

## 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 REPl(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
27 #define endl '\n'
28 #define IOS()
29   ↳ ios_base::sync_with_stdio(0); cin.tie(0)
30
31 const ll MOD = 1000000007;
32 const ll INF = 0x3f3f3f3f3f3f3f3f;
33 const int iNF = 0x3f3f3f3f;
34 const ll MAXN = 100005;
35
36 void solve(){
37 }
38
39 /***** Good Luck :) *****/
40 int main() {
41     TIME(main);
42     IOS();
43     int t = 1;
44     cin >> t;
45     while(t--){
46         solve();
47     }
48     return 0;
49 }

```

## 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 int n, m;
13 vector<vector<pll>> w;
14 vector<ll> d;
15 vector<int> parent;
16
17 void dijkstra(int src){
18     // vector<bool> visit(n+1, false);
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 v = i.first;
33             ll alt = d[u] + i.second;
34             if(alt < d[v]){
35                 d[v] = alt;
36                 parent[v] = u;
37                 pq.push(mp(alt, v));
38             }
39         }
40     }
41 }

```

```

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     ll a, 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
        ↳ is no shortest path from vertex 1 to vertex n
58         cur = n;
59         while(cur != 1){
60             ans.push_back(cur);
61             cur = parent[cur];
62         }
63         ans.push_back(1);
64         int sz = ans.size();
65         for(int i=sz-1; i>=0; i--){
66             cout << ans[i] << " ";
67         }
68         cout << endl;
69     } else {
70         cout << -1 << endl;
71     }
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
28     return node;
29 }
30
31 void modify(Node*
    ↳ node, int L, int R, int i, int d){
32     if(L == R){
33         assert(L == i);
34         node->val += d;
35         return;
36     }
37     int mid = (L+R) >> 1;
38     if(i <= mid){
39         modify(node->lc, L, mid, i, d);
40     } else {
41         modify(node->rc, mid+1, R, i, d);
42     }
43     node->pull();
44 }
45
46 int query(Node*
    ↳ node, int L, int R, int ql, int qr){
47     if(ql > R || qr < L) return 0;
48     if(ql <= L && R <= qr) return node->val;
49     int mid = (L+R) >> 1;
50     return query(node->lc, L, mid, ql, qr) +
51         query(node->rc, mid+1, R, ql, qr);
52 }

```

#### 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 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*
    ↪ node, int L, int R, int ql, int qr, int d){
51     debug(L, R);
52     if(ql > R || qr < L) return;
53     if(ql <= L && R <= qr){
54         node->tag += d;
55         node->val += d * (R - L + 1);
56         return;
57     }
58     push(node, L, R);
59     int mid = (L + R) >> 1;
60     modify(node->lc, L, mid, ql, qr, d);
61     modify(node->rc, mid+1, R, ql, qr, d);
62     node->pull();
63 }
64
65 int query(Node*
    ↪ node, int L, int R, int ql, int qr){
66     if(ql > R || qr < L) return 0;
67     if(ql <= L && R <= qr) return node->val;
68     push(node, L, R);
69     int mid = (L + R) >> 1;
70     return query(node->lc, L, mid, ql,
    ↪ qr) + query(node->rc, mid+1, R, ql, qr);
71 }
72 /***** Good Luck :) *****/
73 int main() {
74     TIME(main);
75     IOS();
76     Node* root = build(1, n);
77     // cout << query(root, 1, n, 1, 5) << endl;
78     modify(root, 1, n, 1, 4, 3);
79
80     // cout << query(root, 1, n, 3, 3) << endl;
81
82     return 0;
83 }

```

## 6 Disjoint Set

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 const int n = 100;
5
6 vector<int> p(n);
7 vector<int> sz(n, 1);
8

```

```

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 }

```

## 7 Fast Power MOD

```

1 // C++ program to find
2 // (a^b)%m for b very large.
3 #include <bits/stdc++.h>
4 #define ll long long int
5 using namespace std;
6
7 // Function to find power
8 ll power(ll x, ll y, ll p)
9 {
10     ll res = 1; // Initialize result
11
12     // Update x if it is more than or
13     // equal to p
14     x = x % p;
15
16     while (y > 0) {
17         // If y is odd, multiply x with the result
18         if (y & 1)
19             res = (res * x) % p;
20
21         // y must be even now
22         y = y >> 1; // y = y/2
23         x = (x * x) % p;
24     }
25     return res;
26 }
27 // Driver Code
28 int main()
29 {
30     ll a = 3;
31
32     // String input as b is very large
33     string b = "1000000000000000000000000";
34
35     ll remainderB = 0;
36     ll MOD = 1000000007;

