
Algos @bandiaoz 2022

for ACM

2022

BANDIAOZ

目录

0.1	DataStruct	1
0.1.1	Chtholly.cpp	1
0.1.2	DSU.cpp	4
0.1.3	LazySegmentTree.cpp	5
0.1.4	Mo.cpp	7
0.1.5	NearestPointPair.cpp	8
0.1.6	PointDivideAndConquer1.cpp	11
0.1.7	PointDivideAndConquer2.cpp	14
0.1.8	Segtree.cpp	17
0.1.9	SegtreeNoneRecursive.cpp	19
0.1.10	SparseTable.cpp	22
0.1.11	TheKthFarPointPair.cpp	23
0.1.12	Trie01.cpp	26
0.1.13	dsu_on_tree.cpp	27
0.1.14	fenwick.cpp	29
0.1.15	fhq-Treap(区间).cpp	29
0.1.16	fhq-Treap.cpp	32
0.1.17	jls 线段树.cpp	35
0.1.18	segment_tree3.cpp	39
0.1.19	主席树.cpp	41
0.1.20	区间覆盖.cpp	43
0.1.21	带权并查集.cpp	46
0.1.22	替罪羊.cpp	47
0.1.23	树剖.cpp	50
0.1.24	笛卡尔树.cpp	55
0.1.25	轻重链剖分.cpp	56
0.2	Geometry	61
0.2.1	Circle.cpp	61
0.2.2	HalfPlane.cpp	67
0.2.3	Line.cpp	69

0.2.4	Point.cpp	72
0.2.5	PolygonAndConvex.cpp	74
0.2.6	Triangle.cpp	79
0.2.7	mygeo.cpp	81
0.3	Graph	99
0.3.1	2sat.cpp	99
0.3.2	Graph.cpp	100
0.3.3	MaxAssignment.cpp	104
0.3.4	Mincost.cpp	108
0.3.5	Tree.cpp	110
0.3.6	dijkstra.cpp	112
0.3.7	dinic.cpp	113
0.3.8	spfa.cpp	115
0.3.9	匈牙利.cpp	116
0.4	Math	117
0.4.1	China.cpp	117
0.4.2	Euler.cpp	118
0.4.3	FFT.cpp	119
0.4.4	Lagrange.cpp	121
0.4.5	Lucas.cpp	122
0.4.6	Miller-Rabin.cpp	125
0.4.7	NTT.cpp	127
0.4.8	basic.cpp	129
0.4.9	binom.cpp	129
0.4.10	exgcd.cpp	131
0.4.11	xor_basis.cpp	133
0.4.12	公式.md	134
0.4.13	区间线性基.cpp	135
0.4.14	取模 gauss.cpp	137
0.4.15	容斥.cpp	138
0.4.16	异或 gauss.cpp	140
0.4.17	斐波那契.cpp	142
0.4.18	求逆元.cpp	144
0.4.19	浮点型 gauss.cpp	145
0.4.20	第二类斯特林数.cpp	146
0.4.21	线性基类.cpp	149
0.4.22	除法分块.cpp	150
0.5	Others	151

0.5.1	BigNum2.cpp	151
0.5.2	Simulated_annealing.cpp	154
0.5.3	Z.cpp	156
0.5.4	bignum.cpp	157
0.5.5	gen.py	162
0.5.6	makestd.cpp	163
0.5.7	pai.py	163
0.5.8	sg 函数.cpp	164
0.5.9	博弈.cpp	165
0.5.10	威佐夫博弈.cpp	166
0.5.11	杜教 BM.cpp	166
0.5.12	欧拉函数.cpp	168
0.6	String	169
0.6.1	AhoCorasick.cpp	169
0.6.2	exkmp.cpp	173
0.6.3	kmp.cpp	174
0.6.4	manacher.cpp	175
0.6.5	后缀数组.cpp	177
0.7	dp	179
0.7.1	数位 dp.cpp	179
0.7.2	最长上升子序列.cpp	179

0.1 DataStruct

0.1.1 Chtholly.cpp

```

1 #include <bits/stdc++.h>
2
3 using namespace std;
4 using ll = long long;
5
6 struct Chtholly {
7     struct node {
8         int l, r;
9         mutable ll v;
10
11         node(int l, int r, ll v) : l(l), r(r), v(v) {}
12         int size() const {
13             return r - l;
14         }
15         bool operator<(const node &A) const {
16             return l < A.l;

```

```

17     }
18 };
19
20 set<node> s;
21 auto insert(int l, int r, ll v) {
22     return s.insert(node(l, r, v));
23 }
24 auto split(int pos) { //拆区间, 将区间分为[l,pos), [pos,r)两段
25     auto it = s.lower_bound(node(pos, -1, 0));
26     if (it != s.end() && it->l == pos) {
27         return it;
28     }
29     --it;
30     int L = it->l, R = it->r;
31     ll V = it->v;
32     s.erase(it);
33     insert(L, pos, V);
34     //返回第二个区间的地址
35     return insert(pos, R, V).first;
36 }
37 void add(int l, int r, ll x) { //区间加
38     for (auto itr = split(r), itl = split(l); itl != itr; ++itl) {
39         itl->v += x;
40     }
41 }
42 void assign_val(int l, int r, ll x) { //区间推平, 全部赋值x
43     auto itr = split(r), itl = split(l); //划分区间, 注意顺序, 否则会引起itl迭代器失效
44     s.erase(itl, itr);
45     insert(l, r, x);
46 }
47 ll ranks(int l, int r, int k) { //区间第k小
48     vector<pair<ll, int>> vp;
49     for (auto itr = split(r), itl = split(l); itl != itr; ++itl) {
50         vp.push_back({itl->v, itl->size()});
51     }
52     sort(vp.begin(), vp.end());
53     for (auto it : vp) {
54         k -= it.second;
55         if (k <= 0) {
56             return it.first;
57         }
58     }
59     assert(false);
60     return -1;
61 }
62 ll sum(int l, int r, int ex, int mod) { //区间幂次和
63     auto powmod = [](ll a, int b, int mod) {
64         ll ans = 1;

```

```

65         for (a %= mod; b; b >>= 1, a = a * a % mod) {
66             if (b & 1) {
67                 ans = ans * a % mod;
68             }
69         }
70         return ans;
71     };
72
73     ll res = 0;
74     for (auto itr = split(r), itl = split(l); itl != itr; ++itl) {
75         res = (res + itl->size() * powmod(itl->v, ex, mod)) % mod;
76     }
77     return res;
78 }
79 };
80
81 const int mod = 1e9 + 7;
82
83 int seed, vmax;
84 int rnd() {
85     int ret = seed;
86     seed = (seed * 7LL + 13) % mod;
87     return ret;
88 }
89
90 int main() {
91     ios::sync_with_stdio(false);
92     cin.tie(nullptr);
93
94     int n, m;
95     cin >> n >> m >> seed >> vmax;
96
97     Chtholly cho;
98     for (int i = 0; i < n; ++i) {
99         int x = rnd() % vmax + 1;
100         cho.insert(i, i + 1, x);
101     }
102
103     while (m--) {
104         int op = rnd() % 4 + 1;
105
106         int l = rnd() % n;
107         int r = rnd() % n;
108         if (l > r) {
109             swap(l, r);
110         }
111         r++;
112

```

```

113     ll x, y;
114     if (op == 3) {
115         x = rnd() % (r - 1) + 1;
116     } else {
117         x = rnd() % vmax + 1;
118     }
119
120     if (op == 4) {
121         y = rnd() % vmax + 1;
122     }
123
124     if (op == 1) {
125         cho.add(l, r, x);
126     } else if (op == 2) {
127         cho.assign_val(l, r, x);
128     } else if (op == 3) {
129         cout << cho.ranks(l, r, x) << "\n";
130     } else {
131         cout << cho.sum(l, r, x, y) << "\n";
132     }
133 }
134
135 return 0;
136 }

```

0.1.2 DSU.cpp

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4  using ll = long long;
5
6  struct DSU {
7      vector<int> f, sz;
8      DSU(int n) : f(n), sz(n, 1) { iota(f.begin(), f.end(), 0); }
9      int findR(int x) { return x == f[x] ? x : f[x] = findR(f[x]); }
10     bool same(int x, int y) { return findR(x) == findR(y); }
11     bool merge(int x, int y) {
12         x = findR(x), y = findR(y);
13         if (x == y) return false;
14         sz[x] += sz[y], f[y] = x;
15         return true;
16     }
17     int size(int x) { return sz[findR(x)]; }
18 };

```


0.1.3 LazySegmentTree.cpp

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4  using ll = long long;
5
6  struct Info {
7      ll val;
8      Info(ll val = 0) : val(val) {}
9      friend Info operator+(const Info &A, const Info &B) {
10         return Info(A.val + B.val);
11     }
12 };
13
14 void apply(Info &a, ll b, int l, int r) {
15     a.val += b * (r - l);
16 }
17
18 void apply(ll &a, ll b, int l, int r) {
19     a += b;
20 }
21
22 template<class Info, class Tag, class Merge = plus<Info>>
23 class LazySegmentTree {
24 private:
25     const int n;
26     const Merge merge{};
27     vector<Info> info; // data of segment tree, 1-index
28     vector<Tag> tag; // lazy tag of segment tree
29
30     /* [x, y) and val: Add val to each element in range of [x, y)
31      * p: The id of subtree, which is an index of vector 'info'.
32      * [l, r): The range of p.
33      */
34     void innerPull(int p) {
35         info[p] = merge(info[p << 1], info[p << 1 | 1]);
36     }
37     void innerApply(int p, const Tag &v, int l, int r) {
38         ::apply(info[p], v, l, r);
39         ::apply(tag[p], v, l, r);
40     }
41     void push(int p, int l, int r) {
42         if (tag[p] != Tag()) {
43             int m = (l + r) / 2;
44             innerApply(p << 1, tag[p], l, m);
45             innerApply(p << 1 | 1, tag[p], m, r);
46             tag[p] = Tag();

```

```

47     }
48 }
49 void innerUpdate(int p, int x, int y, const Tag &v, int l, int r) {
50     if (x <= l && r <= y) {
51         innerApply(p, v, l, r);
52         return;
53     }
54     int m = (l + r) / 2;
55
56     push(p, l, r);
57     if (x < m) innerUpdate(p << 1, x, y, v, l, m);
58     if (y > m) innerUpdate(p << 1 | 1, x, y, v, m, r);
59     innerPull(p);
60 }
61 /* Query the sum-up value of range [x, y). */
62 Info innerQuery(int p, int x, int y, int l, int r) {
63     if (x <= l && r <= y) return info[p];
64     if (x >= r || y <= l) return Info();
65     int m = (l + r) / 2;
66
67     push(p, l, r);
68     return merge(innerQuery(p << 1, x, y, l, m), innerQuery(p << 1 | 1, x, y, m, r));
69 }
70
71 public:
72     LazySegmentTree(int n) : n(n), info(4 << (32 - __builtin_clz(n))), tag(4 << (32 - __builtin_clz(n)
73     ))) {}
74     LazySegmentTree(vector<Info> &init) : LazySegmentTree(init.size()) {
75         function<void(int, int, int)> innerBuild = [&](int p, int l, int r) {
76             if (r - l == 1) {
77                 info[p] = init[l];
78                 return;
79             }
80             int m = (l + r) / 2;
81             innerBuild(p << 1, l, m);
82             innerBuild(p << 1 | 1, m, r);
83             innerPull(p);
84         };
85         innerBuild(1, 0, n);
86     }
87     /* Add val to each element in range of [x, y) */
88     void update(int x, int y, Tag v) {
89         innerUpdate(1, x, y, v, 0, n);
90     }
91     /* Query the sum-up value of range [x, y) */
92     Info query(int x, int y) {
93         return innerQuery(1, x, y, 0, n);
94     }

```

```

94 };
95
96 int main() {
97     ios::sync_with_stdio(false);
98     cin.tie(nullptr);
99
100     int n, m;
101     cin >> n >> m;
102
103     vector<Info> a(n);
104     for (int i = 0; i < n; ++i) {
105         cin >> a[i].val;
106     }
107
108     LazySegmentTree<Info, ll> seg(a);
109     for (int i = 0; i < m; ++i) {
110         ll op, x, y, k;
111         cin >> op >> x >> y;
112         x--;
113         if (op == 1) {
114             cin >> k;
115             seg.update(x, y, k);
116         } else if (op == 2) {
117             cout << seg.query(x, y).val << "\n";
118         }
119     }
120
121     return 0;
122 }
123 // test problem: https://www.luogu.com.cn/problem/P3372

```

0.1.4 Mo.cpp

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4  using ll = long long;
5
6  int main() {
7      ios::sync_with_stdio(false);
8      cin.tie(nullptr);
9
10     int n;
11     cin >> n;
12     vector<int> a(n);
13     for (int i = 0; i < n; ++i) {
14         cin >> a[i];

```

```

15     a[i]--;
16 }
17
18 int q;
19 cin >> q;
20 vector<int> l(q), r(q);
21 for (int i = 0; i < q; ++i) {
22     cin >> l[i] >> r[i];
23     l[i]--;
24 }
25
26 const int B = max(1.0, n / sqrt(q));
27 vector<int> p(q);
28 iota(p.begin(), p.end(), 0);
29 sort(p.begin(), p.end(), [&](int i, int j) {
30     if (l[i] / B == l[j] / B) return r[i] < r[j];
31     else return l[i] < l[j];
32 });
33
34 vector<int> cnt(n);
35 int L = 0, R = 0, res = 0;
36 auto add = [&](int x, int f) {
37     res -= cnt[x] / 2;
38     cnt[x] += f;
39     res += cnt[x] / 2;
40 };
41
42 vector<int> ans(q);
43 for (auto i : p) {
44     while (L > l[i]) add(a[--L], 1);
45     while (R < r[i]) add(a[R++], 1);
46     while (L < l[i]) add(a[L++], -1);
47     while (R > r[i]) add(a[--R], -1);
48     ans[i] = res;
49 }
50
51 for (int i = 0; i < q; ++i) {
52     cout << ans[i] << "\n";
53 }
54
55 return 0;
56 }
57
58 // https://atcoder.jp/contests/abc242/tasks/abc242_g

```

0.1.5 NearestPointPair.cpp

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4  using ll = long long;
5
6  template<typename T, int K = 2>
7  struct KDTree {
8      KDTree(int n) : n(n), lc(n, -1), rc(n, -1), boundary(n, vector<vector<T>>(K, vector<T>(2))){}
9      KDTree(vector<array<T, K>> &st) : KDTree(st.size()) {
10         a = st;
11         function<int(int, int, int)> innerBuild = [&](int l, int r, int div) {
12             if (l >= r) {
13                 return -1;
14             }
15             int mid = (l + r) >> 1;
16             nth_element(a.begin() + l, a.begin() + mid, a.begin() + r, Cmp(div));
17             lc[mid] = innerBuild(l, mid, (div + 1) % K);
18             rc[mid] = innerBuild(mid + 1, r, (div + 1) % K);
19             maintain(mid);
20             return mid;
21         };
22
23         innerBuild(0, n, 0);
24     };
25     void query(int p, T &ans) {
26         innerQuery(0, n, p, ans);
27     }
28 private:
29     const int n;
30     vector<int> lc, rc;
31     vector<vector<vector<T>>> boundary;
32     vector<array<T, K>> a;
33
34     struct Cmp {
35         int div;
36         Cmp(const int &div) : div(div) {}
37         bool operator()(const array<T, K> &A, const array<T, K> &B) {
38             for (int i = 0; i < K; ++i) {
39                 if (A[(i + div) % K] != B[(i + div) % K]) {
40                     return A[(i + div) % K] < B[(i + div) % K];
41                 }
42             }
43             return false;
44         }
45     };
46     bool cmp(const array<T, K> &A, const array<T, K> &B, int div) {
47         Cmp cp(div);
48         return cp(A, B);

```

```

49     }
50     template<typename U> U sqr(U x) { return x * x; }
51     T dis(const array<T, K> &A, const array<T, K> &B) {
52         T ans = 0;
53         for (int i = 0; i < K; ++i) {
54             ans += sqr(A[i] - B[i]);
55         }
56         return ans;
57     }
58     void maintain(int i) {
59         for (int j = 0; j < K; ++j) {
60             boundary[i][j][0] = boundary[i][j][1] = a[i][j];
61             if (lc[i] != -1) {
62                 boundary[i][j][0] = min(boundary[i][j][0], boundary[lc[i]][j][0]);
63                 boundary[i][j][1] = max(boundary[i][j][1], boundary[lc[i]][j][1]);
64             }
65             if (rc[i] != -1) {
66                 boundary[i][j][0] = min(boundary[i][j][0], boundary[rc[i]][j][0]);
67                 boundary[i][j][1] = max(boundary[i][j][1], boundary[rc[i]][j][1]);
68             }
69         }
70     }
71     T fmin(int p, int i) { // the minimum distance to this area
72         // if i == -1, ignore this area when calculating the answer.
73         if (i == -1) {
74             return 1e18;
75         }
76         T ans = 0;
77         for (int j = 0; j < K; ++j) {
78             if (a[p][j] < boundary[i][j][0]) ans += sqr(boundary[i][j][0] - a[p][j]);
79             if (a[p][j] > boundary[i][j][1]) ans += sqr(a[p][j] - boundary[i][j][1]);
80         }
81         return ans;
82     }
83     void innerQuery(int l, int r, int p, T &ans) {
84         if (l >= r) return;
85         int mid = (l + r) >> 1;
86         if (p != mid) {
87             ans = min(ans, dis(a[p], a[mid]));
88         }
89         if (l + 1 == r) return;
90
91         T dl = fmin(p, lc[mid]), dr = fmin(p, rc[mid]);
92         if (dl < ans && dr < ans) {
93             if (dl < dr) {
94                 innerQuery(l, mid, p, ans);
95                 if (dr < ans) {
96                     innerQuery(mid + 1, r, p, ans);

```

```

97         }
98     } else {
99         innerQuery(mid + 1, r, p, ans);
100         if (dl < ans) {
101             innerQuery(l, mid, p, ans);
102         }
103     }
104 } else if (dl < ans) {
105     innerQuery(l, mid, p, ans);
106 } else if (dr < ans) {
107     innerQuery(mid + 1, r, p, ans);
108 }
109 }
110 };
111
112 int main() {
113     ios::sync_with_stdio(false);
114     cin.tie(nullptr);
115
116     int n;
117     cin >> n;
118
119     vector<array<double, 2>> a(n);
120     for (int i = 0; i < n; ++i) {
121         cin >> a[i][0] >> a[i][1];
122     }
123
124     KDTree<double> kdt(a);
125
126     double ans = 2e18;
127     for (int i = 0; i < n; ++i) {
128         kdt.query(i, ans);
129     }
130
131     cout << fixed << setprecision(4) << sqrt(ans) << "\n";
132
133     return 0;
134 }
135
136 // test problem: https://www.luogu.com.cn/problem/P1429

```

0.1.6 PointDivideAndConquer1.cpp

```

1 #include <bits/stdc++.h>
2
3 using namespace std;
4 using ll = long long;

```

```

5
6 template <typename T>
7 struct Fenwick {
8     const int n;
9     vector<T> a;
10    Fenwick(int n) : n(n), a(n) {}
11    void add(int x, T v) {
12        for (int i = x + 1; i <= n; i += i & -i) {
13            a[i - 1] += v;
14        }
15    }
16    // return the sum of [0, x)
17    T sum(int x) {
18        T ans = 0;
19        for (int i = x; i > 0; i -= i & -i) {
20            ans += a[i - 1];
21        }
22        return ans;
23    }
24    // return the sum of [l, r)
25    T rangeSum(int l, int r) {
26        return sum(r) - sum(l);
27    }
28 };
29
30 int main() {
31     ios::sync_with_stdio(false);
32     cin.tie(nullptr);
33
34     int n;
35     cin >> n;
36     vector<vector<pair<int, int>>> g(n);
37     vector<int> w(n - 1);
38     for (int i = 0; i < n - 1; ++i) {
39         int u, v;
40         cin >> u >> v >> w[i];
41         u--, v--;
42         g[u].emplace_back(v, i);
43         g[v].emplace_back(u, i);
44     }
45
46     int k;
47     cin >> k;
48
49     vector<int> sz(n);
50     vector<bool> vis(n);
51     Fenwick<int> fen(k + 1);
52     function<void(int, int, int, int&&)> dfs_rt = [&](int u, int f, int tot, int &rt) {

```



```

53     int maxx = 0;
54     sz[u] = 1;
55     for (auto [v, j] : g[u]) {
56         if (v == f || vis[v]) continue;
57         dfs_rt(v, u, tot, rt);
58         sz[u] += sz[v];
59         maxx = max(maxx, sz[v]);
60     }
61     maxx = max(maxx, tot - sz[u]);
62     if (maxx * 2 <= tot) {
63         rt = u;
64     }
65 };
66
67 function<void(int, int)> dfs_sz = [&](int u, int f) {
68     sz[u] = 1;
69     for (auto [v, j] : g[u]) {
70         if (v == f || vis[v]) continue;
71         dfs_sz(v, u);
72         sz[u] += sz[v];
73     }
74 };
75
76 vector<int> d;
77 function<void(int, int, int)> dfs_dis = [&](int u, int f, int dis) {
78     d.push_back(dis);
79     for (auto [v, j] : g[u]) {
80         if (v == f || vis[v]) continue;
81         dfs_dis(v, u, dis + w[j]);
82     }
83 };
84
85 function<void(int, int, int)> dfs_clear = [&](int u, int f, int dis) {
86     if (dis) fen.add(dis, -1);
87     for (auto [v, j] : g[u]) {
88         if (v == f || vis[v]) continue;
89         dfs_clear(v, u, dis + w[j]);
90     }
91 };
92
93 function<int(int, int)> work = [&](int u, int tot) {
94     int rt = u;
95     dfs_rt(u, -1, tot, rt);
96     dfs_sz(rt, -1);
97     vis[rt] = true;
98
99     int ans = 0;
100    for (auto [v, j] : g[rt]) {

```

```

101         if (vis[v]) continue;
102         d.clear();
103         dfs_dis(v, rt, w[j]);
104         for (auto dd : d) {
105             if (dd <= k) {
106                 ans += fen.sum(k - dd + 1) + 1;
107             }
108         }
109         for (auto dd : d) {
110             fen.add(dd, 1);
111         }
112     }
113     dfs_clear(rt, -1, 0);
114     for (auto [v, j] : g[rt]) {
115         if (vis[v]) continue;
116         ans += work(v, sz[v]);
117     }
118     return ans;
119 };
120
121 cout << work(0, n) << "\n";
122
123 return 0;
124 }
125
126 // test problem: https://www.luogu.com.cn/problem/P4178

```

0.1.7 PointDivideAndConquer2.cpp

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4  using ll = long long;
5
6  int main() {
7      ios::sync_with_stdio(false);
8      cin.tie(nullptr);
9
10     int n, m;
11     cin >> n >> m;
12     vector<vector<pair<int, int>>> g(n);
13     vector<int> w(n);
14     for (int i = 0; i < n - 1; ++i) {
15         int u, v;
16         cin >> u >> v >> w[i];
17         u--, v--;
18         g[u].emplace_back(v, i);

```

```

19     g[v].emplace_back(u, i);
20 }
21
22 vector<int> ans(m), Q(m);
23 for (int i = 0; i < m; ++i) {
24     cin >> Q[i];
25 }
26
27 vector<int> sz(n);
28 vector<bool> vis(n);
29 function<void(int, int, int, int&)> dfs_rt = [&](int u, int f, int tot, int &rt) {
30     int maxx = 0;
31     sz[u] = 1;
32     for (auto [v, j] : g[u]) {
33         if (v == f || vis[v]) continue;
34         dfs_rt(v, u, tot, rt);
35         sz[u] += sz[v];
36         maxx = max(maxx, sz[v]);
37     }
38     maxx = max(maxx, tot - sz[u]);
39     if (maxx * 2 <= tot) {
40         rt = u;
41     }
42 };
43
44 function<void(int, int)> dfs_sz = [&](int u, int f) {
45     sz[u] = 1;
46     for (auto [v, j] : g[u]) {
47         if (v == f || vis[v]) continue;
48         dfs_sz(v, u);
49         sz[u] += sz[v];
50     }
51 };
52
53
54 vector<bool> mpd(10000001);
55 int cnt;
56 vector<int> d(n);
57
58 function<void(int, int, int)> dfs_ans = [&](int u, int f, int dis) {
59     ++cnt;
60     d[u] = dis;
61     for (int i = 0; i < m; ++i) {
62         if (d[u] == Q[i]) {
63             ans[i] = true;
64         } else if (d[u] < Q[i]) {
65             ans[i] |= mpd[Q[i] - d[u]];
66         }

```

```

67     }
68     for (auto [v, j] : g[u]) {
69         if (v == f || vis[v]) continue;
70         dfs_ans(v, u, dis + w[j]);
71     }
72 };
73
74 function<void(int, int, int)> dfs_dis = [&](int u, int f, int flag) {
75     for (int i = 0; i < m; ++i) {
76         if (d[u] <= Q[i]) {
77             mpd[d[u]] = (flag == 1);
78         }
79     }
80     for (auto [v, j] : g[u]) {
81         if (v == f || vis[v]) continue;
82         dfs_dis(v, u, flag);
83     }
84 };
85
86
87 function<void(int, int)> work = [&](int u, int tot) {
88     int rt = u;
89     dfs_rt(u, -1, tot, rt);
90     dfs_sz(rt, -1);
91     vis[rt] = true;
92
93
94     for (auto [v, j] : g[rt]) {
95         if (vis[v]) continue;
96         dfs_ans(v, rt, w[j]);
97         dfs_dis(v, rt, 1);
98     }
99
100     dfs_dis(rt, -1, -1);
101
102     for (auto [v, j] : g[rt]) {
103         if (vis[v]) continue;
104         work(v, sz[v]);
105     }
106 };
107
108 work(0, n);
109
110 for (int i = 0; i < m; ++i) {
111     cout << (ans[i] ? "AYE" : "NAY") << "\n";
112 }
113
114 return 0;

```

115 | }

0.1.8 Segtree.cpp

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4  using ll = long long;
5
6  template<class Info, class Merge = plus<Info>>
7  struct SegmentTree {
8      SegmentTree(int n) : n(n), merge(Merge()), info(4 << (32 - __builtin_clz(n))) {}
9      SegmentTree(vector<Info> init) : SegmentTree(init.size()) {
10         function<void(int, int, int)> build = [&](int p, int l, int r) {
11             if (r - l == 1) {
12                 info[p] = init[l];
13                 return;
14             }
15             int mid = (l + r) / 2;
16             build(p << 1, l, mid);
17             build(p << 1 | 1, mid, r);
18             innerPull(p);
19         };
20         build(1, 0, n);
21     }
22     void modify(int pos, const Info &x) {
23         innerModify(1, 0, n, pos, x);
24     }
25     Info rangeQuery(int l, int r) {
26         return innerRangeQuery(1, 0, n, l, r);
27     }
28 private:
29     const int n;
30     const Merge merge;
31     vector<Info> info;
32     void innerPull(int p) {
33         info[p] = merge(info[p << 1], info[p << 1 | 1]);
34     }
35     void innerModify(int p, int l, int r, int pos, const Info &x) {
36         if (r - l == 1) {
37             info[p] = info[p] + x;
38             return;
39         }
40         int mid = (l + r) / 2;
41         if (pos < mid) {
42             innerModify(p << 1, l, mid, pos, x);

```

```

44     } else {
45         innerModify(p << 1 | 1, mid, r, pos, x);
46     }
47     innerPull(p);
48 }
49 Info innerRangeQuery(int p, int l, int r, int x, int y) {
50     if (l >= y || r <= x) return Info();
51     if (l >= x && r <= y) return info[p];
52     int mid = (l + r) / 2;
53     return merge(innerRangeQuery(p << 1, l, mid, x, y), innerRangeQuery(p << 1 | 1, mid, r, x, y)
54                 );
55 };
56
57 struct Info {
58     int val;
59     Info(int val = 0) : val(val) {}
60     friend Info operator+(const Info &A, const Info &B) {
61         return Info(A.val + B.val);
62     }
63 };
64
65 int main() {
66     ios::sync_with_stdio(false);
67     cin.tie(nullptr);
68
69     int n, m;
70     cin >> n >> m;
71     SegmentTree<Info> seg(n);
72     for (int i = 0; i < n; ++i) {
73         int x;
74         cin >> x;
75         seg.modify(i, x);
76     }
77
78     while (m--) {
79         int op, x, y;
80         cin >> op;
81         if (op == 1) {
82             cin >> x >> y;
83             x--;
84             seg.modify(x, y);
85         } else {
86             cin >> x >> y;
87             x--;
88             cout << seg.rangeQuery(x, y).val << "\n";
89         }
90     }

```

```

91
92     return 0;
93 }
94
95 // test problem: https://www.luogu.com.cn/problem/P3374

```

0.1.9 SegtreeNoneRecursive.cpp

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4  using ll = long long;
5
6  constexpr unsigned ceil_lg(int n) {
7      return n == 0 ? 0 : 32 - __builtin_clz(n - 1);
8  }
9
10 template <typename T> struct Segtree {
11     public:
12     Segtree() : Segtree(0) {}
13     explicit Segtree(int n) : Segtree(vector<typename T::S>(n, T::e())) {}
14     explicit Segtree(const vector<typename T::S>& a) : _n(int(a.size())) {
15         log = ceil_lg(_n);
16         size = 1 << log;
17         d = vector<typename T::S>(2 * size, T::e());
18         for (int i = 0; i < _n; i++) d[size + i] = a[i];
19         for (int i = size - 1; i >= 1; i--) {
20             update(i);
21         }
22     }
23     void set(int p, typename T::S x) {
24         assert(0 <= p && p < _n);
25         p += size;
26         d[p] = x;
27         for (int i = 1; i <= log; i++) update(p >> i);
28     }
29     typename T::S get(int p) const {
30         assert(0 <= p && p < _n);
31         return d[p + size];
32     }
33     typename T::S query(int l, int r) const {
34         assert(0 <= l && l <= r && r <= _n);
35         typename T::S sml = T::e(), smr = T::e();
36         l += size;
37         r += size;
38         while (l < r) {
39             if (l & 1) sml = T::op(sml, d[l++]);
40             if (r & 1) smr = T::op(d[--r], smr);

```

```

40         l >>= 1;
41         r >>= 1;
42     }
43     return T::op(sml, smr);
44 }
45 typename T::S queryAll() const { return d[1]; }
46 template <bool (*f)(typename T::S)> int max_right(int l) const {
47     return max_right(l, [](typename T::S x) { return f(x); });
48 }
49 // r = l or f(op(a[l], ..., a[r - 1])) = true
50 // r = n or f(op(a[l], ..., a[r])) = false
51 template <class F> int max_right(int l, F f) const {
52     assert(0 <= l && l <= _n);
53     assert(f(T::e()));
54     if (l == _n) return _n;
55     l += size;
56     typename T::S sm = T::e();
57     do {
58         while (l % 2 == 0) l >>= 1;
59         if (!f(T::op(sm, d[l]))) {
60             while (l < size) {
61                 l = (2 * l);
62                 if (f(T::op(sm, d[l]))) {
63                     sm = T::op(sm, d[l]);
64                     l++;
65                 }
66             }
67             return l - size;
68         }
69         sm = T::op(sm, d[l]);
70         l++;
71     } while ((l & -l) != 1);
72     return _n;
73 }
74 template <bool (*f)(typename T::S)> int min_left(int r) const {
75     return min_left(r, [](typename T::S x) { return f(x); });
76 }
77 // r = l or f(op(a[l], ..., a[r - 1])) = true
78 // r = n or f(op(a[l - 1], ..., a[r - 1])) = false
79 template <class F> int min_left(int r, F f) const {
80     assert(0 <= r && r <= _n);
81     assert(f(T::e()));
82     if (r == 0) return 0;
83     r += size;
84     typename T::S sm = T::e();
85     do {
86         r--;
87         while (r > 1 && (r % 2)) r >>= 1;

```



```

88         if (!f(T::op(d[r], sm))) {
89             while (r < size) {
90                 r = (2 * r + 1);
91                 if (f(T::op(d[r], sm))) {
92                     sm = T::op(d[r], sm);
93                     r--;
94                 }
95             }
96             return r + 1 - size;
97         }
98         sm = T::op(d[r], sm);
99     } while ((r & -r) != r);
100     return 0;
101 }
102 private:
103     int _n, size, log;
104     vector<typename T::S> d;
105     void update(int k) { d[k] = T::op(d[2 * k], d[2 * k + 1]); }
106 };
107
108 struct SegtreeOP {
109     using S = int;
110     static S e() { return -1; }
111     static S op(const S &x, const S &y) {
112         return max(x, y);
113     }
114 };
115
116 int main() {
117     ios::sync_with_stdio(false);
118     cin.tie(nullptr);
119
120     int n, m;
121     cin >> n >> m;
122     vector<int> a(n);
123     for (int i = 0; i < n; ++i) {
124         cin >> a[i];
125     }
126
127     Segtree<SegtreeOP> seg(a);
128     for (int i = 0; i < m; ++i) {
129         int op;
130         cin >> op;
131
132         if (op == 1) {
133             int x, v;
134             cin >> x >> v;
135             x--;

```

```

136         seg.set(x, v);
137     } else if (op == 2) {
138         int l, r;
139         cin >> l >> r;
140         l--;
141         cout << seg.query(l, r) << "\n";
142     } else {
143         int x, v;
144         cin >> x >> v;
145         x--;
146         cout << seg.max_right(x, [&](int a) { return a < v; }) + 1 << "\n";
147     }
148 }
149
150 return 0;
151 }
152
153 // test problem: https://atcoder.jp/contests/practice2/tasks/practice2\_j
154 // reference: https://atcoder.github.io/ac-library/master/document\_en/segtree.html

```

0.1.10 SparseTable.cpp

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4  using ll = long long;
5
6  // usage:
7  // auto fun = [&](int i, int j) { return min(i, j); };
8  // SparseTable<int, decltype(fun)> st(a, fun);
9  // or:
10 // SparseTable<int> st(a, [&](int i, int j) { return min(i, j); });
11 // __builtin_clz() : Calculate the number of leading zeros
12
13 template <typename T, class F = function<T(const T&, const T&)>>
14 struct SparseTable {
15     int n;
16     vector<vector<T>> mat;
17     F func;
18
19     SparseTable(const vector<T>& a, const F& f) : func(f) {
20         n = static_cast<int>(a.size());
21         int max_log = 32 - __builtin_clz(n);
22         mat.resize(max_log);
23         mat[0] = a;
24         for (int j = 1; j < max_log; j++) {
25             mat[j].resize(n - (1 << j) + 1);

```

```

26         for (int i = 0; i <= n - (1 << j); i++) {
27             mat[j][i] = func(mat[j - 1][i], mat[j - 1][i + (1 << (j - 1))]);
28         }
29     }
30 }
31
32 // return the answer [from, to)
33 T get(int from, int to) const {
34     assert(0 <= from && from <= to && to <= n);
35     int lg = 32 - __builtin_clz(to - from) - 1;
36     return func(mat[lg][from], mat[lg][to - (1 << lg)]);
37 }
38 };

```

0.1.11 TheKthFarPointPair.cpp

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4  using ll = long long;
5
6  template<typename T, int K = 2>
7  struct KDTree {
8      KDTree(int n) : n(n), lc(n, -1), rc(n, -1), boundary(n, vector<vector<T>>(K, vector<T>(2))){}
9      KDTree(vector<array<T, K>> &st) : KDTree(st.size()) {
10         a = st;
11         function<int(int, int, int)> innerBuild = [&](int l, int r, int div) {
12             if (l >= r) {
13                 return -1;
14             }
15             int mid = (l + r) >> 1;
16             nth_element(a.begin() + l, a.begin() + mid, a.begin() + r, Cmp(div));
17             lc[mid] = innerBuild(l, mid, (div + 1) % K);
18             rc[mid] = innerBuild(mid + 1, r, (div + 1) % K);
19             maintain(mid);
20             return mid;
21         };
22
23         innerBuild(0, n, 0);
24     };
25     T query(int k) {
26         priority_queue<T, vector<T>, greater<T>> q;
27         for (int i = 0; i < k; ++i) q.push(0);
28         for (int i = 0; i < n; ++i) {
29             innerQuery(0, n, i, q);
30         }
31         return q.top();

```

```

32     }
33 private:
34     const int n;
35     vector<int> lc, rc;
36     vector<vector<vector<T>>> boundary;
37     vector<array<T, K>> a;
38
39     struct Cmp {
40         int div;
41         Cmp(const int &div) : div(div) {}
42         bool operator()(const array<T, K> &A, const array<T, K> &B) {
43             for (int i = 0; i < K; ++i) {
44                 if (A[(i + div) % K] != B[(i + div) % K]) {
45                     return A[(i + div) % K] < B[(i + div) % K];
46                 }
47             }
48             return false;
49         }
50     };
51     bool cmp(const array<T, K> &A, const array<T, K> &B, int div) {
52         Cmp cp(div);
53         return cp(A, B);
54     }
55     template<typename U> U sqr(U x) { return x * x; }
56     T dis(const array<T, K> &A, const array<T, K> &B) {
57         T ans = 0;
58         for (int i = 0; i < K; ++i) {
59             ans += sqr(A[i] - B[i]);
60         }
61         return ans;
62     }
63     void maintain(int i) {
64         for (int j = 0; j < K; ++j) {
65             boundary[i][j][0] = boundary[i][j][1] = a[i][j];
66             if (lc[i] != -1) {
67                 boundary[i][j][0] = min(boundary[i][j][0], boundary[lc[i]][j][0]);
68                 boundary[i][j][1] = max(boundary[i][j][1], boundary[lc[i]][j][1]);
69             }
70             if (rc[i] != -1) {
71                 boundary[i][j][0] = min(boundary[i][j][0], boundary[rc[i]][j][0]);
72                 boundary[i][j][1] = max(boundary[i][j][1], boundary[rc[i]][j][1]);
73             }
74         }
75     }
76     T fmax(int p, int i) { // the maximum distance to this area
77         // if i == -1, ignore this area when calculating the answer.
78         if (i == -1) {
79             return 0;

```

