

1 Template

```

1 // :80 <enter>
2 #include <bits/stdc++.h>
3 using namespace std;
4 typedef long long ll;
5 typedef pair<int, int> pii;
6 typedef pair<ll, ll> pll;
7 typedef pair<int, ll> pil;
8 typedef pair<ll, int> pli;
9 typedef pair<double, double> pdd;
10 #define SQ(i) ((i)*(i))
11 #define MEM(a, b) memset(a, (b), sizeof(a))
12 #define SZ(i) int(i.size())
13 #define
14     FOR(i, j, k, in) for (int i=j ; i<(k) ; i+=in)
15 #define RFOR(i,
16     j, k, in) for (int i=j ; i>=(k) ; i-=in)
17 #define REP(i, j) FOR(i, 0, j, 1)
18 #define REP1(i, j) FOR(i, 1, j+1, 1)
19 #define RREP(i, j) RFOR(i, j, 0, 1)
20 #define ALL(_a) _a.begin(),_a.end()
21 #define mp make_pair
22 #define pb push_back
23 #define eb emplace_back
24
25 #define endl '\n'
26 #define IOS()
27     ios_base::sync_with_stdio(0);cin.tie(0)
28
29 const ll MOD=1000000007;
30 const ll INF=0x3f3f3f3f3f3f3f3f;
31 const int iNF=0x3f3f3f3f;
32 const ll MAXN=100005;
33
34 void solve(){
35 }
36
37 //***** Good Luck :) *****/
38 int main(){
39     TIME(main);
40     IOS();
41     int t=1;
42     cin>>t;
43     while(t--){
44         solve();
45     }
46
47     return 0;
48 }
```

2 BIT

```

1 class BIT{
2 private:
3     vector<ll> bit;
4 public:
5     BIT():bit(vector<ll>(MAXN, 0)) {}
6     ll lowbit(ll x){
7         return x&(-x);
8     }
```

```

9
10    ll query(ll idx){
11        if(idx == 0) return 0;
12        ll ans = 0;
13        for(; idx>0; idx-=lowbit(idx)){
14            ans += bit[idx];
15        }
16
17        return ans;
18    }
19
20    void modify(ll idx, int val){
21        for(; idx<=n; idx+=lowbit(idx)){
22            bit[idx] += val;
23        }
24    }
25 }
```

3 Dijkstra

```

1 // djisktra
2     // with priority queue and memory optimize
3
4 #include <bits/stdc++.h>
5 using namespace std;
6 typedef long long ll;
7 typedef pair<ll, ll> pll;
8 #define pb push_back
9 #define mp make_pair
10
11 const ll INF=0x3f3f3f3f3f3f3f3f;
12
13 int n, m;
14 vector<vector<pll>> w;
15 vector<ll> d;
16 vector<int> parent;
17
18 void dijkstra(int src){
19     priority_queue<pll,
20     vector<pll>, greater<pll>>
21     pq; // first: weight, second: vertex
22     for(int i=1;i<n;i++){
23         pq.push(mp(INF, i));
24     }
25     d[src]=0;
26     parent[src]=src;
27     while(!pq.empty()){
28         pll edge=pq.top();
29         pq.pop();
30         ll u=edge.second;
31         for(auto i:w[u]){
32             ll w=i.second;
33             ll v=i.first;
34             ll alt=d[u]+i.second;
35             if(alt<d[v]){
36                 d[v]=alt;
37                 parent[v]=u;
38                 pq.push(mp(alt, v));
39             }
40         }
41     }
42 }
```

```

42 int main () {
43
44     cin >> n >> m;
45     w.resize(n+1);
46     d.resize(n+1, INF);
47     parent.resize(n+1, -1);
48     lla, b, tmp;
49     for(int i=0;i<m;i++){
50         cin >> a >> b >> tmp;
51         w[a].pb(mp(b, tmp));
52         w[b].pb(mp(a, tmp));
53     }
54     dijkstra(1); // source = 1
55     vector<int> ans;
56     int cur;
57     if(d[n] != INF){ // if d[n] = INF, there
58         → is no shortest path from vertex 1 to vertex n
59         cur=n;
60         while(cur != 1){
61             ans.push_back(cur);
62             cur=parent[cur];
63         }
64         ans.push_back(1);
65         int sz=ans.size();
66         for(int i=sz-1;i>=0;i--){
67             cout << ans[i] << " ";
68         }
69         cout << endl;
70     } else {
71         cout << -1 << endl;
72     }
73
74     return 0;
75 }
```

4 Segment Tree

```

1 // 利用線段樹解決區間求和 & 單點修改
2 #include <bits/stdc++.h>
3
4 using namespace std;
5
6 struct Node{
7     int val;
8     Node *lc, *rc;
9     void pull(){
10         val=lc->val+rc->val;
11     }
12 };
13
14 const int n=5;
15
16 int v[n+1]={0, 1, 2, 3, 4, 5};
17
18 Node* build(int L, int R){
19     Node *node=new Node();
20     if(L==R){
21         node->val=v[L];
22     }
23     int mid=(L+R)>>1;
24     node->lc=build(L, mid);
25     node->rc=build(mid+1, R);
26     node->pull();
27 }
```

```

27
28     return node;
29 }
30
31 void modify(Node*
32     → node, int L, int R, int i, int d){
33     if(L==R){
34         assert(L==i);
35         node->val+=d;
36         return;
37     }
38     int mid=(L+R)>>1;
39     if(i<=mid){
40         modify(node->lc, L, mid, i, d);
41     } else {
42         modify(node->rc, mid+1, R, i, d);
43     }
44     node->pull();
45 }
46
47 int query(Node*
48     → node, int L, int R, int ql, int qr){
49     if(ql>R || qr<L) return 0;
50     if(ql<=L && R<=qr) return node->val;
51     int mid=(L+R)>>1;
52     return query(node->lc, L, mid, ql, qr)+query(node->rc, mid+1, R, ql, qr);
53 }
```

5 Segment Tree w/ lazy tags

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 typedef pair<int, int> pii;
5 typedef pair<ll, ll> pll;
6 typedef pair<int, ll> pil;
7 typedef pair<ll, int> pli;
8 typedef pair<double, double> pdd;
9
10 struct Node{
11     int val, tag;
12     Node *lc, *rc;
13     Node(){
14         tag=val=0;
15         lc=rc=nullptr;
16     }
17     void pull(){
18         val=lc->val+rc->val;
19     }
20 };
21
22 const int n=5;
23 int v[n+1]={0, 1, 16, 2, 8, 4}; // input, 1-base
24
25 Node* build(int L, int R){
26     Node *node=new Node();
27     if(L==R){
28         node->val=v[L];
29         return node;
30     }
31     int mid=(L+R)>>1;
32     node->lc=build(L, mid);
33     node->rc=build(mid+1, R);
34 }
```

```

34     node->pull();
35     return node;
36 }
37
38 void push(Node* node, int L, int R){
39     if(!node->tag) return;
40     if(L != R){
41         int mid = (L + R) >> 1;
42         node->lc->tag += node->tag;
43         node->rc->tag += node->tag;
44         node->lc->val += node->tag * (mid - L + 1);
45         node->rc->val += node->tag * (R - mid);
46     }
47     node->tag = 0;
48 }
49
50 void modify(Node*
51     ↪ node, int L, int R, int ql, int qr, int d){
52     debug(L, R);
53     if(ql > R || qr < L) return;
54     if(ql <= L && R <= qr){
55         node->tag += d;
56         node->val += d * (R - L + 1);
57         return;
58     }
59     push(node, L, R);
60     int mid = (L + R) >> 1;
61     modify(node->lc, L, mid, ql, qr, d);
62     modify(node->rc, mid + 1, R, ql, qr, d);
63     node->pull();
64 }
65
66 int query(Node*
67     ↪ node, int L, int R, int ql, int qr){
68     if(ql > R || qr < L) return 0;
69     if(ql <= L && R <= qr) return node->val;
70     push(node, L, R);
71     int mid = (L + R) >> 1;
72     return query(node->lc, L, mid, ql,
73     ↪ qr) + query(node->rc, mid + 1, R, ql, qr);
74 }
75 //***** Good Luck :) *****/
76 int main() {
77     TIME(main);
78     IOS();
79     Node* root = build(1, n);
80     //cout << query(root, 1, n, 1, 5) << endl;
81     modify(root, 1, n, 1, 4, 3);
82
83     //cout << query(root, 1, n, 3, 3) << endl;
84
85     return 0;
86 }

```

```
9 void init(){
10    for(int i=0;i<n;i++){
11        p[i]=i;
12    }
13 }
14
15 int find(int x){
16    if(p[x] == x) return x;
17
18    return p[x] = find(p[x]);
19 }
20
21 void merge(int x, int y){
22    int fx = find(x);
23    int fy = find(y);
24    if(fx == fy) return;
25    if(sz[fx] < sz[fy]) swap(fx, fy);
26
27    p[fy] = fx;
28    sz[fx] += sz[fy];
29 }
30
31 int main(){
32
33 }
```

6 Disjoint Set

Fast Power MOD

```

37
38 // Reduce the number B to a small number
39 // using Fermat Little
40 for (int i=0; i<b.length(); i++)
41     remainderB
42     ↪ = (remainderB * 10 + b[i] - '0') % (MOD-1);
43
44 cout << power(a, remainderB, MOD) << endl;
45 return 0;
}

```

8 Kosaraju

```

1 // same scc will number
2
3 #include<bits/stdc++.h>
4
5 using namespace std;
6
7 const int N=100; // # of vertex
8
9 vector<vector<int>> g; // graph
10 vector<vector<int>> r; // reversed graph
11 vector<int> order;
12 vector<int> scc(N);
13 vector<bool> vis(N);
14
15 void RevDfs(int cur){
16     vis[cur]=true;
17     for(int u: r[cur]){
18         if(!vis[u]){
19             RevDfs(u);
20         }
21     }
22     order.push_back(cur); // topological order
23 }
24
25 void Dfs(int cur, int s){
26     scc[cur]=s;
27     for(int u: g[cur]){
28         if(scc[u]==-1) Dfs(u, s);
29     }
30 }
31
32 void Kosaraju(int n){
33     fill(vis.begin(), vis.end(), false);
34     fill(scc.begin(), scc.end(), -1);
35
36     for(int i=0;i<n;i++){
37         if(!vis[i]) RevDfs(i);
38     }
39
40     int n_scc=0;
41     for(int i=n-1;i>=0;i--){
42         int cur=order[i];
43         if(scc[cur]==-1){
44             Dfs(cur, n_scc);
45             n_scc++;
46         }
47     }
48 }

```

9 Treap