```

```

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;
46 }

```

## 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) {}
22 };
23
24 treap mempool[MAXN];
25 treap *ptr;
26 treap *gc; //
27 ⇨ use treap as linked list to garbage collect
28
29 inline void init() {
30     ptr=mempool;
31     gc=NULL;
32 }
33
34 inline void Del(treap *t) {
35     t->l=gc;
36     gc=t;
37 }
38
39 inline treap *New(int v) {
40     if (gc==NULL) {
41         *ptr=treap(v);
42         return ptr++;
43     } else {
44         treap *t=gc;
45         gc=gc->l;
46         *t=treap(v);
47         return t;
48     }
49 }
50
51 inline int size(treap *t) {
52     return t!=NULL?t->sz:0;
53 }
54
55 inline int small(treap *t) {
56     return t!=NULL?t->mn+t->add:INF;
57 }
58
59 inline void pull(treap *t) {
60     if (t==NULL)
61         return;
62     t->sz=1+size(t->l)+size(t->r);

```



```

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(mr, ml)), r);
216             break;
217         }
218     }
219     break;
220 }
221 return 0;
222 }

```

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

```

```

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     }

```

## 13 Mega inversion

```

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:

```



```

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     return res;
23 }
24 };
25
26 void solve(){
27     cin >> n;
28     a.resize(n);
29     for(auto &i: a) cin >> i;
30     BIT bit1, bit2;
31     vector<ll> ans1(MAXN, 0);
32     vector<ll> ans2(MAXN, 0);
33     for(int i=n-1; i>=0; i--){
34         ans1[i] = bit1.query(a[i]-1);
35         bit1.update(a[i], 1);
36     }
37     for(int i=0; i<n; i++){
38         ans2[i] = i - bit2.query(a[i]);
39         bit2.update(a[i], 1);
40     }
41     ll ans = 0;
42     for(int i=0; i<n; i++){
43         ans += ans1[i] * ans2[i];
44     }
45     cout << ans << endl;
46 }
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 }
20 int query(int x){
21     int ret = 0;
22     while(x){
23         ret += bit[x];
24         x -= x & (-x);
25     }
26     return ret;
27 }
28
29 int main() {
30     ios_base::sync_with_stdio(0);
31     cin.tie(0);
32     while(cin >> n){
33         if(n == 0) break;
34         v.clear();
35         ans = 0;
36         for(int i=0; i<n; i++){
37             cin >> a[i];
38             bit[i] = 0;
39             v.push_back(a[i]);
40         }
41         bit[n] = 0;
42         sort(v.begin(), v.end());
43         v.erase(unique(v.begin(),
44             ↪ v.end(), v.end()), v.end());
45         for(int i=0; i<n; i++){
46             ans += i - query(get_id(a[i]));
47             update(get_id(a[i]));
48         }
49         Case++;
50         cout << "Case #" << Case << ": " << ans << "\n";
51     }
52 }

```

## 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 // 份代點默點1始，即 i 0 [1, n]
2 int size[MAXN],
3 ↪ // 點的“大小”（所有子點+點）
4 weight[MAXN], // 點的“重量”
5 centroid[2]; // 用於的重心（存的是點）
6
7 void GetCentroid(int
8     ↪ cur, int fa) { // cur 表示前點 (current)
9     size[cur] = 1;
10    weight[cur] = 0;
11    for(int i = head[cur]; i != -1; i = e[i].nxt) {
12        if(e[i].to !=
13            ↪ fa) { // e[i].to 表示有向所通向的點。
14            GetCentroid(e[i].to, cur);
15            size[cur] += size[e[i].to];
16            weight[cur]
17            ↪ = max(weight[cur], size[e[i].to]);
18        }
19    }
20    weight[cur] = max(weight[cur], n - size[cur]);
21    if(weight[cur]
22        ↪ <= n / 2) { // 依照的重心的定
23        centroid[centroid[0] != 0] = cur;
24    }
25 }
```

## 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+1][j]=true;
22                 } else {
23                     dp[i+1][j]=dp[i][j];
24                 }
25             }
26         }
27     }
28     vector<int> ans;
29     for(int i=n-1, j=c; i>=0; i--){
30         if(has[i][j]){
31             ans.push_back(i);
32             j-=w[i];
33         }
34     }
35     cout << ans.size() << endl;
36     for(auto i: ans) cout << i << " ";
37     cout << endl;
38 }
39
40 /***** Good Luck :) *****/
41 int main() {
42     TIME(main);
43     IOS();
44     // cin >> t;
45     int c, n;
46     while(cin >> c >> n){
47         solve(c, n);
48     }
49     return 0;
50 }