```

80     }
81     T ans = 0;
82     for (int j = 0; j < K; ++j) {
83         ans += max(sqr(a[p][j] - boundary[i][j][0]), sqr(a[p][j] - boundary[i][j][1]));
84     }
85     return ans;
86 }
87 void innerQuery(int l, int r, int p, priority_queue<T, vector<T>, greater<T>> &q) {
88     if (l >= r) return;
89     int mid = (l + r) >> 1;
90     T tmp = dis(a[p], a[mid]);
91     if (tmp > q.top()) {
92         q.pop();
93         q.push(tmp);
94     }
95     T dl = fmax(p, lc[mid]), dr = fmax(p, rc[mid]);
96     if (dl > q.top() && dr > q.top()) {
97         if (dl > dr) {
98             innerQuery(l, mid, p, q);
99             if (dr > q.top()) {
100                 innerQuery(mid + 1, r, p, q);
101             }
102         } else {
103             innerQuery(mid + 1, r, p, q);
104             if (dl > q.top()) {
105                 innerQuery(l, mid, p, q);
106             }
107         }
108     } else if (dl > q.top()) {
109         innerQuery(l, mid, p, q);
110     } else if (dr > q.top()) {
111         innerQuery(mid + 1, r, p, q);
112     }
113 }
114 };
115
116 int main() {
117     ios::sync_with_stdio(false);
118     cin.tie(nullptr);
119
120     int n, k;
121     cin >> n >> k;
122
123     k *= 2;
124
125     vector<array<ll, 2>> a(n);
126     for (int i = 0; i < n; ++i) {
127         cin >> a[i][0] >> a[i][1];

```

```

128     }
129
130     KDTree<ll> kdt(a);
131
132     cout << kdt.query(k) << "\n";
133
134     return 0;
135 }
136
137 // test problem: https://www.luogu.com.cn/problem/P4357

```

0.1.12 Trie01.cpp

```

1 // 01 Trie find maximal xor sum
2 template <typename T, int B = 30>
3 class Trie01 {
4     using Node = array<int, 2>;
5     vector<Node> ch_;
6     void addNode(int fa, int c) {
7         ch_[fa][c] = ch_.size();
8         ch_.emplace_back(Node());
9     }
10
11 public:
12     Trie01() : ch_(1) {}
13     void insert(T x) {
14         for (int i = B, p = 0; i >= 0; --i) {
15             int c = x >> i & 1;
16             if (ch_[p][c] == 0) addNode(p, c);
17             p = ch_[p][c];
18         }
19     }
20     T getMax(T x) {
21         T res = 0;
22         for (int i = B, p = 0; i >= 0; --i) {
23             int c = x >> i & 1;
24             if (ch_[p][c ^ 1]) {
25                 p = ch_[p][c ^ 1];
26                 res |= 1 << i;
27             } else {
28                 p = ch_[p][c];
29             }
30         }
31         return res;
32     }
33     T getMin(T x) {
34         T res = 0;

```

```

35     for (int i = B, p = 0; i >= 0; --i) {
36         int c = x >> i & 1;
37         if (ch_[p][c]) {
38             p = ch_[p][c];
39         } else {
40             p = ch_[p][c ^ 1];
41             res |= 1 << i;
42         }
43     }
44     return res;
45 }
46 };

```

0.1.13 dsu_on_tree.cpp

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4  using ll = long long;
5
6  int main() {
7      ios::sync_with_stdio(false);
8      cin.tie(nullptr);
9
10     int n;
11     cin >> n;
12     vector<int> a(n);
13     vector<vector<int>> g(n);
14     for (int i = 0; i < n; ++i) {
15         cin >> a[i];
16     }
17     for (int i = 0; i < n - 1; ++i) {
18         int u, v;
19         cin >> u >> v;
20         u--, v--;
21         g[u].push_back(v);
22         g[v].push_back(u);
23     }
24
25     vector<int> fa(n, -1), sz(n, 1);
26     function<void(int)> dfs_son = [&](int u) {
27         if (u > 0) {
28             g[u].erase(find(g[u].begin(), g[u].end(), fa[u]));
29         }
30         for (auto &v : g[u]) {
31             fa[v] = u;
32             dfs_son(v);

```

```

33         sz[u] += sz[v];
34         if (sz[v] > sz[g[u][0]]) {
35             swap(v, g[u][0]);
36         }
37     }
38 };
39
40 dfs_son(0);
41
42 int flag = -1, maxx = 0;
43 vector<int> cnt(n + 1);
44 vector<ll> ans(n);
45 ll sum = 0;
46 function<void(int, int)> count = [&](int u, int val) {
47     cnt[a[u]] += val;
48     if (cnt[a[u]] > maxx) {
49         maxx = cnt[a[u]];
50         sum = a[u];
51     } else if (cnt[a[u]] == maxx) {
52         sum += a[u];
53     }
54     for (auto v : g[u]) {
55         if (v == flag) continue;
56         count(v, val);
57     }
58 };
59
60 function<void(int, bool)> dfs_dsu = [&](int u, bool keep) {
61     // 搞轻儿子及其子树答案删贡献
62     for (auto v : g[u]) {
63         if (v == g[u][0]) continue;
64         dfs_dsu(v, 0);
65     }
66     // 搞重儿子及其子树答案不删贡献
67     if (g[u].size()) {
68         dfs_dsu(g[u][0], true);
69         flag = g[u][0];
70     }
71     // 暴力统计u及其所有轻儿子的贡献合并到刚算出的重儿子信息里
72     count(u, 1);
73     flag = -1;
74     ans[u] = sum;
75     // 把需要删除的贡献删一删
76     if (!keep) {
77         count(u, -1);
78         sum = maxx = 0;
79     }
80 };

```



```

81
82     dfs_dsu(0, false);
83
84     for (int i = 0; i < n; ++i) {
85         cout << ans[i] << " \n"[i == n - 1];
86     }
87
88     return 0;
89 }
90
91 // https://codeforces.com/problemset/problem/600/E

```

0.1.14 fenwick.cpp

```

1  template <typename T>
2  struct Fenwick {
3      const int n;
4      vector<T> a;
5      Fenwick(int n) : n(n), a(n) {}
6      void add(int x, T v) {
7          for (int i = x + 1; i <= n; i += i & -i) {
8              a[i - 1] += v;
9          }
10     }
11     // return the sum of [0, x)
12     T sum(int x) {
13         T ans = 0;
14         for (int i = x; i > 0; i -= i & -i) {
15             ans += a[i - 1];
16         }
17         return ans;
18     }
19     // return the sum of [l, r)
20     T rangeSum(int l, int r) {
21         return sum(r) - sum(l);
22     }
23 };

```

0.1.15 fhq-Treap(区间).cpp

```

1  #include <bits/stdc++.h>
2  #define rep(i, a, n) for (int i = a; i <= n; ++i)
3  #define per(i, a, n) for (int i = n; i >= a; --i)
4  #ifdef LOCAL
5  #include "Print.h"
6  #define de(...) W(' ', #__VA_ARGS__, " ") ==, __VA_ARGS__)
7  #else

```

```

8  #define de(...)
9  #endif
10 using namespace std;
11 typedef long long ll;
12 const int maxn = 1e5 + 5;
13 namespace fhq {
14     #define tr t[root]
15     #define lson t[tr.lc]
16     #define rson t[tr.rc]
17     mt19937 rnd(233);
18     struct node {
19         int lc, rc, val, key, sz;
20         bool tag;
21     } t[maxn];
22     int cnt, Root;
23     // 重新计算以 root 为根的子树大小
24     inline void update(int root) { tr.sz = lson.sz + rson.sz + 1; }
25     // 新建一个权值为val的结点
26     int newNode(int val) {
27         t[++cnt] = {0, 0, val, (int)rnd(), 1, 0};
28         return cnt;
29     }
30     inline void pushdown(int root) {
31         swap(tr.lc, tr.rc);
32         lson.tag ^= 1, rson.tag ^= 1;
33         tr.tag = false;
34     }
35     // 合并成小根堆, 参数保证x树的权值严格小于y树的权值
36     int merge(int x, int y) {
37         if (!x || !y) return x + y;
38         if (t[x].key < t[y].key) {
39             if (t[x].tag) pushdown(x);
40             t[x].rc = merge(t[x].rc, y);
41             update(x); return x;
42         } else {
43             if (t[y].tag) pushdown(y);
44             t[y].lc = merge(x, t[y].lc);
45             update(y); return y;
46         }
47     }
48     // 在以 root 为根的子树内树按值分裂, x树的大小等于k
49     void split_sz(int root, int k, int &x, int &y) {
50         if (!root) x = y = 0;
51         else {
52             if (tr.tag) pushdown(root);
53             if (k <= lson.sz) y = root, split_sz(tr.lc, k, x, tr.lc);
54             else x = root, split_sz(tr.rc, k - lson.sz - 1, tr.rc, y);
55             update(root);

```

```

56     }
57 }
58 void reverse(int l, int r) {
59     int x, y, z;
60     split_sz(Root, l - 1, x, y);
61     split_sz(y, r - l + 1, y, z);
62     t[y].tag ^= 1;
63     Root = merge(merge(x, y), z);
64 }
65 void ldr(int root) {
66     if (!root) return;
67     if (tr.tag) pushdown(root);
68     ldr(tr.lc);
69     printf("%d ", tr.val);
70     ldr(tr.rc);
71 }
72 #undef tr
73 #undef lson
74 #undef rson
75 } // namespace fhq
76 int case_Test() {
77     int n, m;
78     scanf("%d%d", &n, &m);
79     rep(i, 1, n) fhq::Root = fhq::merge(fhq::Root, fhq::newNode(i));
80     while (m--) {
81         int l, r;
82         scanf("%d%d", &l, &r);
83         fhq::reverse(l, r);
84     }
85     fhq::ldr(fhq::Root);
86     return 0;
87 }
88 int main() {
89 #ifdef LOCAL
90     freopen("/Users/chenjinglong/Desktop/cpp_code/in.in", "r", stdin);
91     freopen("/Users/chenjinglong/Desktop/cpp_code/out.out", "w", stdout);
92     clock_t start = clock();
93 #endif
94     int _ = 1;
95     // scanf("%d", &_);
96     while (_--) case_Test();
97 #ifdef LOCAL
98     printf("Time used: %.3lfs\n", (double)(clock() - start) / CLOCKS_PER_SEC);
99 #endif
100     return 0;
101 }

```

0.1.16 fhq-Treap.cpp

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4  using ll = long long;
5
6  template<typename key_t>
7  struct Treap {
8      struct Node {
9          key_t key;
10         int pri;
11         int l, r, sz;
12         Node(key_t a, int b) : key(a), pri(b), l(-1), r(-1), sz(1) {}
13     };
14
15     int root = -1;
16     vector<Node> tree;
17
18     // split by key, the key of x treap less than y treap
19     array<int, 2> split(int pos, key_t key) {
20         if (pos == -1) return {-1, -1};
21
22         if (tree[pos].key <= key) {
23             array<int, 2> res = split(tree[pos].r, key);
24             tree[pos].r = res[0];
25             update(pos);
26             return {pos, res[1]};
27         } else {
28             array<int, 2> res = split(tree[pos].l, key);
29             tree[pos].l = res[1];
30             update(pos);
31             return {res[0], pos};
32         }
33     }
34
35     // split by size, the size of x treap equal to sz
36     array<int, 2> split_sz(int pos, int sz) {
37         if (pos == -1) return {-1, -1};
38
39         if (tree[tree[pos].l].sz + 1 <= sz) {
40             array<int, 2> res = split_sz(tree[pos].r, sz - tree[tree[pos].l].sz - 1);
41             tree[pos].r = res[0];
42             update(pos);
43             return {pos, res[1]};
44         } else {
45             array<int, 2> res = split_sz(tree[pos].l, sz);
46             tree[pos].l = res[1];
47             update(pos);

```

```

47         return {res[0], pos};
48     }
49 }
50 // small root heap, the key of x treap less than y treap
51 int merge(int x, int y) {
52     if (x == -1) return y;
53     if (y == -1) return x;
54
55     if (tree[x].pri > tree[y].pri) {
56         swap(x, y);
57     }
58
59     array<int, 2> res = split(y, tree[x].key);
60     tree[x].l = merge(tree[x].l, res[0]);
61     tree[x].r = merge(tree[x].r, res[1]);
62     update(x);
63     return x;
64 }
65 void update(int pos) {
66     tree[pos].sz = tree[tree[pos].l].sz + tree[tree[pos].r].sz + 1;
67 }
68 int create(key_t key) {
69     mt19937 rng((unsigned int) chrono::steady_clock::now().time_since_epoch().count());
70     int pri = (int)(rng() & ((1ll << 31) - 1));
71     tree.emplace_back(key, pri);
72     return (int)tree.size() - 1;
73 }
74 void insert(int &pos, key_t key) {
75     int o = create(key);
76     array<int, 2> res = split(pos, key);
77     pos = merge(merge(res[0], o), res[1]);
78 }
79 // Return rank with power is key
80 int rank(int &pos, key_t key) {
81     array<int, 2> res = split(pos, key - 1);
82     int rk = (res[0] == -1) ? 1 : tree[res[0]].sz + 1;
83     pos = merge(res[0], res[1]);
84     return rk;
85 }
86 // Return the key of the k largest
87 key_t kth(int &pos, int k) {
88     assert(k <= tree[pos].sz);
89     array<int, 2> res1 = split_sz(pos, k);
90     array<int, 2> res2 = split_sz(res1[0], k - 1);
91     key_t key = tree[res2[1]].key;
92     pos = merge(merge(res2[0], res2[1]), res1[1]);
93     return key;
94 }

```

```

95 // Delete one node that equal to key
96 void erase(int &pos, key_t key) {
97     array<int, 2> res1 = split(pos, key);
98     array<int, 2> res2 = split(res1[0], key - 1);
99
100     if (res2[1] != -1) {
101         res2[1] = merge(tree[res2[1]].l, tree[res2[1]].r);
102     }
103
104     pos = merge(merge(res2[0], res2[1]), res1[1]);
105 }
106 // Return the precursor of key
107 key_t pre(int &pos, key_t key) {
108     array<int, 2> res = split(pos, key - 1);
109     key_t ans = kth(res[0], tree[res[0]].sz);
110     pos = merge(res[0], res[1]);
111     return ans;
112 }
113 // Return the next of key
114 key_t nxt(int &pos, key_t key) {
115     array<int, 2> res = split(pos, key);
116     int ans = kth(res[1], 1);
117     pos = merge(res[0], res[1]);
118     return ans;
119 }
120
121 void insert(key_t x) { insert(root, x); }
122 void erase(int x) { erase(root, x); }
123 int rank(key_t x) { return rank(root, x); }
124 key_t kth(int x) { return kth(root, x); }
125 key_t pre(key_t x) { return pre(root, x); }
126 key_t nxt(key_t x) { return nxt(root, x); }
127 };
128
129 int main() {
130     ios::sync_with_stdio(false);
131     cin.tie(nullptr);
132
133     int n;
134     cin >> n;
135
136     Treap<int> T;
137
138     for (int i = 1; i <= n; i++) {
139         int op, x;
140         cin >> op >> x;
141
142         if (op == 1) {

```

```

143         T.insert(x);
144     } else if (op == 2) {
145         T.erase(x);
146     } else if (op == 3) {
147         cout << T.rank(x) << "\n";
148     } else if (op == 4) {
149         cout << T.kth(x) << "\n";
150     } else if (op == 5) {
151         cout << T.pre(x) << "\n";
152     } else if (op == 6) {
153         cout << T.nxt(x) << "\n";
154     }
155 }
156
157 return 0;
158 }
159
160 // test problem: https://loj.ac/p/104

```

0.1.17 jls 线段树.cpp

```

1  #pragma region
2  #include <algorithm>
3  #include <cmath>
4  #include <cstring>
5  #include <iomanip>
6  #include <iostream>
7  #include <map>
8  #include <queue>
9  #include <set>
10 #include <stack>
11 #include <string>
12 #include <vector>
13 using namespace std;
14 typedef long long ll;
15 #define tr t[root]
16 #define lson t[root << 1]
17 #define rson t[root << 1 | 1]
18 #define rep(i, a, n) for (int i = a; i <= n; ++i)
19 #define per(i, a, n) for (int i = n; i >= a; --i)
20 namespace fastIO {
21     #define BUF_SIZE 100000
22     #define OUT_SIZE 100000
23     //fread->R
24     bool IOerror = 0;
25     //inline char nc(){char ch=getchar();if(ch==-1)IOerror=1;return ch;}
26     inline char nc() {

```

```

27     static char buf[BUF_SIZE], *p1 = buf + BUF_SIZE, *pend = buf + BUF_SIZE;
28     if (p1 == pend) {
29         p1 = buf;
30         pend = buf + fread(buf, 1, BUF_SIZE, stdin);
31         if (pend == p1) {
32             IOError = 1;
33             return -1;
34         }
35     }
36     return *p1++;
37 }
38 inline bool blank(char ch) { return ch == ' ' || ch == '\n' || ch == '\r' || ch == '\t'; }
39 template <class T>
40 inline bool R(T &x) {
41     bool sign = 0;
42     char ch = nc();
43     x = 0;
44     for (; blank(ch); ch = nc())
45         ;
46     if (IOError)
47         return false;
48     if (ch == '-')
49         sign = 1, ch = nc();
50     for (; ch >= '0' && ch <= '9'; ch = nc())
51         x = x * 10 + ch - '0';
52     if (sign)
53         x = -x;
54     return true;
55 }
56 inline bool R(double &x) {
57     bool sign = 0;
58     char ch = nc();
59     x = 0;
60     for (; blank(ch); ch = nc())
61         ;
62     if (IOError)
63         return false;
64     if (ch == '-')
65         sign = 1, ch = nc();
66     for (; ch >= '0' && ch <= '9'; ch = nc())
67         x = x * 10 + ch - '0';
68     if (ch == '.') {
69         double tmp = 1;
70         ch = nc();
71         for (; ch >= '0' && ch <= '9'; ch = nc())
72             tmp /= 10.0, x += tmp * (ch - '0');
73     }
74     if (sign)

```



```

75     x = -x;
76     return true;
77 }
78 inline bool R(char *s) {
79     char ch = nc();
80     for (; blank(ch); ch = nc())
81         ;
82     if (IError)
83         return false;
84     for (; !blank(ch) && !IError; ch = nc())
85         *s++ = ch;
86     *s = 0;
87     return true;
88 }
89 inline bool R(char &c) {
90     c = nc();
91     if (IError) {
92         c = -1;
93         return false;
94     }
95     return true;
96 }
97 template <class T, class... U>
98 bool R(T &h, U &... t) { return R(h) && R(t...); }
99 #undef OUT_SIZE
100 #undef BUF_SIZE
101 }; // namespace fastIO
102 using namespace fastIO;
103 template <class T>
104 void _W(const T &x) { cout << x; }
105 void _W(const int &x) { printf("%d", x); }
106 void _W(const int64_t &x) { printf("%lld", x); }
107 void _W(const double &x) { printf("%.16f", x); }
108 void _W(const char &x) { putchar(x); }
109 void _W(const char *x) { printf("%s", x); }
110 template <class T, class U>
111 void _W(const pair<T, U> &x) { _W(x.F), putchar(' '), _W(x.S); }
112 template <class T>
113 void _W(const vector<T> &x) {
114     for (auto i = x.begin(); i != x.end(); _W(*i++))
115         if (i != x.cbegin()) putchar(' ');
116 }
117 void W() {}
118 template <class T, class... U>
119 void W(const T &head, const U &... tail) { _W(head), putchar(sizeof...(tail) ? ' ' : '\n'), W(tail...); }
120 #pragma endregion
121 //HDU - 5306 Gorgeous Sequence(jls线段树)

```

```

122 const int maxn = 1e6 + 5;
123 int n, m, a[maxn];
124 struct segtree {
125     int l, r, maxx, semax, cmax;
126     ll sum;
127 } t[maxn << 2];
128 inline void pushup(int root) {
129     tr.sum = lson.sum + rson.sum;
130     tr.maxx = max(lson.maxx, rson.maxx);
131     tr.semax = max(lson.semax, rson.semax);
132     tr.cmax = 0;
133     if (lson.maxx != rson.maxx) tr.semax = max(tr.semax, min(lson.maxx, rson.maxx));
134     if (tr.maxx == lson.maxx) tr.cmax += lson.cmax;
135     if (tr.maxx == rson.maxx) tr.cmax += rson.cmax;
136 }
137 void build(int root, int l, int r) {
138     tr.l = l, tr.r = r;
139     if (l == r) {
140         tr.sum = tr.maxx = a[l];
141         tr.cmax = 1;
142         tr.semax = -1;
143         return;
144     }
145     int mid = (l + r) >> 1;
146     build(root << 1, l, mid);
147     build(root << 1 | 1, mid + 1, r);
148     pushup(root);
149 }
150 inline void dec_tag(int root, int x) { //更新maxx和sum
151     if (x >= tr.maxx) return;
152     tr.sum += 1LL * (x - tr.maxx) * tr.cmax;
153     tr.maxx = x;
154 }
155 inline void spread(int root) {
156     dec_tag(root << 1, tr.maxx);
157     dec_tag(root << 1 | 1, tr.maxx);
158 }
159 void update(int root, int l, int r, int x) {
160     if (x >= tr.maxx) return; //不会产生影响, 退出
161     if (l <= tr.l && tr.r <= r && x > tr.semax) { //只影响最大值, 更新, 打标记退出
162         dec_tag(root, x);
163         return;
164     }
165     //无法更新, 递归搜索
166     spread(root);
167     int mid = (tr.l + tr.r) >> 1;
168     if (l <= mid) update(root << 1, l, r, x);
169     if (r > mid) update(root << 1 | 1, l, r, x);

```

```

170     pushup(root);
171 }
172 int qmax(int root, int l, int r) {
173     if (l <= tr.l && tr.r <= r) return tr.maxx;
174     spread(root);
175     int mid = (tr.l + tr.r) >> 1;
176     int maxx = 0;
177     if (l <= mid) maxx = max(maxx, qmax(root << 1, l, r));
178     if (r > mid) maxx = max(maxx, qmax(root << 1 | 1, l, r));
179     return maxx;
180 }
181 ll qsum(int root, int l, int r) {
182     if (l <= tr.l && tr.r <= r) return tr.sum;
183     spread(root);
184     ll ans = 0;
185     int mid = (tr.l + tr.r) >> 1;
186     if (l <= mid) ans += qsum(root << 1, l, r);
187     if (r > mid) ans += qsum(root << 1 | 1, l, r);
188     return ans;
189 }
190 int main() {
191     int T;
192     R(T);
193     while (T--) {
194         R(n, m);
195         rep(i, 1, n) R(a[i]);
196         build(1, 1, n);
197         while (m--) {
198             int op, l, r, x;
199             R(op, l, r);
200             if (op == 0) R(x), update(1, l, r, x); //区间 a[i]=min(a[i],x)
201             if (op == 1) W(qmax(1, l, r));
202             if (op == 2) W(qsum(1, l, r));
203         }
204     }
205 }

```

0.1.18 segment_tree3.cpp

```

1 // #pragma GCC optimize(2)
2 #include <algorithm>
3 #include <cstdio>
4 #include <cstdlib>
5 #include <cstring>
6 #include <iostream>
7 #include <vector>
8 using namespace std;

```

```

9 typedef long long ll;
10 const int maxn = 1e6 + 10;
11
12 ll n, m;
13 ll a[maxn];
14 struct segtree {
15     int lc, rc; //记录左右子树所在的索引下标
16     int dat;    //存储要统计的信息
17 } tr[maxn];    //开点
18 int root, tot; //根节点与即时节点
19
20 int build() //开新的节点
21 {
22     tot++; //开辟新空间
23     tr[tot].lc = tr[tot].rc = tr[tot].dat = 0; //初始化
24     return tot; //返回位置(指针)
25 }
26
27 void insert(int p, int l, int r, int val, int dat) //添加新节点, 节点管辖的是[l,r], 修改位置为val,
    加上dat
28 {
29     if (l == r) //递归基, l==r
30     {
31         tr[p].dat += dat; //修改数据域
32         return; //回退
33     }
34     int mid = (l + r) >> 1; //二分
35     //分而治之
36     if (val <= mid) //进入[l,mid]
37     {
38         if (!tr[p].lc)
39             tr[p].lc = build(); //未开辟则开辟新节点
40         insert(tr[p].lc, l, mid, val, dat); //递归下去继续插入
41     } else // [mid+1,r]
42     {
43         if (!tr[p].rc)
44             tr[p].rc = build(); //未开辟则开辟新节点
45         insert(tr[p].rc, mid + 1, r, val, dat); //递归下去继续插入
46     }
47     tr[p].dat = tr[tr[p].lc].dat + tr[tr[p].rc].dat; //合并
48 }
49
50 ll query(int p, int l, int r, int ql, int qr) {
51     if (ql <= l && qr >= r) //递归基, 查询区间包含了统计区间
52     {
53         return tr[p].dat; //回退
54     }
55     ll ans = 0; //统计答案

```

```

56     int mid = (l + r) >> 1; //划分
57     if (ql <= mid)
58         ans += query(tr[p].lc, l, mid, ql, qr); //统计左子树
59     if (qr > mid)
60         ans += query(tr[p].rc, mid + 1, r, ql, qr); //统计右子树
61     return ans; //返回答案
62 }
63
64 int main() {
65     ios::sync_with_stdio(false);
66     cin.tie(0);
67     int T;
68     cin >> T;
69     for (int cas = 1; cas <= T; cas++) {
70         cout << "Case " << cas << ":" << endl;
71
72         root = 0, tot = 0;
73         cin >> n;
74         root = build();
75         for (int i = 1; i <= n; i++)
76             cin >> a[i], insert(root, 1, n, i, a[i]);
77         string s;
78         while (cin >> s) {
79             if (s == "End")
80                 break;
81             else if (s == "Query") {
82                 int l, r;
83                 cin >> l >> r;
84                 cout << query(root, 1, n, l, r) << endl;
85             } else if (s == "Add") {
86                 int x, v;
87                 cin >> x >> v;
88                 insert(root, 1, n, x, v);
89             } else if (s == "Sub") {
90                 int x, v;
91                 cin >> x >> v;
92                 insert(root, 1, n, x, -v);
93             }
94         }
95     }
96 }

```

0.1.19 主席树.cpp

```

1 #include <algorithm>
2 #include <cstdio>
3 #include <cstring>

```

```

4 using namespace std;
5 const int maxn = 1e5 + 5; //数据范围
6 int tot, n, m;
7 int sum[(maxn << 5) + 10], rt[maxn + 10], ls[(maxn << 5) + 10],
8     rs[(maxn << 5) + 10];
9 int a[maxn + 10], ind[maxn + 10], len;
10 inline int getid(const int &val) { //离散化
11     return lower_bound(ind + 1, ind + len + 1, val) - ind;
12 }
13 int build(int l, int r) { //建树
14     int root = ++tot;
15     if (l == r)
16         return root;
17     int mid = (l + r) >> 1;
18     ls[root] = build(l, mid);
19     rs[root] = build(mid + 1, r);
20     return root; //返回该子树的根节点
21 }
22 int update(int k, int l, int r, int root) { //插入操作
23     int dir = ++tot;
24     ls[dir] = ls[root], rs[dir] = rs[root], sum[dir] = sum[root] + 1;
25     if (l == r) return dir;
26     int mid = (l + r) >> 1;
27     if (k <= mid) ls[dir] = update(k, l, mid, ls[dir]);
28     else rs[dir] = update(k, mid + 1, r, rs[dir]);
29     return dir;
30 }
31 int query(int u, int v, int l, int r, int k) { //查询操作
32     int mid = (l + r) >> 1, x = sum[ls[v]] - sum[ls[u]]; //通过区间减法得到左儿子的信息
33     if (l == r) return l;
34     if (k <= x) //说明在左儿子中
35         return query(ls[u], ls[v], l, mid, k);
36     else //说明在右儿子中
37         return query(rs[u], rs[v], mid + 1, r, k - x);
38 }
39 inline void init() {
40     tot = 0;
41     scanf("%d%d", &n, &m);
42     for (int i = 1; i <= n; ++i)
43         scanf("%d", a + i);
44     memcpy(ind, a, sizeof ind);
45     sort(ind + 1, ind + n + 1);
46     len = unique(ind + 1, ind + n + 1) - ind - 1;
47     rt[0] = build(1, len);
48     for (int i = 1; i <= n; ++i)
49         rt[i] = update(getid(a[i]), 1, len, rt[i - 1]);
50 }
51 int l, r, k;

```

```

52 inline int qmin(int k) { return ind[query(rt[l - 1], rt[r], 1, len, k)]; } //回答第k小
53 inline int qmax(int k) { return ind[query(rt[l - 1], rt[r], 1, len, r - l + 2 - k)]; } //回答第k大
54 inline void work() {
55     while (m--) {
56         scanf("%d%d%d", &l, &r, &k);
57         printf("%d\n", ind[query(rt[l - 1], rt[r], 1, len, k)]); //回答询问
58     }
59 }
60 int main() {
61     init();
62     work();
63     return 0;
64 }

```

0.1.20 区间覆盖.cpp

```

1  #include <bits/stdc++.h>
2  #define rep(i, a, n) for (int i = a; i <= n; ++i)
3  #define per(i, a, n) for (int i = n; i >= a; --i)
4  #ifdef LOCAL
5  #include "Print.h"
6  #define de(...) W(' ', #__VA_ARGS__, " ") ==, __VA_ARGS__)
7  #else
8  #define de(...)
9  #endif
10 using namespace std;
11 typedef long long ll;
12 const int maxn = 1e5 + 5;
13 int n, q, a[maxn];
14 vector<int> g[maxn];
15 int sz[maxn], id[maxn], idd[maxn], cnt;
16 void dfs(int u, int f) {
17     sz[u] = 1, id[u] = ++cnt, idd[cnt] = u;
18     for (auto v : g[u]) {
19         if (v == f) continue;
20         dfs(v, u);
21         sz[u] += sz[v];
22     }
23 }
24 struct segtree{
25     #define tr t[root]
26     #define lson t[root << 1]
27     #define rson t[root << 1 | 1]
28     struct node {
29         int l, r, maxx, minn;
30         int add, cov;
31     } t[maxn << 2];

```

```

32 void build(int root, int l, int r) {
33     tr.l = l, tr.r = r, tr.add = 0, tr.cov = -1;
34     if (l == r) {
35         tr.maxx = tr.minn = a[idd[l]];
36         return;
37     }
38     int mid = (l + r) >> 1;
39     build(root << 1, l, mid);
40     build(root << 1 | 1, mid + 1, r);
41     pushup(root);
42 }
43 void pushup(int root) {
44     tr.maxx = max(lson.maxx, rson.maxx);
45     tr.minn = min(lson.minn, rson.minn);
46 }
47 void spdCov(int root) {
48     lson.minn = rson.minn = tr.cov;
49     lson.maxx = rson.maxx = tr.cov;
50     lson.cov = rson.cov = tr.cov;
51 }
52 void spdAdd(int root) {
53     if (~lson.cov) {
54         if (lson.l != lson.r) spdCov(root << 1);
55         lson.cov = -1, lson.add = 0;
56     }
57     if (~rson.cov) {
58         if (rson.l != rson.r) spdCov(root << 1 | 1);
59         rson.cov = -1, rson.add = 0;
60     }
61     lson.minn += tr.add, rson.minn += tr.add;
62     lson.maxx += tr.add, rson.maxx += tr.add;
63     lson.add += tr.add, rson.add += tr.add;
64 }
65 void spread(int root) {
66     if (~tr.cov) {
67         if (tr.l != tr.r) spdCov(root);
68         tr.cov = -1, tr.add = 0;
69     }
70     if (tr.add) {
71         if (tr.l != tr.r) spdAdd(root);
72         tr.add = 0;
73     }
74 }
75 void cov(int root, int l, int r, int x) {
76     spread(root);
77     if (l <= tr.l && tr.r <= r) {
78         tr.minn = x, tr.maxx = x;
79         tr.add = 0, tr.cov = x;

```



```

80         return;
81     }
82     int mid = (tr.l + tr.r) >> 1;
83     if (l <= mid) cov(root << 1, l, r, x);
84     if (r > mid) cov(root << 1 | 1, l, r, x);
85     pushup(root);
86 }
87 void add(int root, int l, int r, int x) {
88     spread(root);
89     if (l <= tr.l && tr.r <= r) {
90         tr.minn += x, tr.maxx += x;
91         tr.add += x;
92         return;
93     }
94     int mid = (tr.l + tr.r) >> 1;
95     if (l <= mid) add(root << 1, l, r, x);
96     if (r > mid) add(root << 1 | 1, l, r, x);
97     pushup(root);
98 }
99 int qmax(int root, int l, int r) {
100     spread(root);
101     if (l <= tr.l && tr.r <= r) return tr.maxx;
102     int mid = (tr.l + tr.r) >> 1, ans = 0;
103     if (l <= mid) ans = max(ans, qmax(root << 1, l, r));
104     if (r > mid) ans = max(ans, qmax(root << 1 | 1, l, r));
105     return ans;
106 }
107 int qmin(int root, int l, int r) {
108     spread(root);
109     if (l <= tr.l && tr.r <= r) return tr.minn;
110     int mid = (tr.l + tr.r) >> 1, ans = 2e9;
111     if (l <= mid) ans = min(ans, qmin(root << 1, l, r));
112     if (r > mid) ans = min(ans, qmin(root << 1 | 1, l, r));
113     return ans;
114 }
115 } Tr;
116 inline void add(int u, int val) { Tr.add(1, id[u], id[u] + sz[u] - 1, val); }
117 inline void cov(int u, int val) { Tr.cov(1, id[u], id[u] + sz[u] - 1, val); }
118 inline int qry(int u) {
119     int l = id[u], r = id[u] + sz[u] - 1;
120     return Tr.qmax(1, l, r) - Tr.qmin(1, l, r);
121 }
122 int case_Test() {
123     scanf("%d%d", &n, &q);
124     rep(i, 1, n) scanf("%d", &a[i]);
125     rep(i, 1, n - 1) {
126         int u, v;
127         scanf("%d%d", &u, &v);

```

```

128     g[u].emplace_back(v);
129     g[v].emplace_back(u);
130 }
131 dfs(1, 0), Tr.build(1, 1, n);
132 while (q--) {
133     int op, x, V;
134     scanf("%d%d", &op, &x);
135     if (op == 0) scanf("%d", &V), add(x, V);
136     if (op == 1) scanf("%d", &V), cov(x, V);
137     if (op == 2) printf("%d\n", qry(x));
138 }
139 return 0;
140 }
141 int main() {
142 #ifdef LOCAL
143     freopen("/Users/chenjinglong/cpp_code/in.in", "r", stdin);
144     freopen("/Users/chenjinglong/cpp_code/out.out", "w", stdout);
145     clock_t start = clock();
146 #endif
147     int _ = 1;
148     // scanf("%d", &_);
149     while (--) case_Test();
150 #ifdef LOCAL
151     printf("Time used: %.3lfs\n", (double)(clock() - start) / CLOCKS_PER_SEC);
152 #endif
153     return 0;
154 }
155 // 【月下“毛景树”】https://www.luogu.com.cn/problem/P4315

```

0.1.21 带权并查集.cpp

```

1 #include <bits/stdc++.h>
2 #define rep(i, a, n) for (int i = a; i <= n; ++i)
3 #define per(i, a, n) for (int i = n; i >= a; --i)
4 #ifdef LOCAL
5 #include "Print.h"
6 #define de(...) W('[' , #__VA_ARGS__ , "]" , __VA_ARGS__)
7 #else
8 #define de(...)
9 #endif
10 using namespace std;
11 typedef long long ll;
12 const int maxn = 3e4 + 5;
13 int fa[maxn], sz[maxn], d[maxn]; // d表示与父亲结点的关系
14 int findR(int x) {
15     if (x == fa[x]) return x;
16     int rt = findR(fa[x]);

```

```

17     d[x] += d[fa[x]];
18     return fa[x] = rt;
19 }
20 void link(int x, int y, int f) {
21     int xx = findR(x), yy = findR(y);
22     fa[xx] = yy, d[xx] += sz[yy];
23     sz[yy] += sz[xx];
24 }
25 int query(int x, int y) {
26     if (x == y) return 0;
27     int xx = findR(x), yy = findR(y);
28     if (xx != yy) return -1;
29     return abs(d[x] - d[y]) - 1;
30 }
31 int main() {
32     int T;
33     scanf("%d", &T);
34     rep(i, 1, maxn - 1) fa[i] = i, sz[i] = 1;
35     while (T--) {
36         char op[5]; int x, y;
37         scanf("%s%d%d", op + 1, &x, &y);
38         if (op[1] == 'M') link(x, y, 1);
39         else printf("%d\n", query(x, y));
40     }
41     return 0;
42 }

```

0.1.22 替罪羊.cpp