```

1 #include<cstdio>
2 #include<algorithm>
3 #include<stack>
4 #include<ctime>
5 #include<cstdlib>
6 #include<queue>
7 #define MAXN 800000
8 #define INF 2147483647
9 using namespace std;
10 struct treap
11 {
12     int v;
13     int sz;
14     int p;
15     int mn;
16     int rev;
17     int add;
18     treap* l, *r;
19     treap() {}
20     treap(int k) : v(k), sz(1), p(rand()),
21     ↪ mn(k), rev(0), add(0), l(NULL), r(NULL) {}
21 };
22
23 treap mempool[MAXN];
24 treap* ptr;
25 treap* gc; //
26     ↪ use treap as linked list to garbage collect
27
28 inline void init(){
29     ptr=mempool;
30     gc=NULL;
31 }
32
33 inline void Del(treap* t){
34     t->l=gc;
35     gc=t;
36 }
37
38 inline treap* New(int v){
39     if(gc==NULL){
40         *ptr=treap(v);
41         return ptr++;
42     } else {
43         treap* t=gc;
44         gc=gc->l;
45         *t=treap(v);
46         return t;
47     }
48 }
49
50 inline int size(treap* t){
51     return t!=NULL ? t->sz : 0;
52 }
53
54 inline int small(treap* t){
55     return t!=NULL ? t->mn+t->add : INF;
56 }
57
58 inline void pull(treap* t){
59     if(t==NULL)
60         return;
61     t->sz=1+size(t->l)+size(t->r);
62 }

```

```

61     t->mn = min(t->v,
62     ↪   min(small(t->l), small(t->r)));
63 }
64 inline void reverse(treap* t) {
65     if (t != NULL)
66         t->rev ^= 1;
67 }
68 inline void addn(treap* t, int v) {
69     if (t != NULL)
70         t->add += v;
71 }
72
73 inline treap* push(treap* t) {
74     if (t != NULL) {
75         if (t->rev) {
76             swap(t->l, t->r);
77             reverse(t->l);
78             reverse(t->r);
79             t->rev = 0;
80         }
81
82         if (t->add) {
83             t->v += t->add;
84             t->mn += t->add;
85             addn(t->l, t->add);
86             addn(t->r, t->add);
87             t->add = 0;
88         }
89     }
90 }
91 return t;
92 }
93
94 void split(treap*
95     ↪   t, int k, treap*& a, treap*& b) {
96     // split first k nodes from t to a, others to b
97     push(t);
98
99     if (t == NULL) {
100        a = b = NULL;
101    } else if (size(t->l) + 1 <= k) {
102        a = t;
103        split(t->r, k - size(t->l) - 1, a->r, b);
104        pull(a);
105    } else {
106        b = t;
107        split(t->l, k, a, b->l);
108        pull(b);
109    }
110
111 treap* merge(treap* a, treap* b) {
112     if (a == NULL)
113         return push(b);
114     else if (b == NULL)
115         return push(a);
116     if (a->p > b->p) {
117         push(a);
118         a->r = merge(a->r, b);
119         pull(a);
120         return a;
121     } else {
122         push(b);
123         b->l = merge(a, b->l);
124         pull(b);
125     }
126 }
127
128 inline void slice(treap* t, int
129     ↪   x, int y, treap*& l, treap*& m, treap*& r) {
130     split(t, x - 1, l, r);
131     split(r, y - x + 1, m, r);
132 }
133 treap* build(int n) {
134     treap* r = NULL;
135     int v;
136     stack<treap*> rc;
137     treap* nt;
138     while (n--) {
139         scanf("%d", &v);
140         nt = New(v);
141         r = NULL;
142         while (!rc.empty() && rc.top()->p < nt->p) {
143             pull(r = rc.top());
144             rc.pop();
145         }
146         nt->l = r;
147         if (!rc.empty())
148             rc.top()->r = nt;
149         rc.push(nt);
150     }
151     while (!rc.empty()) {
152         pull(r = rc.top());
153         rc.pop();
154     }
155     return r;
156 }
157
158 int main()
159 {
160     srand(42);
161     int n, q;
162     char cmd[10];
163     int x, y, v;
164     treap* l, *m, *r;
165     treap* ml, *mr;
166     treap* root;
167
168     while (scanf("%d", &n) == 1) {
169         init();
170         root = build(n);
171         scanf("%d", &q);
172         while (q--) {
173             scanf("%s", cmd);
174             switch (cmd[0]) {
175                 case 'A':
176                     scanf("%d%d%d", &x, &y, &v);
177                     slice(root, x, y, l, m, r);
178                     addn(m, v);
179                     root = merge(merge(l, m), r);
180                     break;
181                 case 'I':
182                     scanf("%d%d", &x, &v);
183                     split(root, x, l, r);
184                     root = merge(merge(l, New(v)), r);
185                     break;
186                 case 'D':
187             }
188         }
189     }
190 }
```

```

187     scanf("%d", &x);
188     slice(root, x, x, l, m, r);
189     Del(m);
190     root=merge(l, r);
191     break;
192 case 'M':
193     scanf("%d%d", &x, &y);
194     slice(root, x, y, l, m, r);
195     printf("%d\n", m->mn);
196     root=merge(merge(l, m), r);
197     break;
198 case 'R':
199     scanf("%d%d", &x, &y);
200     switch(cmd[3]){
201     case 'E':
202         slice(root, x, y, l, m, r);
203         reverse(m);
204         root=merge(merge(l, m), r);
205         break;
206     case 'O':
207         scanf("%d", &v);
208         int len=(y-x+1);
209         v=(v%len
210             + len)%len; // v could be negative?
211         if(v){
212             slice(root, x, y, l, m, r);
213             split(m, len-v, ml, mr);
214             root=
215                 merge(merge(l, merge(ml, mr)), r);
216         }
217         break;
218     }
219 }
220 return 0;
221 }
```

10 FFT

```

1 #include<bits/stdc++.h>
2
3 using namespace std;
4
5 typedef complex<double> cd;
6 const double PI=acos(-1);
7
8 void fft(vector<cd>&a, bool invert){
9     int n=a.size();
10    if(n==1){
11        return;
12    }
13    vector<cd>a0(n/2), a1(n/2);
14    for(int i=0;2*i<n;i++){
15        a0[i]=a[2*i];
16        a1[i]=a[2*i+1];
17    }
18    fft(a0, invert);
19    fft(a1, invert);
20
21    double ang=2 * PI / n * (invert?-1:1);
22    cd w(1), wn(cos(ang), sin(ang));
23    for(int i=0;2*i<n;i++){

```

```

24        a[i]=a0[i]+w*a1[i];
25        a[i+n/2]=a0[i]-w*a1[i];
26        if(invert){
27            a[i]/=2;
28            a[i+n/2]/=2;
29        }
30        w*=wn;
31    }
32 }
33
34 vector<int>
35     multiply(vector<int>&a, vector<int>&b){
36     vector<cd>fa(a.begin(),
37             a.end()), fb(b.begin(), b.end());
38     int n=1;
39     while(n<(int)a.size()+(int)b.size()){
40         n<=1;
41     }
42     fa.resize(n);
43     fb.resize(n);
44
45     fft(fa, false);
46     fft(fb, false);
47
48     for(int i=0;i<n;i++){
49         fa[i]*=fb[i];
50     }
51
52     fft(fa, true);
53
54     vector<int> result(n);
55     for(int i=0;i<n;i++){
56         result[i]=round(fa[i].real());
57     }
58
59     return result;
60 }
61
62 int main(){
63     vector<int>a{1, 2, 1};
64     vector<int>b{2, 4, 6};
65
66     auto res=multiply(a, b);
67     for(auto i: res){
68         cout << i << " ";
69     }
70     cout << endl;
71 }
```

11 pbds ordered set

```

1 // C++ program to demonstrate the
2 // ordered set in GNU C++
3 #include<iostream>
4 using namespace std;
5
6 // Header files, namespaces,
7 // macros as defined above
8 #include<ext/pb_ds/assoc_container.hpp>
9 #include<ext/pb_ds/tree_policy.hpp>
10 using namespace __gnu_pbds;
11

```

```

12 #define
13   ↪ ordered_set
14   ↪ tree<int,
15   ↪ null_type, less<int>,
16   ↪ rb_tree_tag,tree_order_statistics_node_update>
17 // Driver program to test above functions
18 int main()
19 {
20   // Ordered set declared with name o_set
21   ordered_set o_set;
22
23   // insert function to insert in
24   // ordered set same as SET STL
25   o_set.insert(5);
26   o_set.insert(1);
27   o_set.insert(2);
28
29   // Finding the second smallest element
30   // in the set using * because
31   // find_by_order returns an iterator
32   cout << *(o_set.find_by_order(1))
33     << endl;
34
35   // Finding the number of elements
36   // strictly less than k=4
37   cout << o_set.order_of_key(4)
38     << endl;
39
40   // Finding the count of elements less
41   // than or equal to 4 i.e. strictly less
42   // than 5 if integers are present
43   cout << o_set.order_of_key(5)
44     << endl;
45
46   // Deleting 2 from the set if it exists
47   if(o_set.find(2) != o_set.end())
48     o_set.erase(o_set.find(2));
49
50   // Now after deleting 2 from the set
51   // Finding
52   // the second smallest element in the set
53   cout << *(o_set.find_by_order(1))
54     << endl;
55
56   // Finding the number of
57   // elements strictly less than k=4
58   cout << o_set.order_of_key(4)
59     << endl;
60
61   return 0;
62 }
```

```

10   return dsu[a] = find(dsu[a]);
11 }
12 void merge(int a, int b){
13   int fa = find(a);
14   int fb = find(b);
15   if(fa == fb){
16     return;
17   } else {
18     sum[fb] += sum[fa];
19     sz[fb] += sz[fa];
20     dsu[fa] = fb;
21   }
22 }
23 void move(int a, int b){
24   int fa = find(convert[a]);
25   int fb = find(convert[b]);
26   if(fa == fb){
27     return;
28   } else {
29     sz[fa] -= 1;
30     sum[fa] -= a;
31     convert[a] = ++n;
32     sum[convert[a]] = a;
33     sz[convert[a]] = 1;
34     merge(convert[a], convert[b]);
35   }
36 }
37 void solve(){
38   while(cin >> n >> q){
39     for(int i=1;i<MAXN;i++){
40       dsu[i] = sum[i] = convert[i] = i;
41       sz[i] = 1;
42     }
43     int a, b, c;
44     while(q--){
45       cin >> c;
46       if(c == 1){
47         cin >> a >> b;
48         merge(convert[a], convert[b]);
49       } else if(c == 2){
50         cin >> a >> b;
51         move(a, b);
52       } else {
53         cin >> a;
54         a = find(convert[a]);
55         cout << sz[a] << " " << sum[a] << endl;
56       }
57     }
58   }
59 }
60 }
```

12 Almost union-find

```

1 vector<int> dsu(MAXN);
2 vector<int> convert(MAXN);
3 vector<ll> sz(MAXN, 1);
4 vector<ll> sum(MAXN, 0);
5 int n, q;
6 int find(int a){
7   if(dsu[a] == a){
8     return a;
9   }
```

```

1 inline ll lowbit(ll x){
2   return x & (-x);
3 }
4
5 int n;
6 vector<int> a;
7 class BIT{
8 private:
9   vector<ll> sum;
10 public:
```

13 Mega inversion