```

## 20 有限背包

```

1  const int N=100, W=100000;
2  int cost[N], weight[N], number[N];
3  int c[W+1];
4
5  void knapsack(int n, int w)
6  {
7      for(int i=0; i<n; ++i)
8      {

```

```

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];
20 }

```

## 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=100000;
2  int cost[N], weight[N];
3  int c[W+1];
4
5  void knapsack(int n, int w)
6  {
7      memset(c, 0, sizeof(c));
8
9      for(int i=0; i<n; ++i)
10         for(int j=weight[i]; j<=w; ++j)
11             c[j]=max(c[j], c[j-weight[i]]+cost[i]);
12
13     cout << " 最高的價值為" << c[w];
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>&
    ↪ a, vector<int>&prev, int idx){
5     vector<int> tmp;
6     while(idx != -1){
7         tmp.push_back(a[idx]);
8         idx = prev[idx];
9     }
10    reverse(ALL(tmp));
11    for(auto i: tmp) cout<<i<<" ";
12    cout<<endl;
13 }
14
15 void solve(){
16     int n;
17     cin>>n;
18     if(n == 0) exit(0);
19     vector<int> a(n);
20     for(auto &i: a) cin>>i;
21     vector<int> dp(n, 1);
22     vector<int> prev(n, -1);
23     for(int i=0; i<n; i++){
24         for(int j=0; j<i; j++){
25             if(a[j] < a[i] and (dp[j] + 1 >
    ↪ dp[i] or (dp[j] + 1 == dp[i] and (prev[i]
    ↪ != -1 and a[prev[i]] > a[j])))){
26                 dp[i] = dp[j] + 1;
27                 prev[i] = j;
28             }
29         }
30     }
31     int ans = 0;
32     int ans_val = INF;
33     for(int i=0; i<n; i++){
34         if(ans < dp[i]
    ↪ or (ans == dp[i] and ans_val > a[i])){
35             ans = dp[i];
36             ans_val = a[i];
37         }
38     }
39     cout<<ans<<" ";
40     for(int i=0; i<n; i++){
41         if(dp[i] == ans and ans_val == a[i]){
42             print(a, prev, i);
43             break;
44         }
45     }

```

```

46 }
47
48 /***** Good Luck :) *****/
49 int main() {
50     TIME(main);
51     IOS();
52     // cin>>t;
53     while(true){
54         solve();
55     }
56
57     return 0;
58 }

```

### 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
    ↪ FOR(i, j, k, in) for (int i=j; i<(k); i+=in)
13 #define RFOR(i,
    ↪ j, k, in) for (int i=j; i>=(k); i-=in)
14 #define REP(i, j) FOR(i, 0, j, 1)
15 #define REPl(i, j) FOR(i, 1, j+1, 1)
16 #define RREP(i, j) RFOR(i, j, 0, 1)
17 #define ALL(_a) _a.begin(), _a.end()
18 #define mp make_pair
19 #define pb push_back
20 #define eb emplace_back
21 #define X first
22 #define Y second
23 #ifdef tmd
24 #define TIME(i) Timer i(#i)
25 #define debug( ... ) do{\
26     ↪ fprintf(stderr, "%s
    ↪ -
    ↪ %d
    ↪ (%s)
    ↪ =
    ↪ ", __PRETTY_FUNCTION__, __LINE__, #__VA_ARGS__);
    ↪ \
27     _do(__VA_ARGS__); \
28 }while(0)
29 template<typename
    ↪ T> void _do(T&&x){cerr<<x<<endl;}
30 template<typename
    ↪ T, typename ... S> void _do(T&&x, S
    ↪ &&... _t){cerr<<x<<" "; _do(_t ...);}
31 template<typename
    ↪ _a, typename _b> ostream& operator
    ↪ <<(ostream&_s, const pair<_a, _b>&p)
    ↪ {return _s<<"("<<p.X<<" "<<p.Y<<"");}
32 template<typename It>
    ↪ 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
    ↪ &operator<<(ostream& _s, vector<_a>
    ↪ &_c){return _OUTC(_s, ALL(_c));}
43 template<typename _a>
    ↪ ostream& operator<<(ostream& _s, set<_a>
    ↪ &_c){return _OUTC(_s, ALL(_c));}
44 template<typename _a>
    ↪ ostream& operator<<(ostream& _s, deque<_a>
    ↪ &_c){return _OUTC(_s, ALL(_c));}
45 template<typename _a, typename _b> ostream
    ↪ &operator<<(ostream& _s, map<_a, _b>
    ↪ &_c){return _OUTC(_s, ALL(_c));}
46 template<typename _t> void pary(_t
    ↪ _a, _t _b){_OUTC(cerr, _a, _b); cerr<<endl;}
47 #define IOS()
48 class Timer{
49 private:
50     string scope_name;
51     chrono::high_resolution_clock::time_point
    ↪ start_time;
52 public:
53     Timer(string name) : scope_name(name) {
54         start_time =
    ↪ chrono::high_resolution_clock::now();
55     }
56     ~Timer() {
57         auto stop_time =
    ↪ chrono::high_resolution_clock::now();
58
    ↪ auto
    ↪ length
    ↪ =
    ↪ chrono::duration_cast<chrono::microseconds>
    ↪ (-start_time).count();
59         double mlength = double(length) * 0.001;
60         debug(scope_name, mlength);
61     }
62 };
63 #else
64 #define TIME(i)
65 #define debug( ... )
66 #define pary( ... )
67 #define endl '\n'
68 #define IOS()
    ↪ ios_base::sync_with_stdio(0); cin.tie(0)
69 #endif
70
71 const ll MOD = 1000000007;
72 const ll INF = 0x3f3f3f3f3f3f3f3f;
73 const int iNF = 0x3f3f3f3f;
74 const ll MAXN = 100005;
75
76 mt19937 rng(chrono::steady_clock::now()
    ↪ .time_since_epoch().count());
77
78

