字
9                  if (now != pre)
10                     dp[i][now] += dp[i - 1][pre];
11 }
12 int calc(int x) { // 小於等於 x 的數量
13     if (x == 0) return 0;
14     int arr[20] = {}, len = 0;
15     while (x > 0) arr[++len] = x % 10, x /= 10;
16     // 計算 len 位數的數量
17     int cnt = 0, pre = -1;
18     for (int idx = len; idx >= 1; idx--) {
19         for (int now = (idx == len); now < arr[idx]; now++)
20             if (now != pre)
21                 cnt += dp[idx][now];
22         if (arr[idx] == pre) break;
23         pre = arr[idx];
24         if (idx == 1) cnt++; // 特判
25     }
26     // 計算小於 len 位數的數量
27     for (int i = 1; i < len; i++)
28         for (int j = 1; j <= 9; j++)

```

```

29         cnt += dp[i][j];
30     return cnt;
31 }

```

5.4 mcmf

```

1  struct Flow {
2      struct Edge {
3          int u, rc, k, rv;
4      };
5      vector<vector<Edge>> G;
6      vector<int> par, par_eid;
7      Flow(int n) : G(n + 1), par(n + 1), par_eid(n + 1) {}
8      void add(int v, int u, int c, int k) {
9          G[v].push_back({u, c, k, (int)G[u].size()});
10         G[u].push_back({v, 0, -k, (int)G[v].size() - 1});
11     }
12     int spfa(int s, int t) {
13         fill(par.begin(), par.end(), -1);
14         vector<int> dis(par.size(), LONG_LONG_MAX);
15         vector<bool> in_q(par.size(), false);
16         queue<int> Q;
17         dis[s] = 0; in_q[s] = true;
18         Q.push(s);
19         while (!Q.empty()) {
20             int v = Q.front(); Q.pop();
21             in_q[v] = false;
22             for (int i = 0; i < (int)G[v].size(); i++) {
23                 auto [u, rc, k, rv] = G[v][i];
24                 if (rc > 0 && dis[v] + k < dis[u]) {
25                     dis[u] = dis[v] + k;
26                     par[u] = v;
27                     par_eid[u] = i;
28                     if (!in_q[u]) Q.push(u);
29                     in_q[u] = true;
30                 }
31             }
32         }
33         return dis[t];
34     }
35     pair<int, int> flow(int s, int t) {
36         int fl = 0, cost = 0, d;
37         while ((d = spfa(s, t)) < LONG_LONG_MAX) {
38             int cur = LONG_LONG_MAX;
39             for (int v = t; v != s; v = par[v])
40                 cur = min(cur, G[par[v]][par_eid[v]].rc);
41             fl += cur;
42             cost += d * cur;
43             for (int v = t; v != s; v = par[v]) {
44                 G[par[v]][par_eid[v]].rc -= cur;
45                 G[v][G[par[v]][par_eid[v]].rv].rc += cur;
46             }
47         }
48         return {fl, cost};
49     }
50 };

```

5.5 dinic

```

1  struct Flow {
2      struct Edge {
3          // rc : residual capacity
4          int u, rc, rev;
5      };
6      vector<vector<Edge>> G;
7      vector<int> dis, it;
8      Flow(int n) : G(n), dis(n), it(n) {}
9      void add(int v, int u, int c) {
10         G[v].push_back({u, c, sz(G[u])});
11         G[u].push_back({v, 0, sz(G[v]) - 1});
12     }
13     int dfs(int v, int t, int f) {
14         if (v == t || f == 0) return f;
15         for (int &i = it[v]; i < sz(G[v]); i++) {
16             auto &[u, rc, rev] = G[v][i];
17             if (dis[u] != dis[v] + 1) continue;
18             int res = dfs(u, t, min(f, rc));
19             if (res <= 0) continue;
20             rc -= res;
21             G[u][rev].rc += res;
22             return res;
23         }
24         return 0;
25     }
26     int flow(int s, int t) {
27         int ans = 0;
28         for (int l = 30; l >= 0; l--) while (true) {
29             fill(all(dis), INT_MAX);
30             queue<int> Q;
31             Q.push(s);
32             dis[s] = 0;

```



```

33 while (!Q.empty()) {
34     int v = Q.front(); Q.pop();
35     for (auto [u, rc, rev] : G[v]) {
36         if ((rc >> 1) <= 0 || dis[u] < INT_MAX) continue;
37         dis[u] = dis[v] + 1;
38         Q.push(u);
39     }
40 }
41 if (dis[t] == INT_MAX) break;
42
43 fill(all(it), 0);
44 while (true) {
45     int res = dfs(s, t, INT_MAX);
46     if (res <= 0) break;
47     ans += res;
48 }
49 }
50 return ans;
51 }
52 };

```

6 string

6.1 manacher