```

11 BIT(): sum(vector<ll>(MAXN, 0)) {}
12 void update(int idx, int val){
13     for(int i=idx; i<=n; i+=lowbit(i)){
14         sum[i] += val;
15     }
16 }
17 ll query(int idx){
18     ll res = 0;
19     for(int i=idx; i>0; i-=lowbit(i)){
20         res += sum[i];
21     }
22
23     return res;
24 }
25 };
26
27 void solve(){
28     cin >> n;
29     a.resize(n);
30     for(auto &i: a) cin >> i;
31     BIT bit1, bit2;
32     vector<ll> ans1(MAXN, 0);
33     vector<ll> ans2(MAXN, 0);
34     for(int i=n-1; i>=0; i--){
35         ans1[i] = bit1.query(a[i]-1);
36         bit1.update(a[i], 1);
37     }
38     for(int i=0; i<n; i++){
39         ans2[i] = i - bit2.query(a[i]);
40         bit2.update(a[i], 1);
41     }
42     ll ans = 0;
43     for(int i=0; i<n; i++){
44         ans += ans1[i] * ans2[i];
45     }
46     cout << ans << endl;
47 }
48 }
49
50 //***** Good Luck : ) *****/
51 int main () {
52     TIME(main);
53     IOS();
54     int t=1;
55     // cin >> t;
56     while(t--){
57         solve();
58     }
59
60     return 0;
61 }
```

14 逆序數對

```

1 #include <iostream>
2 #include <vector>
3 #include <algorithm>
4 using namespace std;
5
6 int Case, n, a[100005], bit[100005];
7 long long ans;
8 vector<int> v;
9
10 int get_id(int x){
```

```

11     return lower_bound(v.begin(),
12                         v.end(), x)-v.begin()+1;
13 }
14 void update(int x){
15     while(x<=n){
16         bit[x]++;
17         x += x & (-x);
18     }
19     int query(int x){
20         int ret = 0;
21         while(x){
22             ret += bit[x];
23             x -= x & (-x);
24         }
25         return ret;
26     }
27
28     int main() {
29         ios_base::sync_with_stdio(0);
30         cin.tie(0);
31         while(cin >> n){
32             if(n == 0) break;
33             v.clear();
34             ans = 0;
35             for(int i=0; i<n; i++){
36                 cin >> a[i];
37                 bit[i] = 0;
38                 v.push_back(a[i]);
39             }
40             bit[n] = 0;
41             sort(v.begin(), v.end());
42             v.erase(unique(v.begin(),
43                           v.end()), v.end());
43             for(int i=0; i<n; i++){
44                 ans += i - query(get_id(a[i]));
45                 update(get_id(a[i]));
46             }
47             Case++;
48             cout << "Case #" << Case << ": " << ans << "\n";
49         }
50     }
```

15 Batmanacci

```

1 n, k = input().split()
2 n = int(n)
3 k = int(k)
4
5 fib = [0, 1, 1]
6
7 for i in range(3, n+1):
8     fib.append(fib[i-1] + fib[i-2])
9
10 while n > 2:
11     if k > fib[n-2]:
12         k -= fib[n-2]
13         n -= 1
14     else:
15         n -= 2
16
17 if n == 1:
18     print("N")
19 else:
```

```
20 print("A")
```

16 模逆元

```
1 #include <iostream>
2 using namespace std;
3 #define ll long long
4
5 const int mod = 17, maxn = 20;
6
7 ll pre[maxn+1];
8 ll inv[maxn+1];
9 ll prei[maxn+1];
10
11 void build(int n){
12     pre[1] = pre[0] = 1,
13     → inv[1] = inv[0] = 1, prei[1] = prei[0] = 1;
14     for(int i=2; i<=n; i++){
15         pre[i] = pre[i-1] * i % mod;
16         // i 的逆元 inv[i] = -(p/i) * inv[p% i] (mod p)
17         inv[i] = mod - (mod / i * inv[mod % i]) % mod;
18         prei[i] = prei[i-1] * inv[i] % mod;
19     }
20 }
21 ll C(int n, int k){
22     return pre[n] * prei[k] % mod * prei[n-k] % mod;
23 }
24
25 int main(){
26     build(maxn);
27     cout << inv[10] << endl;
28     cout << C(6, 3) << endl;
29 }
```

17 樹重心

```
1 #include <iostream>
2 #include <vector>
3 using namespace std;
4 const int maxn = 100005;
5 int N;
6 vector<int> g[maxn];
7 int cost[maxn]; // 慘度
8
9 int dfs(int now, int pre){
10    // tot: 以 now 為 root 的子樹 size
11    int tot = 1, ret = 0;
12    // now 下方的子樹
13    for(auto nxt: g[now]){
14        if(nxt != pre){
15            ret = dfs(nxt, now);
16            tot += ret;
17            cost[now] = max(cost[now], ret);
18        }
19    }
20    // now 頭上的子樹
21    cost[now] = max(cost[now], N - tot);
22    return tot;
23 }
24
25 int main(){
26     ios_base::sync_with_stdio(0);
```

```
27     cin.tie(0);
28     int T, a, b;
29     cin >> T;
30     while(T--){
31         cin >> N;
32         for(int i=0; i<N; i++){
33             cost[i] = 0;
34             g[i].clear();
35         }
36         for(int i=0; i<N-1; i++){
37             cin >> a >> b;
38             g[a].push_back(b);
39             g[b].push_back(a);
40         }
41         dfs(0, -1); // 將 0 當作 root
42         int mn = 0x7FFFFFFF;
43         int ans = -1;
44         for(int i=0; i<N; i++){
45             if(cost[i] < mn){
46                 mn = cost[i];
47                 ans = i;
48             }
49         }
50         cout << ans << "\n";
51     }
52     return 0;
53 }
```

1 // 份代默點點 1 起始，即 $i \in [1, n]$

2 int size[MAXN],
→ // 點的“大小”（所有子點上點 + 點點）

3 weight[MAXN], // 點的“重量”

4 centroid[2]; // 用于點的重心（存的是點）

5

6 void GetCentroid(int
→ cur, int fa) { // cur 表示前點 (current)

7 size[cur] = 1;
8 weight[cur] = 0;
9 for(int i = head[cur]; i != -1; i = e[i].nxt) {
10 if(e[i].to !=
→ fa) { // e[i].to 表示有向所通向的點。
11 GetCentroid(e[i].to, cur);
12 size[cur] += size[e[i].to];
13 weight[cur]
→ = max(weight[cur], size[e[i].to]);
14 }
15 }
16 }
17 weight[cur] = max(weight[cur], n - size[cur]);
18 if(weight[cur]
→ <= n / 2) { // 依照點的重心的定點
19 centroid[centroid[0] != 0] = cur;
20 }
21 }

18 pbds hashtable

```
1 #include <ext/pb_ds/assoc_container.hpp>
2 using namespace __gnu_pbds;
3 gp_hash_table<int, int> table;
4 cc_hash_table<int, int> table;
```

19 0/1 背包

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 void solve(int c, int n){
5     vector<int> v(n);
6     vector<int> w(n);
7     REP(i, n){
8         cin >> v[i] >> w[i];
9     }
10    vector<vector<int>>
11    → dp(n+1, vector<int>(c+1, 0));
12    vector<vector<bool>>
13    → has(n+1, vector<bool>(c+1, false));
14    for(int i=0; i<n; i++){
15        for(int j=0; j<=c; j++){
16            if(j-w[i]<0){
17                dp[i+1][j]=dp[i][j];
18            } else {
19                if(dp[i][j-w[i]]+v[i]>dp[i][j]){
20                    dp[i+1][j]=dp[i][j-w[i]]+v[i];
21                    has[i][j]=true;
22                } else {
23                    dp[i+1][j]=dp[i][j];
24                }
25            }
26        }
27        vector<int> ans;
28        for(int i=n-1, j=c; i>=0; i--){
29            if(has[i][j]){
30                ans.push_back(i);
31                j-=w[i];
32            }
33        }
34        cout << ans.size() << endl;
35        for(auto i: ans) cout << i << " ";
36        cout << endl;
37    }
38
39 //***** Good Luck : ) *****/
40 int main(){
41     TIME(main);
42     IOS();
43     // cin >> t;
44     int c, n;
45     while(cin >> c >> n){
46         solve(c, n);
47     }
48
49     return 0;
}

```

20 有限背包

```

1 const int N=100, W=1000000;
2 int cost[N], weight[N], number[N];
3 int c[W+1];
4
5 void knapsack(int n, int w)
{
7     for(int i=0; i<n; ++i)
{

```

```

9         int num=min(number[i], w/weight[i]);
10        for(int k=1; num>0; k*=2)
11        {
12            if(k>num) k=num;
13            num-=k;
14            for(int j=w; j>=weight[i]*k; --j)
15            c[j]=max(c[j],
16                      → c[j-weight[i]*k]+cost[i]*k);
17        }
18    }
19    cout << " 最高的價值為" << c[w];
}

```

21 無限背包

$$c(n, w) = \max(c(n-1, w), c(n, w - \text{weight}[n]) + \text{cost}[n])$$

```

1 const int N=100, W=1000000;
2 int cost[N], weight[N];
3 int c[W+1];
4
5 void knapsack(int n, int w)
{
6     memset(c, 0, sizeof(c));
7
8     for(int i=0; i<n; ++i)
9         for(int j=weight[i]; j<=w; ++j)
10            c[j]=max(c[j], c[j-weight[i]]+cost[i]);
11
12    cout << " 最高的價值為" << c[w];
13
14
}

```

22 換錢問題

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 void solve(){
5     vector<int> dp(MAXN, INF);
6     int x;
7     cin >> x;
8     int n;
9     cin >> n;
10    vector<int> a(n);
11    REP(i, n) cin >> a[i];
12    dp[0]=0;
13    for(int i=0; i<n; i++){
14        for(int j=MAXN; j>=0; j--){
15            if(j-a[i]>=0){
16                dp[j]=min(dp[j], dp[j-a[i]]+1);
17            }
18        }
19    }
20    for(int i=x; i<MAXN; i++){
21        if(dp[i]!=INF){
22            cout << i << " " << dp[i] << endl;
23            return;
24        }
25    }
26
27
}

```

```

28 //***** Good Luck :) *****/
29 int main () {
30     TIME(main);
31     IOS();
32     int t=1;
33     cin >> t;
34     while(t--){
35         solve();
36     }
37
38     return 0;
39 }
```

23 Increasing Subsequences