```

```

79 /*
80 find solution of ax + by = gcd(a, b)
81
82 gcd(a, b) = gcd(b, a % b), therefore
83
84 bx + (a % b)y = gcd(b, a % b) = gcd(a, b) = ax + by
85 bx' + (a % b)y' = bx' + (a - floor(a/
    ↪ b) * b)y' = ay' + b(x' - floor(a/b)y') = ax + by
86
87 x = y', y = x' - floor(a/b) * y'
88 */
89
90 void exgcd(int a, int b, int& x, int& y){
91     if(b == 0){
92         x = 1; y = 0;
93         return;
94     }
95     int x1, y1;
96     exgcd(b, a % b, x1, y1);
97     x = y1;
98     y = x1 - (a / b) * y1;
99 }
100
101 // if x < 0, ans = (x + b) % b
102
103 /***** Good Luck :) *****/
104
105 int main() {
106     TIME(main);
107     IOS();
108
109     return 0;
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
18 double operator^(const Pt& p1, const Pt& p2){
19     return p1.X * p2.Y - p1.Y * p2.X;
20 }
21
22 int main(){
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 REPI(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
30   ↪ -
31   ↪ %d
32   ↪ (%s)
33   ↪ =
34   ↪ ", __PRETTY_FUNCTION__, __LINE__, #__VA_ARGS__)
35   ↪ \
36   ↪ _do(__VA_ARGS__); \
37 }while(0)
38 template<typename
39   ↪ T>void _do(T&&x){cerr<<x<<endl;}
40 template<typename
41   ↪ T, typename ... S>void _do(T&&x, S
42   ↪ && ... _t){cerr<<x<<" "; _do(_t ... );}
43 template<typename
44   ↪ _a, typename _b> ostream& operator
45   ↪ <<(ostream&s, const pair<_a, _b>&p)
46   ↪ {return s<<"("<<p.X<<" "<<p.Y<<"")<<endl;}
47 template<typename It>
48   ↪ ostream& _OUTC(ostream&s, It _ita, It _itb)
49 {
50   ↪ _s<<"{"<<endl;
51   ↪ for(It _it=_ita; _it!=_itb; _it++)
52   ↪ {
53   ↪   ↪ _s<<(_it==_ita?"":", ")<<* _it;
54   ↪ }
55   ↪ _s<<"}<<endl;
56   ↪ return _s;
57 }
58 template<typename _a> ostream
59   ↪ &operator<<(ostream&s, vector<_a>
60   ↪ &c){return _OUTC(s, ALL(c));}
61
62 template<typename _a>
63   ↪ ostream&operator<<(ostream&s, set<_a>
64   ↪ &c){return _OUTC(s, ALL(c));}
65 template<typename _a>
66   ↪ ostream&operator<<(ostream&s, deque<_a>
67   ↪ &c){return _OUTC(s, ALL(c));}
68 template<typename _a, typename _b> ostream
69   ↪ &operator<<(ostream&s, map<_a, _b>
70   ↪ &c){return _OUTC(s, ALL(c));}
71 template<typename _t> void pary(_t
72   ↪ _a, _t _b){_OUTC(cerr, _a, _b); cerr<<endl;}
73 #define IOS()
74 class Timer{
75 private:
76   ↪ string scope_name;
77   ↪ chrono::high_resolution_clock::time_point
78   ↪ start_time;
79 public:
80   ↪ Timer(string name): scope_name(name){
81   ↪   ↪ start_time =
82   ↪   ↪ chrono::high_resolution_clock::now();
83   ↪ }
84   ↪ ~Timer() {
85   ↪   ↪ auto stop_time =
86   ↪   ↪ chrono::high_resolution_clock::now();
87   ↪   ↪
88   ↪   ↪ auto
89   ↪   ↪ length
90   ↪   ↪ =
91   ↪   ↪ chrono::duration_cast<chrono::microseconds>(st
92   ↪   ↪ - start_time).count();
93   ↪   ↪ double mlength = double(length) * 0.001;
94   ↪   ↪ debug(scope_name, mlength);
95   ↪ }
96   ↪ };
97 #else
98 #define TIME(i)
99 #define debug( ... )
100 #define pary( ... )
101 #define endl '\n'
102 #define IOS()
103   ↪ ios_base::sync_with_stdio(0); cin.tie(0)
104 #endif
105
106 const ll MOD = 1000000007;
107 const ll INF = 0x3f3f3f3f3f3f3f3f;
108 const int inf = 0x3f3f3f3f;
109 const ll MAXN = 100005;
110
111 mt19937 rng(chrono::steady_clock::now()
112   ↪ .time_since_epoch().count());
113
114 int cross(pii o, pii a, pii b){
115   ↪ int u = a.X - o.X;
116   ↪ int v = a.Y - o.Y;
117   ↪ int s = b.X - o.X;
118   ↪ int t = b.Y - o.Y;
119
120   ↪ return u * t - s * v;
121 }
122
123 bool cmp(pii&a, pii&b){
124   ↪ return (a.X < b.X) || (a.X == b.X && a.Y < b.Y);
125 }