```

1  #include <bits/stdc++.h>
2  #define rep(i, a, n) for (int i = a; i <= n; ++i)
3  #define per(i, a, n) for (int i = n; i >= a; --i)
4  #ifdef LOCAL
5  #include "Print.h"
6  #define de(...) W('[' , #__VA_ARGS__ , "]" = , __VA_ARGS__ )
7  #else
8  #define de(...)
9  #endif
10 using namespace std;
11 typedef long long ll;
12 const int maxn = 1e5 + 5;
13 namespace tzy {
14     #define tr t[root]
15     #define lson t[tr.lc]
16     #define rson t[tr.rc]
17     const double alpha = 0.75;
18     int cnt, Root;

```

```

19 struct node {
20     int val, lc, rc;
21     int num, sz, csz, dsz;
22 } t[maxn];
23 // 重新计算以 root 为根的子树大小
24 void Calc(int root) {
25     tr.sz = lson.sz + rson.sz + 1;
26     tr.csz = lson.csz + rson.csz + tr.num;
27     tr.dsz = lson.dsz + rson.dsz + (tr.num != 0);
28 }
29 // 判断节点 root 是否需要重构
30 inline bool CanRbu(int root) {
31     return tr.num && (max(lson.sz, rson.sz) >= alpha * tr.sz || tr.dsz <= alpha * tr.sz);
32 }
33 int ldr[maxn];
34 // 中序遍历展开以 root 节点为根子树
35 void getLdr(int &len, int root) {
36     if (!root) return;
37     getLdr(len, tr.lc);
38     if (tr.num) ldr[len++] = root;
39     getLdr(len, tr.rc);
40 }
41 // 将 ldr[] 数组内 [l, r) 区间重建成树, 返回根节点
42 int lift(int l, int r) {
43     int mid = (l + r) >> 1, R = ldr[mid];
44     if (l >= r) return 0;
45     t[R].lc = lift(l, mid);
46     t[R].rc = lift(mid + 1, r);
47     Calc(R);
48     return R;
49 }
50 // 重构节点 root 的全过程
51 void rebuild(int &root) {
52     if (!CanRbu(root)) return;
53     int len = 0;
54     getLdr(len, root);
55     root = lift(0, len);
56 }
57 // 在以 root 为根的子树内添加权值为 val 节点
58 void Insert(int &root, int val) {
59     if (!root) {
60         root = ++cnt;
61         if (!Root) Root = 1;
62         tr.val = val, tr.lc = tr.rc = 0;
63         tr.num = tr.sz = tr.csz = tr.dsz = 1;
64     } else {
65         if (val == tr.val) tr.num++;
66         else if (val < tr.val) Insert(tr.lc, val);

```

```

67         else Insert(tr.rc, val);
68         Calc(root), rebuild(root);
69     }
70 }
71 // 在以 root 为根子树移除权值为 val 节点
72 void Del(int &root, int val) {
73     if (!root) return;
74     if (tr.val == val) {
75         if (tr.num) tr.num--;
76     } else {
77         if (val < tr.val) Del(tr.lc, val);
78         else Del(tr.rc, val);
79     }
80     Calc(root), rebuild(root);
81 }
82 // 在以 root 为根子树中, 大于 val 的最小数的名次
83 int MyUprBd(int root, int val) {
84     if (!root) return 1;
85     if (val == tr.val && tr.num) return lson.csz + 1 + tr.num;
86     if (val < tr.val) return MyUprBd(tr.lc, val);
87     return lson.csz + tr.num + MyUprBd(tr.rc, val);
88 }
89 // 权值严格小于某值的最大名次
90 int MyUprGrt(int root, int val) {
91     if (!root) return 0;
92     if (val == tr.val) return lson.csz;
93     if (val < tr.val) return MyUprGrt(tr.lc, val);
94     return lson.csz + tr.num + MyUprGrt(tr.rc, val);
95 }
96 // 以 root 为根的子树中, 名次为 rnk 的权值
97 int Getnum(int root, int rnk) {
98     if (!root) return 0;
99     if (lson.csz < rnk && rnk <= lson.csz + tr.num) return tr.val;
100     if (lson.csz >= rnk) return Getnum(tr.lc, rnk);
101     return Getnum(tr.rc, rnk - lson.csz - tr.num);
102 }
103 inline void insert(int val) { Insert(Root, val); }
104 inline void del(int val) { Del(Root, val); }
105 inline int getnum(int rnk) { return Getnum(Root, rnk); }
106 inline int getrnk(int val) { return MyUprGrt(Root, val) + 1; }
107 inline int lowerRnk(int val) { return MyUprGrt(Root, val); }
108 inline int upperRnk(int val) { return MyUprBd(Root, val); }
109 inline int getpre(int val) { return getnum(lowerRnk(val)); }
110 inline int getnex(int val) { return getnum(upperRnk(val)); }
111 #undef tr
112 #undef lson
113 #undef rson
114 } // namespace tzy

```

```

115 int case_Test() {
116     int _; scanf("%d", &_);
117     while (_--) {
118         int op, x;
119         scanf("%d%d", &op, &x);
120         if (op == 1) tzy::insert(x);
121         if (op == 2) tzy::del(x);
122         if (op == 3) printf("%d\n", tzy::getrnk(x));
123         if (op == 4) printf("%d\n", tzy::getnum(x));
124         if (op == 5) printf("%d\n", tzy::getpre(x));
125         if (op == 6) printf("%d\n", tzy::getnex(x));
126     }
127     return 0;
128 }
129 int main() {
130 #ifdef LOCAL
131     freopen("/Users/chenjinglong/Desktop/cpp_code/in.in", "r", stdin);
132     freopen("/Users/chenjinglong/Desktop/cpp_code/out.out", "w", stdout);
133     clock_t start = clock();
134 #endif
135     int _ = 1;
136     // scanf("%d", &_);
137     while (_--) case_Test();
138 #ifdef LOCAL
139     printf("Time used: %.3lfs\n", (double)(clock() - start) / CLOCKS_PER_SEC);
140 #endif
141     return 0;
142 }

```

0.1.23 树剖.cpp

```

1 #include <bits/stdc++.h>
2
3 using i64 = long long;
4
5 struct Info {
6     int c[2];
7     i64 s[2];
8     Info() : c{}, s{} {}
9     Info(int x, int v) : Info() {
10         c[x] = 1;
11         s[x] = v;
12     }
13 };
14
15 Info operator+(const Info &a, const Info &b) {
16     Info c;

```

```

17     c.c[0] = a.c[0] + b.c[0];
18     c.c[1] = a.c[1] + b.c[1];
19     c.s[0] = a.s[0] + b.s[0];
20     c.s[1] = a.s[1] + b.s[1];
21     return c;
22 }
23
24 void apply(Info &a, int b) {
25     if (b) {
26         std::swap(a.c[0], a.c[1]);
27         std::swap(a.s[0], a.s[1]);
28     }
29 }
30
31 void apply(int &a, int b) {
32     a ^= b;
33 }
34
35 template<class Info, class Tag,
36         class Merge = std::plus<Info>>
37 struct LazySegmentTree {
38     const int n;
39     const Merge merge;
40     std::vector<Info> info;
41     std::vector<Tag> tag;
42     LazySegmentTree(int n) : n(n), merge(Merge()), info(4 << std::lg(n)), tag(4 << std::lg(n)) {}
43     LazySegmentTree(std::vector<Info> init) : LazySegmentTree(init.size()) {
44         std::function<void(int, int, int)> build = [&](int p, int l, int r) {
45             if (r - l == 1) {
46                 info[p] = init[l];
47                 return;
48             }
49             int m = (l + r) / 2;
50             build(2 * p, l, m);
51             build(2 * p + 1, m, r);
52             pull(p);
53         };
54         build(1, 0, n);
55     }
56     void pull(int p) {
57         info[p] = merge(info[2 * p], info[2 * p + 1]);
58     }
59     void apply(int p, const Tag &v) {
60         ::apply(info[p], v);
61         ::apply(tag[p], v);
62     }
63     void push(int p) {
64         apply(2 * p, tag[p]);

```

```

65     apply(2 * p + 1, tag[p]);
66     tag[p] = Tag();
67 }
68 void modify(int p, int l, int r, int x, const Info &v) {
69     if (r - l == 1) {
70         info[p] = v;
71         return;
72     }
73     int m = (l + r) / 2;
74     push(p);
75     if (x < m) {
76         modify(2 * p, l, m, x, v);
77     } else {
78         modify(2 * p + 1, m, r, x, v);
79     }
80     pull(p);
81 }
82 void modify(int p, const Info &v) {
83     modify(1, 0, n, p, v);
84 }
85 Info rangeQuery(int p, int l, int r, int x, int y) {
86     if (l >= y || r <= x) {
87         return Info();
88     }
89     if (l >= x && r <= y) {
90         return info[p];
91     }
92     int m = (l + r) / 2;
93     push(p);
94     return merge(rangeQuery(2 * p, l, m, x, y), rangeQuery(2 * p + 1, m, r, x, y));
95 }
96 Info rangeQuery(int l, int r) {
97     return rangeQuery(1, 0, n, l, r);
98 }
99 bool rangeApply(int p, int l, int r, int x, int y, const Tag &v) {
100     if (l >= y || r <= x) {
101         return true;
102     }
103     if (l >= x && r <= y && info[p].c[0] + info[p].c[1] == r - l) {
104         apply(p, v);
105         return true;
106     }
107     if (l >= x && r <= y && info[p].c[0] + info[p].c[1] == 0) {
108         return false;
109     }
110     int m = (l + r) / 2;
111     push(p);
112     bool res;

```



```

113     if (rangeApply(2 * p + 1, m, r, x, y, v)) {
114         res = rangeApply(2 * p, l, m, x, y, v);
115     } else {
116         res = false;
117     }
118     pull(p);
119     return res;
120 }
121 bool rangeApply(int l, int r, const Tag &v) {
122     return rangeApply(1, 0, n, l, r, v);
123 }
124 };
125
126 int main() {
127     std::ios::sync_with_stdio(false);
128     std::cin.tie(nullptr);
129
130     int n;
131     std::cin >> n;
132
133     std::vector<std::vector<std::pair<int, int>>> adj(n);
134     for (int i = 0; i < n - 1; i++) {
135         int u, v;
136         std::cin >> u >> v;
137         u--;
138         v--;
139
140         adj[u].emplace_back(v, i + 1);
141         adj[v].emplace_back(u, i + 1);
142     }
143
144     std::vector<int> id(n), parent(n, -1), dep(n), top(n), in(n), out(n), siz(n);
145     int clk = 0;
146
147     std::function<void(int)> dfs1 = [&](int u) {
148         if (u > 0) {
149             adj[u].erase(std::find(adj[u].begin(), adj[u].end(), std::pair<parent[u], id[u]>));
150         }
151         siz[u] = 1;
152         for (auto &e : adj[u]) {
153             auto [v, i] = e;
154             id[v] = i;
155             parent[v] = u;
156             dep[v] = dep[u] + 1;
157             dfs1(v);
158             siz[u] += siz[v];
159             if (siz[v] > siz[adj[u][0].first]) {
160                 std::swap(adj[u][0], e);

```

```

161         }
162     }
163 };
164 dfs1(0);
165
166 std::function<void(int)> dfs2 = [&](int u) {
167     in[u] = clk++;
168     for (auto [v, i] : adj[u]) {
169         top[v] = v == adj[u][0].first ? top[u] : v;
170         dfs2(v);
171     }
172     out[u] = clk;
173 };
174 dfs2(0);
175
176 LazySegmentTree<Info, int> seg(n);
177 seg.modify(0, Info(1, 0));
178
179 while (true) {
180     int op;
181     std::cin >> op;
182
183     if (op == 3) {
184         break;
185     }
186
187     if (op == 1) {
188         int x;
189         std::cin >> x;
190         x--;
191         int s = 1;
192         for (auto [v, i] : adj[x]) {
193             s ^= seg.rangeQuery(in[v], in[v] + 1).c[1];
194         }
195
196         seg.modify(in[x], Info(s, id[x]));
197         if (s == 1) {
198             x = parent[x];
199             while (x != -1) {
200                 if (!seg.rangeApply(in[top[x]], in[x] + 1, 1)) {
201                     break;
202                 }
203                 x = parent[top[x]];
204             }
205         }
206     }
207
208     auto info = seg.info[1];

```

```

209         if (info.c[0] != info.c[1]) {
210             std::cout << 0 << std::endl;
211         } else if (op == 1) {
212             std::cout << info.s[1] << std::endl;
213         } else {
214             std::vector<int> ans;
215             for (int i = 0; i < n; i++) {
216                 if (seg.rangeQuery(in[i], in[i] + 1).c[1] == 1) {
217                     ans.push_back(id[i]);
218                 }
219             }
220             std::sort(ans.begin(), ans.end());
221
222             std::cout << ans.size();
223             for (auto x : ans) {
224                 std::cout << " " << x;
225             }
226             std::cout << std::endl;
227         }
228     }
229
230     return 0;
231 }

```

0.1.24 笛卡尔树.cpp

```

1  #include<bits/stdc++.h>
2  #define rep(i, a, n) for (int i = a; i <= n; ++i)
3  #define per(i, a, n) for (int i = n; i >= a; --i)
4  using namespace std;
5  typedef long long ll;
6  const int maxn = 1e7 + 5;
7  int n, a[maxn];
8  int ls[maxn], rs[maxn];
9  int top = 0;
10 // stack<int> st;
11 int st[maxn];
12 // ls代表笛卡尔树每个节点的左孩子，rs代表笛卡尔树每个节点的右孩子
13 // 按照满足二叉搜索树的权值排序，插入在右链
14 // 栈顶元素为当前元素的左孩子
15 // 当前元素为栈顶元素的右孩子
16 int main() {
17     int n;
18     scanf("%d", &n);
19     rep(i, 1, n) {
20         scanf("%d", &a[i]);
21         // while (st.size() && a[st.top()] > a[i]) ls[i] = st.top(), st.pop();

```

```

22     // if (st.size()) rs[st.top()] = i;
23     // st.push(i);
24     while (top && a[st[top]] > a[i]) ls[i] = st[top--];
25     if (top) rs[st[top]] = i;
26     st[++top] = i;
27 }
28 ll lans = 0, rans = 0;
29 rep(i, 1, n) {
30     lans ^= 1LL * i * (ls[i] + 1);
31     rans ^= 1LL * i * (rs[i] + 1);
32 }
33 printf("%lld %lld\n", lans, rans);
34 return 0;
35 }

```

0.1.25 轻重链剖分.cpp

```

1 //洛谷P3384
2 #pragma region
3 #include <algorithm>
4 #include <cmath>
5 #include <cstring>
6 #include <iomanip>
7 #include <iostream>
8 #include <map>
9 #include <queue>
10 #include <set>
11 #include <stack>
12 #include <string>
13 #include <vector>
14 using namespace std;
15 typedef long long ll;
16 #define tr t[root]
17 #define lson t[root << 1]
18 #define rson t[root << 1 | 1]
19 #define rep(i, a, n) for (int i = a; i <= n; ++i)
20 #define per(i, a, n) for (int i = n; i >= a; --i)
21 namespace fastIO {
22     #define BUF_SIZE 100000
23     #define OUT_SIZE 100000
24     //fread->R
25     bool IOerror = 0;
26     //inline char nc(){char ch=getchar();if(ch==-1)IOerror=1;return ch;}
27     inline char nc() {
28         static char buf[BUF_SIZE], *p1 = buf + BUF_SIZE, *pend = buf + BUF_SIZE;
29         if (p1 == pend) {
30             p1 = buf;

```

```

31     pend = buf + fread(buf, 1, BUF_SIZE, stdin);
32     if (pend == p1) {
33         IOError = 1;
34         return -1;
35     }
36 }
37 return *p1++;
38 }
39 inline bool blank(char ch) { return ch == ' ' || ch == '\n' || ch == '\r' || ch == '\t'; }
40 template <class T>
41 inline bool R(T &x) {
42     bool sign = 0;
43     char ch = nc();
44     x = 0;
45     for (; blank(ch); ch = nc())
46         ;
47     if (IOError)
48         return false;
49     if (ch == '-')
50         sign = 1, ch = nc();
51     for (; ch >= '0' && ch <= '9'; ch = nc())
52         x = x * 10 + ch - '0';
53     if (sign)
54         x = -x;
55     return true;
56 }
57 inline bool R(double &x) {
58     bool sign = 0;
59     char ch = nc();
60     x = 0;
61     for (; blank(ch); ch = nc())
62         ;
63     if (IOError)
64         return false;
65     if (ch == '-')
66         sign = 1, ch = nc();
67     for (; ch >= '0' && ch <= '9'; ch = nc())
68         x = x * 10 + ch - '0';
69     if (ch == '.') {
70         double tmp = 1;
71         ch = nc();
72         for (; ch >= '0' && ch <= '9'; ch = nc())
73             tmp /= 10.0, x += tmp * (ch - '0');
74     }
75     if (sign)
76         x = -x;
77     return true;
78 }

```

```

79 inline bool R(char *s) {
80     char ch = nc();
81     for (; blank(ch); ch = nc())
82         ;
83     if (IOerror)
84         return false;
85     for (; !blank(ch) && !IOerror; ch = nc())
86         *s++ = ch;
87     *s = 0;
88     return true;
89 }
90 inline bool R(char &c) {
91     c = nc();
92     if (IOerror) {
93         c = -1;
94         return false;
95     }
96     return true;
97 }
98 template <class T, class... U>
99 bool R(T &h, U &... t) { return R(h) && R(t...); }
100 #undef OUT_SIZE
101 #undef BUF_SIZE
102 }; // namespace fastIO
103 using namespace fastIO;
104 template <class T>
105 void _W(const T &x) { cout << x; }
106 void _W(const int &x) { printf("%d", x); }
107 void _W(const int64_t &x) { printf("%lld", x); }
108 void _W(const double &x) { printf("%.16f", x); }
109 void _W(const char &x) { putchar(x); }
110 void _W(const char *x) { printf("%s", x); }
111 template <class T, class U>
112 void _W(const pair<T, U> &x) { _W(x.F), putchar(' '), _W(x.S); }
113 template <class T>
114 void _W(const vector<T> &x) {
115     for (auto i = x.begin(); i != x.end(); _W(*i++))
116         if (i != x.cbegin()) putchar(' ');
117 }
118 void W() {}
119 template <class T, class... U>
120 void W(const T &head, const U &... tail) { _W(head), putchar(sizeof...(tail) ? ' ' : '\n'), W(tail
    ...); }
121 #pragma endregion
122 const int maxn = 1e5 + 5;
123 int n, m, r, mod;
124 int w[maxn];
125 vector<int> g[maxn];

```

```

126 int fa[maxn], sz[maxn], dep[maxn], son[maxn];
127 int id[maxn], cnt, wt[maxn], top[maxn];
128 void init() {
129     rep(i, 1, n) {
130         g[i].clear();
131         son[i] = 0;
132     }
133 }
134 void dfs1(int u, int f, int deep) {
135     dep[u] = deep, fa[u] = f, sz[u] = 1;
136     for (auto v : g[u]) {
137         if (v == f) continue;
138         dfs1(v, u, deep + 1);
139         sz[u] += sz[v];
140         if (sz[v] > sz[son[u]]) son[u] = v;
141     }
142 }
143 void dfs2(int u, int topf) {
144     id[u] = ++cnt, wt[cnt] = w[u], top[u] = topf;
145     if (!son[u]) return;
146     dfs2(son[u], topf);
147     for (auto v : g[u]) {
148         if (v == fa[u] || v == son[u]) continue;
149         dfs2(v, v);
150     }
151 }
152 struct segtree {
153     int l, r, val, lazy;
154 } t[maxn << 2];
155 void build(int root, int l, int r) {
156     tr.l = l, tr.r = r, tr.lazy = 0;
157     if (l == r) {
158         tr.val = wt[l] % mod;
159         return;
160     }
161     int mid = (l + r) >> 1;
162     build(root << 1, l, mid);
163     build(root << 1 | 1, mid + 1, r);
164     tr.val = (lson.val + rson.val) % mod;
165 }
166 void spread(int root) {
167     if (tr.lazy) {
168         lson.val = (lson.val + tr.lazy * (lson.r - lson.l + 1)) % mod;
169         rson.val = (rson.val + tr.lazy * (rson.r - rson.l + 1)) % mod;
170         lson.lazy = (lson.lazy + tr.lazy) % mod;
171         rson.lazy = (rson.lazy + tr.lazy) % mod;
172         tr.lazy = 0;
173     }

```

```

174 }
175 int query(int root, int l, int r) {
176     if (l <= tr.l && tr.r <= r) return tr.val % mod;
177     spread(root);
178     int ans = 0;
179     int mid = (tr.l + tr.r) >> 1;
180     if (l <= mid) ans = (ans + query(root << 1, l, r)) % mod;
181     if (r > mid) ans = (ans + query(root << 1 | 1, l, r)) % mod;
182     return ans;
183 }
184 void update(int root, int l, int r, int x) {
185     if (l <= tr.l && tr.r <= r) {
186         tr.val = (tr.val + x * (tr.r - tr.l + 1)) % mod;
187         tr.lazy = (tr.lazy + x) % mod;
188         return;
189     }
190     spread(root);
191     int mid = (tr.l + tr.r) >> 1;
192     if (l <= mid) update(root << 1, l, r, x);
193     if (r > mid) update(root << 1 | 1, l, r, x);
194     tr.val = (lson.val + rson.val) % mod;
195 }
196 int qSon(int x) { return query(1, id[x], id[x] + sz[x] - 1); }
197 void updSon(int x, int k) { update(1, id[x], id[x] + sz[x] - 1, k); }
198 int qRange(int x, int y) {
199     int ans = 0;
200     while (top[x] != top[y]) {
201         if (dep[top[x]] < dep[top[y]]) swap(x, y);
202         ans = (ans + query(1, id[top[x]], id[x])) % mod;
203         x = fa[top[x]];
204     }
205     if (dep[x] > dep[y]) swap(x, y);
206     ans = (ans + query(1, id[x], id[y])) % mod;
207     return ans;
208 }
209 void updRange(int x, int y, int k) {
210     k %= mod;
211     while (top[x] != top[y]) {
212         if (dep[top[x]] < dep[top[y]]) swap(x, y);
213         update(1, id[top[x]], id[x], k);
214         x = fa[top[x]];
215     }
216     if (dep[x] > dep[y]) swap(x, y);
217     update(1, id[x], id[y], k);
218 }
219 int main() {
220     R(n, m, r, mod);
221     rep(i, 1, n) R(w[i]);

```



```

222     rep(i, 1, n - 1) {
223         int u, v;
224         R(u, v);
225         g[u].push_back(v);
226         g[v].push_back(u);
227     }
228     dfs1(r, 0, 1);
229     dfs2(r, r);
230     build(1, 1, n);
231     while (m--) {
232         int op, x, y, z;
233         R(op);
234         if (op == 1)
235             R(x, y, z), updRange(x, y, z);
236         else if (op == 2)
237             R(x, y), W(qRange(x, y));
238         else if (op == 3)
239             R(x, y), updSon(x, y);
240         else
241             R(x), W(qSon(x));
242     }
243 }

```

0.2 Geometry

0.2.1 Circle.cpp

```

1  #include "PolygonAndConvex.cpp"
2
3  double sqr(double x) { return x * x; }
4  double mysqrt(double n) {
5      return sqrt(max(0.0, n));
6  } // 防止出现sqrt(-eps)的情况
7
8  struct Circle {
9      Point o;
10     double r;
11     Circle(Point o = Point(), double r = 0) : o(o), r(r) {}
12     bool operator==(const Circle &c) { return o == c.o && !sgn(r - c.r); }
13     double area() { return PI * r * r; }
14     double perimeter() { return r * PI * 2; }
15     // 点在圆内, 不包含边界
16     bool pointIn(const Point &p) { return sgn((p - o).norm() - r) < 0; }
17     // 判直线和圆相交, 包括相切
18     friend int isLineCircleIntersection(Line L, Circle c) {
19         return L.disPointLine(c.o) < c.r + eps;
20     }

```

```

21 // 判线段和圆相交,包括端点和相切
22 friend int isSegCircleIntersection(Line L, Circle c) {
23     double t1 = dis(c.o, L.s) - c.r, t2 = dis(c.o, L.t) - c.r;
24     Point t = c.o;
25     if (t1 < eps || t2 < eps) return t1 > -eps || t2 > -eps;
26     t.x += L.s.y - L.t.y;
27     t.y += L.t.x - L.s.x;
28     return det(L.s - t, c.o - t) * det(L.t - t, c.o - t) < eps && L.disPointLine(c.o) < c.r + eps;
29 }
30 // 判圆和圆相交,包括相切
31 friend int isCirCirIntersection(Circle c1, Circle c2) {
32     return dis(c1.o, c2.o) < c1.r + c2.r + eps &&
33         dis(c1.o, c2.o) > fabs(c1.r - c2.r) - eps;
34 }
35 // 判圆和圆内含
36 friend int isCirCirContain(Circle c1, Circle c2) {
37     return sgn(dis(c1.o, c2.o) + min(c1.r, c2.r) - max(c1.r, c2.r)) <= 0;
38 }
39 // 计算圆上到点p最近点,如p与圆心重合,返回p本身
40 friend Point dotPointCircle(Point p, Circle C) {
41     Point u, v, c = C.o;
42     if (dis(p, c) < eps) return p;
43     u.x = c.x + C.r * fabs(c.x - p.x) / dis(c, p);
44     u.y = c.y + C.r * fabs(c.y - p.y) / dis(c, p) * ((c.x - p.x) * (c.y - p.y) < 0 ? -1 : 1);
45     v.x = c.x - C.r * fabs(c.x - p.x) / dis(c, p);
46     v.y = c.y - C.r * fabs(c.y - p.y) / dis(c, p) * ((c.x - p.x) * (c.y - p.y) < 0 ? -1 : 1);
47     return dis(u, p) < dis(v, p) ? u : v;
48 }
49 // 圆与线段交 用参数方程表示直线:  $P=A+t*(B-A)$ , 带入圆的方程求解t
50 friend vector<Point> segCircleIntersection(const Line &l, const Circle &c) {
51     double dx = l.t.x - l.s.x, dy = l.t.y - l.s.y;
52     double A = dx * dx + dy * dy;
53     double B = 2 * dx * (l.s.x - c.o.x) + 2 * dy * (l.s.y - c.o.y);
54     double C = sqr(l.s.x - c.o.x) + sqr(l.s.y - c.o.y) - sqr(c.r);
55     double delta = B * B - 4 * A * C;
56     vector<Point> res;
57     if (A < eps) return res;
58     if (sgn(delta) >= 0) { // or delta > -eps ?
59         // 可能需要注意delta接近-eps的情况, 所以使用mysqrt
60         double w1 = (-B - mysqrt(delta)) / (2 * A);
61         double w2 = (-B + mysqrt(delta)) / (2 * A);
62         if (sgn(w1 - 1) <= 0 && sgn(w1) >= 0) {
63             res.push_back(l.s + w1 * (l.t - l.s));
64         }
65         if (sgn(w2 - 1) <= 0 && sgn(w2) >= 0 && fabs(w1 - w2) > eps) {
66             res.push_back(l.s + w2 * (l.t - l.s));
67         }
68     }

```

```

68     }
69     return res;
70 }
71 // 圆与直线交
72 friend vector<Point> lineCircleIntersection(const Line &l, const Circle &c) {
73     double dx = l.t.x - l.s.x, dy = l.t.y - l.s.y;
74     double A = dx * dx + dy * dy;
75     double B = 2 * dx * (l.s.x - c.o.x) + 2 * dy * (l.s.y - c.o.y);
76     double C = sqr(l.s.x - c.o.x) + sqr(l.s.y - c.o.y) - sqr(c.r);
77     double delta = B * B - 4 * A * C;
78     vector<Point> res;
79     if (A < eps) return res;
80     if (sgn(delta) >= 0) { // or delta > -eps ?
81         double w1 = (-B - mysqrt(delta)) / (2 * A);
82         double w2 = (-B + mysqrt(delta)) / (2 * A);
83         res.push_back(l.s + w1 * (l.t - l.s));
84         if (fabs(w1 - w2) > eps) res.push_back(l.s + w2 * (l.t - l.s));
85     }
86     return res;
87 }
88 // 计算圆与圆的交点 保证圆不重合
89 friend vector<Point> cirCirIntersection(Circle a, Circle b) {
90     Point c1 = a.o;
91     vector<Point> vec;
92     if (dis(a.o, b.o) + eps > a.r + b.r &&
93         dis(a.o, b.o) < fabs(a.r - b.r) + eps)
94         return vec;
95     Line L;
96     double t = (1.0 + (sqr(a.r) - sqr(b.r)) / sqr(dis(a.o, b.o))) / 2;
97     L.s = c1 + (b.o - a.o) * t;
98     L.t.x = L.s.x + a.o.y - b.o.y;
99     L.t.y = L.s.y - a.o.x + b.o.x;
100    return lineCircleIntersection(L, a);
101 }
102 // 将向量p逆时针旋转angle角度
103 // 求圆外一点对圆(o,r)的切点
104 friend vector<Point> tangentPointCircle(Point poi, Circle C) {
105     Point o = C.o;
106     double r = C.r;
107     vector<Point> vec;
108     double dist = (poi - o).norm();
109     if (dist < r - eps) return vec;
110     if (fabs(dist - r) < eps) {
111         vec.push_back(poi);
112         return vec;
113     }
114     Point res1, res2;
115     double line =

```

```

116         sqrt((poi.x - o.x) * (poi.x - o.x) + (poi.y - o.y) * (poi.y - o.y));
117     double angle = acos(r / line);
118     Point unitVector, lin;
119     lin.x = poi.x - o.x;
120     lin.y = poi.y - o.y;
121     unitVector.x = lin.x / sqrt(lin.x * lin.x + lin.y * lin.y) * r;
122     unitVector.y = lin.y / sqrt(lin.x * lin.x + lin.y * lin.y) * r;
123     res1 = rotate(unitVector, -angle) + o;
124     res2 = rotate(unitVector, angle) + o;
125     vec.push_back(res1);
126     vec.push_back(res2);
127     return vec;
128 }
129 // 扇形面积 a->b
130 double sectorArea(const Point &a, const Point &b) const {
131     double theta = atan2(a.y, a.x) - atan2(b.y, b.x);
132     while (theta < 0) theta += 2 * PI;
133     while (theta > 2.0 * PI) theta -= 2 * PI;
134     theta = min(theta, 2.0 * PI - theta);
135     return sgn(det(a, b)) * theta * r * r / 2.0;
136 }
137 // 与线段AB的交点计算面积 a->b
138 double areaSegCircle(const Line &L) const {
139     Point a = L.s, b = L.t;
140     vector<Point> p = segCircleIntersection(Line(a, b), *this);
141     bool ina = sgn((a - o).norm() - r) < 0;
142     bool inb = sgn((b - o).norm() - r) < 0;
143     if (ina) {
144         if (inb)
145             return det(a - o, b - o) / 2;
146         else
147             return det(a - o, p[0] - o) / 2 + sectorArea(p[0] - o, b - o);
148     } else {
149         if (inb)
150             return det(p[0] - o, b - o) / 2 + sectorArea(a - o, p[0] - o);
151         else {
152             if (p.size() == 2)
153                 return sectorArea(a - o, p[0] - o) +
154                     sectorArea(p[1] - o, b - o) +
155                     det(p[0] - o, p[1] - o) / 2;
156             else
157                 return sectorArea(a - o, b - o);
158         }
159     }
160 }
161
162 // 圆与多边形交, 结果可以尝试 +eps
163 friend double areaPolygonCircle(const Circle &c, const Polygon &a) {

```

```

164     int n = a.p.size();
165
166     double ans = 0;
167     for (int i = 0; i < n; ++i) {
168         if (sgn(det(a.p[i] - c.o, a.p[_next(i)] - c.o)) == 0) {
169             continue;
170         }
171         ans += c.areaSegCircle((a.p[i], a.p[_next(i)]));
172     }
173     return ans;
174 }
175 // 两个圆的公共面积
176 friend double areaCircleCircle(const Circle &A, const Circle &B) {
177     double ans = 0.0;
178     Circle M = (A.r > B.r) ? A : B;
179     Circle N = (A.r > B.r) ? B : A;
180     double D = dis(M.o, N.o);
181     if ((D < M.r + N.r) && (D > M.r - N.r)) {
182         double alpha = 2.0 * acos((M.r * M.r + D * D - N.r * N.r) / (2.0 * M.r * D));
183         double beta = 2.0 * acos((N.r * N.r + D * D - M.r * M.r) / (2.0 * N.r * D));
184         ans = (alpha / (2 * PI)) * M.area() + (beta / (2 * PI)) * N.area() -
185             0.5 * M.r * M.r * sin(alpha) - 0.5 * N.r * N.r * sin(beta);
186     } else if (D <= M.r - N.r) {
187         ans = N.area();
188     }
189     return ans;
190 }
191
192 // 三点求圆
193 Circle getCircle3(const Point &p0, const Point &p1, const Point &p2) {
194     double a1 = p1.x - p0.x, b1 = p1.y - p0.y, c1 = (a1 * a1 + b1 * b1) / 2;
195     double a2 = p2.x - p0.x, b2 = p2.y - p0.y, c2 = (a2 * a2 + b2 * b2) / 2;
196     double d = a1 * b2 - a2 * b1;
197     Point o((p0.x + (c1 * b2 - c2 * b1) / d, p0.y + (a1 * c2 - a2 * c1) / d);
198     return Circle(o, (o - p0).norm());
199 }
200 // 直径上两点求圆
201 Circle getCircle2(const Point &p0, const Point &p1) {
202     Point o((p0.x + p1.x) / 2, (p0.y + p1.y) / 2);
203     return Circle(o, (o - p0).norm());
204 }
205 // 最小圆覆盖 用之前可以随机化random_shuffle
206 Circle minCirCover(vector<Point> &a) {
207     int n = a.size();
208     Circle c(a[0], 0);
209     for (int i = 1; i < n; ++i) {
210         if (!c.pointIn(a[i])) {
211             c.o = a[i];

```

```

212         c.r = 0;
213         for (int j = 0; j < i; ++j) {
214             if (!c.pointIn(a[j])) {
215                 c = getCircle2(a[i], a[j]);
216                 for (int k = 0; k < j; ++k) {
217                     if (!c.pointIn(a[k])) {
218                         c = getCircle3(a[i], a[j], a[k]);
219                     }
220                 }
221             }
222         }
223     }
224 }
225 return c;
226 }
227 // 线段在圆内的长度
228 friend double lengthSegInCircle(Line a, Circle c) {
229     if (c.pointIn(a.s) && c.pointIn(a.t)) return a.norm();
230     vector<Point> vec = segCircleIntersection(a, c);
231     if (vec.size() == 0) return 0;
232     if (vec.size() == 1) {
233         if (c.pointIn(a.s)) return dis(vec[0], a.s);
234         if (c.pointIn(a.t)) return dis(vec[0], a.t);
235         return 0;
236     }
237     return dis(vec[0], vec[1]);
238 }
239 // 多边形在圆内的长度
240 friend double lengthPolygonInCircle(Polygon a, Circle c) {
241     double ans = 0;
242     for (int i = 0; i < a.n; ++i) {
243         Line li;
244         li.s = a.p[i];
245         li.t = a.p[(i + 1) % a.n];
246         ans += lengthSegInCircle(li, c);
247     }
248     return ans;
249 }
250 // 圆b在圆a内的长度
251 friend double lengthCircleInCircle(Circle a, Circle b) {
252     if (a.r > b.r && a.r - b.r + eps > dis(a.o, b.o)) return b.perimeter();
253     vector<Point> vec = cirCirIntersection(a, b);
254     if (vec.size() < 2) return 0;
255     // Line l1 = (vec[0], b.o), l2 = (vec[1], b.o);
256     double ans = b.r * arg_3(vec[0], b.o, vec[1]);
257     if (b.r >= a.r || !a.pointIn(b.o)) return b.r * ans;
258     return b.perimeter() - ans;
259 }

```

260 | };

0.2.2 HalfPlane.cpp