```

1 #include<bits/stdc++.h>
2 using namespace std;
3
4 void print(vector<int>&
5             a, vector<int>& prev, int idx){
6     vector<int> tmp;
7     while(idx != -1){
8         tmp.push_back(a[idx]);
9         idx = prev[idx];
10    }
11    reverse(ALL(tmp));
12    for(auto i:tmp) cout << i << " ";
13    cout << endl;
14 }
15
16 void solve(){
17     int n;
18     cin >> n;
19     if(n == 0) exit(0);
20     vector<int> a(n);
21     for(auto &i: a) cin >> i;
22     vector<int> dp(n, 1);
23     vector<int> prev(n, -1);
24     for(int i=0;i<n;i++){
25         for(int j=0;j<i;j++){
26             if(a[j] < a[i] and (dp[j]+1 >
27                             dp[i] or (dp[j]+1 == dp[i] and (prev[i]
28                             != -1 and a[prev[i]] > a[j])))){
29                 dp[i] = dp[j]+1;
30                 prev[i] = j;
31             }
32         }
33         int ans = 0;
34         int ans_val = INF;
35         for(int i=0;i<n;i++){
36             if(ans < dp[i]
37                 or (ans == dp[i] and ans_val > a[i])){
38                 ans = dp[i];
39                 ans_val = a[i];
40             }
41             cout << ans << " ";
42             if(dp[i] == ans and ans_val == a[i]){
43                 print(a, prev, i);
44                 break;
45             }
46     }
47 }
```

```

46 }
47 //***** Good Luck :) *****/
48 int main () {
49     TIME(main);
50     IOS();
51     // cin >> t;
52     while(true){
53         solve();
54     }
55
56     return 0;
57 }
```

24 Exgcd

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 typedef pair<int, int> pii;
5 typedef pair<ll, ll> pll;
6 typedef pair<int, ll> pil;
7 typedef pair<ll, int> pli;
8 typedef pair<double, double> pdd;
9 #define SQ(i) ((i)*(i))
10 #define MEM(a, b) memset(a, (b), sizeof(a))
11 #define SZ(i) int(i.size())
12 #define
13     FOR(i, j, k, in) for (int i=j ; i<(k) ; i+=in)
14 #define RFOR(i,
15             j, k, in) for (int i=j ; i>=(k) ; i-=in)
16 #define REP(i, j) FOR(i, 0, j, 1)
17 #define REP1(i, j) FOR(i, 1, j+1, 1)
18 #define RREP(i, j) RFOR(i, j, 0, 1)
19 #define ALL(_a) _a.begin(),_a.end()
20 #define mp make_pair
21 #define pb push_back
22 #define eb emplace_back
23 #define X first
24 #define Y second
25 #ifdef tmd
26 #define TIME(i) Timer i(#i)
27 #define debug( ... ) do{\
28     fprintf(stderr,"%s
29     -
30     %d
31     (%s)
32     =
33     ",__PRETTY_FUNCTION__,__LINE__,#__VA_ARGS__);
34     \
35 }while(0)
36 template<typename
37 T>void _do(T &&x){cerr<<_x<<endl;}
38 template<typename
39 T, typename ... S> void _do(T &&x, S
40 && ... _t){cerr<<_x<<" , ";_do(_t ... );}
41 template<typename
42 _a,typename _b> ostream& operator<<(ostream &_s, const pair<_a,_b> &_p)
43 {return _s<<"(" <<_p.X << ", " <<_p.Y << ")";}
44 template<typename It>
45 ostream& _OUTC(ostream &_s, It _ita, It _itb)
```

```

33 {
34     _s<<"{";
35     for(It _it=_ita;_it!=_itb;_it++)
36     {
37         _s<<(_it==_ita?"~":"")<<*_it;
38     }
39     _s<<"}";
40     return _s;
41 }
42 template<typename _a> ostream
43     &operator<<(ostream&_s,vector<_a>
44     &_c){return _OUTC(_s,ALL(_c));}
45 template<typename _a>
46     ostream&operator<<(ostream&_s,set<_a>
47     &_c){return _OUTC(_s,ALL(_c));}
48 template<typename _a>
49     ostream&operator<<(ostream&_s,deque<_a>
50     &_c){return _OUTC(_s,ALL(_c));}
51 template<typename _a,>typename _b> ostream
52     &operator<<(ostream&_s,map<_a,_b>
53     &_c){return _OUTC(_s,ALL(_c));}
54 template<typename _t> void pary(_t
55     _a,_t _b){_OUTC(cerr,_a,_b);cerr<<endl;}
56 #define IOS()
57 class Timer{
58 private:
59     string scope_name;
60     chrono::high_resolution_clock::time_point
61         start_time;
62 public:
63     Timer(string name) : scope_name(name) {
64         start_time=
65             chrono::high_resolution_clock::now();
66     }
67     ~Timer(){
68         auto stop_time=
69             chrono::high_resolution_clock::now();
70
71         auto
72         length
73         =
74             chrono::duration_cast<chrono::microseconds>
75             (-start_time).count();
76         double mlength=double(length)*0.001;
77         debug(scope_name,mlength);
78     }
79 };
80 #else
81 #define TIME(i)
82 #define debug( ... )
83 #define pary( ... )
84 #define endl '\n'
85 #define IOS()
86     ios_base::sync_with_stdio(0);cin.tie(0)
87 #endif
88
89 const ll MOD=1000000007;
90 const ll INF=0x3f3f3f3f3f3f3f3f;
91 const int iNF=0x3f3f3f3f;
92 const ll MAXN=100005;
93
94 mt19937 rng(chrono::steady_clock::now())
95     .time_since_epoch().count());
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110

```

```

79 /*
80 find solution of ax+by=gcd(a, b)
81 gcd(a, b)=gcd(b, a%b), therefore
82 bx+(a%b)y=gcd(b, a%b)=gcd(a, b)=ax+by
83 bx'+(a%b)y'=bx'+(a-floor(a/
84     → b)*b)y'=ay'+b(x'-floor(a/b)y')=ax+by
85 x=y', y=x'-floor(a/b)*y'
86 */
87
88 void exgcd(int a, int b, int &x, int &y){
89     if(b == 0){
90         x=1;y=0;
91         return;
92     }
93     int x1, y1;
94     exgcd(b, a%b, x1, y1);
95     x=y1;
96     y=x1-(a/b)*y1;
97 }
98
99
100 // if x < 0, ans=(x+b)%b
101
102 //***** Good Luck :) *****/
103 int main(){
104     TIME(main);
105     IOS();
106
107     return 0;
108 }
109
110

```

25 計算幾何模板

```

1 #include<bits/stdc++.h>
2 using namespace std;
3 #define X first
4 #define Y second
5 typedef pair<double, double> Pt;
6
7
8 Pt operator+(const Pt&p1, const Pt&p2){
9     return Pt(p1.X+p2.X, p1.Y+p2.Y);
10 }
11 Pt operator-( const Pt&p1 , const Pt&p2){
12     return Pt(p1.X-p2.X, p1.Y-p2.Y);
13 }
14 double operator*( const Pt&p1 , const Pt&p2 ){
15     return p1.X*p2.X+p1.Y*p2.Y;
16 }
17 double operator^( const Pt&p1 , const Pt&p2 ){
18     return p1.X*p2.Y-p1.Y*p2.X;
19 }
20
21 int main(){
22
23 }
24

```

26 Monotone Chain

```

1 // :80<enter>
2 #include <bits/stdc++.h>
3 using namespace std;
4 typedef long long ll;
5 typedef pair<int, int> pii;
6 typedef pair<ll, ll> pll;
7 typedef pair<int, ll> pil;
8 typedef pair<ll, int> pli;
9 typedef pair<double, double> pdd;
10 #define SQ(i) ((i)*(i))
11 #define MEM(a, b) memset(a, (b), sizeof(a))
12 #define SZ(i) int(i.size())
13 #define
14   ← FOR(i, j, k, in) for (int i=j ; i<(k) ; i+=in)
15 #define RFOR(i,
16   ← j, k, in) for (int i=j ; i>=(k) ; i-=in)
17 #define REP(i, j) FOR(i, 0, j, 1)
18 #define REP1(i, j) FOR(i, 1, j+1, 1)
19 #define RREP(i, j) RFOR(i, j, 0, 1)
20 #define ALL(_a) _a.begin(),_a.end()
21 #define mp make_pair
22 #define pb push_back
23 #define eb emplace_back
24 #define X first
25 #define Y second
26 #ifdef jayinnn
27 #define TIME(i) Timer i(#i)
28 #define debug( ... ) do{\
29   ← fprintf(stderr,"%s
29   ← -
29   ← %d
29   ← (%s)
29   ← =
29   ← ", __PRETTY_FUNCTION__, __LINE__, #__VA_ARGS__)
29   ← \
29   ← _do(__VA_ARGS__);\
29 }while(0)
30 template<typename
31   ← T>void _do(T&&_x){cerr<<_x<<endl;}
32 template<typename
33   ← T,typename ... S>void _do(T&&_x,S
34   ← &&... _t){cerr<<_x<<" ,_do(_t ... );}
35 template<typename
36   ← _a,typename _b> ostream& operator
37   ← <<(ostream&_s,const pair<_a,_b>&_p)
38   ← {_return _s<<"("<<_p.X<<","<<_p.Y<<")";}
39 template<typename It>
40   ← ostream& _OUTC(ostream&_s,It _ita,It _itb)
41 {
42   ← _s<<"{";
43   ← for(It _it=_ita;_it!=_itb;_it++)
44   ← {
45   ←   ← _s<<(_it==_ita?" ":"")<<*_it;
46   ← }
47   ← _s<<"}";
48   ← return _s;
49 }
50 template<typename _a> ostream
51   ← &operator <<(ostream&_s,vector<_a>
52   ← &_c){return _OUTC(_s,ALL(_c));}

```

```

44 template<typename _a>
45   ← ostream&operator <<(ostream&_s, set<_a>
46   ← &_c){return _OUTC(_s,ALL(_c));}
47 template<typename _a>
48   ← ostream&operator <<(ostream&_s, deque<_a>
49   ← &_c){return _OUTC(_s,ALL(_c));}
50 template<typename _a,typename _b> ostream
51   ← &operator <<(ostream&_s, map<_a,_b>
52   ← &_c){return _OUTC(_s,ALL(_c));}
53 template<typename _t> void pary(_t
54   ← _a,_t_b){_OUTC(cerr,_a,_b);cerr<<endl;}
55 #define IOS()
56 class Timer{
57 private:
58   string scope_name;
59   chrono::high_resolution_clock::time_point
60   ← start_time;
61 public:
62   Timer(string name) : scope_name(name) {
63     start_time =
64       ← chrono::high_resolution_clock::now();
65   }
66   ~Timer(){
67     auto stop_time=
68       ← chrono::high_resolution_clock::now();
69
69   ← auto
70   ← length
71   ← =
72   ← chrono::duration_cast<chrono::microseconds>(
73   ← -start_time).count();
74   ← double mlength=double(length)*0.001;
75   ← debug(scope_name, mlength);
76 }
77 #else
78 #define TIME(i)
79 #define debug( ... )
80 #define pary( ... )
81 #define endl '\n'
82 #define IOS()
83   ← ios_base::sync_with_stdio(0);cin.tie(0)
84 #endif
85
86 const ll MOD=1000000007;
87 const ll INF=0x3f3f3f3f3f3f3f3f;
88 const int iNF=0x3f3f3f3f;
89 const ll MAXN=100005;
90
91 mt19937 rng(chrono::steady_clock::now()
92   ← .time_since_epoch().count());
93
94 int cross(pio, piia, piib){
95   ← int u=a.X-o.X;
96   ← int v=a.Y-o.Y;
97   ← int s=b.X-o.X;
98   ← int t=b.Y-o.Y;
99
100  ← return u*t-s*v;
101 }
102
103 bool cmp(pii&a, pii&b){
104   ← return (a.X<b.X) || (a.X==b.X && a.Y<b.Y);
105 }