```

```

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
152

```

## 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
16     ↪ .tie(NULL);
17 #define FILE()
18     ↪ freopen("a.in", "r", stdin); freopen("out
19     ↪ .txt", "w", stdout);
20 #define endl '\n'
21 #define F first
22 #define S second
23 #define pb push_back
24 #define pf push_front
25 #define all(a) a.begin(), a.end()
26 #define rall(a) a.rbegin(), a.rend()
27 #define sz(a) (int)a.size()
28 #define FOR(i, a, b) for(int i = (a); i < (b); i++)
29 using vii = vector<vector<int>>;
30 using vi = vector<int>;
31 using mii = map<int, int>;
32 using cd = complex<double>;
33 typedef long long ll;
34 typedef unsigned long long ull;
35 typedef pair<int, int> pii;
36 typedef pair<long, long> pll;
37 const double PI = acos(-1);
38 const ll Mod = 1e9+7;
39 //const ll Inf = 2e18+9;
40 //const int N = 1e5+9;
41 //const int dx[4] = {0, 1, 0, -1};
42 //const int dy[4] = {1, 0, -1, 0};
43 //const int dx8[8] = {-1, -1, -1, 0, 1, 1, 1, 0};
44 //const int dy8[8] = {-1, 0, 1, 1, 1, 0, -1, -1};
45 ll gcd(ll a, ll
46     ↪ b){if(b==0) return a; return gcd(b, a%b);}
47 ll lcm(ll a, ll b){return a*b/gcd(a, b);}
48 int mul(int a, int b){return (1LL*a*b)%Mod;}
49 ll add(ll a, ll b){a+=b; if(a
50     ↪ >= Mod) a-=Mod; if(a<0) a+=Mod; return a;}
51 ll sub(ll a, ll
52     ↪ b){return (a-b)%Mod + ((a>=b)?0:Mod);}
53 //
54     ↪ -----
55
56 ll cross(pll a, pll b, pll c){
57     return (a.F-b.F)*(a.S-c.S)
58     ↪ - (a.S-b.S)*(a.F-c.F);
59 }
60
61 void solve(){
62     int n;
63     scanf("%d", &n);
64

```