```

1  #include "PolygonAndConvex.cpp"
2
3  const int inf = 1e9;
4
5  struct HalfPlane : public Line { // 半平面
6      //  $ax + by + c \leq 0$ 
7      double a, b, c;
8      // s->t 的左侧表示半平面
9      HalfPlane(const Point &s = Point(), const Point &t = Point()) : Line(s, t) {
10         a = t.y - s.y;
11         b = s.x - t.x;
12         c = det(t, s);
13     }
14     HalfPlane(double a, double b, double c) : a(a), b(b), c(c) {}
15     // 求点p带入直线方程的值
16     double calc(const Point &p) const { return p.x * a + p.y * b + c; }
17     // 好像跟lineIntersection一样, 那个是4个点计算。这个是用abc与两点进行计算
18     friend Point halfxLine(const HalfPlane &h, const Line &l) {
19         Point res;
20         double t1 = h.calc(l.s), t2 = h.calc(l.t);
21         res.x = (t2 * l.s.x - t1 * l.t.x) / (t2 - t1);
22         res.y = (t2 * l.s.y - t1 * l.t.y) / (t2 - t1);
23         return res;
24     }
25     // 用 abc 进行计算 尚未测试
26     friend Point halfxHalf(const HalfPlane &h1, const HalfPlane &h2) {
27         return Point(
28             (h1.b * h2.c - h1.c * h2.b) / (h1.a * h2.b - h2.a * h1.b) + eps,
29             (h1.a * h2.c - h2.a * h1.c) / (h1.b * h2.a - h1.a * h2.b) + eps);
30     }
31     // 凸多边形与半平面交(cut)
32     friend Convex halfxConvex(const HalfPlane &h, const Convex &c) {
33         Convex res;
34         for (int i = 0; i < c.n; ++i) {
35             if (h.calc(c.p[i]) < -eps)
36                 res.p.push_back(c.p[i]);
37             else {
38                 int j = i - 1;
39                 if (j < 0) j = c.n - 1;
40                 if (h.calc(c.p[j]) < -eps)
41                     res.p.push_back(halfxLine(h, Line(c.p[j], c.p[i])));
42                 j = i + 1;
43                 if (j == c.n) j = 0;

```

```

44         if (h.calc(c.p[j]) < -eps) {
45             res.p.push_back(halfxLine(h, Line(c.p[i], c.p[j])));
46         }
47     }
48 }
49 res.n = res.p.size();
50 return res;
51 }
52 // 点在半平面内
53 friend int satisfy(const Point &p, const HalfPlane &h) {
54     return sgn(det(p - h.s, h.t - h.s)) <= 0;
55 }
56 friend bool operator<(const HalfPlane &h1, const HalfPlane &h2) {
57     int res = sgn(h1.vec().arg() - h2.vec().arg());
58     return res == 0 ? satisfy(h1.s, h2) : res < 0;
59 }
60 // 半平面交出的凸多边形
61 friend Convex halfx(vector<HalfPlane> &v) {
62     sort(v.begin(), v.end());
63     deque<HalfPlane> q;
64     deque<Point> ans;
65     q.push_back(v[0]);
66     for (int i = 1; i < v.size(); ++i) {
67         if (sgn(v[i].vec().arg() - v[i - 1].vec().arg()) == 0) continue;
68         while (ans.size() > 0 && !satisfy(ans.back(), v[i])) {
69             ans.pop_back();
70             q.pop_back();
71         }
72         while (ans.size() > 0 && !satisfy(ans.front(), v[i])) {
73             ans.pop_front();
74             q.pop_front();
75         }
76         ans.push_back(lineIntersection(q.back(), v[i]));
77         q.push_back(v[i]);
78     }
79     while (ans.size() > 0 && !satisfy(ans.back(), q.front())) {
80         ans.pop_back();
81         q.pop_back();
82     }
83     while (ans.size() > 0 && !satisfy(ans.front(), q.back())) {
84         ans.pop_front();
85         q.pop_front();
86     }
87     ans.push_back(lineIntersection(q.back(), q.front()));
88     Convex c(ans.size());
89     int i = 0;
90     for (deque<Point>::iterator it = ans.begin(); it != ans.end();
91         ++it, ++i) {

```



```

92         c.p[i] = *it;
93     }
94     return c;
95 }
96 };
97 // 多边形的核, 逆时针
98 Convex core(const Polygon &a) {
99     Convex res;
100     res.p.push_back(Point(-inf, -inf));
101     res.p.push_back(Point(inf, -inf));
102     res.p.push_back(Point(inf, inf));
103     res.p.push_back(Point(-inf, inf));
104     res.n = 4;
105     for (int i = 0; i < a.n; i++) {
106         res = halfxConvex(HalfPlane(a.p[i], a.p[(i + 1) % a.n]), res);
107     }
108     return res;
109 }
110 // 凸多边形交出的凸多边形
111 Convex convexxConvex(Convex &c1, Convex &c2) {
112     vector<HalfPlane> h;
113     for (int i = 0; i < c1.p.size(); ++i)
114         h.push_back(HalfPlane(c1.p[i], c1.p[(i + 1) % c1.p.size()]));
115     for (int i = 0; i < c2.p.size(); i++)
116         h.push_back(HalfPlane(c2.p[i], c2.p[(i + 1) % c2.p.size()]));
117     return halfx(h);
118 }

```

0.2.3 Line.cpp

```

1  #include "Point.cpp"
2
3  const double PI = acos(-1);
4  struct Line {
5      int id;
6      Point s, t;
7      Line(const Point &s = Point(), const Point &t = Point()) : s(s), t(t) {}
8
9      Point vec() const { return t - s; }           // 化成矢量
10     double norm() const { return vec().norm(); }   // 线段长度
11     // 点是否在直线上
12     bool pointOnLine(const Point &p) {
13         return sgn(det(p - s, t - s)) == 0;
14     }
15     // 点是否在线段上, 含线段端点
16     bool pointOnSeg(const Point &p) {
17         return pointOnLine(p) && sgn(dot(p - s, p - t)) <= 0;

```

```

18     }
19     // 点是否在线段上, 不含线段端点
20     bool pointOnSegInterval(const Point &p) {
21         return pointOnLine(p) && sgn(dot(p - s, p - t) < 0);
22     }
23     // 点到直线的垂足
24     Point pedalPointLine(const Point &p) {
25         return s + vec() * ((dot(p - s, vec()) / norm()) / norm());
26     }
27     // 点到直线的距离
28     double disPointLine(const Point &p) {
29         return fabs(det(p - s, vec()) / norm());
30     }
31     // 点到线段的距离
32     double disPointSeg(const Point &p) {
33         if (sgn(dot(p - s, t - s)) < 0) return (p - s).norm();
34         if (sgn(dot(p - t, s - t)) < 0) return (p - t).norm();
35         return disPointLine(p);
36     }
37     // 计算点 p 与直线的关系, 返回ONLINE、LEFT、RIGHT 上0 左-1 右1
38     int relation(const Point &p) { return sgn(det(t - s, p - s)); }
39     // 判断 a, b 是否在直线的同侧或者同时在直线上
40     bool sameSide(const Point &a, const Point &b) {
41         return relation(a) == relation(b);
42     }
43     // 二维平面上点 p 关于直线的对称点
44     Point symPoint(const Point &p) {
45         return 2.0 * s - p + 2.0 * (t - s) * dot(p - s, t - s) / ((t.x - s.x) * (t.x - s.x) + (t.y -
            s.y) * (t.y - s.y));
46     }
47     // 判断两直线是否平行
48     friend bool isParallel(const Line &l1, const Line &l2) {
49         return sgn(det(l1.vec(), l2.vec())) == 0;
50     }
51     // 利用相似三角形对应成比例求两直线的交点
52     friend Point lineIntersection(const Line &l1, const Line &l2) {
53         double s1 = det(l1.s - l2.s, l2.vec());
54         double s2 = det(l1.t - l2.s, l2.vec());
55         return (l1.t * s1 - l1.s * s2) / (s1 - s2);
56     }
57     // 求两直线交点的另一种方法
58     friend Point getLineIntersection(const Line &u, const Line &v) {
59         return u.s + (u.t - u.s) * det(u.s - v.s, v.s - v.t) /
            det(u.s - u.t, v.s - v.t);
60     }
61     }
62     // 判断直线l1和线段l2是否相交
63     friend bool isLineSegIntersection(Line l1, Line l2) {
64         return l1.relation(l2.s) * l1.relation(l2.t) <= 0;

```

```

65     }
66     // 判断线段交，返回是否有交点
67     friend bool isSegIntersection(Line l1, Line l2) {
68         if (!sgn(det(l2.s - l1.s, l1.vec())) &&
69             !sgn(det(l2.t - l1.t, l1.vec()))) {
70             return l1.pointOnSeg(l2.s) || l1.pointOnSeg(l2.t) ||
71                 l2.pointOnSeg(l1.s) || l2.pointOnSeg(l1.t);
72         }
73         return !l1.sameSide(l2.s, l2.t) && !l2.sameSide(l1.s, l1.t);
74     }
75
76     // 规范相交，两线段仅有一个非端点处的交点
77     // 判断线段相交，并求线段交点，1规范相交，2相交，0不交
78     friend int segSegIntersection(Line l1, Line l2, Point &p) {
79         Point a, b, c, d;
80         a = l1.s;
81         b = l1.t;
82         c = l2.s;
83         d = l2.t;
84         double s1, s2, s3, s4;
85         int d1, d2, d3, d4;
86         d1 = sgn(s1 = det(b - a, c - a)); // l1.relation(l2.s);
87         d2 = sgn(s2 = det(b - a, d - a)); // l1.relation(l2.t);
88         d3 = sgn(s3 = det(d - c, a - c)); // l2.relation(l1.s);
89         d4 = sgn(s4 = det(d - c, b - c)); // l2.relation(l1.t);
90
91         // 若规范相交则求交点的代码
92         if (d1 * d2 < 0 && d3 * d4 < 0) {
93             p.x = (c.x * s2 - d.x * s1) / (s2 - s1);
94             p.y = (c.y * s2 - d.y * s1) / (s2 - s1);
95             return 1;
96         }
97
98         // 判断非规范相交
99         // d1 == 0, 则证明a, b, c三点共线;
100        // 如果sgn(dot(a - c, b - c)) < 0, 则说明点c在点a, b的中间;
101        // 如果sgn(dot(a - c, b - c)) == 0, 则说明点c与线段ab的端点a, 或者b重合。
102        // 如果sgn(dot(a - c, b - c)) > 0, 则说明点c在线段ab的外面。
103        if ((d1 == 0 && sgn(dot(a - c, b - c)) <= 0) ||
104            (d2 == 0 && sgn(dot(a - d, b - d)) <= 0) ||
105            (d3 == 0 && sgn(dot(c - a, d - a)) <= 0) ||
106            (d4 == 0 && sgn(dot(c - b, d - b)) <= 0)) {
107            return 2;
108        }
109        return 0;
110    }
111
112    // 直线沿法向量(指向直线逆时针方向, 若需要顺时针则移动 -d) 移动 d 距离

```

```

113     friend Line move(const Line &l, const double &d) {
114         Point t = l.vec();
115         t = t / t.norm();
116         t = rotate(t, PI / 2);
117         return Line(l.s + t * d, l.t + t * d);
118     }
119     // 计算线段 l1 到线段 l2 的最短距离
120     friend double disSegSeg(Line &l1, Line &l2) {
121         double d1, d2, d3, d4;
122         if (isSegIntersection(l1, l2))
123             return 0;
124         else {
125             d1 = l2.disPointSeg(l1.s);
126             d2 = l2.disPointSeg(l1.t);
127             d3 = l1.disPointSeg(l2.s);
128             d4 = l1.disPointSeg(l2.t);
129             return min(min(d1, d2), min(d3, d4));
130         }
131     }
132     // 两直线的夹角, 返回 [0, PI] 弧度
133     friend double argLineLine(Line l1, Line l2) {
134         Point u = l1.vec();
135         Point v = l2.vec();
136         return acos(dot(u, v) / (u.norm() * v.norm()));
137     }
138 };

```

0.2.4 Point.cpp

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4  using ll = long long;
5
6  const double eps = 1e-8;
7
8  int sgn(double x) { return abs(x) < eps ? 0 : (x > 0 ? 1 : -1); }
9
10 struct Point { // Point & Vector
11     double x, y;
12     Point(const double &x = 0, const double &y = 0) : x(x), y(y) {}
13
14     friend Point operator+(const Point &a, const Point &b) {
15         return Point(a.x + b.x, a.y + b.y);
16     }
17     friend Point operator-(const Point &a, const Point &b) {
18         return Point(a.x - b.x, a.y - b.y);

```

```

19     }
20     friend Point operator*(const double &c, const Point &a) {
21         return Point(c * a.x, c * a.y);
22     }
23     friend Point operator*(const Point &a, const double &c) {
24         return Point(c * a.x, c * a.y);
25     }
26     friend Point operator/(const Point &a, const double &c) {
27         return Point(a.x / c, a.y / c);
28     }
29     friend Point rotate(const Point &v, double theta) { // 向量逆时针旋转theta弧度
30         return Point(v.x * cos(theta) - v.y * sin(theta),
31                     v.x * sin(theta) + v.y * cos(theta));
32     }
33     friend Point rotateAroundPoint(Point &v, Point &p, double theta) {
34         return rotate(v - p, theta) + p;
35     }
36     friend bool operator==(const Point &a, const Point &b) {
37         return !sgn(a.x - b.x) && !sgn(a.y - b.y);
38     }
39     friend bool operator<(const Point &a, const Point &b) {
40         return sgn(a.x - b.x) < 0 || (!sgn(a.x - b.x) && sgn(a.y - b.y) < 0);
41     }
42     // 向量模
43     double norm() { return sqrt(x * x + y * y); }
44     // 向量叉积
45     friend double det(const Point &a, const Point &b) {
46         return a.x * b.y - a.y * b.x;
47     }
48     // 向量点积
49     friend double dot(const Point &a, const Point &b) {
50         return a.x * b.x + a.y * b.y;
51     }
52     // 两点间距离
53     friend double dis(const Point &a, const Point &b) {
54         return sqrt((a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b.y));
55     }
56     friend Point intersection(Point u1, Point u2, Point v1, Point v2) { // 线段交点，线段有交点才可用
57         return u1 + (u2 - u1) * det(u1 - v1, v1 - v2) / det(u1 - u2, v1 - v2);
58     }
59     double arg() { return atan2(y, x); } // 返回弧度
60     friend double arg_2(Point u, Point v) {
61         return acos(dot(u, v) / (u.norm() * v.norm()));
62     } // 两向量之间的夹角
63     friend double arg_3(const Point &a, const Point &b, const Point &c) {
64         return arg_2(a - b, c - b);
65     } // abc

```

66 |};

0.2.5 PolygonAndConvex.cpp

```

1  #include "Line.cpp"
2
3  struct Polygon {
4  #define _next(i) ((i + 1) % n)
5      int n;
6      vector<Point> p;
7
8      Polygon(vector<Point> &v) : p(v) { n = p.size(); }
9      Polygon(int n = 0) : n(n) { p.resize(n); }
10
11     void addPoint(Point &a) {
12         p.push_back(a);
13         n++;
14     }
15     // 多边形周长
16     double perimeter() {
17         double sum = 0;
18         for (int i = 0; i < n; ++i) sum += (p[_next(i)] - p[i]).norm();
19         return sum;
20     }
21     // 多边形面积
22     double area() {
23         double sum = 0;
24         for (int i = 0; i < n; ++i) sum += det(p[i], p[_next(i)]);
25         return fabs(sum) / 2;
26     } // eps
27     // 判断点与多边形的位置关系 0外, 1内, 2边上
28     int pointIn(const Point &t) {
29         int num = 0;
30         for (int i = 0; i < n; i++) {
31             if (Line(p[i], p[_next(i)]).pointOnSeg(t)) return 2;
32             int k = sgn(det(p[_next(i)] - p[i], t - p[i]));
33             int d1 = sgn(p[i].y - t.y);
34             int d2 = sgn(p[_next(i)].y - t.y);
35             if (k > 0 && d1 <= 0 && d2 > 0) num++;
36             if (k < 0 && d2 <= 0 && d1 > 0) num--;
37         }
38         return num % 2;
39     }
40     // 多边形重心
41     Point baryCenter() {
42         Point ans;
43         if (sgn(area()) == 0) return ans;

```

```

44     for (int i = 0; i < n; ++i)
45         ans = ans + (p[i] + p[_next(i)]) * det(p[i], p[_next(i)]);
46     return ans / area() / 6 + eps; // 要加eps吗?
47 }
48 // 判断多边形是否为凸多边形 (需要已经排好序)
49 bool isConvex() { //不允许3点共线
50     int s[3] = {1, 1, 1};
51     for (int i = 0; i < n && (s[0] || s[2]) && s[1]; ++i) {
52         s[1 + sgn(det(p[_next(i)] - p[i], p[_next(_next(i))] - p[i]))] = 0;
53     }
54     return (s[0] || s[2]) && s[1];
55 }
56 bool isConvex_3() { // 允许3点共线
57     int s[3] = {1, 1, 1};
58     for (int i = 0; i < n && (s[0] || s[2]); ++i) {
59         s[1 + sgn(det(p[_next(i)] - p[i], p[_next(_next(i))] - p[i]))] = 0;
60     }
61     return (s[0] || s[2]);
62 }
63 // 多边形边界上格点的数量
64 long long borderPointNum() {
65     long long num = 0;
66     for (int i = 0; i < n; ++i) {
67         num += gcd((long long)fabs(p[_next(i)].x - p[i].x),
68                 (long long)fabs(p[_next(i)].y - p[i].y));
69     }
70     return num;
71 }
72 // 多边形内格点数量
73 long long inSidePointNum() {
74     return (long long)(area()) + 1 - borderPointNum() / 2;
75 }
76 // 点 p 在以 l1l2 为对角线的矩形内边界上
77 inline int dotOnlineIn(Point p, Point l1, Point l2) {
78     return sgn(det(p - l2, l1 - l2)) && (l1.x - p.x) * (l2.x - p.x) < eps &&
79         (l1.y - p.y) * (l2.y - p.y) < eps;
80 }
81 // 判线段在任意多边形内,顶点按顺时针或逆时针给出,与边界相交返回1
82 int insidePolygon(Line l) {
83     vector<Point> t;
84     Point tt, l1 = l.s, l2 = l.t;
85     if (!pointIn(l.s) || !pointIn(l.t)) return 0;
86     for (int i = 0; i < n; ++i) {
87         if (l.sameSide(p[i], p[(i + 1) % n]) &&
88             l.sameSide(p[i], p[(i + 1) % n]))
89             return 0;
90         else if (dotOnlineIn(l1, p[i], p[(i + 1) % n]))
91             t.push_back(l1);

```

```

92         else if (dotOnlineIn(l2, p[i], p[(i + 1) % n]))
93             t.push_back(l2);
94         else if (dotOnlineIn(p[i], l1, l2))
95             t.push_back(p[i]);
96     }
97     for (int i = 0; i < t.size(); ++i) {
98         for (int j = i + 1; j < t.size(); ++j) {
99             if (!pointIn((t[i] + t[j]) / 2)) return 0;
100         }
101     }
102     return 1;
103 }
104 };
105
106 struct Convex : public Polygon {
107     Convex(int n = 0) : Polygon(n) {}
108     Convex(vector<Point> &a) { // 传入n个点构造凸包
109         Convex res(a.size() * 2 + 7);
110         sort(a.begin(), a.end());
111         a.erase(unique(a.begin(), a.end()), a.end()); // 去重点
112         int m = 0;
113         for (int i = 0; i < a.size(); ++i) {
114             // <0 则允许3点共线, <=0 则不允许
115             while (m > 1 && sgn(det(res.p[m - 1] - res.p[m - 2], a[i] - res.p[m - 2])) <= 0)
116                 m--;
117             res.p[m++] = a[i];
118         }
119         int k = m;
120         for (int i = a.size() - 2; i >= 0; --i) {
121             while (m > k && sgn(det(res.p[m - 1] - res.p[m - 2], a[i] - res.p[m - 2])) <= 0) {
122                 m--;
123             }
124             res.p[m++] = a[i];
125         }
126         if (m > 1) m--;
127         res.p.resize(m);
128         res.n = m;
129         *this = res;
130     }
131
132     // 需要先求凸包, 若凸包每条边除端点外都有点, 则可唯一确定凸包
133     bool isUnique(vector<Point> &v) {
134         if (sgn(area()) == 0) return 0;
135         for (int i = 0; i < n; ++i) {
136             Line l(p[i], p[_next(i)]);
137             bool flag = 0;
138             for (int j = 0; j < v.size(); ++j) {
139                 if (l.pointOnSegInterval(v[j])) {

```



```

140         flag = 1;
141         break;
142     }
143 }
144 if (!flag) return 0;
145 }
146 return 1;
147 }
148 // O(n)时间内判断点是否在凸包内 包含边
149 bool containon(const Point &a) {
150     for (int sign = 0, i = 0; i < n; ++i) {
151         int x = sgn(det(p[i] - a, p[_next(i)] - a));
152         if (x == 0) continue; // return 0; // 改成不包含边
153         if (!sign)
154             sign = x;
155         else if (sign != x)
156             return 0;
157     }
158     return 1;
159 }
160 // O(logn)时间内判断点是否在凸包内
161 bool containologn(const Point &a) {
162     Point g = (p[0] + p[n / 3] + p[2.0 * n / 3]) / 3.0;
163     int l = 0, r = n;
164     while (l + 1 < r) {
165         int m = (l + r) >> 1;
166         if (sgn(det(p[l] - g, p[m] - g)) > 0) {
167             if (sgn(det(p[l] - g, a - g)) >= 0 &&
168                 sgn(det(p[m] - g, a - g)) < 0)
169                 r = m;
170             else
171                 l = m;
172         } else {
173             if (sgn(det(p[l] - g, a - g)) < 0 &&
174                 sgn(det(p[m] - g, a - g)) >= 0)
175                 l = m;
176             else
177                 r = m;
178         }
179     }
180     return sgn(det(p[r % n] - a, p[l] - a)) - 1;
181 }
182 // 最远点对 (直径)
183 int fir, sec; // 最远的两个点对应标号
184 double diameter() {
185     double mx = 0;
186     if (n == 1) {
187         fir = sec = 0;

```

```

188         return mx;
189     }
190     for (int i = 0, j = 1; i < n; ++i) {
191         while (sgn(det(p[_next(i)] - p[i], p[j] - p[i]) -
192             det(p[_next(i)] - p[i], p[_next(j)] - p[i])) < 0) {
193             j = _next(j);
194         }
195         double d = dis(p[i], p[j]);
196         if (d > mx) {
197             mx = d;
198             fir = i;
199             sec = j;
200         }
201         d = dis(p[_next(i)], p[_next(j)]);
202         if (d > mx) {
203             mx = d;
204             fir = _next(i);
205             sec = _next(j);
206         }
207     }
208     return mx;
209 }
210
211 // 凸包是否与直线有交点O(log(n)), 需要On的预处理, 适合判断与直线集是否有交点
212 vector<double> ang; // 角度
213 bool isinitangle;
214 int finda(const double &x) {
215     return upper_bound(ang.begin(), ang.end(), x) - ang.begin();
216 }
217 double getAngle(const Point &p) { // 获取向量角度[0, 2PI]
218     double res = atan2(p.y, p.x); // (-PI, PI]
219     // if (res < 0) res += 2 * pi; //为何不可以
220     if (res < -PI / 2 + eps) res += 2 * PI; // eps修正精度
221     return res;
222 }
223 void initAngle() {
224     for (int i = 0; i < n; ++i) {
225         ang.push_back(getAngle(p[_next(i)] - p[i]));
226     }
227     isinitangle = 1;
228 }
229 bool isxLine(const Line &l) {
230     if (!isinitangle) initAngle();
231     int i = finda(getAngle(l.t - l.s));
232     int j = finda(getAngle(l.s - l.t));
233     if (sgn(det(l.t - l.s, p[i] - l.s) * det(l.t - l.s, p[j] - l.s) >= 0))
234         return 0;
235     return 1;

```

```

236     }
237 };

```

0.2.6 Triangle.cpp

```

1  #include "Line.cpp"
2
3  struct Triangle {
4      Triangle(const Point &a, const Point &b, const Point &c)
5          : a(a), b(b), c(c){};
6      Point a, b, c;
7      double getArea() { return det(b - a, c - a) * sin(arg_2(b - c, c - a)); }
8      // 外心
9      Point outCenter() {
10         Line u, v;
11         u.s = (a + b) / 2;
12         u.t.x = u.s.x - a.y + b.y;
13         u.t.y = u.s.y + a.x - b.x;
14         v.s = (a + c) / 2;
15         v.t.x = v.s.x - a.y + c.y;
16         v.t.y = v.s.y + a.x - c.x;
17         return lineIntersection(u, v);
18     }
19     // 内心
20     Point inCenter() {
21         Line u, v;
22         u.s = a;
23         double m = atan2(b.y - a.y, b.x - a.x);
24         double n = atan2(c.y - a.y, c.x - a.x);
25         u.t.x = u.s.x + cos((m + n) / 2);
26         u.t.y = u.s.y + sin((m + n) / 2);
27         v.s = b;
28         m = atan2(a.y - b.y, a.x - b.x);
29         n = atan2(c.y - b.y, c.x - b.x);
30         v.t.x = v.s.x + cos((m + n) / 2);
31         v.t.y = v.s.y + sin((m + n) / 2);
32         return lineIntersection(u, v);
33     }
34     // 垂心
35     Point perpenCenter() {
36         Line u, v;
37         u.s = c;
38         u.t.x = u.s.x - a.y + b.y;
39         u.t.y = u.s.y + a.x - b.x;
40         v.s = b;
41         v.t.x = v.s.x - a.y + c.y;
42         v.t.y = v.s.y + a.x - c.x;

```

```

43     return lineIntersection(u, v);
44 }
45
46 // 重心
47 // 到三角形三顶点距离的平方和最小的点
48 // 三角形内到三边距离之积最大的点
49 Point baryCenter() {
50     Line u((a + b) / 2, c), v((a + c) / 2, b);
51     return lineIntersection(u, v);
52 }
53
54 // 费马点 到三角形三顶点距离之和最小的点
55 Point fermentPoint() {
56     if (arg_3(a, b, c) >= 2 * PI / 3) return b;
57     if (arg_3(b, a, c) >= 2 * PI / 3) return a;
58     if (arg_3(a, c, b) >= 2 * PI / 3) return c;
59     Point ab = (a + b) / 2, ac = (a + c) / 2;
60     Point z1 = sqrt(3.0) * (a - ab), z2 = sqrt(3.0) * (a - ac);
61     z1 = rotate(z1, PI / 2);
62     z2 = rotate(z2, PI / 2);
63     if (arg_2(z1, c - ab) < PI / 2) {
64         z1.x = -z1.x;
65         z1.y = -z1.y;
66     }
67     if (arg_2(z2, b - ac) < PI / 2) {
68         z2.x = -z2.x;
69         z2.y = -z2.y;
70     }
71     return intersection(c, ab + z1, b, ac + z2);
72 }
73 // 模拟退火求费马点
74 Point FermatPoint() {
75     Point u, v;
76     double step = fabs(a.x) + fabs(a.y) + fabs(b.x) + fabs(b.y) + fabs(c.x) + fabs(c.y);
77     u = (a + b + c) / 3;
78     while (step > 1e-10)
79         for (int k = 0; k < 10; step /= 2, ++k)
80             for (int i = -1; i <= 1; ++i) {
81                 for (int j = -1; j <= 1; ++j) {
82                     v.x = u.x + step * i;
83                     v.y = u.y + step * j;
84                     if (dis(u, a) + dis(u, b) + dis(u, c) > dis(v, a) + dis(v, b) + dis(v, c)) {
85                         u = v;
86                     }
87                 }
88             }
89     return u;
90 }

```

91 |};

0.2.7 mygeo.cpp

```

1  #include <bits/stdc++.h>
2  using namespace std;
3
4  #define mp make_pair
5  #define fi first
6  #define se second
7  #define pb push_back
8  typedef double db;
9  const db eps = 1e-6;
10 const db pi = acos(-1);
11 int sign(db k) {
12     if (k > eps)
13         return 1;
14     else if (k < -eps)
15         return -1;
16     return 0;
17 }
18 int cmp(db k1, db k2) { return sign(k1 - k2); }
19 int inmid(db k1, db k2, db k3) {
20     return sign(k1 - k3) * sign(k2 - k3) <= 0;
21 } // k3 在 [k1,k2] 内
22 struct point {
23     db x, y;
24     point operator+(const point &k1) const {
25         return (point){k1.x + x, k1.y + y};
26     }
27     point operator-(const point &k1) const {
28         return (point){x - k1.x, y - k1.y};
29     }
30     point operator*(db k1) const { return (point){x * k1, y * k1}; }
31     point operator/(db k1) const { return (point){x / k1, y / k1}; }
32     int operator==(const point &k1) const {
33         return cmp(x, k1.x) == 0 && cmp(y, k1.y) == 0;
34     }
35     // 逆时针旋转
36     point turn(db k1) {
37         return (point){x * cos(k1) - y * sin(k1), x * sin(k1) + y * cos(k1)};
38     }
39     point turn90() { return (point){-y, x}; }
40     bool operator<(const point k1) const {
41         int a = cmp(x, k1.x);
42         if (a == -1)
43             return 1;

```

```

44     else if (a == 1)
45         return 0;
46     else
47         return cmp(y, k1.y) == -1;
48 }
49 db abs() { return sqrt(x * x + y * y); }
50 db abs2() { return x * x + y * y; }
51 db dis(point k1) { return ((*this) - k1).abs(); }
52 point unit() {
53     db w = abs();
54     return (point){x / w, y / w};
55 }
56 void scan() {
57     double k1, k2;
58     scanf("%lf%lf", &k1, &k2);
59     x = k1;
60     y = k2;
61 }
62 void print() { printf("%.11lf %.11lf\n", x, y); }
63 db getw() { return atan2(y, x); }
64 point getdel() {
65     if (sign(x) == -1 || (sign(x) == 0 && sign(y) == -1))
66         return (*this) * (-1);
67     else
68         return (*this);
69 }
70 int getP() const { return sign(y) == 1 || (sign(y) == 0 && sign(x) == -1); }
71 };
72 int inmid(point k1, point k2, point k3) {
73     return inmid(k1.x, k2.x, k3.x) && inmid(k1.y, k2.y, k3.y);
74 }
75 db cross(point k1, point k2) { return k1.x * k2.y - k1.y * k2.x; }
76 db dot(point k1, point k2) { return k1.x * k2.x + k1.y * k2.y; }
77 db rad(point k1, point k2) { return atan2(cross(k1, k2), dot(k1, k2)); }
78 // -pi -> pi
79 int compareangle(point k1, point k2) {
80     return k1.getP() < k2.getP() ||
81         (k1.getP() == k2.getP() && sign(cross(k1, k2)) > 0);
82 }
83 point proj(point k1, point k2, point q) { // q 到直线 k1,k2 的投影
84     point k = k2 - k1;
85     return k1 + k * (dot(q - k1, k) / k.abs2());
86 }
87 point reflect(point k1, point k2, point q) { return proj(k1, k2, q) * 2 - q; }
88 int clockwise(point k1, point k2,
89     point k3) { // k1 k2 k3 逆时针 1 顺时针 -1 否则 0
90     return sign(cross(k2 - k1, k3 - k1));
91 }

```

```

92 int checkLL(point k1, point k2, point k3,
93     point k4) { // 求直线 (L) 线段 (S)k1,k2 和 k3,k4 的交点
94     return cmp(cross(k3 - k1, k4 - k1), cross(k3 - k2, k4 - k2)) != 0;
95 }
96 point getLL(point k1, point k2, point k3, point k4) {
97     db w1 = cross(k1 - k3, k4 - k3), w2 = cross(k4 - k3, k2 - k3);
98     return (k1 * w2 + k2 * w1) / (w1 + w2);
99 }
100 int intersect(db l1, db r1, db l2, db r2) {
101     if (l1 > r1) swap(l1, r1);
102     if (l2 > r2) swap(l2, r2);
103     return cmp(r1, l2) != -1 && cmp(r2, l1) != -1;
104 }
105 int checkSS(point k1, point k2, point k3, point k4) {
106     return intersect(k1.x, k2.x, k3.x, k4.x) &&
107         intersect(k1.y, k2.y, k3.y, k4.y) &&
108         sign(cross(k3 - k1, k4 - k1)) * sign(cross(k3 - k2, k4 - k2)) <= 0 &&
109         sign(cross(k1 - k3, k2 - k3)) * sign(cross(k1 - k4, k2 - k4)) <= 0;
110 }
111 db disSP(point k1, point k2, point q) {
112     point k3 = proj(k1, k2, q);
113     if (inmid(k1, k2, k3))
114         return q.dis(k3);
115     else
116         return min(q.dis(k1), q.dis(k2));
117 }
118 db disSS(point k1, point k2, point k3, point k4) {
119     if (checkSS(k1, k2, k3, k4))
120         return 0;
121     else
122         return min(min(disSP(k1, k2, k3), disSP(k1, k2, k4)),
123             min(disSP(k3, k4, k1), disSP(k3, k4, k2)));
124 }
125 int onS(point k1, point k2, point q) {
126     return inmid(k1, k2, q) && sign(cross(k1 - q, k2 - k1)) == 0;
127 }
128 struct circle {
129     point o;
130     db r;
131     void scan() {
132         o.scan();
133         scanf("%lf", &r);
134     }
135     int inside(point k) { return cmp(r, o.dis(k)); }
136 };
137 struct line {
138     // p[0]->p[1]
139     point p[2];

```

```

140     line(point k1, point k2) {
141         p[0] = k1;
142         p[1] = k2;
143     }
144     point &operator[](int k) { return p[k]; }
145     int include(point k) { return sign(cross(p[1] - p[0], k - p[0])) > 0; }
146     point dir() { return p[1] - p[0]; }
147     line push() { // 向外 ( 左手边 ) 平移 eps
148         const db eps = 1e-6;
149         point delta = (p[1] - p[0]).turn90().unit() * eps;
150         return {p[0] - delta, p[1] - delta};
151     }
152 };
153 point getLL(line k1, line k2) { return getLL(k1[0], k1[1], k2[0], k2[1]); }
154 int parallel(line k1, line k2) { return sign(cross(k1.dir(), k2.dir())) == 0; }
155 int sameDir(line k1, line k2) {
156     return parallel(k1, k2) && sign(dot(k1.dir(), k2.dir())) == 1;
157 }
158 int operator<(line k1, line k2) {
159     if (sameDir(k1, k2)) return k2.include(k1[0]);
160     return compareangle(k1.dir(), k2.dir());
161 }
162 int checkpos(line k1, line k2, line k3) { return k3.include(getLL(k1, k2)); }
163 vector<line> getHL(
164     vector<line> &L) { // 求半平面交 , 半平面是逆时针方向 , 输出按照逆时针
165     sort(L.begin(), L.end());
166     deque<line> q;
167     for (int i = 0; i < (int)L.size(); i++) {
168         if (i && sameDir(L[i], L[i - 1])) continue;
169         while (q.size() > 1 &&
170             !checkpos(q[q.size() - 2], q[q.size() - 1], L[i]))
171             q.pop_back();
172         while (q.size() > 1 && !checkpos(q[1], q[0], L[i])) q.pop_front();
173         q.push_back(L[i]);
174     }
175     while (q.size() > 2 && !checkpos(q[q.size() - 2], q[q.size() - 1], q[0]))
176         q.pop_back();
177     while (q.size() > 2 && !checkpos(q[1], q[0], q[q.size() - 1]))
178         q.pop_front();
179     vector<line> ans;
180     for (int i = 0; i < q.size(); i++) ans.push_back(q[i]);
181     return ans;
182 }
183 db closepoint(vector<point> &A, int l,
184     int r) { // 最近点对 , 先要按照 x 坐标排序
185     if (r - l <= 5) {
186         db ans = 1e20;
187         for (int i = l; i <= r; i++)

```



```

188         for (int j = i + 1; j <= r; j++) ans = min(ans, A[i].dis(A[j]));
189     return ans;
190 }
191 int mid = (l + r) >> 1;
192 db ans = min(closepoint(A, l, mid), closepoint(A, mid + 1, r));
193 vector<point> B;
194 for (int i = l; i <= r; i++)
195     if (abs(A[i].x - A[mid].x) <= ans) B.push_back(A[i]);
196 sort(B.begin(), B.end(), [](point k1, point k2) { return k1.y < k2.y; });
197 for (int i = 0; i < B.size(); i++)
198     for (int j = i + 1; j < B.size() && B[j].y - B[i].y < ans; j++)
199         ans = min(ans, B[i].dis(B[j]));
200 return ans;
201 }
202 int checkposCC(circle k1, circle k2) { // 返回两个圆的公切线数量
203     if (cmp(k1.r, k2.r) == -1) swap(k1, k2);
204     db dis = k1.o.dis(k2.o);
205     int w1 = cmp(dis, k1.r + k2.r), w2 = cmp(dis, k1.r - k2.r);
206     if (w1 > 0)
207         return 4;
208     else if (w1 == 0)
209         return 3;
210     else if (w2 > 0)
211         return 2;
212     else if (w2 == 0)
213         return 1;
214     else
215         return 0;
216 }
217 vector<point> getCL(circle k1, point k2,
218     point k3) { // 沿着 k2->k3 方向给出, 相切给出两个
219     point k = proj(k2, k3, k1.o);
220     db d = k1.r * k1.r - (k - k1.o).abs2();
221     if (sign(d) == -1) return {};
222     point del = (k3 - k2).unit() * sqrt(max((db)0.0, d));
223     return {k - del, k + del};
224 }
225 vector<point> getCC(circle k1,
226     circle k2) { // 沿圆 k1 逆时针给出, 相切给出两个
227     int pd = checkposCC(k1, k2);
228     if (pd == 0 || pd == 4) return {};
229     db a = (k2.o - k1.o).abs2(), cosA = (k1.r * k1.r + a - k2.r * k2.r) /
230         (2 * k1.r * sqrt(max(a, (db)0.0)));
231     db b = k1.r * cosA, c = sqrt(max((db)0.0, k1.r * k1.r - b * b));
232     point k = (k2.o - k1.o).unit(), m = k1.o + k * b, del = k.turn90() * c;
233     return {m - del, m + del};
234 }
235 vector<point> TangentCP(circle k1, point k2) { // 沿圆 k1 逆时针给出