```

```

92 void solve(int n){
93     vector<pii> a;
94     set<pii> s;
95     int x, y;
96     for(int i=0;i<n;i++){
97         cin >> x >> y;
98         s.insert({x, y});
99     }
100    for(auto i: s){
101        a.push_back(i);
102    }
103    n=a.size();
104    if(n<=2){
105        cout << n << endl;
106        for(auto i: a){
107            cout << i.X << " " << i.Y << endl;
108        }
109        return;
110    }
111    sort(ALL(a), cmp);
112    debug(a);
113    debug(a);
114    int idx=0;
115    vector<pii> ch(2*n);
116    for(int i=0;i<n;i++){
117        while(idx>=2 &&
118            cross(ch[idx-2], ch[idx-1], a[i])<=0){
119            idx--;
120        }
121        ch[idx++]=a[i];
122    }
123    for(int i=n-2, t=idx+1; i>=0; i--){
124        while(idx>=t &&
125            cross(ch[idx-2], ch[idx-1], a[i])<=0){
126            idx--;
127        }
128        ch[idx++]=a[i];
129    }
130    idx--;
131    cout << idx << endl;
132    for(int i=0; i<idx; i++){
133        cout << ch[i].X << " " << ch[i].Y << endl;
134    }
135 }

136

137
138 //***** Good Luck :) *****/
139 int main(){
140     TIME(main);
141     IOS();
142     int n;
143
144     while(cin >> n){
145         if(n==0) break;
146         solve(n);
147     }
148
149     return 0;
150 }
151

```

27 旋轉卡尺

```

1 #include <iostream>
2 #include <cmath>
3 #include <algorithm>
4 #include <vector>
5 #include <map>
6 #include <stack>
7 #include <set>
8 #include <queue>
9 #include <list>
10 #include <string.h>
11 #include <complex>
12 #include <sstream>
13 using namespace std;
14 #define INITIO()
15     → ios_base::sync_with_stdio(false); cin
15     → .tie(NULL);
15 #define FILEC()
15     → freopen("a.in", "r", stdin); freopen("out"
15     → .txt", "w", stdout);
16 #define endl '\n'
17 #define F first
18 #define S second
19 #define pb push_back
20 #define pf push_front
21 #define all(a) a.begin(), a.end()
22 #define rall(a) a.rbegin(), a.rend()
23 #define sz(a) (int)a.size()
24 #define FOR(i,a,b) for(int i=(a); i<(b); i++)
25 using vii = vector<vector<int>>;
26 using vi = vector<int>;
27 using mii = map<int, int>;
28 using cd = complex<double>;
29 typedef long long ll;
30 typedef unsigned long long ull;
31 typedef pair<int, int> pii;
32 typedef pair<long, long> pll;
33 const double PI = acos(-1);
34 const ll Mod = 1e9+7;
35 //const ll Inf = 2e18+9;
36 //const int N = 1e5+9;
37 //const int dx[4] = {0, 1, 0, -1};
38 //const int dy[4] = {1, 0, -1, 0};
39 //const int dx8[8] = {-1, -1, -1, 0, 1, 1, 1, 0};
40 //const int dy8[8] = {-1, 0, 1, 1, 1, 0, -1, -1};
41 ll gcd(ll a, ll
41     → b){if(b==0) return a; return gcd(b, a%b);}
42 ll lcm(ll a, ll b){return a*b/gcd(a, b);}
43 int mul(int a, int b){return (1LL*a*b)%Mod;}
44 ll add(ll a, ll b){a+=b;if(a
44     → >= Mod) a-=Mod;if(a<0) a+=Mod;return a;}
45 ll sub(ll a, ll
45     → b){return (a-b)%Mod+((a>=b)?0:Mod);}
46 //
46     → -----
47
48 ll cross(plla, pllb, pllc){
49     return (a.F-b.F)*(a.S-c.S)
49     → -(a.S-b.S)*(a.F-c.F);
50 }
51
52 void solve(){
53     int n;
54     scanf("%d", &n);

```

```

55  vector<pll>a;
56  set<pii>s;
57  FOR(i,0,n){
58      int x,y;
59      scanf("%d %d", &x, &y);
60      if(s.count({x,y}))continue;
61      a.pb({x,y});
62      s.insert({x,y});
63  }
64  n=sz(a);
65  sort(all(a));
66  vector<pll>ans;
67  FOR(i,0,n){
68      while(sz(ans)>=2 &&
69          cross(ans[sz(ans)-1], ans[sz(ans)-2],
70          a[i])<=0)ans.pop_back();
71      ans.pb(a[i]);
72  }
73  for(int i=n-2, t=sz(ans)+1; i>=0; i--){
74      while(sz(ans)>=t &&
75          cross(ans[sz(ans)-1], ans[sz(ans)-2],
76          a[i])<=0)ans.pop_back();
77      ans.pb(a[i]);
78  }
79  if(sz(ans)>1)ans.pop_back();
80  a=ans;
81  n=sz(a);
82  ll best=0;
83  //FOR(i,0,n)cout<<a[i].F<<'
84  //    '<<a[i].S<<endl;
85  for(int i=0; i<n-2; i++){
86      int k=i+1;
87      for(int j=i+2; j<n; j++){
88          ll area=abs(cross(a[i], a[k], a[j]));
89          while(1){
90              k++;
91              ll newarea=abs(cross(a[i], a[k], a[j]));
92              if(newarea<=area || k>=j)break;
93              area=newarea;
94          }
95          k=max(i+1, k-1);
96          best=max(area, best);
97      }
98      printf("%.5f\n", 0.5*best);
99  }
100
101 int main(){
102     INITIO()
103     //int t; cin>>t;
104     //while(t--){
105         solve();
106     //}
107     return 0;
108 }
```

28 Optimal Triangulation

$d(i, j) = \max(d(i, k) + d(k, j) + w(i, j, k) | i < k < j)$, $w(i, j, k)$ is the weight function

```

1 int n;
2 vector<pii>a;
3 double area(int i, int j, int k){
```

```

4     return abs((a[j].F
5             - a[i].F)*(a[k].S - a[i].S)*1.0 - (a[j].S
6             - a[i].S)*(a[k].F - a[i].F)*1.0)/2.0;
7 }
8
9 bool judge(int a, int b, int c){
10    for(int i=0; i<n; i++){
11        if(i==a || i==b || i==c)continue;
12        double s=area(a,b,i) +
13            area(a,c,i)+area(b,c,i)-area(a,b,c);
14        if(fabs(s)<0.01)return false;
15    }
16    return true;
17 }
18 double dp(int i, int j){
19    if(i+1>=j)return d[i][j]=0;
20    if(d[i][j]!=-1.0)return d[i][j];
21    double ans=1e9*1.0;
22    for(int k=i+1; k<j; k++){
23        if(judge(i,j,k))ans
24            =min(ans, max(dp(i,k), max(dp(k,j),
25                area(i,k,j))));
26    }
27    return d[i][j]=ans;
28 }
29
30 int main(){
31     ios_base::sync_with_stdio(false);cin
32     .tie(NULL);
33     //freopen("a.in", "r", stdin);freopen("out"
34     .txt", "w", stdout);
35     int t; cin>>t;
36     while(t--){
37         cin>>n;
38         a.resize(n);FOR(i,0,n)cin>>a[i].F>>a[i].S;
39         FOR(i,0,n)FOR(j,0,n)d[i][j]=-1.0;
40         printf("%.1f\n", dp(0,n-1));
41     }
42     return 0;
43 }
```

29 Catalan Number

When counting the way of triangulation of a convex hull. $f(n) = f(n-1)*f(2) + f(3)*f(n-2) + \dots + f(n-1)*f(2)$

30 Optimal Matrix Chain Multiplication

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 int p[105];
5
6 int dp(int i, int j){
7     if(j<=i+1) return 0;
8     int mn=100000000;
9     for(int k=i+1; k<j; k++){
10         int cost=dp(i,k)+dp(k,j)+p[i]*p[k]*p[j];
11         mn=min(cost, mn);
12     }
13 }
```

```

13     return mn;
14 }
15
16 int main(){
17     int n;
18     cin>>n;
19     for(int i=0;i<n;i++)cin>>p[i];
20     cout<<dp(0,n-1)<<endl;
21     return 0;
22 }
```

31 Check if point belongs to the convex polygon ($\log N$ for each query)

```

1 struct pt {
2     long long x, y;
3     pt() {}
4     pt(long long _x, long long _y) : x(_x), y(_y) {}
5     pt operator+(const
6         → pt&p) const { return pt(x+p.x, y+p.y); }
7     pt operator-(const
8         → pt&p) const { return pt(x-p.x, y-p.y); }
9     long long cross(const
10        → pt&p) const { return x*p.y - y*p.x; }
11     long long dot(const
12        → pt&p) const { return x*p.x + y*p.y; }
13     long long cross(const pt &a, const pt &b) const
14        → { return (a-*this).cross(b-*this); }
15     long long dot(const pt &a, const pt &b)
16        → const { return (a-*this).dot(b-*this); }
17     long long sqrLen()
18        → const { return this->dot(*this); }
19 };
20
21 bool lexComp(const pt&l, const pt&r) {
22     return l.x < r.x || (l.x == r.x && l.y < r.y);
23 }
24
25 int sgn(long long
26        → val) { return val > 0 ? 1 : (val == 0 ? 0 : -1); }
27
28 vector<pt> seq;
29 pt translation;
30 int n;
```

31 bool pointInTriangle(pt a, pt b, pt c, pt point) {
32 long long s1 = abs(a.cross(b, c));
33 long long s2 = abs(point.cross(a,
34 → b)) + abs(point.cross(b,
35 → c)) + abs(point.cross(c, a));
36 return s1 == s2;
37 }

38 void prepare(vector<pt> &points) {
39 n = points.size();
40 int pos = 0;
41 for (int i = 1; i < n; i++) {
42 if (lexComp(points[i], points[pos]))
43 pos = i;
44 }
45 rotate(points.begin(),
46 → points.begin() + pos, points.end());
47 n--;

```

40     seq.resize(n);
41     for(int i=0; i<n; i++)
42         seq[i]=points[i+1]-points[0];
43     translation=points[0];
44 }