```

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 &&
        ↪ cross(ans[sz(ans)-1], ans[sz(ans)-2],
        ↪ a[i]) <= 0) ans.pop_back();
69     ans.pb(a[i]);
70 }
71 for(int i=n-2, t=sz(ans)+1; i>=0; i--){
72     while(sz(ans)>=t &&
        ↪ cross(ans[sz(ans)-1], ans[sz(ans)-2],
        ↪ a[i]) <= 0) ans.pop_back();
73     ans.pb(a[i]);
74 }
75 if(sz(ans)>1) ans.pop_back();
76 a=ans;
77 n=sz(a);
78 ll best=0;
79 //FOR(i, 0, n) cout<<a[i].F<<
    ↪ '<<a[i].S<<endl;
80 for(int i=0; i<n-2; i++){
81     int k=i+1;
82     for(int j=i+2; j<n; j++){
83         ll area=abs(cross(a[i], a[k], a[j]));
84         while(1){
85             k++;
86             ll newarea=abs(cross(a[i], a[k], a[j]));
87             if(newarea <= area || k>=j) break;
88             area=newarea;
89         }
90         k=max(i+1, k-1);
91         best=max(area, best);
92     }
93 }
94 printf("%.5f\n", 0.5*best);
95 }
96
97 int main(){
98     INITIO()
99     //int t; cin>>t;
100    //while(t--){
101        solve();
102    //}
103    return 0;
104 }

```

## 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
    ↪ -a[i].F)*(a[k].S-a[i].S)*1.0 - (a[j].S
    ↪ -a[i].S)*(a[k].F-a[i].F)*1.0)/2.0;
5 }
6
7 bool judge(int a, int b, int c){
8     for(int i=0; i<n; i++){
9         if(i==a || i==b || i==c) continue;
10        double s=area(a, b, i) +
    ↪ area(a, c, i) + area(b, c, i) - area(a, b, c);
11        if(fabs(s) < 0.01) return false;
12    }
13    return true;
14 }
15
16 double d[55][55];
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
    ↪ = min(ans, max(dp(i, k), max(dp(k, j),
    ↪ area(i, k, j)))));
24     }
25     return d[i][j]=ans;
26 }
27
28 int main(){
29     ios_base::sync_with_stdio(false); cin
    ↪ .tie(NULL);
30     //freopen("a.in", "r", stdin); freopen("out
    ↪ .txt", "w", stdout);
31     int t; cin>>t;
32     while(t--){
33         cin>>n;
34         a.resize(n); FOR(i, 0, n) cin>>a[i]
    ↪ .F>>a[i].S;
35         FOR(i, 0, n) FOR(j, 0, n) d[i][j] = -1.0;
36         printf("%.1f\n", dp(0, n-1));
37     }
38     return 0;
39 }
40 }

```

## 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=10000000;
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 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 (logN 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    ↪ { return (a-*this).dot(b-*this); }
17    long long sqrLen()
18    ↪ { 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
32 bool pointInTriangle(pt a, pt b, pt c, pt point) {
33     long long s1=abs(a.cross(b, c));
34     long long s2=abs(point.cross(a,
35     ↪ b))+abs(point.cross(b,
36     ↪ c))+abs(point.cross(c, a));
37     return s1==s2;
38 }
39
40 void prepare(vector<pt> &points) {
41     n=points.size();
42     int pos=0;
43     for(int i=1;i<n;i++){
44         if(lexComp(points[i], points[pos]))
45             pos=i;
46     }
47     rotate(points.begin(),
48     ↪ points.begin()+pos, points.end());
49     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
46 bool pointInConvexPolygon(pt point) {
47     point=point-translation;
48     if(seq[0].cross(point)!=1 &&
49     ↪ sgn(seq[0].cross(point))
50     ↪ ↪ !=sgn(seq[0].cross(seq[n-1])))
51     ↪ return false;
52     if(seq[n-1].cross(point)!=0 &&
53     ↪ sgn(seq[n-1].cross(point))
54     ↪ ↪ !=sgn(seq[n-1].cross(seq[0])))
55     ↪ return false;
56
57     if(seq[0].cross(point)==0)
58     ↪ return seq[0].sqrLen()>=point.sqrLen();
59
60     int l=0, r=n-1;
61     while(r-l>1){
62         int mid=(l+r)/2;
63         int pos=mid;
64         if(seq[pos].cross(point)>=0)
65             l=mid;
66         else
67             r=mid;
68     }
69     int pos=l;
70     return pointInTriangle(seq[pos],
71     ↪ seq[pos+1], pt(0, 0), point);
72 }

```

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,llw;
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         //FOR(i,0,n) cout<<dis[i]<<endl;
29         return dis[ed] <= t;
30     }
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     cout<<ans<<endl;
65 }

```

## 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     int k=sz(a);
50     FOR(i, 0, sz(pok[id]))a.pb(1);
51     for(int i=0;i<pok[id].size();i++){
52         a[i+k]=0;
53         f(a, id+1);
54         a[i+k]=1;
55     }
56 }
57
58 void solve(){
59     cin>>n;
60     p.resize(n+1);
61     v.resize(n);
62     FOR(i, 0, n){
63         int r,c;cin>>r>>c;
64         string s;cin>>s;
65         if(mp.count(s)==0){
66             mp[s]=cnt++;
67         }
68         pok[mp[s]].pb(i);
69         v[i]={r,c};
70     }
71     p[0]={0,0};
72     for(int i=0,k=1;i<cnt;i++){
73         for(int j=0;j<pok[i].size();j++){
74             p[k++]=v[pok[i][j]];
75

```