```

```

236     db a = (k2 - k1.o).abs(), b = k1.r * k1.r / a,
237     c = sqrt(max((db)0.0, k1.r * k1.r - b * b));
238     point k = (k2 - k1.o).unit(), m = k1.o + k * b, del = k.turn90() * c;
239     return {m - del, m + del};
240 }
241 vector<line> TangentoutCC(circle k1, circle k2) {
242     int pd = checkposCC(k1, k2);
243     if (pd == 0) return {};
244     if (pd == 1) {
245         point k = getCC(k1, k2)[0];
246         return {(line){k, k}};
247     }
248     if (cmp(k1.r, k2.r) == 0) {
249         point del = (k2.o - k1.o).unit().turn90().getdel();
250         return {(line){k1.o - del * k1.r, k2.o - del * k2.r},
251             (line){k1.o + del * k1.r, k2.o + del * k2.r}};
252     } else {
253         point p = (k2.o * k1.r - k1.o * k2.r) / (k1.r - k2.r);
254         vector<point> A = TangentCP(k1, p), B = TangentCP(k2, p);
255         vector<line> ans;
256         for (int i = 0; i < A.size(); i++) ans.push_back((line){A[i], B[i]});
257         return ans;
258     }
259 }
260 vector<line> TangentinCC(circle k1, circle k2) {
261     int pd = checkposCC(k1, k2);
262     if (pd <= 2) return {};
263     if (pd == 3) {
264         point k = getCC(k1, k2)[0];
265         return {(line){k, k}};
266     }
267     point p = (k2.o * k1.r + k1.o * k2.r) / (k1.r + k2.r);
268     vector<point> A = TangentCP(k1, p), B = TangentCP(k2, p);
269     vector<line> ans;
270     for (int i = 0; i < A.size(); i++) ans.push_back((line){A[i], B[i]});
271     return ans;
272 }
273 vector<line> TangentCC(circle k1, circle k2) {
274     int flag = 0;
275     if (k1.r < k2.r) swap(k1, k2), flag = 1;
276     vector<line> A = TangentoutCC(k1, k2), B = TangentinCC(k1, k2);
277     for (line k : B) A.push_back(k);
278     if (flag)
279         for (line &k : A) swap(k[0], k[1]);
280     return A;
281 }
282 db getarea(circle k1, point k2, point k3) {
283     // 圆 k1 与三角形 k2 k3 k1.o 的有向面积交

```

```

284     point k = k1.o;
285     k1.o = k1.o - k;
286     k2 = k2 - k;
287     k3 = k3 - k;
288     int pd1 = k1.inside(k2), pd2 = k1.inside(k3);
289     vector<point> A = getCL(k1, k2, k3);
290     if (pd1 >= 0) {
291         if (pd2 >= 0) return cross(k2, k3) / 2;
292         return k1.r * k1.r * rad(A[1], k3) / 2 + cross(k2, A[1]) / 2;
293     } else if (pd2 >= 0) {
294         return k1.r * k1.r * rad(k2, A[0]) / 2 + cross(A[0], k3) / 2;
295     } else {
296         int pd = cmp(k1.r, disSP(k2, k3, k1.o));
297         if (pd <= 0) return k1.r * k1.r * rad(k2, k3) / 2;
298         return cross(A[0], A[1]) / 2 +
299             k1.r * k1.r * (rad(k2, A[0]) + rad(A[1], k3)) / 2;
300     }
301 }
302 circle getcircle(point k1, point k2, point k3) {
303     db a1 = k2.x - k1.x, b1 = k2.y - k1.y, c1 = (a1 * a1 + b1 * b1) / 2;
304     db a2 = k3.x - k1.x, b2 = k3.y - k1.y, c2 = (a2 * a2 + b2 * b2) / 2;
305     db d = a1 * b2 - a2 * b1;
306     point o =
307         (point){k1.x + (c1 * b2 - c2 * b1) / d, k1.y + (a1 * c2 - a2 * c1) / d};
308     return (circle){o, k1.dis(o)};
309 }
310 circle getScircle(vector<point> A) {
311     // random_shuffle(A.begin(), A.end());
312     circle ans = (circle){A[0], 0};
313     for (int i = 1; i < A.size(); i++)
314         if (ans.inside(A[i]) == -1) {
315             ans = (circle){A[i], 0};
316             for (int j = 0; j < i; j++)
317                 if (ans.inside(A[j]) == -1) {
318                     ans.o = (A[i] + A[j]) / 2;
319                     ans.r = ans.o.dis(A[i]);
320                     for (int k = 0; k < j; k++)
321                         if (ans.inside(A[k]) == -1)
322                             ans = getcircle(A[i], A[j], A[k]);
323                 }
324         }
325     return ans;
326 }
327 db area(vector<point> A) { // 多边形用 vector<point> 表示 , 逆时针
328     db ans = 0;
329     for (int i = 0; i < A.size(); i++)
330         ans += cross(A[i], A[(i + 1) % A.size()]);
331     return ans / 2;

```

```

332 }
333 int checkconvex(vector<point> A) {
334     int n = A.size();
335     A.push_back(A[0]);
336     A.push_back(A[1]);
337     for (int i = 0; i < n; i++)
338         if (sign(cross(A[i + 1] - A[i], A[i + 2] - A[i])) == -1) return 0;
339     return 1;
340 }
341 int contain(vector<point> A, point q) { // 2 内部 1 边界 0 外部
342     int pd = 0;
343     A.push_back(A[0]);
344     for (int i = 1; i < A.size(); i++) {
345         point u = A[i - 1], v = A[i];
346         if (onS(u, v, q)) return 1;
347         if (cmp(u.y, v.y) > 0) swap(u, v);
348         if (cmp(u.y, q.y) >= 0 || cmp(v.y, q.y) < 0) continue;
349         if (sign(cross(u - v, q - v)) < 0) pd ^= 1;
350     }
351     return pd << 1;
352 }
353 vector<point> ConvexHull(vector<point> A,
354                         int flag = 1) { // flag=0 不严格 flag=1 严格
355     int n = A.size();
356     vector<point> ans(n * 2);
357     sort(A.begin(), A.end());
358     int now = -1;
359     for (int i = 0; i < A.size(); i++) {
360         while (now > 0 &&
361                sign(cross(ans[now] - ans[now - 1], A[i] - ans[now - 1])) < flag)
362             now--;
363         ans[++now] = A[i];
364     }
365     int pre = now;
366     for (int i = n - 2; i >= 0; i--) {
367         while (now > pre &&
368                sign(cross(ans[now] - ans[now - 1], A[i] - ans[now - 1])) < flag)
369             now--;
370         ans[++now] = A[i];
371     }
372     ans.resize(now);
373     return ans;
374 }
375 db convexDiameter(vector<point> A) {
376     int now = 0, n = A.size();
377     db ans = 0;
378     for (int i = 0; i < A.size(); i++) {
379         now = max(now, i);

```

```

380     while (1) {
381         db k1 = A[i].dis(A[now % n]), k2 = A[i].dis(A[(now + 1) % n]);
382         ans = max(ans, max(k1, k2));
383         if (k2 > k1)
384             now++;
385         else
386             break;
387     }
388 }
389 return ans;
390 }
391 vector<point> convexcut(vector<point> A, point k1, point k2) {
392     // 保留 k1,k2,p 逆时针的所有点
393     int n = A.size();
394     A.push_back(A[0]);
395     vector<point> ans;
396     for (int i = 0; i < n; i++) {
397         int w1 = clockwise(k1, k2, A[i]), w2 = clockwise(k1, k2, A[i + 1]);
398         if (w1 >= 0) ans.push_back(A[i]);
399         if (w1 * w2 < 0) ans.push_back(getLL(k1, k2, A[i], A[i + 1]));
400     }
401     return ans;
402 }
403 int checkPoS(vector<point> A, point k1, point k2) {
404     // 多边形 A 和直线 ( 线段 )k1->k2 严格相交 , 注释部分为线段
405     struct ins {
406         point m, u, v;
407         int operator<(const ins &k) const { return m < k.m; }
408     };
409     vector<ins> B;
410     // if (contain(A,k1)==2||contain(A,k2)==2) return 1;
411     vector<point> poly = A;
412     A.push_back(A[0]);
413     for (int i = 1; i < A.size(); i++)
414         if (checkLL(A[i - 1], A[i], k1, k2)) {
415             point m = getLL(A[i - 1], A[i], k1, k2);
416             if (inmid(A[i - 1], A[i], m) /*&&inmid(k1,k2,m)*/)
417                 B.push_back((ins){m, A[i - 1], A[i]});
418         }
419     if (B.size() == 0) return 0;
420     sort(B.begin(), B.end());
421     int now = 1;
422     while (now < B.size() && B[now].m == B[0].m) now++;
423     if (now == B.size()) return 0;
424     int flag = contain(poly, (B[0].m + B[now].m) / 2);
425     if (flag == 2) return 1;
426     point d = B[now].m - B[0].m;
427     for (int i = now; i < B.size(); i++) {

```

```

428     if (!(B[i].m == B[i - 1].m) && flag == 2) return 1;
429     int tag = sign(cross(B[i].v - B[i].u, B[i].m + d - B[i].u));
430     if (B[i].m == B[i].u || B[i].m == B[i].v)
431         flag += tag;
432     else
433         flag += tag * 2;
434 }
435 // return 0;
436 return flag == 2;
437 }
438 int checkinp(point r, point l, point m) {
439     if (compareangle(l, r)) {
440         return compareangle(l, m) && compareangle(m, r);
441     }
442     return compareangle(l, m) || compareangle(m, r);
443 }
444 int checkPosFast(vector<point> A, point k1,
445     point k2) { // 快速检查线段是否和多边形严格相交
446     if (contain(A, k1) == 2 || contain(A, k2) == 2) return 1;
447     if (k1 == k2) return 0;
448     A.push_back(A[0]);
449     A.push_back(A[1]);
450     for (int i = 1; i + 1 < A.size(); i++)
451         if (checkLL(A[i - 1], A[i], k1, k2)) {
452             point now = getLL(A[i - 1], A[i], k1, k2);
453             if (inmid(A[i - 1], A[i], now) == 0 || inmid(k1, k2, now) == 0)
454                 continue;
455             if (now == A[i]) {
456                 if (A[i] == k2) continue;
457                 point pre = A[i - 1], ne = A[i + 1];
458                 if (checkinp(pre - now, ne - now, k2 - now)) return 1;
459             } else if (now == k1) {
460                 if (k1 == A[i - 1] || k1 == A[i]) continue;
461                 if (checkinp(A[i - 1] - k1, A[i] - k1, k2 - k1)) return 1;
462             } else if (now == k2 || now == A[i - 1])
463                 continue;
464             else
465                 return 1;
466         }
467     return 0;
468 }
469 // 拆分凸包成上下凸壳 凸包尽量都随机旋转一个角度来避免出现相同横坐标
470 // 尽量特判只有一个点的情况 凸包逆时针
471 void getUDP(vector<point> A, vector<point> &U, vector<point> &D) {
472     db l = 1e100, r = -1e100;
473     for (int i = 0; i < A.size(); i++) l = min(l, A[i].x), r = max(r, A[i].x);
474     int wherel, wherer;
475     for (int i = 0; i < A.size(); i++)

```

```

476         if (cmp(A[i].x, l) == 0) wherel = i;
477     for (int i = A.size(); i; i--)
478         if (cmp(A[i - 1].x, r) == 0) wherer = i - 1;
479     U.clear();
480     D.clear();
481     int now = wherel;
482     while (1) {
483         D.push_back(A[now]);
484         if (now == wherer) break;
485         now++;
486         if (now >= A.size()) now = 0;
487     }
488     now = wherer;
489     while (1) {
490         U.push_back(A[now]);
491         if (now == wherer) break;
492         now--;
493         if (now < 0) now = A.size() - 1;
494     }
495 }
496 // 需要保证凸包点数大于等于 3, 2 内部, 1 边界, 0 外部
497 int containCoP(const vector<point> &U, const vector<point> &D, point k) {
498     db lx = U[0].x, rx = U[U.size() - 1].x;
499     if (k == U[0] || k == U[U.size() - 1]) return 1;
500     if (cmp(k.x, lx) == -1 || cmp(k.x, rx) == 1) return 0;
501     int where1 =
502         lower_bound(U.begin(), U.end(), (point){k.x, -1e100}) - U.begin();
503     int where2 =
504         lower_bound(D.begin(), D.end(), (point){k.x, -1e100}) - D.begin();
505     int w1 = clockwise(U[where1 - 1], U[where1], k),
506         w2 = clockwise(D[where2 - 1], D[where2], k);
507     if (w1 == 1 || w2 == -1)
508         return 0;
509     else if (w1 == 0 || w2 == 0)
510         return 1;
511     return 2;
512 }
513 // d 是方向, 输出上方切点和下方切点
514 pair<point, point> getTangentCow(const vector<point> &U, const vector<point> &D,
515                                 point d) {
516     if (sign(d.x) < 0 || (sign(d.x) == 0 && sign(d.y) < 0)) d = d * (-1);
517     point whereU, whereD;
518     if (sign(d.x) == 0) return mp(U[0], U[U.size() - 1]);
519     int l = 0, r = U.size() - 1, ans = 0;
520     while (l < r) {
521         int mid = (l + r) >> 1;
522         if (sign(cross(U[mid + 1] - U[mid], d)) <= 0)
523             l = mid + 1, ans = mid + 1;

```

```

524         else
525             r = mid;
526     }
527     whereU = U[ans];
528     l = 0, r = D.size() - 1, ans = 0;
529     while (l < r) {
530         int mid = (l + r) >> 1;
531         if (sign(cross(D[mid + 1] - D[mid], d)) >= 0)
532             l = mid + 1, ans = mid + 1;
533         else
534             r = mid;
535     }
536     whereD = D[ans];
537     return mp(whereU, whereD);
538 }
539 // 先检查 contain, 逆时针给出
540 pair<point, point> getTangentCoP(const vector<point> &U, const vector<point> &D,
541                                 point k) {
542     db lx = U[0].x, rx = U[U.size() - 1].x;
543     if (k.x < lx) {
544         int l = 0, r = U.size() - 1, ans = U.size() - 1;
545         while (l < r) {
546             int mid = (l + r) >> 1;
547             if (clockwise(k, U[mid], U[mid + 1]) == 1)
548                 l = mid + 1;
549             else
550                 ans = mid, r = mid;
551         }
552         point w1 = U[ans];
553         l = 0, r = D.size() - 1, ans = D.size() - 1;
554         while (l < r) {
555             int mid = (l + r) >> 1;
556             if (clockwise(k, D[mid], D[mid + 1]) == -1)
557                 l = mid + 1;
558             else
559                 ans = mid, r = mid;
560         }
561         point w2 = D[ans];
562         return mp(w1, w2);
563     } else if (k.x > rx) {
564         int l = 1, r = U.size(), ans = 0;
565         while (l < r) {
566             int mid = (l + r) >> 1;
567             if (clockwise(k, U[mid], U[mid - 1]) == -1)
568                 r = mid;
569             else
570                 ans = mid, l = mid + 1;
571         }

```



```

572     point w1 = U[ans];
573     l = 1, r = D.size(), ans = 0;
574     while (l < r) {
575         int mid = (l + r) >> 1;
576         if (clockwise(k, D[mid], D[mid - 1]) == 1)
577             r = mid;
578         else
579             ans = mid, l = mid + 1;
580     }
581     point w2 = D[ans];
582     return mp(w2, w1);
583 } else {
584     int where1 =
585         lower_bound(U.begin(), U.end(), (point){k.x, -1e100}) - U.begin();
586     int where2 =
587         lower_bound(D.begin(), D.end(), (point){k.x, -1e100}) - D.begin();
588     if ((k.x == lx && k.y > U[0].y) ||
589         (where1 && clockwise(U[where1 - 1], U[where1], k) == 1)) {
590         int l = 1, r = where1 + 1, ans = 0;
591         while (l < r) {
592             int mid = (l + r) >> 1;
593             if (clockwise(k, U[mid], U[mid - 1]) == 1)
594                 ans = mid, l = mid + 1;
595             else
596                 r = mid;
597         }
598         point w1 = U[ans];
599         l = where1, r = U.size() - 1, ans = U.size() - 1;
600         while (l < r) {
601             int mid = (l + r) >> 1;
602             if (clockwise(k, U[mid], U[mid + 1]) == 1)
603                 l = mid + 1;
604             else
605                 ans = mid, r = mid;
606         }
607         point w2 = U[ans];
608         return mp(w2, w1);
609     } else {
610         int l = 1, r = where2 + 1, ans = 0;
611         while (l < r) {
612             int mid = (l + r) >> 1;
613             if (clockwise(k, D[mid], D[mid - 1]) == -1)
614                 ans = mid, l = mid + 1;
615             else
616                 r = mid;
617         }
618         point w1 = D[ans];
619         l = where2, r = D.size() - 1, ans = D.size() - 1;

```

```

620         while (l < r) {
621             int mid = (l + r) >> 1;
622             if (clockwise(k, D[mid], D[mid + 1]) == -1)
623                 l = mid + 1;
624             else
625                 ans = mid, r = mid;
626         }
627         point w2 = D[ans];
628         return mp(w1, w2);
629     }
630 }
631 }
632 struct P3 {
633     db x, y, z;
634     P3 operator+(P3 k1) { return (P3){x + k1.x, y + k1.y, z + k1.z}; }
635     P3 operator-(P3 k1) { return (P3){x - k1.x, y - k1.y, z - k1.z}; }
636     P3 operator*(db k1) { return (P3){x * k1, y * k1, z * k1}; }
637     P3 operator/(db k1) { return (P3){x / k1, y / k1, z / k1}; }
638     db abs2() { return x * x + y * y + z * z; }
639     db abs() { return sqrt(x * x + y * y + z * z); }
640     P3 unit() { return (*this) / abs(); }
641     int operator<(const P3 k1) const {
642         if (cmp(x, k1.x) != 0) return x < k1.x;
643         if (cmp(y, k1.y) != 0) return y < k1.y;
644         return cmp(z, k1.z) == -1;
645     }
646     int operator==(const P3 k1) {
647         return cmp(x, k1.x) == 0 && cmp(y, k1.y) == 0 && cmp(z, k1.z) == 0;
648     }
649     void scan() {
650         double k1, k2, k3;
651         scanf("%lf%lf%lf", &k1, &k2, &k3);
652         x = k1;
653         y = k2;
654         z = k3;
655     }
656 };
657 P3 cross(P3 k1, P3 k2) {
658     return (P3){k1.y * k2.z - k1.z * k2.y, k1.z * k2.x - k1.x * k2.z,
659                 k1.x * k2.y - k1.y * k2.x};
660 }
661 db dot(P3 k1, P3 k2) { return k1.x * k2.x + k1.y * k2.y + k1.z * k2.z; }
662 // p=(3,4,5),l=(13,19,21),theta=85 ans=(2.83,4.62,1.77)
663 P3 turn3D(db k1, P3 l, P3 p) {
664     l = l.unit();
665     P3 ans;
666     db c = cos(k1), s = sin(k1);
667     ans.x = p.x * (l.x * l.x * (1 - c) + c) +

```

```

668         p.y * (l.x * l.y * (1 - c) - l.z * s) +
669         p.z * (l.x * l.z * (1 - c) + l.y * s);
670     ans.y = p.x * (l.x * l.y * (1 - c) + l.z * s) +
671         p.y * (l.y * l.y * (1 - c) + c) +
672         p.z * (l.y * l.z * (1 - c) - l.x * s);
673     ans.z = p.x * (l.x * l.z * (1 - c) - l.y * s) +
674         p.y * (l.y * l.z * (1 - c) + l.x * s) +
675         p.z * (l.x * l.x * (1 - c) + c);
676     return ans;
677 }
678 typedef vector<P3> VP;
679 typedef vector<VP> VVP;
680 db Acos(db x) { return acos(max(-(db)1, min(x, (db)1))); }
681 // 球面距离 , 圆心原点 , 半径 1
682 db Odist(P3 a, P3 b) {
683     db r = Acos(dot(a, b));
684     return r;
685 }
686 db r;
687 P3 rnd;
688 vector<db> solve(db a, db b, db c) {
689     db r = sqrt(a * a + b * b), th = atan2(b, a);
690     if (cmp(c, -r) == -1)
691         return {0};
692     else if (cmp(r, c) <= 0)
693         return {1};
694     else {
695         db tr = pi - Acos(c / r);
696         return {th + pi - tr, th + pi + tr};
697     }
698 }
699 vector<db> jiao(P3 a, P3 b) {
700     // dot(rd+x*cos(t)+y*sin(t),b) >= cos(r)
701     if (cmp(Odist(a, b), 2 * r) > 0) return {0};
702     P3 rd = a * cos(r), z = a.unit(), y = cross(z, rnd).unit(),
703         x = cross(y, z).unit();
704     vector<db> ret = solve(-(dot(x, b) * sin(r)), -(dot(y, b) * sin(r)),
705         -(cos(r) - dot(rd, b)));
706     return ret;
707 }
708 db norm(db x, db l = 0, db r = 2 * pi) { // change x into [l,r)
709     while (cmp(x, l) == -1) x += (r - l);
710     while (cmp(x, r) >= 0) x -= (r - l);
711     return x;
712 }
713 db disLP(P3 k1, P3 k2, P3 q) {
714     return (cross(k2 - k1, q - k1)).abs() / (k2 - k1).abs();
715 }

```

```

716 db disLL(P3 k1, P3 k2, P3 k3, P3 k4) {
717     P3 dir = cross(k2 - k1, k4 - k3);
718     if (sign(dir.abs()) == 0) return disLP(k1, k2, k3);
719     return fabs(dot(dir.unit(), k1 - k2));
720 }
721 VP getFL(P3 p, P3 dir, P3 k1, P3 k2) {
722     db a = dot(k2 - p, dir), b = dot(k1 - p, dir), d = a - b;
723     if (sign(fabs(d)) == 0) return {};
724     return {(k1 * a - k2 * b) / d};
725 }
726 VP getFF(P3 p1, P3 dir1, P3 p2, P3 dir2) { // 返回一条线
727     P3 e = cross(dir1, dir2), v = cross(dir1, e);
728     db d = dot(dir2, v);
729     if (sign(abs(d)) == 0) return {};
730     P3 q = p1 + v * dot(dir2, p2 - p1) / d;
731     return {q, q + e};
732 }
733 // 3D Covex Hull Template
734 db getV(P3 k1, P3 k2, P3 k3, P3 k4) { // get the Volume
735     return dot(cross(k2 - k1, k3 - k1), k4 - k1);
736 }
737 db rand_db() { return 1.0 * rand() / RAND_MAX; }
738 VP convexHull2D(VP A, P3 dir) {
739     P3 x = {(db)rand(), (db)rand(), (db)rand()};
740     x = x.unit();
741     x = cross(x, dir).unit();
742     P3 y = cross(x, dir).unit();
743     P3 vec = dir.unit() * dot(A[0], dir);
744     vector<point> B;
745     for (int i = 0; i < A.size(); i++)
746         B.push_back((point){dot(A[i], x), dot(A[i], y)});
747     B = ConvexHull(B);
748     A.clear();
749     for (int i = 0; i < B.size(); i++)
750         A.push_back(x * B[i].x + y * B[i].y + vec);
751     return A;
752 }
753 namespace CH3 {
754 VVP ret;
755 set<pair<int, int> > e;
756 int n;
757 VP p, q;
758 void wrap(int a, int b) {
759     if (e.find({a, b}) == e.end()) {
760         int c = -1;
761         for (int i = 0; i < n; i++)
762             if (i != a && i != b) {
763                 if (c == -1 || sign(getV(q[c], q[a], q[b], q[i])) > 0) c = i;

```

```

764     }
765     if (c != -1) {
766         ret.push_back({p[a], p[b], p[c]});
767         e.insert({a, b});
768         e.insert({b, c});
769         e.insert({c, a});
770         wrap(c, b);
771         wrap(a, c);
772     }
773 }
774 }
775 VVP ConvexHull3D(VP _p) {
776     p = q = _p;
777     n = p.size();
778     ret.clear();
779     e.clear();
780     for (auto &i : q)
781         i = i + (P3){rand_db() * 1e-4, rand_db() * 1e-4, rand_db() * 1e-4};
782     for (int i = 1; i < n; i++)
783         if (q[i].x < q[0].x) swap(p[0], p[i]), swap(q[0], q[i]);
784     for (int i = 2; i < n; i++)
785         if ((q[i].x - q[0].x) * (q[1].y - q[0].y) >
786             (q[i].y - q[0].y) * (q[1].x - q[0].x))
787             swap(q[1], q[i]), swap(p[1], p[i]);
788     wrap(0, 1);
789     return ret;
790 }
791 } // namespace CH3
792 VVP reduceCH(VVP A) {
793     VVP ret;
794     map<P3, VP> M;
795     for (VP nowF : A) {
796         P3 dir = cross(nowF[1] - nowF[0], nowF[2] - nowF[0]).unit();
797         for (P3 k1 : nowF) M[dir].pb(k1);
798     }
799     for (pair<P3, VP> nowF : M) ret.pb(convexHull2D(nowF.se, nowF.fi));
800     return ret;
801 }
802 // 把一个面变成 ( 点 , 法向量 ) 的形式
803 pair<P3, P3> getF(VP F) {
804     return mp(F[0], cross(F[1] - F[0], F[2] - F[0]).unit());
805 }
806 // 3D Cut 保留 dot(dir,x-p)>=0 的部分
807 VVP ConvexCut3D(VVP A, P3 p, P3 dir) {
808     VVP ret;
809     VP sec;
810     for (VP nowF : A) {
811         int n = nowF.size();

```

```

812     VP ans;
813     int dif = 0;
814     for (int i = 0; i < n; i++) {
815         int d1 = sign(dot(dir, nowF[i] - p));
816         int d2 = sign(dot(dir, nowF[(i + 1) % n] - p));
817         if (d1 >= 0) ans.pb(nowF[i]);
818         if (d1 * d2 < 0) {
819             P3 q = getFL(p, dir, nowF[i], nowF[(i + 1) % n])[0];
820             ans.push_back(q);
821             sec.push_back(q);
822         }
823         if (d1 == 0)
824             sec.push_back(nowF[i]);
825         else
826             dif = 1;
827         dif |= (sign(dot(dir, cross(nowF[(i + 1) % n] - nowF[i],
828                                     nowF[(i + 1) % n] - nowF[i]))) == -1);
829     }
830     if (ans.size() > 0 && dif) ret.push_back(ans);
831 }
832 if (sec.size() > 0) ret.push_back(convexHull2D(sec, dir));
833 return ret;
834 }
835 db vol(VVP A) {
836     if (A.size() == 0) return 0;
837     P3 p = A[0][0];
838     db ans = 0;
839     for (VP nowF : A)
840         for (int i = 2; i < nowF.size(); i++)
841             ans += abs(getV(p, nowF[0], nowF[i - 1], nowF[i]));
842     return ans / 6;
843 }
844 VVP init(db INF) {
845     VVP pss(6, VP(4));
846     pss[0][0] = pss[1][0] = pss[2][0] = {-INF, -INF, -INF};
847     pss[0][3] = pss[1][1] = pss[5][2] = {-INF, -INF, INF};
848     pss[0][1] = pss[2][3] = pss[4][2] = {-INF, INF, -INF};
849     pss[0][2] = pss[5][3] = pss[4][1] = {-INF, INF, INF};
850     pss[1][3] = pss[2][1] = pss[3][2] = {INF, -INF, -INF};
851     pss[1][2] = pss[5][1] = pss[3][3] = {INF, -INF, INF};
852     pss[2][2] = pss[4][3] = pss[3][1] = {INF, INF, -INF};
853     pss[5][0] = pss[4][0] = pss[3][0] = {INF, INF, INF};
854     return pss;
855 }

```

0.3 Graph

0.3.1 2sat.cpp

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4  using ll = long long;
5
6  struct TwoSat {
7      int n;
8      vector<vector<int>> G;
9      vector<bool> ans;
10     TwoSat(int n) : n(n), G(2 * n), ans(n) {}
11     void addClause(int u, bool f, int v, bool g) {
12         G[2 * u + !f].push_back(2 * v + g);
13         G[2 * v + !g].push_back(2 * u + f);
14     }
15     bool satisfiable() {
16         vector<int> id(2 * n, -1), dfn(2 * n, -1), low(2 * n, -1);
17         vector<int> stk;
18         int now = 0, cnt = 0;
19         function<void(int)> tarjan = [&](int u) {
20             stk.push_back(u);
21             dfn[u] = low[u] = now++;
22             for (auto v : G[u]) {
23                 if (dfn[v] == -1) {
24                     tarjan(v);
25                     low[u] = min(low[u], low[v]);
26                 } else if (id[v] == -1) {
27                     low[u] = min(low[u], dfn[v]);
28                 }
29             }
30             if (dfn[u] == low[u]) {
31                 int v;
32                 do {
33                     v = stk.back();
34                     stk.pop_back();
35                     id[v] = cnt;
36                 } while (v != u);
37                 ++cnt;
38             }
39         };
40         for (int i = 0; i < 2 * n; ++i) if (dfn[i] == -1) tarjan(i);
41         for (int i = 0; i < n; ++i) {
42             if (id[2 * i] == id[2 * i + 1]) return false;
43             ans[i] = id[2 * i] > id[2 * i + 1];
44         }

```

```
45         return true;
46     }
47     vector<bool> answer() { return ans; }
48 };
```

0.3.2 Graph.cpp

```
1  #include <bits/stdc++.h>
2
3  using namespace std;
4  using ll = long long;
5
6  template <typename T>
7  class graph {
8      public:
9          struct edge {
10              int from;
11              int to;
12              T cost;
13          };
14
15          vector<edge> edges;
16          vector<vector<int>> g;
17          int n;
18
19          graph(int _n) : n(_n) { g.resize(n); }
20
21          virtual int add(int from, int to, T cost) = 0;
22      };
23
24  template <typename T>
25  class forest : public graph<T> {
26      public:
27          using graph<T>::edges;
28          using graph<T>::g;
29          using graph<T>::n;
30
31          forest(int _n) : graph<T>(_n) {}
32
33          int add(int from, int to, T cost = 1) {
34              assert(0 <= from && from < n && 0 <= to && to < n);
35              int id = (int)edges.size();
36              assert(id < n - 1);
37              g[from].push_back(id);
38              g[to].push_back(id);
39              edges.push_back({from, to, cost});
40              return id;
41          }
42      };
43
44  }
```



```

41     }
42 };
43
44 template <typename T>
45 class dfs_forest : public forest<T> {
46     public:
47         using forest<T>::edges;
48         using forest<T>::g;
49         using forest<T>::n;
50
51         vector<int> pv;
52         vector<int> pe;
53         vector<int> order;
54         vector<int> pos;
55         vector<int> end;
56         vector<int> sz;
57         vector<int> root;
58         vector<int> depth;
59         vector<T> dist;
60
61         dfs_forest(int _n) : forest<T>(_n) {}
62
63         void init() {
64             pv = vector<int>(n, -1);
65             pe = vector<int>(n, -1);
66             order.clear();
67             pos = vector<int>(n, -1);
68             end = vector<int>(n, -1);
69             sz = vector<int>(n, 0);
70             root = vector<int>(n, -1);
71             depth = vector<int>(n, -1);
72             dist = vector<T>(n);
73         }
74
75         void clear() {
76             pv.clear();
77             pe.clear();
78             order.clear();
79             pos.clear();
80             end.clear();
81             sz.clear();
82             root.clear();
83             depth.clear();
84             dist.clear();
85         }
86
87     private:
88         void do_dfs(int v) {

```

```

89     pos[v] = (int)order.size();
90     order.push_back(v);
91     sz[v] = 1;
92     for (int id : g[v]) {
93         if (id == pe[v]) {
94             continue;
95         }
96         auto &e = edges[id];
97         int to = e.from ^ e.to ^ v;
98         depth[to] = depth[v] + 1;
99         dist[to] = dist[v] + e.cost;
100        pv[to] = v;
101        pe[to] = id;
102        root[to] = (root[v] != -1 ? root[v] : to);
103        do_dfs(to);
104        sz[v] += sz[to];
105    }
106    end[v] = (int)order.size() - 1;
107 }
108
109 void do_dfs_from(int v) {
110     depth[v] = 0;
111     dist[v] = T{};
112     root[v] = v;
113     pv[v] = pe[v] = -1;
114     do_dfs(v);
115 }
116
117 public:
118 void dfs(int v, bool clear_order = true) {
119     if (pv.empty()) {
120         init();
121     } else {
122         if (clear_order) {
123             order.clear();
124         }
125     }
126     do_dfs_from(v);
127 }
128
129 void dfs_all() {
130     init();
131     for (int v = 0; v < n; v++) {
132         if (depth[v] == -1) {
133             do_dfs_from(v);
134         }
135     }
136     assert((int)order.size() == n);

```

```

137     }
138 };
139
140 template <typename T>
141 class lca_forest : public dfs_forest<T> {
142     public:
143         using dfs_forest<T>::edges;
144         using dfs_forest<T>::g;
145         using dfs_forest<T>::n;
146         using dfs_forest<T>::pv;
147         using dfs_forest<T>::pos;
148         using dfs_forest<T>::end;
149         using dfs_forest<T>::depth;
150
151         int h;
152         vector<vector<int>>> pr;
153
154         lca_forest(int _n) : dfs_forest<T>(_n) {}
155
156         inline void build_lca() {
157             assert(!pv.empty());
158             int max_depth = 0;
159             for (int i = 0; i < n; i++) {
160                 max_depth = max(max_depth, depth[i]);
161             }
162             h = 1;
163             while ((1 << h) <= max_depth) {
164                 h++;
165             }
166             pr.resize(n);
167             for (int i = 0; i < n; i++) {
168                 pr[i].resize(h);
169                 pr[i][0] = pv[i];
170             }
171             for (int j = 1; j < h; j++) {
172                 for (int i = 0; i < n; i++) {
173                     pr[i][j] = (pr[i][j - 1] == -1 ? -1 : pr[pr[i][j - 1]][j - 1]);
174                 }
175             }
176         }
177
178         inline bool anc(int x, int y) {
179             return (pos[x] <= pos[y] && end[y] <= end[x]);
180         }
181
182         inline int go_up(int x, int up) {
183             assert(!pr.empty());
184             up = min(up, (1 << h) - 1);

```

```

185     for (int j = h - 1; j >= 0; j--) {
186         if (up & (1 << j)) {
187             x = pr[x][j];
188             if (x == -1) {
189                 break;
190             }
191         }
192     }
193     return x;
194 }
195
196 inline int lca(int x, int y) {
197     assert(!pr.empty());
198     if (anc(x, y)) {
199         return x;
200     }
201     if (anc(y, x)) {
202         return y;
203     }
204     for (int j = h - 1; j >= 0; j--) {
205         if (pr[x][j] != -1 && !anc(pr[x][j], y)) {
206             x = pr[x][j];
207         }
208     }
209     return pr[x][0];
210 }
211 };

```

0.3.3 MaxAssignment.cpp

```

1  #include <bits/stdc++.h>
2
3  using i64 = long long;
4
5  template<class T>
6  struct MaxAssignment {
7      public:
8          T solve(int nx, int ny, std::vector<std::vector<T>> a) {
9              assert(0 <= nx && nx <= ny);
10             assert(int(a.size()) == nx);
11             for (int i = 0; i < nx; ++i) {
12                 assert(int(a[i].size()) == ny);
13                 for (auto x : a[i])
14                     assert(x >= 0);
15             }
16
17             auto update = [&](int x) {

```

```

18         for (int y = 0; y < ny; ++y) {
19             if (lx[x] + ly[y] - a[x][y] < slack[y]) {
20                 slack[y] = lx[x] + ly[y] - a[x][y];
21                 slackx[y] = x;
22             }
23         }
24     };
25
26     costs.resize(nx + 1);
27     costs[0] = 0;
28     lx.assign(nx, std::numeric_limits<T>::max());
29     ly.assign(ny, 0);
30     xy.assign(nx, -1);
31     yx.assign(ny, -1);
32     slackx.resize(ny);
33     for (int cur = 0; cur < nx; ++cur) {
34         std::queue<int> que;
35         visx.assign(nx, false);
36         visy.assign(ny, false);
37         slack.assign(ny, std::numeric_limits<T>::max());
38         p.assign(nx, -1);
39
40         for (int x = 0; x < nx; ++x) {
41             if (xy[x] == -1) {
42                 que.push(x);
43                 visx[x] = true;
44                 update(x);
45             }
46         }
47
48         int ex, ey;
49         bool found = false;
50         while (!found) {
51             while (!que.empty() && !found) {
52                 auto x = que.front();
53                 que.pop();
54                 for (int y = 0; y < ny; ++y) {
55                     if (a[x][y] == lx[x] + ly[y] && !visy[y]) {
56                         if (yx[y] == -1) {
57                             ex = x;
58                             ey = y;
59                             found = true;
60                             break;
61                         }
62                         que.push(yx[y]);
63                         p[yx[y]] = x;
64                         visy[y] = visx[yx[y]] = true;
65                         update(yx[y]);

```