45 bool pointInConvexPolygon(pt point) {
46     point=point-translation;
47     if (seq[0].cross(point) != 1 &&
48         sgn(seq[0].cross(point))
49         → != sgn(seq[0].cross(seq[n-1])))
50         return false;
51     if (seq[n-1].cross(point) != 0 &&
52         sgn(seq[n-1].cross(point))
53         → != sgn(seq[n-1].cross(seq[0])))
54         return false;
55     if (seq[0].cross(point) == 0)
56         return seq[0].sqrLen() >= point.sqrLen();
57
58     int l=0, r=n-1;
59     while(r-l>1){
60         int mid=(l+r)/2;
61         int pos=mid;
62         if (seq[pos].cross(point) >= 0)
63             l=mid;
64         else
65             r=mid;
66     }
67     int pos=l;
68     return pointInTriangle(seq[pos],
69        → seq[pos+1], pt(0, 0), point);
70 }
```

32 Bungee Builder (monotonic stack)

```

1 void solve(){
2     int n; cin>>n;
3     vector<int> a(n); FOR(i, 0, n) cin>>a[i];
4     stack<pii> s;
5     int ans=0;
6     FOR(i, 0, n){
7         int bot=a[i];
8         while(sz(s)){
9             pii x=s.top(); s.pop();
10            bot=min(bot, x.S);
11            if(x.F > a[i]){
12                s.push({x.F, bot});
13                ans=max(ans, a[i]-bot);
14                break;
15            }
16            ans=max(ans, x.first-bot);
17        }
18        s.push({a[i], a[i]});
19    }
20    cout<<ans<<endl;
21 }
```

33 Arachnophobia (dijkstra + binary search)

```

1 int n,m,t,st,ed,k;
2 struct Edge{
3     int v;ll w;
4     bool operator<(const Edge &cmp) const {
5         return cmp.w<w;
6     }
7 };
8
9 vector<Edge> g[100005];
10 ll dis[100005],spiderdis[100005];
11
12 bool check(ll val){
13     if(spiderdis[st]<val) return 0;
14     if(spiderdis[ed]<val) return 0;
15     FOR(i,0,n)dis[i]=Inf;
16     priority_queue<Edge> pq;
17     pq.push({st,0});
18     while(pq.size()){
19         auto node=pq.top(); pq.pop();
20         if(dis[node.v]<=node.w) continue;
21         dis[node.v]=node.w;
22         for(auto it:g[node.v]){
23             if(spiderdis[it.v]
24                 >=val && dis[it.v]>it.w+node.w){
25                 pq.push({it.v,it.w+node.w});
26             }
27         }
28     }
29 //FOR(i,0,n)cout<<dis[i]<<endl;
30     return dis[ed]<=t;
31 }
32 void solve(){
33     cin>>n>>m>>t;
34     while(m--){
35         int u,v,w;
36         cin>>u>>v>>w;
37         g[u].pb({v,w});
38         g[v].pb({u,w});
39     }
40     cin>>st>>ed;
41     cin>>k;
42     priority_queue<Edge> pq;
43     while(k--){
44         int x;cin>>x;
45         pq.push({x,0});
46     }
47     FOR(i,0,n)spiderdis[i]=Inf;
48     while(pq.size()){
49         auto node=pq.top(); pq.pop();
50         if(spiderdis[node.v]<=node.w) continue;
51         spiderdis[node.v]=node.w;
52         for(auto it:g[node.v]){
53             if(spiderdis[it.v]>it.w+node.w){
54                 pq.push({it.v,it.w+node.w});
55             }
56         }
57     }
58     ll ans=0,step=1;
59     FOR(i,0,62)step*=2;
60     while(step>0){
61         if(check(ans+step)) ans+=step;
62         step=(step>>1);
63     }
64 }
65 cout<<ans<<endl;
66 }
```

34 Ascending Photo (Do some transition and DP)

```

1 void solve(){
2     int n,cnt=0;cin>>n;
3     vector<int> h;
4     set<int> st;
5     map<int,int> mp;
6     FOR(i,0,n){
7         int x;cin>>x;
8         st.insert(x);
9         if(h.empty() || h.back()!=x)h.pb(x);
10    }
11    for(auto it:st){
12        if(!mp.count(it))mp[it]=cnt++;
13    }
14    n=sz(h);
15    vector<vector<int>> pos(cnt);
16    FOR(i,0,n){
17        h[i]=mp[h[i]];
18        pos[h[i]].pb(i);
19    }
20    pii best[2]={{0,n},{0,n}};
21    FOR(i,0,cnt-1){
22        pii nbest[2]={best[0],best[1]};
23        FOR(j,0,sz(pos[i])){
24            int p=pos[i][j];
25            if(p==n-1 || h[p]+1!=h[p+1]) continue;
26            pii s(0,n);
27            if(p!=best[0].second) s=best[0];
28            else s=best[1];
29            s.first++;
30            s.second=p+1;
31            if(pos[i+1].size()==1) s.second=n;
32            if(s>nbest[0]){
33                nbest[1]=nbest[0];
34                nbest[0]=s;
35            }
36            else if(s>nbest[1]) nbest[1]=s;
37        }
38        best[0]=nbest[0];
39        best[1]=nbest[1];
40    }
41    cout<<n-1-best[0].first<<endl;
42 }
```

35 Pokemongogo (TSP)

```

1 int n,ans,cnt=0 , g[22][22]={}, dp[1<<22][22];
2 map<string,int> mp;
3 vector<int> pok[22];
4 vector<pii> p, v;
5
6 int dfs(int i,int j){
7     if(dp[i][j]!=-1) return dp[i][j];
8     if(i==
9         << (1<<(n+1))-1&&j==0) return dp[i][j]=0;
10    int res = 1e9+5;
11    for(int k=0;k<=n;k++){
12        if(((i>>k)&1)){
```

```

12     res=min(res,
13         →   dfs((i | (1<<k)), k)+g[j][k]);
14   }
15 */
16 int b=i;
17 while(b){
18   cout<<(b&1);
19   b/=2;
20 }
21 cout<<' '<<j<<' ';
22 cout<<res<<endl;
23 */
24
25 return dp[i][j]=res;
26 }
27
28 void f(vector<int>a, int id){
29   if(a.size()==n+1){
30
31     for(int i=0;i<=n;i++){
32       if(a[i])continue;
33       a[i]=1;
34
35       int cn=0;
36       for(int i=0, k=1;i<sz(a);i++, k*=2)
37         cn+=a[i]*k;
38       if(dp[cn][i]!=-1){
39         ans=min(ans, dp[cn][i]+g[0][i]);
40         //for(int i=0;i<sz(a);i++)cout<<a[i];
41         //cout<<' '<<i<<' ';
42         //cout<<dp[cn][i]<<' ';
43         //cout<<endl;
44       }
45       a[i]=0;
46     }
47     //cout<<endl;
48
49   }
50   int k=sz(a);
51   FOR(i,0,sz(pok[id]))a.pb(1);
52   for(int i=0;i<pok[id].size();i++){
53     a[i+k]=0;
54     f(a, id+1);
55     a[i+k]=1;
56   }
57 }
58
59 void solve(){
60   cin>>n;
61   p.resize(n+1);
62   v.resize(n);
63   FOR(i,0,n){
64     int r,c;cin>>r>>c;
65     string s;cin>>s;
66     if(mp.count(s)==0){
67       mp[s]=cnt++;
68     }
69     pok[mp[s]].pb(i);
70     v[i]={r,c};
71   }
72   p[0]={0,0};
73   for(int i=0,k=1;i<cnt;i++){
74     for(int j=0;j<pok[i].size();j++){
75       p[k++]=v[pok[i][j]];
76     }
77   }
78 }
79 FOR(i,0,n+1){
80   FOR(j,0,n+1){
81     g[i][j]=abs(p[i].F-p[j].F)
82         → +abs(p[i].S-p[j].S);
83     //cout<<g[i][j]<<' ';
84   }
85   //cout<<endl;
86 }
87 memset(dp, -1, sizeof(dp));
88 dfs(0, 0);
89 ans=1e9+5;
90 f({0}, 0);
91 cout<<ans<<endl;
92 }
```

```

76   }
77 }
78 FOR(i,0,n+1){
79   FOR(j,0,n+1){
80     g[i][j]=abs(p[i].F-p[j].F)
81         → +abs(p[i].S-p[j].S);
82     //cout<<g[i][j]<<' ';
83   }
84   //cout<<endl;
85 }
86 memset(dp, -1, sizeof(dp));
87 dfs(0, 0);
88 ans=1e9+5;
89 f({0}, 0);
90 cout<<ans<<endl;
91 }
```

36 British Menu (Dp + SCC)

```

1 #include <cmath>
2 #include <cstdio>
3 #include <cstring>
4 #include <iostream>
5 #include <algorithm>
6 #include <vector>
7 #define MAX_V 200005
8 using namespace std;
9 int n, ans, ans1[MAX_V], ans2[MAX_V], c[MAX_V]
10    → [6], cnum[MAX_V], id[MAX_V], dis[MAX_V]
11    → [6], m;
12 vector<int> G[MAX_V];
13 vector<int> rG[MAX_V];
14 vector<int> vs;
15 bool used[MAX_V];
16 int cmp[MAX_V];
17 void add_edge(int from, int to)
18 {
19   G[from].push_back(to);
20   rG[to].push_back(from);
21 }
22 void dfs(int v)
23 {
24   used[v]=1;
25   for(int i=0;i<G[v].size();++i)
26   {
27     if(!used[G[v][i]])
28       dfs(G[v][i]);
29   }
30   vs.push_back(v);
31 }
32 void rdfs(int v, int k)
33 {
34   used[v]=1;
35   cmp[v]=k;
36   for(int i=0;i<rG[v].size();++i)
37   {
38     if(!used[rG[v][i]])
39       rdfs(rG[v][i], k);
40   }
41 }
42 int scc()
43 {
44   memset(used, 0, sizeof used);
45   vs.clear();
46 }
```

```

44     for(int v=1;v<=n;v++)
45     {
46         if(!used[v]) dfs(v);
47     }
48     memset(used,0,sizeof used);
49     int k=1;
50     for(int i=vs.size()-1;i>=0;--i)
51     {
52         if(!used[vs[i]])
53             rdfs(vs[i],k++);
54     }
55     return k;
56 }
57 void cal(int now,int nowid,int f,int nowdis)
58 {
59     used[now]=1;
60     dis[now]
61     ↪ [nowid]=max(dis[now][nowid],nowdis);
62     for(int i=0;i<G[now].size();++i)
63     if(cmp[G[now][i]]==f&&!used[G[now][i]])
64     {
65         cal(G[now][i],nowid,f,nowdis+1);
66     }
67     used[now]=0;
68     return;
69 }
70 int get2(int now)
71 {
72     if(ans1[now]==-1) return ans1[now];
73     int nowans=1;
74     for(int i=0;i<rG[now].size();i++)
75     if(cmp[rG[now][i]]!=cmp[now])
76     {
77         nowans=max(nowans,get2(rG[now][i])+1);
78     }
79     return ans1[now]=nowans;
80 }
81 int get2(int now)
82 {
83     if(ans2[now]==-1) return ans2[now];
84     int nowans=-1;
85     for(int i=1;i<=cnum[cmp[now]];++i)
86     {
87         nowans=max(nowans,get1(c[cmp[now]]
88         ↪ [i])+dis[now][i]);
89     }
90     return ans2[now]=nowans;
91 }
92 int main()
93 {
94     scanf("%d%d",&n,&m);
95     int u,v;
96     for(int i=1;i<=m;++i)
97     {
98         scanf("%d%d",&u,&v);
99         add_edge(u,v);
100    }
101    scc();
102    memset(used,0,sizeof used);
103    for(int i=1;i<=n;++i)
104    {
105        ans1[i]=ans2[i]=-1;
106        c[cmp[i]][++cnum[cmp[i]]]=i;
107        id[i]=cnum[cmp[i]];
108    }
109 }
110 }
```