```

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();

```

```

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 int get1(int now)
72 {
73     if(ans1[now]!=-1) return ans1[now];
74     int nowans=1;
75     for(int i=0;i<rG[now].size();i++)
76         if(cmp[rG[now][i]]!=cmp[now])
77         {
78             nowans=max(nowans,get2(rG[now][i])+1);
79         }
80     return ans1[now]=nowans;
81 }
82 int get2(int now)
83 {
84     if(ans2[now]!=-1) return ans2[now];
85     int nowans=-1;
86     for(int i=1;i<=cnum[cmp[now]];++i)
87     {
88         nowans=max(nowans,get1(c[cmp[now]]
89         ↳ [i])+dis[now][i]);
90     }
91     return ans2[now]=nowans;
92 }
93 int main()
94 {
95     scanf("%d%d",&n,&m);
96     int u,v;
97     for(int i=1;i<=m;++i)
98     {
99         scanf("%d%d",&u,&v);
100         add_edge(u,v);
101     }
102     scc();
103     memset(used,0,sizeof used);
104     for(int i=1;i<=n;++i)
105     {
106         ans1[i]=ans2[i]=-1;
107         c[cmp[i]][++cnum[cmp[i]]]=i;
108         id[i]=cnum[cmp[i]];

```

```

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 return 0;
115 }
116

```

### 37 Flight (Tree diameter/radius)

```

1 int n,vis[2505],dis[2505],
2   ↳ siz[2505],h1[2505],
3   ↳ h2[2505],c1[2505],c2[2505],p[2505];
4 vector<int> edge[2505];
5
6 pii treedia(int x){
7     for(int i=1;i<=n;i++){
8         vis[i]=0;
9         dis[i]=0;
10    }
11    queue<int> q;
12    q.push(x);
13    while(q.size()){
14        int top=q.front();
15        q.pop();
16        vis[top]=1;
17        for(auto it:edge[top]){
18            if(!vis[it]){
19                dis[it]=dis[top]+1;
20                q.push(it);
21            }
22        }
23    }
24    for(int i=1;i<=n;i++){
25        if(dis[x]<dis[i])x=i;
26    }
27    for(int i=1;i<=n;i++){
28        vis[i]=0;
29        dis[i]=0;
30    }
31    q.push(x);
32    while(q.size()){
33        int top=q.front();
34        q.pop();
35        vis[top]=1;
36        for(auto it:edge[top]){
37            if(!vis[it]){
38                dis[it]=dis[top]+1;
39                q.push(it);
40            }
41        }
42    }
43    int id=0;
44    for(int i=1;i<=n;i++){
45        if(dis[id]<dis[i])id=i;
46    }
47    int ans=0;
48    for(int i=1;i<=n;i++){
49        if(dis[i]
50        ↳ ==dis[id]/2+(dis[id]%2?1:0))ans=i;
51    }
52 }

```