```

66         }
67     }
68 }
69 if (found)
70     break;
71
72 T delta = std::numeric_limits<T>::max();
73 for (int y = 0; y < ny; ++y)
74     if (!visy[y])
75         delta = std::min(delta, slack[y]);
76 for (int x = 0; x < nx; ++x)
77     if (visx[x])
78         lx[x] -= delta;
79 for (int y = 0; y < ny; ++y) {
80     if (visy[y]) {
81         ly[y] += delta;
82     } else {
83         slack[y] -= delta;
84     }
85 }
86 for (int y = 0; y < ny; ++y) {
87     if (!visy[y] && slack[y] == 0) {
88         if (yx[y] == -1) {
89             ex = slackx[y];
90             ey = y;
91             found = true;
92             break;
93         }
94         que.push(yx[y]);
95         p[yx[y]] = slackx[y];
96         visy[y] = visx[yx[y]] = true;
97         update(yx[y]);
98     }
99 }
100 }
101
102 costs[cur + 1] = costs[cur];
103 for (int x = ex, y = ey, ty; x != -1; x = p[x], y = ty) {
104     costs[cur + 1] += a[x][y];
105     if (xy[x] != -1)
106         costs[cur + 1] -= a[x][xy[x]];
107     ty = xy[x];
108     xy[x] = y;
109     yx[y] = x;
110 }
111 }
112 return costs[nx];
113 }

```

```

114     std::vector<int> assignment() {
115         return xy;
116     }
117     std::pair<std::vector<T>, std::vector<T>> labels() {
118         return std::make_pair(lx, ly);
119     }
120     std::vector<T> weights() {
121         return costs;
122     }
123 private:
124     std::vector<T> lx, ly, slack, costs;
125     std::vector<int> xy, yx, p, slackx;
126     std::vector<bool> visx, visy;
127 };
128
129 constexpr i64 inf = 1E12;
130
131 int main() {
132     std::ios::sync_with_stdio(false);
133     std::cin.tie(nullptr);
134
135     int n;
136     std::cin >> n;
137
138     std::vector cost(150, std::vector<i64>(150));
139     for (int i = 0; i < n; i++) {
140         int a, b, c;
141         std::cin >> a >> b >> c;
142         a--;
143         b--;
144         cost[a][b] = std::max(cost[a][b], inf + c);
145     }
146
147     MaxAssignment<i64> m;
148     m.solve(150, 150, cost);
149
150     int k = 0;
151     auto ans = m.weights();
152     while (k < 150 && ans[k + 1] >= inf * (k + 1)) {
153         k++;
154     }
155
156     std::cout << k << "\n";
157     for (int i = 1; i <= k; i++) {
158         std::cout << ans[i] - inf * i << "\n";
159     }
160
161     return 0;

```

```

162 }
163
164 //test problem: https://atcoder.jp/contests/abc247/tasks/abc247\_g

```

0.3.4 Mincost.cpp

```

1  #include <bits/stdc++.h>
2  using namespace std;
3
4  using ll = long long;
5
6  template <typename cap_t, typename cost_t>
7  struct Mincost {
8      static constexpr cost_t INF = numeric_limits<cost_t>::max();
9      int n;
10     struct Edge {
11         int to;
12         cap_t cap;
13         cost_t cost;
14         Edge(int to, cap_t cap, cost_t cost) : to(to), cap(cap), cost(cost) {}
15     };
16     vector<Edge> e;
17     vector<vector<int>> g;
18     vector<int> cur, pre;
19     vector<bool> vis;
20     vector<cost_t> dis;
21     Mincost(int n) : n(n), g(n), vis(n) {}
22     void addEdge(int u, int v, cap_t c, cost_t w) {
23         g[u].push_back(e.size());
24         e.emplace_back(v, c, w);
25         g[v].push_back(e.size());
26         e.emplace_back(u, 0, -w);
27     }
28     bool spfa(int s, int t) {
29         pre.assign(n, -1);
30         dis.assign(n, INF);
31         queue<int> que;
32         que.push(s);
33         dis[s] = 0;
34         while (!que.empty()) {
35             int u = que.front();
36             que.pop();
37             vis[u] = false;
38             for (auto j : g[u]) {
39                 auto [v, c, w] = e[j];
40                 if (c > 0 && dis[v] > dis[u] + w) {
41                     dis[v] = dis[u] + w;

```



```

42         pre[v] = j;
43         if (!vis[v]) {
44             que.push(v);
45             vis[v] = true;
46         }
47     }
48 }
49 }
50 return dis[t] != INF;
51 }
52 pair<cap_t, cost_t> dfs(int u, int t, cap_t f) {
53     if (u == t) return {f, 0};
54     vis[u] = true;
55     cap_t r = f;
56     cost_t p = 0;
57     for (int &i = cur[u]; i < int(g[u].size()); ++ i) {
58         int j = g[u][i];
59         auto [v, c, w] = e[j];
60         if (!vis[v] && c > 0 && dis[v] == dis[u] + w) {
61             auto a = dfs(v, t, min(c, r));
62             e[j].cap -= a.first;
63             e[j ^ 1].cap += a.first;
64             r -= a.first;
65             p += a.first * w + a.second;
66             if (r == 0) break;
67         }
68     }
69     vis[u] = false;
70     return {f - r, p};
71 }
72 void augment(int s, int t, pair<cap_t, cost_t> &ans) {
73     int p = t;
74     cap_t _f = INF;
75     while (pre[p] != -1) {
76         _f = min(_f, e[pre[p]].cap);
77         p = e[pre[p] ^ 1].to;
78     }
79     ans.first += _f;
80     ans.second += _f * dis[t];
81     p = t;
82     while(pre[p] != -1) {
83         e[pre[p]].cap -= _f;
84         e[pre[p] ^ 1].cap += _f;
85         p = e[pre[p] ^ 1].to;
86     }
87 }
88 // select dfs or augment
89 // dfs() can multiple augment

```

```

90 // augment() can augment a minimum cost flow
91 pair<cap_t, cost_t> maxFlowMinCost(int s, int t) {
92     pair<cap_t, cost_t> ans = {0, 0};
93     while (spfa(s, t)) {
94         cur.assign(n, 0);
95         auto res = dfs(s, t, INF);
96         ans.first += res.first;
97         ans.second += res.second;
98
99         // augment(s, t, ans);
100     }
101     return ans;
102 }
103 };
104
105 int main() {
106     ios::sync_with_stdio(false);
107     cin.tie(nullptr);
108
109     int n, m;
110     cin >> n >> m;
111
112     Mincost<ll, ll> flow(n);
113     const int source = 0, sink = n - 1;
114
115     for (int i = 0; i < m; ++i) {
116         int u, v;
117         ll c, w;
118         cin >> u >> v >> c >> w;
119         u--, v--;
120         flow.addEdge(u, v, c, w);
121     }
122
123     auto ans = flow.maxFlowMinCost(source, sink);
124     cout << ans.first << " " << ans.second << "\n";
125
126     return 0;
127 };
128
129 // test problem: https://loj.ac/p/102

```

0.3.5 Tree.cpp

```

1 #include <bits/stdc++.h>
2
3 using namespace std;
4 using ll = long long;

```

```

5
6 struct Tree {
7     vector<int> sz, top, dep, parent, in, out;
8     int cur;
9     vector<vector<int>> e;
10    Tree(int n) : sz(n), top(n), dep(n), parent(n, -1), in(n), out(0), cur(0), e(n) {}
11    void addEdge(int u, int v) {
12        e[u].push_back(v);
13        e[v].push_back(u);
14    }
15    void init() {
16        dfsSz(0);
17        dfsHLD(0);
18    }
19    void dfsSz(int u) {
20        if (parent[u] != -1) {
21            e[u].erase(find(e[u].begin(), e[u].end(), parent[u]));
22        }
23        sz[u] = 1;
24        for (int &v : e[u]) {
25            parent[v] = u;
26            dep[v] = dep[u] + 1;
27            dfsSz(v);
28            sz[u] += sz[v];
29            if (sz[v] > sz[e[u][0]]) {
30                swap(v, e[u][0]);
31            }
32        }
33    }
34    void dfsHLD(int u) {
35        in[u] = cur++;
36        for (int v : e[u]) {
37            top[v] = (v == e[u][0] ? top[u] : v);
38            dfsHLD(v);
39        }
40        out[u] = cur;
41    }
42    int lca(int u, int v) {
43        while (top[u] != top[v]) {
44            if (dep[top[u]] < dep[top[v]]) {
45                swap(u, v);
46            }
47            u = parent[top[u]];
48        }
49        return dep[u] < dep[v] ? u : v;
50    }
51 };

```

0.3.6 dijkstra.cpp

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4  using ll = long long;
5
6  int main() {
7      ios::sync_with_stdio(false);
8      cin.tie(nullptr);
9
10     int n, m, s;
11     cin >> n >> m >> s; s--;
12     vector<vector<pair<int, int>>> g(n);
13     vector<int> w(m);
14     for (int i = 0; i < m; ++i) {
15         int u, v;
16         cin >> u >> v >> w[i];
17         u--, v--;
18         g[u].emplace_back(v, i);
19     }
20
21     auto dijkstra = [&]() {
22         vector<int> dis(n, -1);
23         priority_queue<pair<int, int>> h;
24         h.emplace(0, s);
25         while (!h.empty()) {
26             auto [d, u] = h.top();
27             h.pop();
28             if (dis[u] != -1) continue;
29             dis[u] = -d;
30             for (auto [v, j] : g[u]) {
31                 h.emplace(d - w[j], v);
32             }
33         }
34         return dis;
35     };
36
37     auto dis = dijkstra();
38     for (int i = 0; i < n; ++i) {
39         cout << dis[i] << " \n"[i == n - 1];
40     }
41
42     return 0;
43 }
44
45 // test problem: https://www.luogu.com.cn/problem/P4779

```

0.3.7 dinic.cpp

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4  using ll = long long;
5
6  template<class cap_t>
7  struct Flow {
8      static constexpr cap_t INF = numeric_limits<cap_t>::max();
9      int n;
10     struct Edge {
11         int to;
12         cap_t cap;
13         Edge(int to, cap_t cap) : to(to), cap(cap) {}
14     };
15     vector<Edge> e;
16     vector<vector<int>> g;
17     vector<int> cur, h;
18     Flow(int n) : n(n), g(n) {}
19     bool bfs(int s, int t) {
20         h.assign(n, -1);
21         queue<int> que;
22         h[s] = 0;
23         que.push(s);
24         while (!que.empty()) {
25             int u = que.front();
26             que.pop();
27             for (int j : g[u]) {
28                 int v = e[j].to;
29                 cap_t c = e[j].cap;
30                 if (c > 0 && h[v] == -1) {
31                     h[v] = h[u] + 1;
32                     if (v == t) return true;
33                     que.push(v);
34                 }
35             }
36         }
37         return false;
38     }
39     cap_t dfs(int u, int t, cap_t f) {
40         if (u == t) return f;
41         cap_t r = f;
42         for (int &i = cur[u]; i < int(g[u].size()); ++i) {
43             int j = g[u][i];
44             int v = e[j].to;
45             cap_t c = e[j].cap;
46             if (c > 0 && h[v] == h[u] + 1) {

```

```

47         cap_t a = dfs(v, t, min(r, c));
48         e[j].cap -= a;
49         e[j ^ 1].cap += a;
50         r -= a;
51         if (r == 0) return f;
52     }
53 }
54 return f - r;
55 }
56 void addEdge(int u, int v, cap_t c) {
57     g[u].push_back(e.size());
58     e.emplace_back(v, c);
59     g[v].push_back(e.size());
60     e.emplace_back(u, 0);
61 }
62 cap_t maxFlow(int s, int t) {
63     cap_t ans = 0;
64     while (bfs(s, t)) {
65         cur.assign(n, 0);
66         ans += dfs(s, t, INF);
67     }
68     return ans;
69 }
70 };
71
72 int main() {
73     ios::sync_with_stdio(false);
74     cin.tie(nullptr);
75
76     int n, m, source, sink;
77     cin >> n >> m >> source >> sink;
78     source--, sink--;
79     Flow<ll> flow(n);
80     for (int i = 0; i < m; ++i) {
81         int u, v, c;
82         cin >> u >> v >> c;
83         u--, v--;
84         flow.addEdge(u, v, c);
85     }
86
87     cout << flow.maxFlow(source, sink) << "\n";
88
89     return 0;
90 }
91
92 // test problem: https://loj.ac/p/101

```

0.3.8 spfa.cpp

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4  using ll = long long;
5
6  const int inf = 1e9;
7
8  void solve() {
9      int n, m;
10     cin >> n >> m;
11
12     vector<vector<pair<int, int>>> g(n);
13     vector<int> w(m);
14     for (int i = 0; i < m; ++i) {
15         int u, v;
16         cin >> u >> v >> w[i];
17         u--, v--;
18         g[u].emplace_back(v, i);
19         if (w[i] >= 0) {
20             g[v].emplace_back(u, i);
21         }
22     }
23
24     auto spfa = [&](int s) { // true: no negative ring
25         vector<int> dis(n, inf), cnt(n);
26         vector<bool> vis(n);
27         dis[s] = 0;
28         vis[s] = true;
29         queue<int> q;
30         q.push(s);
31
32         while (!q.empty()) {
33             int u = q.front();
34             q.pop();
35             vis[u] = false;
36             for (auto [v, j] : g[u]) {
37                 if (dis[v] > dis[u] + w[j]) {
38                     dis[v] = dis[u] + w[j];
39                     cnt[v] = cnt[u] + 1;
40                     if (cnt[v] >= n) {
41                         return false;
42                     }
43                     if (vis[v] == false) {
44                         q.push(v);
45                         vis[v] = true;
46                     }

```

```
47         }
48     }
49 }
50
51     return true;
52 };
53
54     cout << (spfa(0) ? "NO\n" : "YES\n");
55 }
56
57 int main() {
58     ios::sync_with_stdio(false);
59     cin.tie(nullptr);
60
61     int t;
62     cin >> t;
63
64     while (t--) {
65         solve();
66     }
67
68     return 0;
69 }
70
71 // test problem: https://www.luogu.com.cn/problem/P3385
```

0.3.9 匈牙利.cpp

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  typedef long long ll;
4  const int maxn = 505;
5  int n1, n2, m, match[maxn];
6  vector<int> g[maxn];
7  bool vis[maxn];
8  bool find(int u) {
9      for (auto v : g[u]) {
10         if (vis[v]) continue;
11         vis[v] = 1;
12         if (match[v] == 0 || find(match[v])) {
13             match[v] = u;
14             return 1;
15         }
16     }
17     return 0;
18 }
19 int main() {
```



```

20     scanf("%d%d%d", &n1, &n2, &m);
21     while (m--) {
22         int u, v;
23         scanf("%d%d", &u, &v);
24         g[u].push_back(v);
25     }
26     int ans = 0;
27     for (int i = 1; i <= n1; ++i) {
28         memset(vis, false, sizeof(vis));
29         if (find(i)) ++ans;
30     }
31     printf("%d\n", ans);
32     return 0;
33 }

```

0.4 Math

0.4.1 China.cpp

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  #define IO ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
4  typedef long long ll;
5  using namespace std;
6  /**
7   *gcd(a,mod)=d;则存在x,y,使d=ax+by
8   *extended_euclid(a,mod)=ax+by
9   */
10 ll extended_euclid(ll a, ll mod, ll &x, ll &y)
11 { //扩张欧几里的算法
12     int d;
13     if (mod == 0)
14     {
15         x = 1;
16         y = 0;
17         return a;
18     }
19     d = extended_euclid(mod, a % mod, y, x);
20     y = y - a / mod * x;
21     return d;
22 }
23 /**
24  *x=mod[i](modw[i]) 0<i<len
25  *prime[i]>0
26  */
27 ll chinese_remainder(int mod[], int prime[], int len)
28 {

```

```

29     ll res, i, d, x, y, n, m;
30     res = 0;
31     n = 1;
32     for (i = 0; i < len; i++)
33         n *= prime[i];
34     for (i = 0; i < len; i++)
35     {
36         m = n / prime[i];
37         extended_euclid(prime[i], m, x, y);
38         res = (res + y * m * mod[i]) % n;
39     }
40     return (n + res % n) % n;
41 }
42
43 int main()
44 {
45     int len, mod[12], prime[12];
46     while (cin >> len)
47     {
48         for (int i = 0; i < len; i++)
49             cin >> prime[i] >> mod[i];
50         cout << chinese_remainder(mod, prime, len) << endl;
51     }
52 }

```

0.4.2 Euler.cpp

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4  using ll = long long;
5
6  // 表示的是小于等于n和n互质的数的个数。
7  int euler_phi(int n) {
8      int ans = n;
9      for (int i = 2; i * i <= n; i++)
10         if (n % i == 0) {
11             ans = ans / i * (i - 1);
12             while (n % i == 0) n /= i;
13         }
14     if (n > 1) ans = ans / n * (n - 1);
15     return ans;
16 }
17
18 vector<int> phi_table(int n) {
19     vector<int> phi(n + 1);
20     phi[1] = 1;

```

```

21     for (int i = 2; i <= n; i++) {
22         if (phi[i]) continue;
23         for (int j = i; j <= n; j += i) {
24             if (!phi[j]) phi[j] = j;
25             phi[j] = phi[j] / i * (i - 1);
26         }
27     }
28     return phi;
29 }

```

0.4.3 FFT.cpp

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  #define PI acos(-1.0)
4  const int maxn = 5e5 + 5;
5  const int INF = 0x3f3f3f3f;
6  const int MOD = 1e9 + 7;
7  struct Complex { //复数结构体
8      double r, i;
9      Complex(double _r = 0.0, double _i = 0.0) { r = _r, i = _i; }
10     Complex operator+(const Complex &b) { return Complex(r + b.r, i + b.i); }
11     Complex operator-(const Complex &b) { return Complex(r - b.r, i - b.i); }
12     Complex operator*(const Complex &b) { return Complex(r * b.r - i * b.i, r * b.i + i * b.r); }
13 };
14 /*
15  * 进行FFT和IFFT前的反转变换。
16  * 位置i和 (i二进制反转后位置) 互换
17  * len必须去2的幂
18  */
19 /*
20 做FFT
21 len必须为2^k形式,
22 on==1时是DFT, on==-1时是IDFT
23 */
24 int rev[maxn];
25 void FFT(Complex y[], int len, int on) {
26     int bit = 0;
27     while ((1 << bit) < len)
28         bit++;
29     for (int i = 0; i < len - 1; i++) { //对每一位y处理出递归结束后的位置,然后进行交换
30         rev[i] = (rev[i] >> 1) >> 1 | ((i & 1) << (bit - 1));
31         if (i < rev[i]) swap(y[i], y[rev[i]]); //不加这条if会交换两次 (就是没交换)
32     }
33     for (int h = 2; h <= len; h <= 1) { //h为合并后的区间长度
34         Complex wn(cos(-on * 2 * PI / h), sin(-on * 2 * PI / h)); //单位根
35         for (int j = 0; j < len; j += h) { //遍历每个区间的开端

```

```

36         Complex w(1, 0);
37         for (int k = j; k < j + h / 2; k++) { //对小区间进行计算
38             Complex u = y[k];
39             Complex t = w * y[k + h / 2];
40             y[k] = u + t;           //前半区间
41             y[k + h / 2] = u - t;   //后半区间
42             w = w * wn;
43         }
44     }
45 }
46 if (on == -1)
47     for (int i = 0; i < len; i++)
48         y[i].r /= len;
49 }
50
51 char s1[maxn], s2[maxn];
52 int ans[maxn];
53 Complex a[maxn], b[maxn];
54 int main() {
55     int i, len1, len2, len;
56     while (~scanf("%s%s", s1, s2)) {
57         len1 = strlen(s1);
58         len2 = strlen(s2);
59         len = 1;
60         while (len < (len1 << 1) || len < (len2 << 1))
61             len <<= 1;
62         for (i = 0; i < len1; i++)
63             a[i] = Complex(s1[len1 - i - 1] - '0', 0);
64         for (; i < len; i++)
65             a[i] = Complex(0, 0);
66         for (i = 0; i < len2; i++)
67             b[i] = Complex(s2[len2 - i - 1] - '0', 0);
68         for (; i < len; i++)
69             b[i] = Complex(0, 0);
70         FFT(a, len, 1);
71         FFT(b, len, 1);
72         for (i = 0; i < len; i++)
73             a[i] = a[i] * b[i];
74         FFT(a, len, -1);
75         for (i = 0; i < len; i++)
76             ans[i] = (int)(a[i].r + 0.5);
77         len = len1 + len2 - 1;
78         for (i = 0; i < len; i++) {
79             ans[i + 1] += ans[i] / 10;
80             ans[i] %= 10;
81         }
82         for (i = len; ans[i] <= 0 && i > 0; i--)
83             ;

```

```

84         for (; i >= 0; i--)
85             printf("%d", ans[i]);
86         putchar('\n');
87     }
88 }

```

0.4.4 Lagrange.cpp

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  typedef long long ll;
4  const ll mod = 1e9 + 7;
5  const int maxn = 1e6 + 10;
6  ll t, n, m, l, r;
7  ll a[maxn], sum[maxn], pre[maxn], suf[maxn], fac[maxn];
8  ll ksm(ll x, ll n, ll p) //计算x的n次幂%p
9  {
10     ll res = 1;
11     while (n) {
12         if (n & 1)
13             res = (res * x) % p;
14         x = (x * x) % p;
15         n >>= 1;
16     }
17     return res;
18 }
19 /* cal 函数, 用从 0 到 n, 一共 n+1 个数,
20    传入的 a 数组代表 yi
21    插值出原来的 n 次方的多项式
22    并返回 原来的多项式在 x 出的取值。
23 */
24 ll cal(ll x, ll *a, ll n) {
25     if (x <= n)
26         return a[x];
27     ll ans = 0;
28     pre[0] = x;
29     suf[n + 1] = 1;
30     for (int i = 1; i <= n; i++) //求前缀积
31         pre[i] = pre[i - 1] * (x - i) % mod;
32     for (int i = n; i >= 0; i--) //求后缀积
33         suf[i] = suf[i + 1] * (x - i) % mod;
34     for (int i = 0; i <= n; i++) { //对于每一个去计算
35         ll f = fac[n - i] * fac[i] % mod; //分母
36         if ((n - i) % 2 == 1)
37             f *= -1; //正负
38         if (i == 0)
39             ans = (ans + a[i] * f % mod * 1LL * suf[i + 1] % mod) % mod; //乘上分子和Y值

```

```

40         else
41             ans = (ans + a[i] * f % mod * pre[i - 1] % mod * suf[i + 1] % mod) % mod;
42     }
43     return (ans + mod) % mod; //可能为负值, 加mod
44 }
45
46 void init() {
47     fac[0] = 1;
48     for (int i = 1; i < maxn; i++) //求N的阶乘
49         fac[i] = fac[i - 1] * i % mod;
50     for (int i = 0; i < maxn; i++) //费马小定理求对应逆元
51         fac[i] = ksm(fac[i], mod - 2, mod);
52 }
53
54 int main() {
55     init();
56     ll n, k;
57     scanf("%lld %lld", &n, &k);
58     ll sum = 0;
59     a[0] = 0;
60     for (int i = 1; i <= k + 2; i++) {
61         sum = (sum + ksm(i, k, mod)) % mod;
62         a[i] = sum;
63     }
64     printf("%lld\n", cal(n, a, k + 1));
65 }

```

0.4.5 Lucas.cpp

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4  using ll = long long;
5
6  int P = 1e9 + 7;
7  // assume -P <= x < P
8  int norm(int x) {
9      if (x < 0) x += P;
10     if (x >= P) x -= P;
11     return x;
12 }
13 template<class T>
14 T power(T a, ll b) {
15     T res = 1;
16     for (; b; b /= 2, a *= a) {
17         if (b % 2) res *= a;
18     }

```

```

19     return res;
20 }
21 struct Z {
22     int x;
23     Z(int x = 0) : x(norm(x)) {}
24     Z(int64_t x) : x(x % P) {}
25     int val() const {
26         return x;
27     }
28     Z operator-() const {
29         return Z(norm(P - x));
30     }
31     Z inv() const {
32         assert(x != 0);
33         return power(*this, P - 2);
34     }
35     Z &operator*=(const Z &rhs) {
36         x = int64_t(x) * rhs.x % P;
37         return *this;
38     }
39     Z &operator+=(const Z &rhs) {
40         x = norm(x + rhs.x);
41         return *this;
42     }
43     Z &operator-=(const Z &rhs) {
44         x = norm(x - rhs.x);
45         return *this;
46     }
47     Z &operator/=(const Z &rhs) {
48         return *this *= rhs.inv();
49     }
50     friend Z operator*(const Z &lhs, const Z &rhs) {
51         Z res = lhs;
52         res *= rhs;
53         return res;
54     }
55     friend Z operator+(const Z &lhs, const Z &rhs) {
56         Z res = lhs;
57         res += rhs;
58         return res;
59     }
60     friend Z operator-(const Z &lhs, const Z &rhs) {
61         Z res = lhs;
62         res -= rhs;
63         return res;
64     }
65     friend Z operator/(const Z &lhs, const Z &rhs) {
66         Z res = lhs;

```

```

67         res /= rhs;
68         return res;
69     }
70     friend istream &operator>>(istream &is, Z &a) {
71         int64_t v;
72         is >> v;
73         a = Z(v);
74         return is;
75     }
76     friend ostream &operator<<(ostream &os, const Z &a) {
77         return os << a.val();
78     }
79 };
80
81 struct Binom {
82     const int N;
83     vector<Z> fac, invfac;
84     Binom(int n) : N(n), fac(N + 1), invfac(N + 1) {
85         fac[0] = 1;
86         for (int i = 1; i <= N; i++) {
87             fac[i] = fac[i - 1] * i;
88         }
89         invfac[N] = fac[N].inv();
90         for (int i = N; i; i--) {
91             invfac[i - 1] = invfac[i] * i;
92         }
93     }
94
95     Z get(int n, int m) {
96         if (m < 0 || n < m) return Z(0);
97         return fac[n] * invfac[m] * invfac[n - m];
98     };
99 };
100
101 void solve() {
102     int n, m;
103     cin >> n >> m >> P;
104
105     Binom binom(P - 1);
106
107     function<ll(int, int, int)> Lucas = [&](int n, int m, int P) {
108         if (m == 0) return 1LL;
109         return 1LL * binom.get(n % P, m % P).val() * Lucas(n / P, m / P, P) % P;
110     };
111
112     cout << Lucas(n + m, m, P) << "\n";
113 }
114

```



```

115 int main() {
116     ios::sync_with_stdio(false);
117     cin.tie(nullptr);
118
119     int t;
120     cin >> t;
121
122     while (t--) {
123         solve();
124     }
125
126     return 0;
127 }
128 // test problem: https://www.luogu.com.cn/problem/P3807

```

0.4.6 Miller-Rabin.cpp

```

1  #include <bits/stdc++.h>
2  using namespace std;
3
4  uint64_t mod_mul64(uint64_t a, uint64_t b, uint64_t mod) {
5      assert(a < mod && b < mod);
6
7      if (mod <= 1LLU << 32)
8          return a * b % mod;
9
10     if (mod <= 1LLU << 63) {
11         uint64_t q = uint64_t((long double) a * b / mod);
12         uint64_t result = a * b - q * mod;
13
14         if (result > 1LLU << 63) {
15             result += mod;
16         } else if (result >= mod) {
17             result -= mod;
18         }
19
20         return result;
21     }
22
23     #ifdef __SIZEOF_INT128__
24         return uint64_t((__uint128_t(a) * b % mod));
25     #endif
26
27     assert(false);
28 }
29
30 uint64_t mod_pow64(uint64_t a, uint64_t b, uint64_t mod) {

```

```

31     uint64_t result = 1;
32     while (b > 0) {
33         if (b & 1) {
34             result = mod_mul64(result, a, mod);
35         }
36         a = mod_mul64(a, a, mod);
37         b >>= 1;
38     }
39     return result;
40 }
41
42 bool miller_rabin(uint64_t n) {
43     if (n < 2)
44         return false;
45
46     // Check small primes.
47     for (uint64_t p : {2, 3, 5, 7, 11, 13, 17, 19, 23, 29})
48         if (n % p == 0)
49             return n == p;
50
51     // https://miller-rabin.appspot.com/
52     auto get_miller_rabin_bases = [&]() -> vector<uint64_t> {
53         if (n < 341531) return {9345883071009581737LLU};
54         if (n < 1050535501) return {336781006125, 9639812373923155};
55         if (n < 350269456337) return {4230279247111683200, 14694767155120705706LLU,
56             16641139526367750375LLU};
57         if (n < 55245642489451) return {2, 141889084524735, 1199124725622454117, 11096072698276303650
58             LLU};
59         if (n < 7999252175582851) return {2, 4130806001517, 149795463772692060, 186635894390467037,
60             3967304179347715805};
61         if (n < 585226005592931977) return {2, 123635709730000, 9233062284813009, 43835965440333360,
62             761179012939631437, 1263739024124850375};
63         return {2, 325, 9375, 28178, 450775, 9780504, 1795265022};
64     };
65
66     int r = __builtin_ctzll(n - 1);
67     uint64_t d = (n - 1) >> r;
68
69     for (uint64_t a : get_miller_rabin_bases()) {
70         if (a % n == 0)
71             continue;
72
73         uint64_t x = mod_pow64(a % n, d, n);
74
75         if (x == 1 || x == n - 1)
76             continue;
77
78         for (int i = 0; i < r - 1 && x != n - 1; i++)

```

```

75         x = mod_mul64(x, x, n);
76
77         if (x != n - 1)
78             return false;
79     }
80
81     return true;
82 }
83
84 // Solution to https://www.spoj.com/problems/PON/
85 int main() {
86     ios::sync_with_stdio(false);
87     cin.tie(nullptr);
88
89     int t;
90     cin >> t;
91
92     while (t--) {
93         uint64_t n;
94         cin >> n;
95         cout << (miller_rabin(n) ? "YES" : "NO") << '\n';
96     }
97
98     return 0;
99 }

```

0.4.7 NTT.cpp

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  #define ll long long
4  const int maxn = 2e5 + 10;
5  const ll mod = 998244353, g = 3;
6  int rev[maxn];
7  ll ksm(ll x, ll n, ll mod) { //计算x的n次幂%mod
8      ll res = 1;
9      while (n) {
10         if (n & 1) res = (res * x) % mod;
11         x = (x * x) % mod;
12         n >>= 1;
13     }
14     return res;
15 }
16 void NTT(ll y[], int len, int on) {
17     int bit = 0;
18     while ((1 << bit) < len) bit++;
19     for (int i = 0; i <= len - 1; i++) //对每一位y处理出递归结束后的位置,然后进行交换

```

```

20     {
21         rev[i] = (rev[i >> 1] >> 1) | ((i & 1) << (bit - 1));
22         if (i < rev[i])
23             swap(y[i], y[rev[i]]); //不加这条if会交换两次（就是没交换）
24     }
25     //前面和FFT相同
26     for (int h = 2; h <= len; h <= 1) // h为合并后的区间长度
27     {
28         ll wn = ksm(g, (mod - 1) / h, mod); //单位根
29         if (on == -1) wn = ksm(wn, mod - 2, mod); //求逆元
30         for (int j = 0; j < len; j += h) //遍历每个区间的开端
31         {
32             ll w = 1;
33             for (int k = j; k < j + h / 2; k++) //对小区间进行计算
34             {
35                 ll u = y[k];
36                 ll t = (w * y[k + h / 2]) % mod;
37                 y[k] = (u + t) % mod; //前半区间
38                 y[k + h / 2] = (u - t + mod) % mod; //后半区间
39                 w = (w * wn) % mod;
40             }
41         }
42     }
43     if (on == -1) {
44         ll t = ksm(len, mod - 2, mod);
45         for (int i = 0; i < len; i++) y[i] = (y[i] * t) % mod;
46     }
47 }
48 char st[maxn], st1[maxn];
49 ll A[maxn], B[maxn];
50 int n;
51 int main() {
52     while (~scanf("%s %s", st, st1)) {
53         int len = strlen(st), len1 = strlen(st1);
54         n = 1;
55         while (n < (len << 1) || n < (len1 << 1)) n <= 1;
56         for (int i = 0; i < len; i++) A[len - 1 - i] = st[i] - '0';
57         for (int i = len; i <= n; i++) A[i] = 0;
58         for (int i = 0; i < len1; i++) B[len1 - 1 - i] = st1[i] - '0';
59         for (int i = len1; i <= n; i++) B[i] = 0;
60         NTT(A, n, 1);
61         NTT(B, n, 1);
62         for (int i = 0; i <= n - 1; i++) A[i] = A[i] * B[i] % mod;
63         NTT(A, n, -1);
64         for (int i = 0; i <= n - 1; i++) {
65             A[i + 1] += A[i] / 10;
66             A[i] %= 10;
67         }

```

```

68     n--;
69     while (A[n] / 10) A[n + 1] += A[n] / 10, A[n++] %= 10;
70     while (!A[n] && n > 0) n--;
71     for (int i = n; i >= 0; i--) printf("%lld", A[i]);
72     printf("\n");
73 }
74 return 0;
75 }

```

0.4.8 basic.cpp

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4  using ll = long long;
5
6  template <typename T>
7  T floor(T a, T n) {
8      if (n < 0) {
9          n = -n;
10         a = -a;
11     }
12     return a < 0 ? (a - n + 1) / n : a / n;
13 }
14
15 template <typename T>
16 T ceil(T a, T n) {
17     if (n < 0) {
18         n = -n;
19         a = -a;
20     }
21     return a < 0 ? a / n : (a + n - 1) / n;
22 }

```

0.4.9 binom.cpp

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4  using ll = long long;
5
6  constexpr int mod = 1e9 + 7;
7  // assume -mod <= x < 2mod
8  int norm(int x) {
9      if (x < 0) x += mod;
10     if (x >= mod) x -= mod;
11     return x;

```

```
12 }
13 template<class T>
14 T power(T a, ll b) {
15     T res = 1;
16     for (; b; b /= 2, a *= a) {
17         if (b % 2) res *= a;
18     }
19     return res;
20 }
21 struct Z {
22     int x;
23     Z(int x = 0) : x(norm(x)) {}
24     Z(int64_t x) : x(x % mod) {}
25     int val() const {
26         return x;
27     }
28     Z operator-() const {
29         return Z(norm(mod - x));
30     }
31     Z inv() const {
32         assert(x != 0);
33         return power(*this, mod - 2);
34     }
35     Z &operator*=(const Z &rhs) {
36         x = int64_t(x) * rhs.x % mod;
37         return *this;
38     }
39     Z &operator+=(const Z &rhs) {
40         x = norm(x + rhs.x);
41         return *this;
42     }
43     Z &operator-=(const Z &rhs) {
44         x = norm(x - rhs.x);
45         return *this;
46     }
47     Z &operator/=(const Z &rhs) {
48         return *this *= rhs.inv();
49     }
50     friend Z operator*(const Z &lhs, const Z &rhs) {
51         Z res = lhs;
52         res *= rhs;
53         return res;
54     }
55     friend Z operator+(const Z &lhs, const Z &rhs) {
56         Z res = lhs;
57         res += rhs;
58         return res;
59     }
```

```

60     friend Z operator-(const Z &lhs, const Z &rhs) {
61         Z res = lhs;
62         res -= rhs;
63         return res;
64     }
65     friend Z operator/(const Z &lhs, const Z &rhs) {
66         Z res = lhs;
67         res /= rhs;
68         return res;
69     }
70     friend istream &operator>>(istream &is, Z &a) {
71         int64_t v;
72         is >> v;
73         a = Z(v);
74         return is;
75     }
76     friend ostream &operator<<(ostream &os, const Z &a) {
77         return os << a.val();
78     }
79 };
80
81 struct Binom {
82     const int N;
83     vector<Z> fac, invfac;
84     Binom(int n) : N(n), fac(N + 1), invfac(N + 1) {
85         fac[0] = 1;
86         for (int i = 1; i <= N; i++) {
87             fac[i] = fac[i - 1] * i;
88         }
89         invfac[N] = fac[N].inv();
90         for (int i = N; i; i--) {
91             invfac[i - 1] = invfac[i] * i;
92         }
93     }
94
95     Z get(int n, int m) {
96         if (m < 0 || n < m) return Z(0);
97         return fac[n] * invfac[m] * invfac[n - m];
98     };
99 };

```

0.4.10 exgcd.cpp

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4  using ll = long long;

```

```

5
6 void solve() {
7     ll a, b, c;
8     cin >> a >> b >> c;
9
10    // ax + by = gcd(a, b)
11    // return tuple(d, x, y)
12    function<tuple<int64_t, int64_t, int64_t>(int64_t, int64_t)> exgcd = [&](int64_t a, int64_t b) {
13        if (b == 0) {
14            return tuple(a, (int64_t)1, (int64_t)0);
15        }
16        auto [d, x, y] = exgcd(b, a % b);
17        return tuple(d, y, x - a / b * y);
18    };
19
20    auto [d, x, y] = exgcd(a, b);
21
22    if (c % d != 0) {
23        cout << "-1\n";
24    } else {
25        x *= c / d;
26        y *= c / d;
27
28        ll dx = b / d;
29        ll dy = a / d;
30
31        ll l = ceil(1.0 * (-x + 1) / dx);
32        ll r = floor(1.0 * (y - 1) / dy);
33
34        if (l > r) {
35            cout << x + l * dx << " " << y - r * dy << "\n";
36        } else {
37            ll minx = x + l * dx, maxx = x + r * dx;
38            ll miny = y - r * dy, maxy = y - l * dy;
39            cout << r - l + 1 << " " << minx << " " << miny << " " << maxx << " " << maxy << "\n";
40        }
41    }
42 }
43
44 int main() {
45     ios::sync_with_stdio(false);
46     cin.tie(nullptr);
47
48     int t;
49     cin >> t;
50
51     while (t--) {
52         solve();