```

107     cal(i,id[i],cmp[i],0);
108 }
109 for(int i=1;i<=n;++i)
110 {
111     ans=max(ans,get2(i));
112 }
113 cout<<ans;
114 }
115 }
116 }
```

37 Flight (Tree diameter/radius)

```

1 int n,vis[2505],dis[2505],
2   ↪ siz[2505],h1[2505],
2   ↪ h2[2505],c1[2505],c2[2505],p[2505];
3 vector<int> edge[2505];
4
5 pii treedia(int x){
6     for(int i=1;i<=n;i++){
7         vis[i]=0;
8         dis[i]=0;
9     }
10    queue<int> q;
11    q.push(x);
12    while(q.size()){
13        int top=q.front();
14        q.pop();
15        vis[top]=1;
16        for(auto it:edge[top]){
17            if(!vis[it]){
18                dis[it]=dis[top]+1;
19                q.push(it);
20            }
21        }
22        for(int i=1;i<=n;i++){
23            if(dis[x]<dis[i])x=i;
24        }
25        for(int i=1;i<=n;i++){
26            vis[i]=0;
27            dis[i]=0;
28        }
29        q.push(x);
30        while(q.size()){
31            int top=q.front();
32            q.pop();
33            vis[top]=1;
34            for(auto it:edge[top]){
35                if(!vis[it]){
36                    dis[it]=dis[top]+1;
37                    q.push(it);
38                }
39            }
40        }
41        int id=0;
42        for(int i=1;i<=n;i++){
43            if(dis[id]<dis[i])id=i;
44        }
45        int ans=0;
46        for(int i=1;i<=n;i++){
47            if(dis[i]
48            ↪ ==dis[id]/2+(dis[id]%2?1:0))ans=i;
49        }
50    }
51 }
```

```

49     if(dis[id] == 0)ans = x;
50     return {ans, id};
51 }
52
53 void record(int x, int height, int child)
54 {
55     if(height > h1[x])
56     {
57         h2[x] = h1[x]; c2[x] = c1[x];
58         h1[x] = height; c1[x] = child;
59     }
60     else if(height > h2[x])
61     {
62         h2[x] = height; c2[x] = child;
63     }
64 }
65
66 void dfs1(int x){
67     h1[x] = h2[x] = 0;
68     for(auto it:edge[x]){
69         if(p[x] != it){
70             p[it] = x;
71             dfs1(it);
72             record(x,h1[it]+1,it);
73         }
74     }
75 }
76
77 void dfs2(int x){
78     if(p[x] != x){
79         int y = p[x];
80         if(c1[y] == x)record(x,h2[y]+1,y);
81         else record(x,h1[y]+1,y);
82     }
83     for(auto it:edge[x]){
84         if(it != p[x]){
85             dfs2(it);
86         }
87     }
88 }
89
90 int treecentroid(int x){
91     int ans=x;
92     FOR(i,1,n+1)
93         → h1[i] = h2[i] = c1[i] = c2[i] = p[i] = 0;
94     p[x] = x;
95     dfs1(x);
96     dfs2(x);
97     for(int i=1;i<=n;i++){
98         //cout<<h1[i]<<' ';
99         if(h1[i] && h1[i] < h1[ans]){
100            ans=i;
101        }
102    }
103    //cout<<endl;
104    return ans;
105 }
106
107 void solve(){
108     cin>>n;
109     vector<pii> ed;
110     FOR(i,0,n-1){
111         int u,v;cin>>u>>v;
112         edge[u].pb(v);
113         edge[v].pb(u);
114     }
115     ed.pb({u,v});
116 }
117
118 int ans=1e9+5, a[4]={};
119 for(auto [u,v]:ed){
120     for(auto it=edge[u].begin();it!
121         → =edge[u].end();it++){
122         if(*it==v){
123             edge[u].erase(it);
124             break;
125         }
126     }
127     for(auto it=edge[v].begin();it!
128         → =edge[v].end();it++){
129         if(*it==u){
130             edge[v].erase(it);
131             break;
132         }
133     }
134     int k=treecentroid(u);
135     int j=treecentroid(v);
136     //cout<<k<<' '<<j<<endl;
137     edge[k].pb(j);
138     edge[j].pb(k);
139     int y=treedia(u).second;
140     if(ans > dis[y]){
141         ans=dis[y];
142         a[0]=u;
143         a[1]=v;
144         a[2]=k;
145         a[3]=j;
146     }
147     for(auto it=edge[k].begin();it!
148         → =edge[k].end();it++){
149         if(*it==j){
150             edge[k].erase(it);
151             break;
152         }
153     }
154     for(auto it=edge[j].begin();it!
155         → =edge[j].end();it++){
156         if(*it==k){
157             edge[j].erase(it);
158             break;
159         }
160     }
161     cout<<ans<<endl;
162     cout<<a[0]<<' '<<a[1]<<endl;
163     cout<<a[2]<<' '<<a[3]<<endl;
164 }
```

38 Rooted Subtree (Combinatorics + LCA)

```

1 vector<int> e[N];
2 int n,m,dep[N]={},siz[N]={},p[20][N]={};
3
4 void dfs(int x){
5     siz[x]=1;
6     for(auto it:e[x]){
7         if(p[0][x]!=it){
8             p[0][it]=x;
9             dep[it]=dep[x]+1;
10        }
11    }
12 }
```

```

10     dfs(it);
11     siz[x] += siz[it];
12   }
13 }
14 }

15 int lca(int a,int b){
16   if(dep[a]>dep[b])swap(a,b);
17   int u=a,v=b;
18   if(dep[a]!=dep[b]){
19     int dif=dep[b]-dep[a];
20     for(int i=0;i<20;i++){
21       if(dif&1)b=p[i][b];
22       dif>>=1;
23     }
24   }
25   if(a==b) return dep[v]-dep[u];
26   for(int i=19;i>=0;i--){
27     if(p[i][a]!=p[i][b]){
28       a=p[i][a];
29       b=p[i][b];
30     }
31   }
32   return dep[u]+dep[v]-2*dep[p[0][a]];
33 }
34 }

35 void solve(){
36   cin>>n>>m;
37   FOR(i,0,n-1){
38     int u,v;
39     cin>>u>>v;
40     e[u].pb(v);
41     e[v].pb(u);
42   }
43   p[0][1]=1;
44   dfs(1);
45   for(int i=1;i<20;i++){
46     for(int j=1;j<=n;j++){
47       p[i][j]=p[i-1][p[i-1][j]];
48     }
49   }
50   while(m--){
51     int u,v;cin>>u>>v;
52     ll x=1ll*lca(u,v)+1;
53     cout<<x*(x+1)/2+n-x<<endl;
54   }
55 }
56 }
```

39 Stogovi (LCA + DSU)

```

1 vector<int> e[N];
2 int n,m,dep[N]={},siz[N]={},p[20]
3   [N]={},f[N]={};
4
5 void dfs(int x){
6   siz[x]=1;
7   for(auto it:e[x]){
8     dep[it]=dep[x]+1;
9     dfs(it);
10    siz[x]+=siz[it];
11  }
12 }
13 int lca(int a,int b){

14   if(dep[a]>dep[b])swap(a,b);
15   if(dep[a]!=dep[b]){
16     int dif=dep[b]-dep[a];
17     for(int i=0;i<20;i++){
18       if(dif&1)b=p[i][b];
19       dif>>=1;
20     }
21   }
22   if(a==b) return a;
23   for(int i=19;i>=0;i--){
24     if(p[i][a]!=p[i][b]){
25       a=p[i][a];
26       b=p[i][b];
27     }
28   }
29   return p[0][a];
30 }

31 void prelca(){
32   dfs(0);
33   for(int i=1;i<20;i++){
34     for(int j=1;j<=n;j++){
35       p[i][j]=p[i-1][p[i-1][j]];
36     }
37   }
38 }

39 int find(int x){
40   if(x==f[x])return x;
41   return f[x]=find(f[x]);
42 }

43 void solve(){
44   cin>>n;
45   FOR(i,1,n+1)f[i]=i;
46   vector<pii> ans;
47   FOR(i,1,n+1){
48     char c;
49     cin>>c;
50     int v,w;
51     if(c=='a'){
52       cin>>v;
53       v=find(v);
54       p[0][i]=v;
55       e[v].pb(i);
56     }
57     else if(c=='b'){
58       cin>>v;
59       v=find(v);
60       f[i]=p[0][v];
61       ans.pb({v,-1});
62     }
63   }
64 }

65 else {
66   cin>>v>>w;
67   v=find(v);
68   w=find(w);
69   f[i]=v;
70   ans.pb({v,w});
71 }

72 }

73 prelca();
74 for(auto [v,w]:ans){
75   if(w==-1){
76     cout<<v<<endl;
77   }
78 }
```

```

79     else{
80         int x=lca(v,w);
81         cout<<dep[x]<<endl;
82     }
83 }

```

40 LCA

```

1 // :80<enter>
2 #include<bits/stdc++.h>
3 using namespace std;
4
5 vector<vector<int>> adj;
6 vector<vector<int>> p;
7 vector<int> d;
8
9 void dfs(int cur, int par, int dep){
10    d[cur]=dep;
11    p[cur][0]=par;
12    for(auto i: adj[cur]){
13        if(i!=par){
14            dfs(i, cur, dep+1);
15        }
16    }
17 }
18
19 int lca(int x, int y){
20    if(d[x]>d[y]) swap(x, y);
21    if(d[x] != d[y]){
22        int diff=d[y]-d[x];
23        for(int i=0;i<20;i++){
24            if(diff&1) y=p[y][i];
25            diff>=1;
26        }
27        if(x==y) return x;
28        for(int i=19;i>=0;i--){
29            if(p[x][i] != p[y][i]){
30                x=p[x][i];
31                y=p[y][i];
32            }
33        }
34    }
35
36    return p[x][0];
37 }
38
39 void solve(){
40    int n;
41    cin>>n;
42    adj.resize(n+1);
43    p.resize(n+1, vector<int>(20, 0));
44    d.resize(n+1);
45    int x, y;
46    for(int i=0;i<n-1;i++){
47        cin>>x>>y;
48        adj[x].pb(y);
49        adj[y].pb(x);
50    }
51    dfs(1, 0, 1);
52    for(int i=1;i<20;i++){
53        for(int j=1;j<=n;j++){
54            p[j][i]=p[p[j][i-1]][i-1];
55        }
56    }

```