```

1 string s;
2 int extend(int l, int r, int N) {
3     int i = 0;
4     while (l - i >= 0 && r + i < N && s[l - i] == s[r + i])
5         i++;
6     return i;
7 }
8 int Longest_Palindromic_Substring(string &t) {
9     int N = t.length();
10    s.resize(2 * N + 1, '$');
11    for (int i = 0; i < N; i++)
12        s[2 * i + 1] = t[i];
13    N = 2 * N + 1;
14    vector<int> res;
15    res.resize(N, 0);
16    res[0] = 1;
17    for (int i = 1, mid = 0, R = 0; i < N; i++) {
18        int j = mid - (i - mid);
19        int lst = R + 1 - i;
20        if (i > R) {
21            res[i] = extend(i, i, N);
22            mid = i;
23            R = i + res[i] - 1;
24        } else if (res[j] == lst) {
25            res[i] = lst + extend(i - R, i + R, N);
26            mid = i;
27            R = i + res[i] - 1;
28        } else {
29            res[i] = min(res[j], lst);
30        }
31    }
32    int mx = -1, idx = -1;
33    for (int i = 0; i < N; i++) {
34        if (res[i] > mx) {
35            mx = res[i];
36            idx = i;
37        }
38    }
39    cout << "Longest Palindromic Substring Length is " << (mx + 1) / 2 << "\n";
40    for (int i = idx - res[idx] + 1; i <= idx + res[idx] + 1; i++)
41        if (i & 1)
42            cout << s[i];
43    cout << "\n";
44 }

```

6.2 string trie

```

1 struct node {
2     node* link[26] = {nullptr};
3     bool flag = false;
4     bool contain_key(char ch) { return (link[ch - 'a'] != nullptr); }
5     void put(char ch, node* _node) { link[ch - 'a'] = _node; }
6     node* get(char ch) { return link[ch - 'a']; }
7     void set_end() { flag = true; }
8     bool is_end() { return flag; }
9 };
10 struct Trie {
11     node* root;
12     Trie() { root = new node(); }
13     void insert(string s) { /// insert a string
14         node* now = root;

```

```

15         for (int i = 0; i < (int)s.size(); i++) {
16             if (!now->contain_key(s[i])) now->put(s[i], new node());
17             now = now->get(s[i]);
18         }
19         now->set_end();
20     }
21     bool query(string s) { /// return if there is a word of the
22         given string
23         node* now = root;
24         for (int i = 0; i < (int)s.size(); i++) {
25             if (!now->contain_key(s[i])) return false;
26             now = now->get(s[i]);
27         }
28         return now->is_end();
29     }
30     /// return if there is any word that start with the given
31     prefix
32     bool query_prefix(string s) {
33         node* now = root;
34         for (int i = 0; i < (int)s.size(); i++) {
35             if (!now->contain_key(s[i])) return false;
36             now = now->get(s[i]);
37         }
38         return true;
39     }
40 };

```

6.3 kmp

```

1 vector<int> F;
2 void build_failure_function(string &s) {
3     F.clear();
4     F.resize(s.size(), -1);
5     for (int i = 1; i < s.size(); i++) {
6         int j = F[i - 1];
7         while (j != -1 && s[i] != s[j + 1])
8             j = F[j];
9         if (s[i] == s[j + 1])
10            F[i] = j + 1;
11    }
12 }
13 void KMP_matching(string &a, string &b) {
14     /// i -> a 的指針 · j -> b 的指針
15     for (int i = 0, j = -1; i < a.size(); i++) {
16         while (j != -1 && a[i] != b[j + 1]) /// 匹配失敗
17             j = F[j];
18         if (a[i] == b[j + 1]) /// 匹配成功
19             j++;
20         if (j + 1 == b.size()) { /// 找到了 · 當作匹配失敗 重新
21             匹配 找下一個
22             cout << "found a matching start at " << i - j << " \n";
23             j = F[j];
24         }
25    }
26 }

```

6.4 hash