```



```

53     }
54
55     return 0;
56 }
57
58 // test problem: https://www.luogu.com.cn/problem/P5656

```

0.4.11 xor_basis.cpp

```

1  template<typename T, int BITS = 30>
2  struct xor_basis {
3      // A list of basis values sorted in decreasing order, where each value has a unique highest bit.
4      vector<T> basis(BITS);
5      int n = 0;
6
7      T min_value(T start) const {
8          if (n == BITS) {
9              return 0;
10         }
11         for (int i = 0; i < n; i++) {
12             start = min(start, start ^ basis[i]);
13         }
14         return start;
15     }
16
17     T max_value(T start = 0) const {
18         if (n == BITS) {
19             return (T(1) << BITS) - 1;
20         }
21         for (int i = 0; i < n; i++) {
22             start = max(start, start ^ basis[i]);
23         }
24         return start;
25     }
26
27     bool add(T x) {
28         x = min_value(x);
29         if (x == 0) {
30             return false;
31         }
32
33         basis[n++] = x;
34         int k = n - 1;
35
36         // Insertion sort.
37         while (k > 0 && basis[k] > basis[k - 1]) {
38             swap(basis[k], basis[k - 1]);

```

```

39         k--;
40     }
41
42     // Remove the highest bit of x from other basis elements.
43     // TODO: this can be removed for speed if desired.
44     for (int i = k - 1; i >= 0; i--) {
45         basis[i] = min(basis[i], basis[i] ^ x);
46     }
47
48     return true;
49 }
50
51 void merge(const xor_basis<T> &other) {
52     for (int i = 0; i < other.n && n < BITS; i++) {
53         add(other.basis[i]);
54     }
55 }
56
57 void merge(const xor_basis<T> &a, const xor_basis<T> &b) {
58     if (a.n > b.n) {
59         *this = a;
60         merge(b);
61     } else {
62         *this = b;
63         merge(a);
64     }
65 }
66 };

```

0.4.12 公式.md

```

1  ### 威尔逊定理:
2   $((k-1)!) \equiv -1 \pmod k$  , 其中  $k$  是质数 否则为 0 注意特判  $k=4$  时答案为 2。
3
4  ### 费马小定理:
5  如果  $p$  是一个质数, 而整数  $a$  不是  $p$  的倍数 (不成立则  $a^p \equiv a \pmod p$ ), 则有  $a^{p-1} \equiv 1 \pmod p$ 。
6
7  ### 欧拉定理:
8  若正整数  $a, n$  互质, 则  $a^{\phi(n)} \equiv 1 \pmod n$  其中  $\phi(n)$  是欧拉函数  $\phi(n)$  与  $n$  互质的数。
9
10 ### 第一类斯特林数:
11 设有多项式  $x(x-1)(x-2)\cdots(x-n+1)$ , 它的展开式形如  $s_n x^n - s_{n-1} x^{n-1} + s_{n-2} x^{n-2} - \cdots$ 。不考虑各项系数的符号, 将  $x^r$  的系数的绝对值记做  $s(n, r)$ , 称为第一类 Stirling 数。
12  $s(n, r)$  也是把  $n$  个不同的球排成  $r$  个非空循环排列的方法数。
13
14 ### 除法分块
15 计算  $\sum_{k=1}^n \left\lfloor \frac{n}{k} \right\rfloor$ 

```

- 16 如果直接暴力计算那么复杂度会达到 $O(n)$ ，但是可以发现 $\left\lfloor \frac{n}{k} \right\rfloor$ 的取值大约只有 \sqrt{n} 种可能，所以我们可以把 $\left\lfloor \frac{n}{k} \right\rfloor$ 取值相同的部分一起计算，这样复杂度就只有 $O(\sqrt{n})$ 。
- 17 具体操作如下：
- 18 我们可以维护一个区间 $[l, r]$ ，满足对于 $\forall i \in [l, r]$ ，保证 $\left\lfloor \frac{n}{i} \right\rfloor$ 的取值相同。首先 l 的初值为 1 ，那么可以计算到 $r = \left\lfloor \frac{n}{\left\lfloor \frac{n}{l} \right\rfloor} \right\rfloor$ ，保证在这个区间上 $\left\lfloor \frac{n}{i} \right\rfloor$ 的取值都等于 $\left\lfloor \frac{n}{l} \right\rfloor$ 。那么在这一段区间上对答案的贡献就为 $\left\lfloor \frac{n}{l} \right\rfloor \cdot (r - l + 1)$ 。然后 $l = r + 1$ ，再计算 r ，重复这样的过程即可在 $O(\sqrt{n})$ 的时间复杂度完成计算。
- 19 对于 $\sum_{k=1}^r \left\lfloor \frac{n}{k} \right\rfloor$ 的计算，我们可以发现改变求和上限不影响算法的正确性，所以可以用前缀相减的方法，即 $solve(r) - solve(l - 1)$ 来进行计算。

0.4.13 区间线性基.cpp

```

1  #pragma region
2  #include <algorithm>
3  #include <cmath>
4  #include <cstring>
5  #include <iomanip>
6  #include <iostream>
7  #include <map>
8  #include <queue>
9  #include <stack>
10 #include <string>
11 #include <vector>
12 typedef long long ll;
13 #define rep(i, a, n) for (ll i = a; i <= n; ++i)
14 #define per(i, a, n) for (ll i = n; i >= a; --i)
15 #define IO ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
16 using namespace std;
17 #pragma endregion
18 const int maxn = 5e5 + 5;
19 ll b[maxn][32], pos[maxn][32]; //插入a[i]后[1,i]直间的线性基第j位的数字是由原数组中哪一个数字得到
    的(最右)
20 inline bool insert(int r, ll x) { //维护区间[1,r]内的线性基
21     for (int i = 0; i <= 31; i++)
22         b[r][i] = b[r - 1][i], pos[r][i] = pos[r - 1][i];
23     ll tmp = r;
24     for (int i = 31; i >= 0; i--) {
25         if (x & (1LL << i)) {
26             if (b[r][i]) {
27                 if (pos[r][i] < tmp) {
28                     swap(pos[r][i], tmp);
29                     swap(b[r][i], x);
30                 }
31                 x ^= b[r][i];

```

```

32         } else {
33             b[r][i] = x;
34             pos[r][i] = tmp;
35             return 1;
36         }
37     }
38 }
39 return 0;
40 }
41 ll get_max(int l, int r) { //求区间[l,r]内的最大异或值
42     int ans = 0;
43     for (int i = 31; i >= 0; i--)
44         if (pos[r][i] >= 1 && (ans ^ b[r][i]) > ans)
45             ans ^= b[r][i];
46     return ans;
47 }
48 int main() {
49     IO;
50     int T;
51     cin >> T;
52     while (T--) {
53         int n, m, ans = 0;
54         cin >> n >> m;
55         for (int i = 1; i <= n; i++) {
56             ll x;
57             cin >> x;
58             insert(i, x);
59         }
60         for (int i = 0; i < m; i++) {
61             ll op, l, r;
62             cin >> op;
63             if (!op) {
64                 cin >> l >> r;
65                 l = (l ^ ans) % n + 1;
66                 r = (r ^ ans) % n + 1;
67                 if (l > r) swap(l, r);
68                 ans = get_max(l, r);
69                 cout << ans << endl;
70             } else {
71                 ll x;
72                 cin >> x;
73                 x ^= ans;
74                 insert(++n, x);
75             }
76         }
77     }
78 }

```

0.4.14 取模 gauss.cpp

```

1 ll a[55][55], x[55];
2 ll lcm(ll a, ll b) {
3     return a / __gcd(a, b) * b;
4 }
5 ll pow2(ll a, ll b) {
6     ll res = 1;
7     while (b) {
8         if (b & 1) res = res * a % mod;
9         a = a * a % mod;
10        b >>= 1;
11    }
12    return res;
13 }
14 ll inv(ll a, ll m) {
15     return pow2(a, mod - 2);
16 }
17 ll Gauss(ll m, ll n) {
18     ll r = 0, c = 0;
19     while (r < m && c < n) {
20         ll id = r;
21         for (ll i = r + 1; i < m; ++i)
22             if (abs(a[i][c]) > abs(a[id][c]))
23                 id = i;
24         if (id != r)
25             for (ll i = 0; i <= n; ++i)
26                 swap(a[r][i], a[id][i]);
27         if (abs(a[r][c]) != 0) {
28             for (ll i = r + 1; i < m; ++i) {
29                 if (abs(a[i][c]) == 0) continue;
30                 ll LCM = lcm(abs(a[i][c]), abs(a[r][c]));
31                 ll ta = LCM / abs(a[i][c]);
32                 ll tb = LCM / abs(a[r][c]);
33                 if (a[i][c] * a[r][c] < 0) tb = -tb;
34                 for (ll j = c; j <= n; ++j)
35                     a[i][j] = ((a[i][j] * ta - a[r][j] * tb) % mod + mod) % mod;
36             }
37             ++r;
38         }
39         ++c;
40     }
41     for (ll i = r; i < m; ++i)
42         if (a[i][n] != 0) return -1;
43     if (r < n) return n - r; //矩阵的秩等于未知数即可，不需要等于m。
44     for (ll i = n - 1; i >= 0; --i) {
45         ll tmp = a[i][n];
46         for (ll j = i + 1; j < n; ++j) {

```

```

47         if (a[i][j] != 0) {
48             tmp -= a[i][j] * x[j];
49             tmp = (tmp % mod + mod) % mod;
50         }
51     }
52     x[i] = (tmp * inv(a[i][i], mod)) % mod;
53     debug(i, x[i])
54 }
55 return 0;
56 }

```

0.4.15 容斥.cpp

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4  using ll = long long;
5
6  constexpr int mod = 998244353;
7  // assume -mod <= x < 2mod
8  int norm(int x) {
9      if (x < 0) x += mod;
10     if (x >= mod) x -= mod;
11     return x;
12 }
13 template<class T>
14 T power(T a, int b) {
15     T res = 1;
16     for (; b; b /= 2, a *= a)
17         if (b % 2) res *= a;
18     return res;
19 }
20 struct Z {
21     int x;
22     Z(int x = 0) : x(norm(x)) {}
23     Z(ll x) : x(x % mod) {}
24
25     int val() const {
26         return x;
27     }
28     Z operator-() const {
29         return Z(norm(mod - x));
30     }
31     Z inv() const {
32         assert(x != 0);
33         return power(*this, mod - 2);
34     }

```

```

35     Z &operator*=(const Z &rhs) {
36         x = ll(x) * rhs.x % mod;
37         return *this;
38     }
39     Z &operator+=(const Z &rhs) {
40         x = norm(x + rhs.x);
41         return *this;
42     }
43     Z &operator-=(const Z &rhs) {
44         x = norm(x - rhs.x);
45         return *this;
46     }
47     Z &operator/=(const Z &rhs) {
48         return *this *= rhs.inv();
49     }
50     friend Z operator*(const Z &lhs, const Z &rhs) {
51         Z res = lhs;
52         res *= rhs;
53         return res;
54     }
55     friend Z operator+(const Z &lhs, const Z &rhs) {
56         Z res = lhs;
57         res += rhs;
58         return res;
59     }
60     friend Z operator-(const Z &lhs, const Z &rhs) {
61         Z res = lhs;
62         res -= rhs;
63         return res;
64     }
65     friend Z operator/(const Z &lhs, const Z &rhs) {
66         Z res = lhs;
67         res /= rhs;
68         return res;
69     }
70 };
71
72 int main() {
73     ios::sync_with_stdio(false);
74     cin.tie(nullptr);
75
76     int n, L;
77     cin >> n >> L;
78     vector<int> s(n);
79     for (int i = 0; i < n; ++i) {
80         string t;
81         cin >> t;
82         for (auto c : t) {

```

```

83         s[i] |= 1 << (c - 'a');
84     }
85 }
86
87 auto cul = [&](int cur) {
88     int ans = 0;
89     while (cur) {
90         ans += cur & 1;
91         cur >>= 1;
92     }
93     return ans;
94 };
95
96 Z ans = 0;
97 vector<Z> f(1 << n);
98 for (int mask = 1; mask < (1 << n); ++mask) {
99     int cur = (1 << 26) - 1;
100    for (int i = 0; i < n; ++i) {
101        if (mask >> i & 1) {
102            cur &= s[i];
103        }
104    }
105    f[mask] = power(Z(cul(cur)), L);
106    ans += (cul(mask) & 1 ? 1 : -1) * f[mask];
107 }
108
109 cout << ans.val() << "\n";
110
111 return 0;
112 }
113
114 // test problem: https://atcoder.jp/contests/abc246/tasks/abc246\_f

```

0.4.16 异或 gauss.cpp

```

1  #include <math.h>
2  #include <stdio.h>
3  #include <string.h>
4
5  #include <algorithm>
6  #include <iostream>
7  using namespace std;
8
9  const int MAXN = 50;
10
11 int a[MAXN][MAXN]; //增广矩阵
12 int x[MAXN];        //解集

```



```

13 int free_x[MAXN]; //标记是否是不确定的变元
14
15 // 高斯消元法解方程组(Gauss-Jordan elimination).(-2表示有浮点数解, 但无整数解,
16 //-1表示无解, 0表示唯一解, 大于0表示无穷解, 并返回自由变元的个数)
17 //有equ个方程, var个变元。增广矩阵行数为equ,分别为0到equ-1,列数为var+1,分别为0到var.
18 int Gauss(int equ, int var) {
19     int i, j, k;
20     int max_r; // 当前这列绝对值最大的行.
21     int col; //当前处理的列
22
23     for (int i = 0; i <= var; i++) {
24         x[i] = 0;
25         free_x[i] = 1;
26     }
27
28     //转换为阶梯阵.
29     col = 0; // 当前处理的列
30     for (k = 0; k < equ && col < var; k++, col++) { // 枚举当前处理的行.
31         // 找到该col列元素绝对值最大的那行与第k行交换.(为了在除法时减小误差)
32         max_r = k;
33         for (i = k + 1; i < equ; i++) {
34             if (abs(a[i][col]) > abs(a[max_r][col]))
35                 max_r = i;
36         }
37         if (max_r != k) { // 与第k行交换.
38             for (j = k; j < var + 1; j++)
39                 swap(a[k][j], a[max_r][j]);
40         }
41         if (a[k][col] == 0) { // 说明该col列第k行以下全是0了, 则处理当前行的下一列.
42             k--;
43             continue;
44         }
45         for (i = k + 1; i < equ; i++) { // 枚举要删去的行.
46             if (a[i][col] != 0) {
47                 for (j = col; j < var + 1; j++)
48                     a[i][j] ^= a[k][j];
49             }
50         }
51     }
52     // 1. 无解的情况: 化简的增广阵中存在(0, 0, ..., a)这样的行(a != 0).
53     for (i = k; i < equ; i++) { // 对于无穷解来说, 如果要判断哪些是自由变元, 那么初等行变换中的交换
        就会影响, 则要记录交换.
54         if (a[i][col] != 0)
55             return -1;
56     }
57     return var - k;
58 }
59

```

```

60 int start[MAXN];
61 int en[MAXN];
62
63 int main() {
64     // freopen("in.txt","r",stdin);
65     // freopen("out.txt","w",stdout);
66     int u, v;
67     int T;
68     int n;
69     scanf("%d", &T);
70     while (T--) {
71         scanf("%d", &n);
72         for (int i = 0; i < n; i++)
73             scanf("%d", &start[i]);
74         for (int i = 0; i < n; i++)
75             scanf("%d", &en[i]);
76         memset(a, 0, sizeof(a));
77         while (scanf("%d%d", &u, &v)) {
78             if (u == 0 && v == 0)
79                 break;
80             a[v - 1][u - 1] = 1;
81         }
82         for (int i = 0; i < n; i++)
83             a[i][i] = 1;
84         for (int i = 0; i < n; i++)
85             a[i][n] = start[i] ^ en[i];
86         int ans = Gauss(n, n);
87         if (ans == -1)
88             printf("Oh,it's impossible~!!\n");
89         else
90             printf("%d\n", 1 << ans);
91     }
92     return 0;
93 }

```

0.4.17 斐波那契.cpp

```

1  #include <bits/stdc++.h>
2  #define rep(i, a, n) for (int i = a; i <= n; ++i)
3  #define per(i, a, n) for (int i = n; i >= a; --i)
4  #ifdef LOCAL
5  #include "Print.h"
6  #define de(...) W('[', #__VA_ARGS__, " ] =", __VA_ARGS__)
7  #else
8  #define de(...)
9  #endif
10 using namespace std;

```

```

11 typedef long long ll;
12 const int maxn = 2e5 + 5;
13 const ll mod = 1e9 + 9;
14 void add(ll &x, ll y) { if ((x += y) >= mod) x -= mod; }
15 void sub(ll &x, ll y) { if ((x -= y) < 0) x += mod; }
16 struct mat {
17     ll a[3][3];
18     mat(int op) {
19         if (op == 1) a[1][1] = a[2][2] = 1, a[1][2] = a[2][1] = 0;
20         if (op == 0) a[1][1] = a[1][2] = a[2][1] = a[2][2] = 0;
21     }
22     mat operator*(const mat &A) {
23         mat ans(0);
24         rep(i, 1, 2) rep(j, 1, 2) rep(k, 1, 2)
25             add(ans.a[i][j], a[i][k] * A.a[k][j] % mod);
26         return ans;
27     }
28 };
29 mat powmod(mat a, ll b) {
30     mat ans(1);
31     while (b) {
32         if (b & 1) ans = ans * a;
33         b >>= 1; a = a * a;
34     }
35     return ans;
36 }
37 ll powmod(ll a, ll b) {
38     ll ans = 1;
39     while (b) {
40         if (b & 1) ans = ans * a % mod;
41         b >>= 1; a = a * a % mod;
42     }
43     return ans;
44 }
45 int case_Test() {
46     auto f = [&](ll n) -> ll {
47         if (n == 1) return 1;
48         if (n == 2) return 2;
49         mat A(0);
50         A.a[1][1] = A.a[1][2] = A.a[2][1] = 1;
51         A = powmod(A, n - 2);
52         return (A.a[1][1] * 2 + A.a[1][2]) % mod;
53     };
54     ll n;
55     scanf("%lld", &n);
56     printf("%lld\n", f(n));
57     return 0;
58 }

```

```

59 int main() {
60 #ifdef LOCAL
61     freopen("/Users/chenjinglong/cpp_code/in.in", "r", stdin);
62     freopen("/Users/chenjinglong/cpp_code/out.out", "w", stdout);
63     clock_t start = clock();
64 #endif
65     int _ = 1;
66     scanf("%d", &_);
67     while (--) case_Test();
68 #ifdef LOCAL
69     printf("Time used: %.3lfs\n", (double)(clock() - start) / CLOCKS_PER_SEC);
70 #endif
71     return 0;
72 }

```

0.4.18 求逆元.cpp

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  typedef long long ll;
4  const ll Mod = 1e9 + 7;
5  ll exgcd(ll a, ll b, ll &x, ll &y) { //扩展欧几里得算法
6      if (b == 0) {
7          x = 1, y = 0;
8          return a;
9      }
10     ll ret = exgcd(b, a % b, y, x);
11     y -= a / b * x;
12     return ret;
13 }
14 ll getInv(int a, int mod) { //求a在mod下的逆元, 不存在逆元返回-1s
15     ll x, y;
16     ll d = exgcd(a, mod, x, y);
17     return d == 1 ? (x % mod + mod) % mod : -1;
18 }
19
20 int main() {
21     ll x = getInv(24, Mod);
22     int T;
23     scanf("%d", &T);
24     while (T--) {
25         ll n;
26         ll ans = 1;
27         scanf("%lld", &n);
28         for (ll i = n; i <= n + 3; ++i)
29             ans = (ans * i) % Mod;
30         ans = ans * x % Mod;

```

```

31     printf("%lld\n", ans);
32 }
33 }

```

0.4.19 浮点型 gauss.cpp

```

1  #include <cmath>
2  #include <iostream>
3  using namespace std;
4  #define eps 1e-9
5  const int maxn = 5e2 + 5;
6  double a[maxn][maxn]; //增广矩阵
7  double x[maxn];       //解集
8  int n;
9  int gauss() {
10     for (int i = 1; i <= n; i++) { //遍历每一行
11         int max_r = i; //找到i+1--n中a[j][i]绝对值最大的一行
12         for (int j = i + 1; j <= n; j++) //遍历i+1至n去寻找
13             if (fabs(a[j][i]) > fabs(a[max_r][i]))
14                 max_r = j;
15
16         for (int k = 1; k <= n + 1; k++) //将max_r与i交换
17             swap(a[max_r][k], a[i][k]);
18         if (fabs(a[i][i]) < eps)
19             continue; //若a[i][i]<0,则说明该行全为0,跳过
20         double p = a[i][i]; //取出a[i][i]
21         for (int j = 1; j <= n + 1; j++)
22             a[i][j] /= p; //归一化
23         for (int j = i + 1; j <= n; j++) //消元,消掉其余的a[j][i]
24             {
25                 if (i != j) {
26                     double tmp = a[j][i];
27                     for (int k = 1; k <= n + 1; k++)
28                         a[j][k] -= a[i][k] * tmp; //套公式a[j][k]=a[j][k]-(a[i][k]/a[i][i])*a[j][i];
29                 }
30             }
31     }
32     int free_num = 0; //自由元数量统计
33     for (int i = 1; i <= n; i++) {
34         int ans = 0;
35         for (int j = 1; j <= n + 1; j++) //统计一行中零元数量
36             if (fabs(a[i][j]) < eps) ans++; //注意精度修正
37         if (ans == n && a[i][n + 1])
38             return -1; //若为无解情形
39         if (ans == n + 1)
40             free_num++; //无数组解的情形
41     }

```

```

42     if (!free_num) { //唯一解回带求解
43         for (int i = n - 1; i >= 1; i--)
44             for (int j = i + 1; j <= n; j++)
45                 a[i][n + 1] -= a[j][n + 1] * a[i][j]; //从下往上, 从右往左
46         for (int i = 1; i <= n; i++)
47             x[i] = a[i][n + 1];
48         return free_num;
49     }
50 }
51
52 int main() {
53     cin >> n;
54     for (int i = 1; i <= n; i++)
55         for (int j = 1; j <= n + 1; j++)
56             cin >> a[i][j];
57     int t = gauss();
58     if (t == 0) {
59         for (int i = 1; i <= n; i++) {
60             if (fabs(x[i]) < eps)
61                 printf("0\n");
62             else
63                 printf("%.2f\n", x[i]);
64         }
65     } else
66         cout << "No Solution\n";
67 }

```

0.4.20 第二类斯特林数.cpp

```

1  #pragma region
2  #include <algorithm>
3  #include <cmath>
4  #include <cstring>
5  #include <iomanip>
6  #include <iostream>
7  #include <map>
8  #include <queue>
9  #include <set>
10 #include <stack>
11 #include <string>
12 #include <unordered_map>
13 #include <vector>
14 using namespace std;
15 typedef long long ll;
16 #define rep(i, a, n) for (int i = a; i <= n; ++i)
17 #define per(i, a, n) for (int i = n; i >= a; --i)
18 namespace fastIO {

```

```

19 #define BUF_SIZE 100000
20 #define OUT_SIZE 100000
21 //fread->R
22 bool IOError = 0;
23 //inline char nc(){char ch=getchar();if(ch==-1)IOError=1;return ch;}
24 inline char nc() {
25     static char buf[BUF_SIZE], *p1 = buf + BUF_SIZE, *pend = buf + BUF_SIZE;
26     if (p1 == pend) {
27         p1 = buf;
28         pend = buf + fread(buf, 1, BUF_SIZE, stdin);
29         if (pend == p1) {
30             IOError = 1;
31             return -1;
32         }
33     }
34     return *p1++;
35 }
36 inline bool blank(char ch) { return ch == ' ' || ch == '\n' || ch == '\r' || ch == '\t'; }
37 template <class T>
38 inline bool R(T &x) {
39     bool sign = 0;
40     char ch = nc();
41     x = 0;
42     for (; blank(ch); ch = nc())
43         ;
44     if (IOError) return false;
45     if (ch == '-') sign = 1, ch = nc();
46     for (; ch >= '0' && ch <= '9'; ch = nc()) x = x * 10 + ch - '0';
47     if (sign) x = -x;
48     return true;
49 }
50 inline bool R(double &x) {
51     bool sign = 0;
52     char ch = nc();
53     x = 0;
54     for (; blank(ch); ch = nc())
55         ;
56     if (IOError) return false;
57     if (ch == '-') sign = 1, ch = nc();
58     for (; ch >= '0' && ch <= '9'; ch = nc()) x = x * 10 + ch - '0';
59     if (ch == '.') {
60         double tmp = 1;
61         ch = nc();
62         for (; ch >= '0' && ch <= '9'; ch = nc())
63             tmp /= 10.0, x += tmp * (ch - '0');
64     }
65     if (sign)
66         x = -x;

```

```

67     return true;
68 }
69 inline bool R(char *s) {
70     char ch = nc();
71     for (; blank(ch); ch = nc())
72         ;
73     if (IOError)
74         return false;
75     for (; !blank(ch) && !IOError; ch = nc())
76         *s++ = ch;
77     *s = 0;
78     return true;
79 }
80 inline bool R(char &c) {
81     c = nc();
82     if (IOError) {
83         c = -1;
84         return false;
85     }
86     return true;
87 }
88 template <class T, class... U>
89 bool R(T &h, U &... t) { return R(h) && R(t...); }
90 #undef OUT_SIZE
91 #undef BUF_SIZE
92 }; // namespace fastIO
93 using namespace fastIO;
94 template <class T>
95 void _W(const T &x) { cout << x; }
96 void _W(const int &x) { printf("%d", x); }
97 void _W(const int64_t &x) { printf("%lld", x); }
98 void _W(const double &x) { printf("%.16f", x); }
99 void _W(const char &x) { putchar(x); }
100 void _W(const char *x) { printf("%s", x); }
101 template <class T, class U>
102 void _W(const pair<T, U> &x) { _W(x.F), putchar(' '), _W(x.S); }
103 template <class T>
104 void _W(const vector<T> &x) {
105     for (auto i = x.begin(); i != x.end(); _W(*i++))
106         if (i != x.cbegin()) putchar(' ');
107 }
108 void W() {}
109 template <class T, class... U>
110 void W(const T &head, const U &... tail) { _W(head), putchar(sizeof...(tail) ? ' ' : '\n'), W(tail
    ...); }
111 #pragma endregion
112 const int maxn = 1005;
113 const ll mod = 1e9 + 7;

```



```

114
115 ll Stirling[maxn][maxn], fac[maxn];
116 void init() {
117     fac[1] = 1;
118     rep(i, 2, 1000) fac[i] = fac[i - 1] * i % mod;
119     Stirling[0][0] = 0;
120     Stirling[1][1] = 1;
121     for (ll i = 2; i < maxn; i++)
122         for (ll j = 1; j <= i; j++)
123             Stirling[i][j] = (Stirling[i - 1][j - 1] + j * Stirling[i - 1][j]) % mod;
124 }

```

0.4.21 线性基类.cpp

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4  using ll = long long;
5
6  struct L_B {
7      ll b[61], p[61]; //线性基, 重构线性基
8      int cnt, flag;    //重构线性基的大小, 是否异或为0
9      L_B() {
10         memset(b, 0, sizeof(b));
11         memset(p, 0, sizeof(p));
12         cnt = 0, flag = 0;
13     }
14     inline bool insert(ll x) {
15         for (int i = 60; i >= 0 && x; i--)
16             if (x & (1LL << i)) {
17                 if (b[i]) x ^= b[i];
18                 else {
19                     b[i] = x;
20                     return true;
21                 }
22             }
23         flag = 1;
24         return false;
25     }
26     ll qmax() {
27         ll ans = 0;
28         for (int i = 60; i >= 0; i--)
29             if ((ans ^ b[i]) > ans) ans ^= b[i];
30         return ans;
31     }
32     ll qmin() {
33         if (flag) return 0;

```

```

34     for (int i = 0; i <= 60; i++)
35         if (b[i]) return b[i];
36     return 0;
37 }
38 inline void rebuild() {
39     for (int i = 60; i >= 1; i--) {
40         if (b[i])
41             for (int j = i - 1; j >= 0; j--)
42                 if (b[i] & (1LL << j)) b[i] ^= b[j];
43     }
44     //异或p[i]对名次的贡献为1<<i
45     for (int i = 0; i <= 60; i++)
46         if (b[i]) p[cnt++] = b[i];
47 }
48 ll kth(ll k) {
49     if (flag) --k;
50     if (!k) return 0;
51     ll ans = 0;
52     if (k >= (1LL << cnt)) return -1;
53     for (int i = 0; i <= cnt; ++i)
54         if (k & (1LL << i)) ans ^= p[i];
55     return ans;
56 }
57 };
58 L_B merge(const L_B &n1, const L_B &n2) {
59     L_B ans = n1;
60     for (int i = 60; i >= 0; i--)
61         if (n2.b[i]) ans.insert(n2.b[i]);
62     ans.flag = n1.flag | n2.flag;
63     return ans;
64 }

```

0.4.22 除法分块.cpp

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4  using ll = long long;
5
6  int main() {
7      ios::sync_with_stdio(false);
8      cin.tie(nullptr);
9
10     // n / l = n / (l + 1) = ... = n / r, 1 <= l <= r <= k
11     auto block = [&](int n, int k) {
12         vector<array<int, 2>> ans;
13         for (int l = 1, r; l <= k; l = r + 1) {

```

```

14         r = (n / l ? min(k, n / (n / l)) : k);
15         ans.push_back({l, r});
16     }
17     for (auto [l, r] : ans) {
18         cout << l << " " << r << " " << n / l << "\n";
19     }
20 };
21
22 block(24, 24);
23
24 return 0;
25 }

```

0.5 Others

0.5.1 BigNum2.cpp

```

1 // #include <bits/stdc++.h>
2 #include <iostream>
3 #include <vector>
4 using namespace std;
5 struct BigNum : vector<int> //用标准库vector做基类，完美解决位数问题，同时更易于实现
6 {
7     //将低精度转高精度的初始化，可以自动被编译器调用
8     //因此无需单独写高精度数和低精度数的运算函数，十分方便
9     BigNum(int n = 0) //默认初始化为0，但0的保存形式为空
10    {
11        push_back(n);
12        check();
13    }
14    BigNum &check() //在各类运算中经常用到的进位小函数，不妨内置
15    {
16        while (!empty() && !back())
17            pop_back(); //去除最高位可能存在的0
18        if (empty())
19            return *this;
20        for (int i = 1; i < size(); ++i) //处理进位
21        {
22            (*this)[i] += (*this)[i - 1] / 10;
23            (*this)[i - 1] %= 10;
24        }
25        while (back() >= 10) {
26            push_back(back() / 10);
27            (*this)[size() - 2] %= 10;
28        }
29        return *this; //为使用方便，将进位后的自身返回引用
30    }

```

```

31 };
32 //输入输出
33 istream &operator>>(istream &is, BigNum &n) {
34     string s;
35     is >> s;
36     n.clear();
37     for (int i = s.size() - 1; i >= 0; --i)
38         n.push_back(s[i] - '0');
39     return is;
40 }
41 ostream &operator<<(ostream &os, const BigNum &n) {
42     if (n.empty())
43         os << 0;
44     for (int i = n.size() - 1; i >= 0; --i)
45         os << n[i];
46     return os;
47 }
48 //比较, 只需要写两个, 其他的直接代入即可
49 //常量引用当参数, 避免拷贝更高效
50 bool operator!=(const BigNum &a, const BigNum &b) {
51     if (a.size() != b.size())
52         return 1;
53     for (int i = a.size() - 1; i >= 0; --i)
54         if (a[i] != b[i])
55             return 1;
56     return 0;
57 }
58 bool operator==(const BigNum &a, const BigNum &b) {
59     return !(a != b);
60 }
61 bool operator<(const BigNum &a, const BigNum &b) {
62     if (a.size() != b.size())
63         return a.size() < b.size();
64     for (int i = a.size() - 1; i >= 0; --i)
65         if (a[i] != b[i])
66             return a[i] < b[i];
67     return 0;
68 }
69 bool operator>(const BigNum &a, const BigNum &b) {
70     return b < a;
71 }
72 bool operator<=(const BigNum &a, const BigNum &b) {
73     return !(a > b);
74 }
75 bool operator>=(const BigNum &a, const BigNum &b) {
76     return !(a < b);
77 }
78 //加法, 先实现+=, 这样更简洁高效

```

```

79 BigNum &operator+=(BigNum &a, const BigNum &b) {
80     if (a.size() < b.size())
81         a.resize(b.size());
82     for (int i = 0; i != b.size(); ++i)
83         a[i] += b[i];
84     return a.check();
85 }
86 BigNum operator+(BigNum a, const BigNum &b) {
87     return a += b;
88 }
89 //减法, 返回差的绝对值, 由于后面有交换, 故参数不用引用
90 BigNum &operator-=(BigNum &a, BigNum b) {
91     if (a < b)
92         swap(a, b);
93     for (int i = 0; i != b.size(); a[i] -= b[i], ++i)
94         if (a[i] < b[i]) //需要借位
95         {
96             int j = i + 1;
97             while (!a[j])
98                 ++j;
99             while (j > i) {
100                 --a[j];
101                 a[--j] += 10;
102             }
103         }
104     return a.check();
105 }
106 BigNum operator-(BigNum a, const BigNum &b) {
107     return a -= b;
108 }
109 //乘法不能先实现*=, 原因自己想
110 BigNum operator*(const BigNum &a, const BigNum &b) {
111     BigNum n;
112     n.assign(a.size() + b.size() - 1, 0);
113     for (int i = 0; i != a.size(); ++i)
114         for (int j = 0; j != b.size(); ++j)
115             n[i + j] += a[i] * b[j];
116     return n.check();
117 }
118 BigNum &operator*=(BigNum &a, const BigNum &b) {
119     return a = a * b;
120 }
121 //除法和取模先实现一个带余除法函数
122 BigNum divmod(BigNum &a, const BigNum &b) {
123     BigNum ans;
124     for (int t = a.size() - b.size(); a >= b; --t) {
125         BigNum d;
126         d.assign(t + 1, 0);

```

```

127         d.back() = 1;
128         BigNum c = b * d;
129         while (a >= c) {
130             a -= c;
131             ans += d;
132         }
133     }
134     return ans;
135 }
136 BigNum operator/(BigNum a, const BigNum &b) {
137     return divmod(a, b);
138 }
139 BigNum &operator/=(BigNum &a, const BigNum &b) {
140     return a = a / b;
141 }
142 BigNum &operator%=(BigNum &a, const BigNum &b) {
143     divmod(a, b);
144     return a;
145 }
146 BigNum operator%(BigNum a, const BigNum &b) {
147     return a %= b;
148 }
149 //顺手实现一个快速幂，可以看到和普通快速幂几乎无异
150 BigNum pow(const BigNum &n, const BigNum &k) {
151     if (k.empty())
152         return 1;
153     if (k == 2)
154         return n * n;
155     if (k.back() % 2)
156         return n * pow(n, k - 1);
157     return pow(pow(n, k / 2), 2);
158 }
159
160 int main() {
161 }

```

0.5.2 Simulated_annealing.cpp

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4  using ll = long long;
5
6  const double eps = 1e-8;
7
8  int main() {
9      ios::sync_with_stdio(false);

```

```

10     cin.tie(nullptr);
11
12     int n;
13     cin >> n;
14
15     vector<tuple<int, int, int>> a(n);
16     for (int i = 0; i < n; ++i) {
17         int x, y, z;
18         cin >> x >> y >> z;
19         a[i] = tuple(x, y, z);
20     }
21
22     auto solve = [&]() {
23         double step = 10000, ans = 1e30;
24         tuple<double, double, double> tp;
25         int pos = 0;
26
27         auto dis = [&](auto A, auto B) {
28             auto [x1, y1, z1] = A;
29             auto [x2, y2, z2] = B;
30             return sqrt((x2 - x1) * (x2 - x1) + (y2 - y1) * (y2 - y1) + (z2 - z1) * (z2 - z1));
31         };
32
33         while (step > eps) {
34             for (int i = 0; i < n; ++i) { // 找一个最远的点
35                 if (dis(tp, a[pos]) < dis(tp, a[i])) {
36                     pos = i;
37                 }
38             }
39             double mt = dis(tp, a[pos]);
40             ans = min(ans, mt);
41             auto [x, y, z] = tp;
42             auto [px, py, pz] = a[pos];
43             x += (px - x) / mt * step;
44             y += (py - y) / mt * step;
45             z += (pz - z) / mt * step;
46             tp = tuple(x, y, z);
47
48             step *= 0.98;
49         }
50         return ans;
51     };
52
53     cout << fixed << setprecision(8) << solve() << "\n";
54
55     return 0;
56 }
57

```

58 // test problem: <https://vjudge.net/problem/Gym-101981D>

0.5.3 Z.cpp

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4  using ll = long long;
5
6  constexpr int mod = 1e9 + 7;
7  // assume -mod <= x < 2mod
8  int norm(int x) {
9      if (x < 0) x += mod;
10     if (x >= mod) x -= mod;
11     return x;
12 }
13 template<class T>
14 T power(T a, int64_t b) {
15     T res = 1;
16     for (; b; b /= 2, a *= a) {
17         if (b % 2) res *= a;
18     }
19     return res;
20 }
21 struct Z {
22     int x;
23     Z(int x = 0) : x(norm(x)) {}
24     Z(int64_t x) : x(x % mod) {}
25     int val() const {
26         return x;
27     }
28     Z operator-() const {
29         return Z(norm(mod - x));
30     }
31     Z inv() const {
32         assert(x != 0);
33         return power(*this, mod - 2);
34     }
35     Z &operator*=(const Z &rhs) {
36         x = int64_t(x) * rhs.x % mod;
37         return *this;
38     }
39     Z &operator+=(const Z &rhs) {
40         x = norm(x + rhs.x);
41         return *this;
42     }
43     Z &operator-=(const Z &rhs) {