```

56    }
57    llans=0;
58    debug(d);
59    for(int i=1;i<=n;i++){
60        for(int j=2*i;j<=n;j+=i){
61            debug(i, j, lca(i, j));
62            lldis=d[i]-2*d[lca(i, j)]+d[j]+1;
63            debug(dis);
64            ans+=dis;
65        }
66    }
67    cout << ans << endl;
68 }

```

41 Edmonds-Karp

Time complexity: $O(E^2V)$

```

1 // :80<enter>
2 #include<bits/stdc++.h>
3 using namespace std;
4
5 class Edge{
6 public:
7     int to, cap, rev;
8     Edge(int _to, int _cap, int
9           → _rev): to(_to), cap(_cap), rev(_rev) {}
10
11 vector<vector<Edge>> G(MAXN);
12 vector<bool> vis(MAXN, false);
13
14
15 void add_edge(int u, int v, int cap){
16    G[u].push_back(Edge(v, cap, G[v].size()));
17    G[v].push_back(Edge(u, 0, G[u].size()-1));
18 }
19
20 int dfs(int cur, int t, int f){
21    if(cur==t){
22        return f;
23    }
24    vis[cur]=true;
25    int cur_sz=G[cur].size();
26    for(int i=0;i<cur_sz;i++){
27        Edge& e=G[cur][i];
28        if(vis[e.to]==false && e.cap>0){
29            int d=dfs(e.to, t, min(f, e.cap));
30            if(d>0){
31                e.cap-=d;
32                G[e.to][e.rev].cap+=d;
33                return d;
34            }
35        }
36    }
37    return 0;
38 }
39
40 void solve(){
41    int n, m, k;
42    cin>>n>>m>>k;
43    /**
44     * node 0: s
45     * node 5000: t

```

```

46 * node 1: tmp
47 * node 1001 ~ 1999: hero
48 * node 2001 ~ 2999: monster
49 * s -> tmp: n+k
50 * tmp -> heroes: 2
51 * heroes -> monster: 1
52 * monster -> t: 1
53 */
54 int q;
55 for(int i=1;i<=n;i++){
56     cin >> q;
57     int tmp;
58     while(q--){
59         cin >> tmp;
60         add_edge(1000+i, 2000+tmp, 1);
61     }
62 }
63 add_edge(0, 1, n+k);
64 for(int i=1;i<=n;i++){
65     add_edge(1, 1000+i, 2);
66 }
67 for(int i=1;i<=m;i++){
68     add_edge(2000+i, 5000, 1);
69 }
70 int ans = 0;
71 while(1){
72     fill(ALL(vis), false);
73     int f = dfs(0, 5000, INF);
74     if(f == 0){
75         cout << ans << endl;
76         return;
77     }
78     ans += f;
79 }
80 }
81 }
```

42 Dinic

Time complexity: $O(V^2E)$

```

1 // :80<enter>
2 #include<bits/stdc++.h>
3 using namespace std;
4
5 int n, m;
6
7 class Edge{
8 public:
9     int to, rev, flow, cap;
10    Edge(int _to,
11          → int _rev, int _flow, int _cap): to(_to),
12          → rev(_rev), flow(_flow), cap(_cap) {}
13 };
14
15 class Graph{
16 public:
17     int v;
18     vector<int> level;
19     vector<vector<Edge>> adj;
20     vector<bool> vis;
21     Graph(int _v){
22         v = _v;
23         adj = vector<vector<Edge>>(v);
```

```

22     level = vector<int>(v);
23     vis = vector<bool>(v, false);
24 }
25
26 void addEdge(int x, int y, int cap){
27     Edge forward(y, adj[y].size(), 0, cap);
28     Edge backward(x, adj[x].size(), 0, 0);
29     adj[x].push_back(forward);
30     adj[y].push_back(backward);
31 }
32
33 bool bfs(int s, int t){
34     fill(level.begin(), level.end(), -1);
35     level[s] = 0;
36     queue<int> q;
37     q.push(s);
38     while(!q.empty()){
39         int cur = q.front();
40         q.pop();
41         for(auto &i: adj[cur]){
42             if(level[i.to] == -1 && i.flow < i.cap){
43                 level[i.to] = level[cur] + 1;
44                 q.push(i.to);
45             }
46         }
47     }
48
49     return level[t] != -1;
50 }
51
52 int sendFlow(int
53             → u, int flow, int t, vector<int>& idx){
54     if(u == t) return flow;
55     for(; idx[u] < (int)adj[u].size(); idx[u]++){
56         Edge& e = adj[u][idx[u]];
57
58         if(level[e.to]
59             → == level[u] + 1 && e.flow < e.cap){
60             int cur_flow = min(flow, e.cap - e.flow);
61             int tmp_flow
62             → = sendFlow(e.to, cur_flow, t, idx);
63
64             if(tmp_flow > 0){
65                 e.flow += tmp_flow;
66                 adj[e.to][e.rev].flow -= tmp_flow;
67                 return tmp_flow;
68             }
69         }
70     }
71
72     return 0;
73 }
74
75 int dinic(int s, int t){
76     for(int i=0;i<v;i++){
77         for(Edge& j: adj[i]){
78             j.flow = 0;
79         }
80     }
81     if(s == t) return -1;
82     int total = 0;
83     while(bfs(s, t)){
84         vector<int> idx(v, 0);
85         while(int flow = sendFlow(s, INF, t, idx)){
86             total += flow;
```

```

84     }
85 }
86
87     return total;
88 }
89 }
90
91 void solve(){
92     cin >> n >> m;
93     Graph g(2*n+1);
94     int u, v;
95     // for(int i=1; i<=n; i++){
96     //     g.addEdge(i, n+i, 1);
97     //     g.addEdge(n+i, i, 1);
98     //}
99     for(int i=0; i<m; i++){
100         cin >> u >> v;
101        // g.addEdge(n+u, v, 1);
102        g.addEdge(v, u, 1);
103        g.addEdge(u, v, 1);
104    }
105    int ans = INF;
106    for(int i=1; i<=n; i++){
107        for(int j=1; j<=n; j++){
108            if(i != j)
109                ans = min(ans, g.dinic(i, j));
110        }
111    }
112
113    cout << ans << endl;
114 }
115
116 //***** Good Luck : ) *****/
117 int main() {
118     TIME(main);
119     IOS();
120     int t=1;
121     // cin >> t;
122     while(t--){
123         solve();
124     }
125
126     return 0;
127 }
```

43 KMP

```

1 int F[MAXN];
2 vector<int> match(auto A, auto B) {
3     const int Asz = A.size(), Bsz = B.size();
4     vector<int> ans {};
5     F[0] = -1, F[1] = 0;
6     for (int i = 1, j = 0; i < Bsz; F[++i] = ++j) {
7         if (B[i] == B[j]) F[i] = F[j]; // optimize
8         while (j != -1 and B[i] != B[j]) j = F[j];
9     }
10    for (int i = 0, j = 0; i < Asz; ++i) {
11        while (j != -1 and A[i] != B[j]) j = F[j];
12        if (++j
13            == Bsz) ans.emplace_back(i - j), j = F[j];
14    }
15    return ans;
}
```

44 Bipartite Matching

```

1 array<int, SZ> mp;
2 array<bool, SZ> vis;
3 bool dfs(int now) {
4     if (vis[now]) return false;
5     vis[now] = true;
6     for (int i = 0; i < n; i++) {
7         if (!G[now][i]) continue;
8         if (mp[i] == -1 or dfs(mp[i]))
9             return mp[i] = now, true;
10    }
11    return false;
12 }
13 int solve() {
14     mp.fill(-1);
15     int r = 0;
16     for (int i = 0; i < n; i++) {
17         vis.fill(false);
18         if (dfs(i)) r++;
19     }
20     return r;
21 }
```

45 Josephus

```

1 int josephus(int n, int k) {
2     int res = 0;
3     for (int i = 1; i <= n; ++i) res = (res + k) % i;
4     return res;
5 }
```

46 Undo Disjoint Set

```

1 struct DisjointSet
2 {
3     ^^I // save() is like recursive
4     ^^I // undo() is like return
5     int n, fa[MAXN], sz[MAXN];
6     vector<pair<int*, int>> h;
7     vector<int> sp;
8     void init(int tn) {
9         n = tn;
10        for (int i = 0; i < n; i++) sz[fa[i]] = 1;
11        sp.clear();
12        h.clear();
13    }
14    void assign(int *k, int v) {
15        h.PB({k, *k});
16        *k = v;
17    }
18    void save() {
19        sp.PB(SZ(h));
20    }
21    void undo() {
22        assert(!sp.empty());
23        int last = sp.back();
24        sp.pop_back();
25        while (SZ(h) != last)
26        {
27            auto x = h.back();
28            h.pop_back();
29            *x.F = x.S;
42 }
```

```

30     }
31 }
32
33 int f(int x) {
34     while (fa[x] != x) x = fa[x];
35     return x;
36 }
37
38 void uni(int x, int y) {
39     x = f(x);
40     y = f(y);
41     if (x == y) return;
42     if (sz[x] < sz[y]) swap(x, y);
43     assign(&sz[x], sz[x] + sz[y]);
44     assign(&fa[y], x);
45 }
46 } djs;

```

```

9     low = mid + 1;
10    else
11        high = mid - 1;
12    }
13    return -1;
14 }

```

47 Fast GCD

```

1 ll fast_gcd(ll x, ll y)
2 {
3     ll g = 1;
4     while (x && y)
5     {
6         const int c = __builtin_ctzll(x | y);
7         g <= c;
8         x >= c;
9         y >= c;
10        x >= __builtin_ctzll(x);
11        y >= __builtin_ctzll(y);
12        if (x < y) swap(x, y);
13        x -= y;
14    }
15
16    return g * (x + y);
17 }

```

48 CRT

```

1 ll solve_crt(ll x1, ll m1, ll x2, ll m2){
2     ll g = __gcd(m1, m2);
3     if ((x2 - x1) % g) return -1; // no sol
4     m1 /= g;
5     m2 /= g;
6     auto [pf, ps] = extgcd(m1, m2);
7     ll lcm = m1 / g * m2;
8     ll res = pf * (x2 - x1) * m1 + x1;
9     return (res % lcm + lcm) % lcm;
10 }

```

49 Binary Search

```

1 int binary_search(const
2     vector<int>& data, int key) {
3     int low = 0;
4     int high = data.size() - 1;
5     while (low <= high) {
6         int mid = int((low + high) / 2);
7         if (key == data[mid])
8             return mid;
9         else if (key > data[mid])

```