```

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);

```

```

113      ed.pb({u, v});
114  }
115  int ans=1e9+5, a[4]={};
116  for(auto [u, v]: ed){
117      for(auto it= edge[u].begin(); it!
118          ↪ =edge[u].end(); it++){
119          if(*it == v){
120              edge[u].erase(it);
121              break;
122          }
123      }
124      for(auto it= edge[v].begin(); it!
125          ↪ =edge[v].end(); it++){
126          if(*it == u){
127              edge[v].erase(it);
128              break;
129          }
130      }
131      int k = treecentroid(u);
132      int j = treecentroid(v);
133      //cout<<k<<' '<<j<<endl;
134      edge[k].pb(j);
135      edge[j].pb(k);
136      int y = treedia(u).second;
137      if(ans > dis[y]){
138          ans = dis[y];
139          a[0] = u;
140          a[1] = v;
141          a[2] = k;
142          a[3] = j;
143      }
144      for(auto it= edge[k].begin(); it!
145          ↪ =edge[k].end(); it++){
146          if(*it == j){
147              edge[k].erase(it);
148              break;
149          }
150      }
151      for(auto it= edge[j].begin(); it!
152          ↪ =edge[j].end(); it++){
153          if(*it == k){
154              edge[j].erase(it);
155              break;
156          }
157      }
158      edge[u].pb(v);
159      edge[v].pb(u);
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     dfs(it);
11     siz[x] += siz[it];
12 }
13 }
14 }
15
16 int lca(int a, int b){
17     if(dep[a] > dep[b]) swap(a, b);
18     int u=a, v=b;
19     if(dep[a] != dep[b]){
20         int dif = dep[b] - dep[a];
21         for(int i=0; i<20; i++){
22             if(dif&1) b = p[i][b];
23             dif>>=1;
24         }
25     }
26     if(a==b) return dep[v]-dep[u];
27     for(int i=19; i>=0; i--){
28         if(p[i][a] != p[i][b]){
29             a = p[i][a];
30             b = p[i][b];
31         }
32     }
33     return dep[u]+dep[v]-2*dep[p[0][a]];
34 }
35
36 void solve(){
37     cin>>n>>m;
38     FOR(i, 0, n-1){
39         int u, v;
40         cin>>u>>v;
41         e[u].pb(v);
42         e[v].pb(u);
43     }
44     p[0][1]=1;
45     dfs(1);
46     for(int i=1; i<20; i++){
47         for(int j=1; j<=n; j++){
48             p[i][j] = p[i-1][p[i-1][j]];
49         }
50     }
51     while(m--){
52         int u, v; cin>>u>>v;
53         ll x = 1ll * lca(u, v) + 1;
54         cout<<x*(x+1)/2 + n - x<<endl;
55     }
56 }

```

## 39 Stogovi (LCA + DSU)

```

1 vector<int> e[N];
2 int n, m, dep[N]={}, siz[N]={}, p[20]
   ↳ [N]={}, f[N]={};
3
4 void dfs(int x){
5     siz[x] = 1;
6     for(auto it: e[x]){
7         dep[it] = dep[x] + 1;
8         dfs(it);
9         siz[x] += siz[it];
10    }
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
32 void prelca(){
33     dfs(0);
34     for(int i=1; i<20; i++){
35         for(int j=1; j<=n; j++){
36             p[i][j] = p[i-1][p[i-1][j]];
37         }
38     }
39 }
40
41 int find(int x){
42     if(x == f[x]) return x;
43     return f[x] = find(f[x]);
44 }
45
46 void solve(){
47     cin>>n;
48     FOR(i, 1, n+1) f[i] = i;
49     vector<pii> ans;
50     FOR(i, 1, n+1){
51         char c;
52         cin>>c;
53         int v, w;
54         if(c=='a'){
55             cin>>v;
56             v = find(v);
57             p[0][i] = v;
58             e[v].pb(i);
59         }
60         else if(c=='b'){
61             cin>>v;
62             v = find(v);
63             f[i] = p[0][v];
64             ans.pb({v, -1});
65         }
66         else {
67             cin>>v>>w;
68             v = find(v);
69             w = find(w);
70             f[i] = v;
71             ans.pb({v, w});
72         }
73     }
74     prelca();
75     for(auto [v, w]: ans){
76         if(w == -1){
77             cout<<v<<endl;
78         }

```

```

79     else{
80         int x = lca(v, w);
81         cout << dep[x] << endl;
82     }
83 }
84 }

```

## 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     }
28     if(x == y) return x;
29     for(int i=19; i>=0; i--){
30         if(p[x][i] != p[y][i]){
31             x = p[x][i];
32             y = p[y][i];
33         }
34     }
35     return p[x][0];
36 }
37
38 void solve(){
39     int n;
40     cin >> n;
41     adj.resize(n+1);
42     p.resize(n+1, vector<int>(20, 0));
43     d.resize(n+1);
44     int x, y;
45     for(int i=0; i<n-1; i++){
46         cin >> x >> y;
47         adj[x].pb(y);
48         adj[y].pb(x);
49     }
50     dfs(1, 0, 1);
51     for(int i=1; i<20; i++){
52         for(int j=1; j<=n; j++){
53             p[j][i] = p[p[j][i-1]][i-1];
54         }
55     }

```

```

56 }
57 ll ans = 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         ll dis = 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
12 vector<vector<Edge>> G(MAXN);
13 vector<bool> vis(MAXN, false);
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);
24     }
25 }

```

```

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     return 0;
72 }
73
74 int dinic(int s, int t){
75     for(int i=0; i<v; i++){
76         for(Edge& j: adj[i]){
77             j.flow = 0;
78         }
79     }
80     if(s == t) return -1;
81     int total = 0;
82     while(bfs(s, t)){
83         vector<int> idx(v, 0);
84         while(int flow = sendFlow(s, INF, t, idx)){
85             total += flow;
86         }
87     }
88     return total;
89 }

```

```

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;
16 }

```

### 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]=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;

```

```

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

```