```



```

44     x = norm(x - rhs.x);
45     return *this;
46 }
47 Z &operator/=(const Z &rhs) {
48     return *this *= rhs.inv();
49 }
50 friend Z operator*(const Z &lhs, const Z &rhs) {
51     Z res = lhs;
52     res *= rhs;
53     return res;
54 }
55 friend Z operator+(const Z &lhs, const Z &rhs) {
56     Z res = lhs;
57     res += rhs;
58     return res;
59 }
60 friend Z operator-(const Z &lhs, const Z &rhs) {
61     Z res = lhs;
62     res -= rhs;
63     return res;
64 }
65 friend Z operator/(const Z &lhs, const Z &rhs) {
66     Z res = lhs;
67     res /= rhs;
68     return res;
69 }
70 friend istream &operator>>(istream &is, Z &a) {
71     int64_t v;
72     is >> v;
73     a = Z(v);
74     return is;
75 }
76 friend ostream &operator<<(ostream &os, const Z &a) {
77     return os << a.val();
78 }
79 };

```

0.5.4 bignum.cpp

```

1 #include <cstring>
2 #include <iostream>
3 using namespace std;
4
5 class BigNum {
6     private:
7         int a[1000];
8         int len;

```

```

9
10 public:
11     BigNum() {
12         len = 1;
13         memset(a, 0, sizeof(a));
14     }
15     BigNum(const int b);
16     BigNum(char *s);
17     BigNum(const BigNum &T);
18     BigNum &operator=(const BigNum &n);
19
20     friend istream &operator>>(istream &, BigNum &);
21     friend ostream &operator<<(ostream &, BigNum &);
22
23     BigNum operator+(const BigNum &T) const;
24     BigNum operator-(const BigNum &T) const;
25     BigNum operator*(const BigNum &T) const;
26     BigNum operator/(const int &b) const;
27     BigNum operator|(const BigNum &T) const;
28     BigNum operator%(const BigNum &T) const;
29
30     bool operator>(const BigNum &T) const;
31     bool operator>(const int &t) const;
32 };
33
34 BigNum::BigNum(const int b) {
35     len = 0;
36     memset(a, 0, sizeof(a));
37     int t = b;
38     while (t) {
39         int x = t % 10;
40         a[len++] = x;
41         t /= 10;
42     }
43 }
44 BigNum::BigNum(char *s) {
45     memset(a, 0, sizeof(a));
46     int l = strlen(s);
47     len = l;
48     int cnt = 0;
49     for (int i = l - 1; i >= 0; --i)
50         a[cnt++] = s[i] - '0';
51 }
52 BigNum::BigNum(const BigNum &T) : len(T.len) {
53     memset(a, 0, sizeof(a));
54     for (int i = 0; i < len; ++i)
55         a[i] = T.a[i];
56 }

```

```

57 | BigNum &BigNum::operator=(const BigNum &n) {
58 |     len = n.len;
59 |     memset(a, 0, sizeof(a));
60 |     for (int i = 0; i < len; ++i)
61 |         a[i] = n.a[i];
62 |     return *this;
63 | }
64 | istream &operator>>(istream &in, BigNum &b) {
65 |     char ch[1000];
66 |     in >> ch;
67 |     int l = strlen(ch);
68 |     int count = 0;
69 |     for (int i = l - 1; i > 0; --i) {
70 |         b.a[count++] = ch[i] - '0';
71 |     }
72 |     if (ch[0] == '-')
73 |         b.a[count - 1] = 0 - b.a[count - 1];
74 |     else
75 |         b.a[count++] = ch[0] - '0';
76 |     b.len = count;
77 |     return in;
78 | }
79 | ostream &operator<<(ostream &out, BigNum &b) {
80 |     for (int i = b.len - 1; i >= 0; --i)
81 |         cout << b.a[i];
82 |     return out;
83 | }
84 | BigNum BigNum::operator+(const BigNum &T) const {
85 |     BigNum t(*this);
86 |     int big;
87 |     big = T.len > len ? T.len : len;
88 |     for (int i = 0; i < big; ++i) {
89 |         t.a[i] += T.a[i];
90 |         if (t.a[i] >= 10) {
91 |             t.a[i + 1]++;
92 |             t.a[i] -= 10;
93 |         }
94 |     }
95 |     if (t.a[big] != 0)
96 |         t.len = big + 1;
97 |     else
98 |         t.len = big;
99 |     return t;
100 | }
101 | BigNum BigNum::operator-(const BigNum &T) const {
102 |     int big;
103 |     bool flag;
104 |     BigNum t1, t2;

```

```

105     if (*this > T) {
106         t1 = *this;
107         t2 = T;
108         flag = 0;
109     } else {
110         t1 = T;
111         t2 = *this;
112         flag = 1;
113     }
114     big = t1.len;
115     for (int i = 0; i < big; ++i) {
116         if (t1.a[i] < t2.a[i]) {
117             int j = i + 1;
118             while (t1.a[j] == 0)
119                 j++;
120             t1.a[j--]--;
121             while (j > i)
122                 t1.a[j--] += 9;
123             t1.a[i] += 10 - t2.a[i];
124         } else
125             t1.a[i] -= t2.a[i];
126     }
127     t1.len = big;
128     while (t1.a[t1.len - 1] == 0 && t1.len > 1) {
129         t1.len--;
130         big--;
131     }
132     if (flag)
133         t1.a[big - 1] = 0 - t1.a[big - 1];
134     return t1;
135 }
136 BigNum BigNum::operator*(const BigNum &T) const {
137     BigNum ret;
138     int up;
139     int temp, temp1;
140     int i, j;
141     for (i = 0; i < len; ++i) {
142         up = 0;
143         for (j = 0; j < T.len; ++j) {
144             temp = a[i] * T.a[j] + ret.a[i + j] + up;
145             if (temp >= 10) {
146                 temp1 = temp % 10;
147                 up = temp / 10;
148                 ret.a[i + j] = temp1;
149             } else {
150                 up = 0;
151                 ret.a[i + j] = temp;
152             }

```

```

153     }
154     if (up != 0)
155         ret.a[i + j] = up;
156 }
157 ret.len = i + j;
158 while (ret.a[ret.len - 1] == 0 && ret.len > 1)
159     ret.len--;
160 return ret;
161 }
162 BigNum BigNum::operator/(const int &b) const {
163     BigNum ret;
164     int down = 0;
165     for (int i = len - 1; i >= 0; --i) {
166         ret.a[i] = (a[i] + down * 10) / b;
167         down = a[i] + down * 10 - ret.a[i] * b;
168     }
169     ret.len = len;
170     while (ret.a[ret.len - 1] == 0 && ret.len > 1)
171         ret.len--;
172     return ret;
173 }
174 BigNum BigNum::operator|(const BigNum &T) const {
175     BigNum ans;
176     BigNum a = *this, b = T;
177     int len1 = len, len2 = T.len;
178     int t = len1 - len2;
179     BigNum x = 1;
180     BigNum ten = 10;
181     for (int i = 0; i < t; ++i) {
182         b = b * ten;
183         x = x * ten;
184     }
185     while (a > T || (!(a > T) && !(T > a))) {
186         while (a > b || (!(a > b) && !(b > a))) {
187             a = a - b;
188             ans = ans + x;
189         }
190         b = b / 10;
191         x = x / 10;
192     }
193     return ans;
194 }
195 BigNum BigNum::operator%(const BigNum &T) const {
196     BigNum ans;
197     BigNum a = *this, b = T;
198     int len1 = len, len2 = T.len;
199     int t = len1 - len2;
200     BigNum x = 1;

```

```

201     BigNum ten = 10;
202     for (int i = 0; i < t; ++i) {
203         b = b * ten;
204         x = x * ten;
205     }
206     while (a > T || (!(a > T) && !(T > a))) {
207         while (a > b || (!(a > b) && !(b > a))) {
208             a = a - b;
209             ans = ans + x;
210         }
211         b = b / 10;
212         x = x / 10;
213     }
214     return a;
215 }
216 bool BigNum::operator>(const BigNum &T) const {
217     int ln;
218     if (len > T.len)
219         return true;
220     else if (len < T.len)
221         return false;
222
223     ln = len - 1;
224     while (a[ln] == T.a[ln] && ln >= 0)
225         ln--;
226     if (ln >= 0 && a[ln] > T.a[ln])
227         return true;
228     else
229         return false;
230 }
231 bool BigNum::operator>(const int &t) const {
232     BigNum b(t);
233     return *this > b;
234 }
235
236 int main() {
237
238 }

```

0.5.5 gen.py

```

1  from random import *
2
3  # make data randint(1, r)
4
5  n = randint(1, 100000)
6

```

```
7 | print(n)
```

0.5.6 makestd.cpp

```
1 | #include <bits/stdc++.h>
2 |
3 | using namespace std;
4 | using ll = long long;
5 |
6 | int main() {
7 |     ios::sync_with_stdio(false);
8 |     cin.tie(nullptr);
9 |
10 |     string s;
11 |     while (getline(cin, s)) {
12 |         cout << "\"";
13 |         for (auto it : s) {
14 |             if (it == '"' || it == '\\')
15 |                 cout << "\\\"";
16 |             cout << it;
17 |         }
18 |         cout << "\", ";
19 |         cout << endl;
20 |     }
21 |     return 0;
22 | }
```

0.5.7 pai.py

```
1 | import os
2 |
3 | stdName = "A"
4 | bfName = "B"
5 | dirName = "pai"
6 |
7 | os.system("g++ -std=c++20 -Wall {0:}.cpp -o std".format(stdName))
8 | os.system("g++ -std=c++20 -Wall {0:}.cpp -o bf".format(bfName))
9 |
10 | os.system("mkdir {0:}".format(dirName))
11 | os.system("mv std {0:}".format(dirName))
12 | os.system("mv bf {0:}".format(dirName))
13 |
14 | tc = 0
15 | while True:
16 |     os.system("python gen.py > ./{0:}/in.in".format(dirName))
17 |     os.system("time ./{0:}/std < ./{0:}/in.in > ./{0:}/std.out".format(dirName))
18 |     os.system("./{0:}/bf < ./{0:}/in.in > ./{0:}/bf.out".format(dirName))
```

```

19     if os.system("diff ./{0:}/bf.out ./{0:}/std.out".format(dirName)):
20         print("WA")
21         exit(0)
22     else:
23         tc += 1
24         print("AC #", tc)

```

0.5.8 sg 函数.cpp

```

1  #include <algorithm>
2  #include <cstring>
3  #include <iostream>
4  using namespace std;
5  #define IO ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
6  typedef long long ll;
7  const int maxm = 1e4 + 5;
8  const int maxn = 105;
9  int k;
10 int f[maxn], sg[maxm], vis[maxm]; //记录后继状态
11 void dosg() {
12     sg[0] = 0;
13     memset(vis, -1, sizeof(vis));
14     for (int i = 1; i < maxm; ++i) {
15         for (int j = 1; f[j] <= i && j <= k; ++j)
16             vis[sg[i - f[j]]] = i;
17         int j = 0;
18         while (vis[j] == i)
19             ++j;
20         sg[i] = j;
21     }
22 }
23
24 int main() {
25     while (scanf("%d", &k) && k) {
26         for (int i = 1; i <= k; ++i)
27             scanf("%d", &f[i]);
28         sort(f + 1, f + 1 + k);
29         dosg();
30         int m;
31         scanf("%d", &m);
32         while (m--) {
33             int n;
34             scanf("%d", &n);
35             int ans = 0;
36             for (int i = 1; i <= n; ++i) {
37                 int x;
38                 scanf("%d", &x);

```



```

39         ans ^= sg[x];
40     }
41     printf("%c", ans ? 'W' : 'L');
42     // cout << (ans ? "W" : "L");
43 }
44 printf("\n");
45 }
46 }

```

0.5.9 博弈.cpp

```

1  #include <cmath>
2  #include <cstring>
3  #include <iostream>
4  #define gold (sqrt(5.0) + 1) / 2
5  using namespace std;
6  typedef long long ll;
7
8  int sg[1005];
9  const int N = 30;
10 int f[N];
11 int s[1005];
12 void DoSg(int num) {
13     int i, j;
14     memset(sg, 0, sizeof(sg));
15     for (i = 1; i <= num; ++i) {
16         memset(s, 0, sizeof(s));
17         for (j = 0; f[j] <= i && j < N; ++j) {
18             s[sg[i - f[j]]] = 1;
19         }
20         for (j = 0; ; ++j) {
21             if (!s[j]) {
22                 sg[i] = j;
23                 break;
24             }
25         }
26     }
27 }
28
29 int main() {
30     ios::sync_with_stdio(false);
31     f[0] = 1;
32     f[1] = 1;
33     for (int i = 2; i <= 30; ++i) {
34         f[i] = f[i - 1] + f[i - 2];
35     }
36     DoSg(1000);

```

```

37     int n, m, k;
38     while (cin >> n >> m >> k) {
39         if (n == 0 && m == 0 && k == 0)
40             break;
41         if (sg[n] ^ sg[m] ^ sg[k])
42             cout << "Fibo" << endl;
43         else
44             cout << "Nqcci" << endl;
45     }
46 }

```

0.5.10 威佐夫博弈.cpp

```

1  #include <algorithm>
2  #include <cmath>
3  #include <iostream>
4  #define gold (sqrt(5.0) + 1) / 2
5  using namespace std;
6  typedef long long ll;
7
8  int main() {
9      ios::sync_with_stdio(false);
10     int a, b;
11     while (cin >> a >> b) {
12         int big = max(a, b);
13         int small = min(a, b);
14         double now = double(big - small) * gold;
15         if ((int)now == small)
16             cout << 0 << endl; //后手必胜
17         else
18             cout << 1 << endl; //先手必胜
19     }
20 }

```

0.5.11 杜教 BM.cpp

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  #define rep(i, a, n) for (long long i = a; i < n; i++)
4  #define per(i, a, n) for (long long i = n - 1; i >= a; i--)
5  #define pb push_back
6  #define all(x) (x).begin(), (x).end()
7  #define SZ(x) ((long long)(x).size())
8  typedef vector<long long> VI;
9  typedef long long ll;
10 typedef pair<long long, long long> PII;
11 const ll mod = 1e9 + 7;

```

```

12 ll powmod(ll a, ll b) {
13     ll res = 1;
14     a %= mod;
15     assert(b >= 0);
16     for (; b; b >>= 1) {
17         if (b & 1)
18             res = res * a % mod;
19         a = a * a % mod;
20     }
21     return res;
22 }
23 // head
24
25 namespace linear_seq {
26     const long long N = 10010;
27     ll res[N], base[N], _c[N], _md[N];
28
29     vector<long long> Md;
30     void mul(ll *a, ll *b, long long k) {
31         rep(i, 0, k + k) _c[i] = 0;
32         rep(i, 0, k) if (a[i]) rep(j, 0, k)
33             _c[i + j] = (_c[i + j] + a[i] * b[j]) % mod;
34         for (long long i = k + k - 1; i >= k; i--)
35             if (_c[i])
36                 rep(j, 0, SZ(Md)) _c[i - k + Md[j]] = (_c[i - k + Md[j]] - _c[i] * _md[Md[j]]) % mod;
37         rep(i, 0, k) a[i] = _c[i];
38     }
39     long long solve(ll n, VI a, VI b) { // a 系数 b 初值 b[n+1]=a[0]*b[n]+...
40         // printf("%d\n", SZ(b));
41         ll ans = 0, pnt = 0;
42         long long k = SZ(a);
43         assert(SZ(a) == SZ(b));
44         rep(i, 0, k) _md[k - 1 - i] = -a[i];
45         _md[k] = 1;
46         Md.clear();
47         rep(i, 0, k) if (_md[i] != 0) Md.push_back(i);
48         rep(i, 0, k) res[i] = base[i] = 0;
49         res[0] = 1;
50         while ((1ll << pnt) <= n) pnt++;
51         for (long long p = pnt; p >= 0; p--) {
52             mul(res, res, k);
53             if ((n >> p) & 1) {
54                 for (long long i = k - 1; i >= 0; i--)
55                     res[i + 1] = res[i];
56                 res[0] = 0;
57                 rep(j, 0, SZ(Md)) res[Md[j]] = (res[Md[j]] - res[k] * _md[Md[j]]) % mod;
58             }
59         }

```

```

60     rep(i, 0, k) ans = (ans + res[i] * b[i]) % mod;
61     if (ans < 0) ans += mod;
62     return ans;
63 }
64 VI BM(VI s) {
65     VI C(1, 1), B(1, 1);
66     long long L = 0, m = 1, b = 1;
67     rep(n, 0, SZ(s)) {
68         ll d = 0;
69         rep(i, 0, L + 1) d = (d + (ll)C[i] * s[n - i]) % mod;
70         if (d == 0)
71             ++m;
72         else if (2 * L <= n) {
73             VI T = C;
74             ll c = mod - d * powmod(b, mod - 2) % mod;
75             while (SZ(C) < SZ(B) + m)
76                 C.pb(0);
77             rep(i, 0, SZ(B)) C[i + m] = (C[i + m] + c * B[i]) % mod;
78             L = n + 1 - L;
79             B = T;
80             b = d;
81             m = 1;
82         } else {
83             ll c = mod - d * powmod(b, mod - 2) % mod;
84             while (SZ(C) < SZ(B) + m) C.pb(0);
85             rep(i, 0, SZ(B)) C[i + m] = (C[i + m] + c * B[i]) % mod;
86             ++m;
87         }
88     }
89     return C;
90 }
91 long long gao(VI a, ll n) {
92     VI c = BM(a);
93     c.erase(c.begin());
94     rep(i, 0, SZ(c)) c[i] = (mod - c[i]) % mod;
95     return solve(n, c, VI(a.begin(), a.begin() + SZ(c)));
96 }
97 }; // namespace linear_seq
98
99 int main() {
100     int n;
101     cin >> n;
102     cout << linear_seq::gao(VI{0, 1, 5, 18, 58, 177, 522, 1503, 4252, 11869}, n - 1) << "\n";
103 }

```

0.5.12 欧拉函数.cpp

```

1 //我们首先应该要知道欧拉函数的通项公式
2 // (n)=n*(1-1/p1)*(1-1/p2)*(1-1/p3)*(1-1/p4).....(1-1/pn), 其中pi为n的质因数
3 //求n以内与n互质的数的个数
4 long long euler(long long n)
5 {
6     long long ans = n;
7     for (int i = 2; i * i <= n; i++)
8     {
9         if (n % i == 0)
10        {
11            ans -= ans / i;    //等价于通项, 把n乘进去
12            while (n % i == 0) //确保下一个i是n的素因数
13                n /= i;
14        }
15    }
16    if (n > 1)
17        ans -= ans / n; //最后可能还剩下一个素因数没有除
18    return ans;
19 }

```

0.6 String

0.6.1 AhoCorasick.cpp

```

1 #include <bits/stdc++.h>
2
3 using namespace std;
4 using ll = long long;
5
6 /** Modified from:
7  * https://github.com/kth-competitive-programming/kactl/blob/master/content/strings/AhoCorasick.h
8  * Try to handle duplicated patterns beforehand, otherwise change 'end' to
9  * vector; empty patterns are not allowed. Time: construction takes  $O(26N)$ ,
10 * where  $N = \sum$  of length of patterns. find(x) is  $O(N)$ , where  $N = \text{length of } x$ .
11 * findAll is  $O(N+M)$  where  $M$  is number of occurrence of all pattern (up to  $N \cdot \sqrt{N}$ ) */
12 struct AhoCorasick {
13     enum { alpha = 26, first = 'a' }; // change this!
14     struct Node {
15         // back: failure link, points to longest suffix that is in the trie.
16         // end: longest pattern that ends here, is -1 if no pattern ends here.
17         // nmatches: number of (patterns that is a suffix of current node)/(duplicated patterns),
18             depends on needs.
19         // output: output link, points to the longest pattern that is a suffix of current node
20         int back, end = -1, nmatches = 0, output = -1;
21         array<int, alpha> ch;
22         Node(int v = -1) { fill(ch.begin(), ch.end(), v); }
23     };
24 };

```

```

23     vector<Node> N;
24     int n;
25     AhoCorasick() : N(1), n(0) {}
26     void insert(string &s) {
27         assert(!s.empty());
28         int p = 0;
29         for (char c : s) {
30             if (N[p].ch[c - first] == -1) {
31                 N[p].ch[c - first] = N.size();
32                 N.emplace_back();
33             }
34             p = N[p].ch[c - first];
35         }
36         N[p].end = n++;
37         N[p].nmatches++;
38     }
39     void build() {
40         N[0].back = (int)N.size();
41         N.emplace_back(0);
42         queue<int> q;
43         q.push(0);
44         while (!q.empty()) {
45             int p = q.front();
46             q.pop();
47             for (int i = 0; i < alpha; i++) {
48                 int pnx = N[N[p].back].ch[i];
49                 auto &nxt = N[N[p].ch[i]];
50                 if (N[p].ch[i] == -1) N[p].ch[i] = pnx;
51                 else {
52                     nxt.back = pnx;
53                     // if prev is an end node, then set output to prev node,
54                     // otherwise set to output link of prev node
55                     nxt.output = N[pnx].end == -1 ? N[pnx].output : pnx;
56                     // if we don't want to distinguish info of patterns that is
57                     // a suffix of current node, we can add info to the ch
58                     // node like this: nxt.nmatches+=N[pnx].nmatches;
59                     q.push(N[p].ch[i]);
60                 }
61             }
62         }
63     }
64     // for each position, finds the longest pattern that ends here
65     vector<int> find(const string &text) {
66         int len = text.length();
67         vector<int> res(len);
68         int p = 0;
69         for (int i = 0; i < len; i++) {
70             p = N[p].ch[text[i] - first];

```

```

71         res[i] = N[p].end;
72     }
73     return res;
74 }
75 // for each position, finds the all that ends here
76 vector<vector<int>> find_all(const string &text) {
77     int len = text.length();
78     vector<vector<int>> res(len);
79     int p = 0;
80     for (int i = 0; i < len; i++) {
81         p = N[p].ch[text[i] - first];
82         res[i].push_back(N[p].end);
83         for (int ind = N[p].output; ind != -1; ind = N[ind].output) {
84             assert(N[ind].end != -1);
85             res[i].push_back(N[ind].end);
86         }
87     }
88     return res;
89 }
90 int find_cnt(const string &text) {
91     int len = text.length();
92     vector<int> num(n + 1, 0);
93     int p = 0, ans = 0;
94     for (int i = 0; i < len; i++) {
95         p = N[p].ch[text[i] - first];
96         if (N[p].end != -1) {
97             if (!num[N[p].end]) {
98                 num[N[p].end]++;
99                 ans += N[p].nmatches;
100             }
101         }
102         for (int ind = N[p].output; ind != -1; ind = N[ind].output) {
103             if (!num[N[ind].end]) {
104                 num[N[ind].end]++;
105                 ans += N[ind].nmatches;
106             }
107         }
108     }
109     return ans;
110 }
111 pair<int, vector<int>> find_maxcnt(const string &text) {
112     int len = text.length();
113     vector<int> num(n + 1, 0);
114     int p = 0, ans = 0;
115     for (int i = 0; i < len; i++) {
116         p = N[p].ch[text[i] - first];
117         if (N[p].end != -1) {
118             if (!num[N[p].end]) {

```

```
119         num[N[p].end]++;
120         ans = max(ans, N[p].nmatches);
121     }
122 }
123 for (int ind = N[p].output; ind != -1; ind = N[ind].output) {
124     if (!num[N[ind].end]) {
125         num[N[ind].end]++;
126         ans += N[ind].nmatches;
127     }
128 }
129 }
130 vector<int> idx;
131 for (int i = 0; i < n; i++) {
132     if (num[i] == ans) {
133         idx.push_back(i);
134     }
135 }
136 return pair(ans, idx);
137 }
138 };
139
140 int main() {
141     ios::sync_with_stdio(false);
142     cin.tie(nullptr);
143
144     int n;
145     cin >> n;
146
147     AhoCorasick ac;
148     for (int i = 0; i < n; ++i) {
149         string s;
150         cin >> s;
151         ac.insert(s);
152     }
153
154     ac.build();
155
156     string t;
157     cin >> t;
158
159     cout << ac.find_cnt(t) << "\n";
160
161     return 0;
162 }
163
164 // test problem: https://www.luogu.com.cn/problem/P3808
```


0.6.2 exkmp.cpp

```

1  #include <cstdio>
2  #include <cstring>
3  #include <iostream>
4  #include <string>
5
6  using namespace std;
7  const int K = 100005;
8  int nt[K], extand[K];
9  char S[K], T[K];
10 void Getnext(char *T, int *next) {
11     int len = strlen(T), a = 0;
12     next[0] = len;
13     while (a < len - 1 && T[a] == T[a + 1])
14         a++;
15     next[1] = a;
16     a = 1;
17     for (int k = 2; k < len; k++) {
18         int p = a + next[a] - 1, L = next[k - a];
19         if ((k - 1) + L >= p) {
20             int j = (p - k + 1) > 0 ? (p - k + 1) : 0;
21             while (k + j < len && T[k + j] == T[j])
22                 j++;
23             next[k] = j;
24             a = k;
25         } else
26             next[k] = L;
27     }
28 }
29 void GetExtand(char *S, char *T, int *next) {
30     Getnext(T, next);
31     int slen = strlen(S), tlen = strlen(T), a = 0;
32     int MinLen = slen < tlen ? slen : tlen;
33     while (a < MinLen && S[a] == T[a])
34         a++;
35     extand[0] = a;
36     a = 0;
37     for (int k = 1; k < slen; k++) {
38         int p = a + extand[a] - 1, L = next[k - a];
39         if ((k - 1) + L >= p) {
40             int j = (p - k + 1) > 0 ? (p - k + 1) : 0;
41             while (k + j < slen && j < tlen && S[k + j] == T[j])
42                 j++;
43             extand[k] = j;
44             a = k;
45         } else
46             extand[k] = L;

```

```
47     }
48 }
49 int main() {
50     while (scanf("%s%s", S, T) == 2) {
51         GetExtand(S, T, nt);
52         for (int i = 0; i < strlen(T); i++)
53             printf("%d ", nt[i]);
54         puts("");
55         for (int i = 0; i < strlen(S); i++)
56             printf("%d ", extand[i]);
57         puts("");
58     }
59 }
```

0.6.3 kmp.cpp

```
1  #include <bits/stdc++.h>
2
3  using namespace std;
4  using ll = long long;
5
6  vector<int> prefixFunction(string s) {
7      int n = (int)s.size();
8      vector<int> p(n);
9      for (int i = 1; i < n; ++i) {
10         int j = p[i - 1];
11         while (j > 0 && s[i] != s[j]) j = p[j - 1];
12         if (s[i] == s[j]) ++j;
13         p[i] = j;
14     }
15     return p;
16 }
17
18 // KMP based on prefixFunction. return all match postion in t
19 // also can create string st = s + '#' + t, and call prefixFunction(st),
20 // if p[i] == s.length(), it's a successful match: s in t
21 vector<int> kmp(string s, string t) {
22     vector<int> ans;
23     int n = (int)s.size(), m = (int)t.size();
24     if (n > m) return ans;
25     auto p = prefixFunction(s);
26     for (int i = 0, j = 0; i < m; ++i) {
27         while (j > 0 && s[j] != t[i]) j = p[j - 1];
28         if (s[j] == t[i] && ++j == n) ans.emplace_back(i - n + 1);
29     }
30     return ans;
31 }
```

```

32
33 int main() {
34     ios::sync_with_stdio(false);
35     cin.tie(nullptr);
36
37     string t, s;
38     cin >> t >> s;
39
40     string st = s + '#' + t;
41     auto ans = prefixFunction(st);
42     for (int i = s.length() + 1; i < st.length(); ++i) {
43         if (ans[i] == s.length()) {
44             cout << i - 2 * s.length() + 1 << "\n";
45         }
46     }
47
48     for (int i = 0; i < s.length(); ++i) {
49         cout << ans[i] << " \n"[i == s.length() - 1];
50     }
51
52     return 0;
53 }
54
55 // test problem: https://www.luogu.com.cn/problem/P3375

```

0.6.4 manacher.cpp

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4  using ll = long long;
5
6  template <typename T>
7  vector<int> manacher(int n, const T &s) {
8      if (n == 0) {
9          return vector<int>();
10     }
11     vector<int> res(2 * n - 1, 0);
12     int l = -1, r = -1;
13     for (int z = 0; z < 2 * n - 1; z++) {
14         int i = (z + 1) >> 1;
15         int j = z >> 1;
16         int p = (i >= r ? 0 : min(r - i, res[2 * (l + r) - z]));
17         while (j + p + 1 < n && i - p - 1 >= 0) {
18             if (!s[j + p + 1] == s[i - p - 1]) {
19                 break;
20             }

```

```

21         p++;
22     }
23     if (j + p > r) {
24         l = i - p;
25         r = j + p;
26     }
27     res[z] = p;
28 }
29 return res;
30 // res[2 * i] = odd radius in position i
31 // res[2 * i + 1] = even radius between positions i and i + 1
32 // s = "abaa" -> res = {0, 0, 1, 0, 0, 1, 0}
33 // s = "aaa" -> res = {0, 1, 1, 1, 0}
34 // in other words, for every z from 0 to 2 * n - 2:
35 // calculate i = (z + 1) >> 1 and j = z >> 1
36 // now there is a palindrome from i - res[z] to j + res[z]
37 // (watch out for i > j and res[z] = 0)
38 }
39 template <typename T>
40 vector<int> manacher(const T &s) {
41     return manacher((int)s.size(), s);
42 }
43
44 int main() {
45     ios::sync_with_stdio(false);
46     cin.tie(nullptr);
47
48     string s;
49     cin >> s;
50     int n = s.length();
51
52     auto ans = manacher(s);
53
54     int len = 0, id = -1;
55     for (int z = 0; z < 2 * n - 1; ++z) {
56         if (z % 2 == 0 && 1 + 2 * ans[z] > len) { // odd length of palindrome
57             len = 1 + 2 * ans[z];
58             id = z / 2 - ans[z];
59         } else if (z % 2 == 1 && 2 * ans[z] > len) { // even length of palindrome
60             len = 2 * ans[z];
61             id = z / 2 - ans[z] + 1;
62         }
63     }
64
65     cout << s.substr(id, len) << "\n";
66
67     return 0;
68 }

```

0.6.5 后缀数组.cpp

```

1  #include <cstdio>
2  #include <iostream>
3  #include <cstdlib>
4  #include <cstring>
5  using namespace std;
6  typedef long long ll;
7  const int N = 2e5 + 10;
8  int n, mx, mn;
9  int a[N];
10 char s[N];
11 int SA[N], rnk[N], height[N], sum[N], tp[N];
12 //rnk[i] 表示 i 在 s 中的排名, SA[i] 表示排名为 i 的字符在 s 中的位置, Height[i] 表示 i 和 i-1 的 LCP
13 //sum[i] 表示 s[0..i-1] 中每个字符出现的次数, tp[i] 表示 s[i] 在 s[0..i-1] 中的排名, SA 表示 s 的逆序
14
15 bool cmp(int *f, int x, int y, int w)
16 {
17     return f[x] == f[y] && f[x + w] == f[y + w];
18 }
19
20 void get_SA(char *s, int n, int m)
21 {
22     // 初始化
23     for (int i = 0; i < m; i++)
24         sum[i] = 0; // 0
25     for (int i = 0; i < n; i++)
26         sum[rnk[i] = s[i]]++; // 统计每个字符出现的次数
27     for (int i = 1; i < m; i++)
28         sum[i] += sum[i - 1]; // sum[i] 表示排名小于等于 i 的字符的个数
29     for (int i = n - 1; i >= 0; i--)
30         SA[--sum[rnk[i]]] = i; // 将每个字符的排名存入 SA
31     // SA[i] 表示排名为 i 的字符在 s 中的位置, SA[--sum[rnk[i]]] = i 表示将 s[i] 的排名存入 SA
32     for (int len = 1; len <= n; len <<= 1)
33     {
34         int p = 0;
35         // 计算 Height
36         for (int i = n - len; i < n; i++)
37             tp[p++] = i; // 将 s[i] 的排名存入 tp
38         for (int i = 0; i < n; i++)
39         {
40             if (SA[i] >= len)
41                 tp[p++] = SA[i] - len;
42         }
43         // tp[i] 表示 s[i] 的排名, tp[p++] = SA[i] - len 表示将 s[SA[i] - len] 的排名存入 tp
44         // 计算 Height
45         for (int i = 0; i < m; i++)
46             sum[i] = 0;

```

```

47     for (int i = 0; i < n; i++)
48         sum[rnk[tp[i]]]++;
49     for (int i = 1; i < m; i++)
50         sum[i] += sum[i - 1];
51     for (int i = n - 1; i >= 0; i--)
52         SA[--sum[rnk[tp[i]]]] = tp[i];
53     //, % SA° rnk      % rnk
54     swap(rnk, tp); //»»»° tp    rnk
55     p = 1;
56     rnk[SA[0]] = 0;
57     for (int i = 1; i < n; i++)
58     {
59         rnk[SA[i]] = cmp(tp, SA[i - 1], SA[i], len) ? p - 1 : p++; //    ° rnk[i]° rnk[i-1]
60     }
61     if (p >= n)
62         break;
63     m = p; // ´
64 }
65 // height
66 int k = 0;
67 n--;
68 for (int i = 0; i <= n; i++)
69     rnk[SA[i]] = i;
70 for (int i = 0; i < n; i++)
71 {
72     if (k)
73         k--;
74     int j = SA[rnk[i] - 1];
75     while (s[i + k] == s[j + k])
76         k++;
77     height[rnk[i]] = k;
78 }
79 }
80
81 void check()
82 {
83     // getchar();//°    û    ô
84     scanf("%s", s);
85     int n = strlen(s);
86     get_SA(s, n + 1, 'z' + 1);
87     ll res = 0;
88     for (int i = 1; i <= n; ++i)
89         res += n - SA[i] - height[i];
90     printf("%lld\n", res);
91 }
92 //    °&¬    @p    -
93
94 int main()

```

```

95 {
96     int t;
97     scanf("%d", &t);
98     while (t--)
99         check();
100 }

```

0.7 dp

0.7.1 数位 dp.cpp

```

1  #include <algorithm>
2  #include <cstring>
3  #include <iostream>
4  #include <map>
5  using namespace std;
6  typedef long long ll;
7  int a, b, num[20], dp[20][2];
8  int dfs(int len, bool if6, bool lim) {
9      if (len == 0) return 1;
10     if (!lim && dp[len][if6]) return dp[len][if6];
11     int cnt = 0, maxx = (lim ? num[len] : 9);
12     for (int i = 0; i <= maxx; i++) {
13         if (i == 4 || (if6 && i == 2)) continue;
14         cnt += dfs(len - 1, i == 6, lim && i == maxx);
15     }
16     return lim ? cnt : dp[len][if6] = cnt;
17 }
18
19 int solve(int x) {
20     memset(num, 0, sizeof(num));
21     int k = 0;
22     while (x) {
23         num[++k] = x % 10;
24         x /= 10;
25     }
26     return dfs(k, false, true);
27 }
28
29 int main() {
30     scanf("%d%d", &a, &b);
31     printf("%d\n", solve(b) - solve(a - 1));
32 }

```

0.7.2 最长上升子序列.cpp

```
1 #include <bits/stdc++.h>
2
3 using namespace std;
4 using ll = long long;
5
6 int main() {
7     ios::sync_with_stdio(false);
8     cin.tie(nullptr);
9
10    int n;
11    cin >> n;
12    vector<int> a(n);
13    for (int i = 0; i < n; ++i) {
14        cin >> a[i];
15    }
16
17    // 最长（不严格）上升子序列
18    vector<int> dp(n, 1e9), pre(n);
19    for (int i = 0; i < n; ++i) {
20        *upper_bound(dp.begin(), dp.end(), a[i]) = a[i];
21        pre[i] = lower_bound(dp.begin(), dp.end(), 1e9) - dp.begin();
22    }
23
24    int ans = *max_element(pre.begin(), pre.end());
25
26    cout << ans << "\n";
27
28    return 0;
29 }
```