```

1 const int p = 1e6 + 3;
2 const int mod = 1e9 + 7;
3 int hash(string &s) {
4     int res = 0;
5     for (int i = 0; i < s.size(); i++) {
6         res *= p;
7         res += s[i] - 'a' + 1;
8         res %= mod;
9     }
10    return res;
11 }
12 int pow_p[1000005];
13 vector<int> rh;
14 void build(string &s) {
15     rh.resize(s.size() + 1);
16     pow_p[0] = 1;
17     for (int i = 1; i <= s.size(); i++)
18         pow_p[i] = (pow_p[i - 1] * p) % mod;
19     rh[0] = s[0] - 'a' + 1;
20     for (int i = 1; i < s.size(); i++) {
21         rh[i] = rh[i - 1] * p + (s[i] - 'a' + 1);
22         rh[i] %= mod;
23     }
24 }
25 int query(int l, int r) {
26     int res = rh[r] - (l > 0 ? rh[l - 1] * pow_p[r - l + 1] : 0);
27     res = (res % mod + mod) % mod;
28     return res;
29 }

```

6.5 xor trie

```

1 struct node {
2     node* link[2];
3     bool contain_key(int val) { return (link[val] != nullptr); }
4     node* get(int val) { return link[val]; }
5     void put(int val, node* _node) { link[val] = _node; }
6 };
7 struct Trie {
8     node* root;
9     Trie() { root = new node(); }
10    void update(int x) {
11        node* now = root;
12        for (int i = 30; i >= 0; i--) {
13            if (x & (1 << i)) {
14                if (!now->contain_key(1)) now->put(1, new node());
15                now = now->get(1);
16            } else {
17                if (!now->contain_key(0)) now->put(0, new node());
18                now = now->get(0);
19            }
20        }
21    }
22    int query(int x) { /// query MAX XOR with number x
23        node* now = root;
24        int mx = 0;
25        for (int i = 30; i >= 0; i--) {
26            int nowbit = (x >> i) & 1;
27            int target = nowbit ^ 1;
28            if (now->contain_key(target)) {
29                mx += (1 << i);
30                now = now->get(target);
31            } else {
32                now = now->get(nowbit);
33            }
34        }
35        return mx;
36    }
37 };

```

6.6 z

```

1 vector<int> Z;
2 void calculate_Z(string &s) {
3     Z.clear();
4     Z.resize(s.size(), 0);
5     int l = 0, r = 0;
6     for (int i = 1; i < s.size(); i++) {
7         if (i <= r) /// 估算下界
8             Z[i] = min(Z[i - 1], r - i + 1);
9         while (i + Z[i] < s.size() && s[i + Z[i]] == s[Z[i]]) ///
10            暴力檢查 Z(i) 是否可以變更大
11            Z[i]++;
12         if (i + Z[i] - 1 > r) { /// 更新 "看到最右邊的區間 [l, r]"
13             l = i; r = i + Z[i] - 1;
14         }
15     }
16 }
17 void Z_matching(string &a, string &b) {
18     string res = "";
19     int i = 0;
20     for (char ch : b)
21         res[i++] = ch;
22     res[i++] = '_';
23     for (char ch : a)
24         res[i++] = ch;
25     calculate_Z(res);
26     for (int i = 0; i < Z.size(); i++) {
27         if (Z[i] == b.size())
28             cout << "found a matching start at " << i - b.size() - 1
29             << "\n";
30     }
31 }

```

7 tree

7.1 hld

```

1 const int N = 100005;
2 vector<int> G[N];
3 struct HLD {
4     vector<int> pa, sz, depth, mxson, topf, id;
5     int n, idcnt = 0;
6     HLD(int _n) : n(_n), pa(_n + 1), sz(_n + 1), depth(_n + 1),
7         mxson(_n + 1), topf(_n + 1), id(_n + 1) {}
8     void dfs1(int v = 1, int p = -1) {
9         pa[v] = p; sz[v] = 1; mxson[v] = 0;

```

```

9         depth[v] = (p == -1 ? 0 : depth[p] + 1);
10        for (int u : G[v]) {
11            if (u == p) continue;
12            dfs1(u, v);
13            sz[v] += sz[u];
14            if (sz[u] > sz[mxson[v]]) mxson[v] = u;
15        }
16    }
17    void dfs2(int v = 1, int top = 1) {
18        id[v] = ++idcnt;
19        topf[v] = top;
20        if (mxson[v]) dfs2(mxson[v], top);
21        for (int u : G[v]) {
22            if (u == mxson[v] || u == pa[v]) continue;
23            dfs2(u, u);
24        }
25    }
26    /// query 為區間資料結構
27    int path_query(int a, int b) {
28        int res = 0;
29        while (topf[a] != topf[b]) { /// 若不在同一條鍊上
30            if (depth[topf[a]] < depth[topf[b]]) swap(a, b);
31            res = max(res, 011); /// query : l = id[topf[a]], r = id[a]
32            a = pa[topf[a]];
33        }
34        /// 此時已在同一條鍊上
35        if (depth[a] < depth[b]) swap(a, b);
36        res = max(res, 011); /// query : l = id[b], r = id[a]
37        return res;
38    }
39 };

```