Team Notebook

Kotlin Enjoyers - Universitas Indonesia

November 25, 2022

Contents

1 CRT

```cpp
#include <bits/stdc++.h>
using namespace std;
const int N = 20;
long long GCD(long long a, long long b) { return (b == 0) ? a : GCD(b, a % b); }
inline long long LCM(long long a, long long b) { return a / GCD(a, b) * b; }
inline long long normalize(long long x, long long mod) { x %= mod; if (x < 0) x += mod; return x; }
struct GCD_type { long long x, y, d; };
GCD_type ex_GCD(long long a, long long b)
{
    if (b == 0) return {1, 0, a};
    GCD_type pom = ex_GCD(b, a % b);
    return {pom.y, pom.x - a / b * pom.y, pom.d};
}
int testCases;
int t;
long long a[N], n[N], ans, lcm;

// format input :
//  x dan MOD

int main()
{
    ios_base::sync_with_stdio(0);
    cin.tie(0);
    cin >> t;
    for(int i = 1; i <= t; i++) cin >> a[i] >> n[i], normalize(a[i], n[i]);
    ans = a[1];
    lcm = n[1];
    for(int i = 2; i <= t; i++)
    {
        auto pom = ex_GCD(lcm, n[i]);
        int x1 = pom.x;
        int d = pom.d;
        if((a[i] - ans) % d != 0) return cerr << "No solutions" << endl, 0;
        ans = normalize(ans + x1 * (a[i] - ans) / d % (n[i] / d) * lcm, lcm * n[i] / d);
        lcm = LCM(lcm, n[i]); // you can save time by replacing above lcm * n[i] /d by lcm = lcm * n[i] / d
    }
    cout << ans << " " << lcm << endl;

    return 0;
}
```

2 Dinic

```cpp
struct FlowEdge {
    int v, u;
    long long cap, flow = 0;
    FlowEdge(int v, int u, long long cap) : v(v), u(u), cap(cap) {}
};

struct Dinic {
    const long long flow_inf = 1e18;
    vector<FlowEdge> edges;
    vector<vector<int>> adj;
    int n, m = 0;
    int s, t;
    vector<int> level, ptr;
    queue<int> q;

    Dinic(int n, int s, int t) : n(n), s(s), t(t) {
        adj.resize(n);
        level.resize(n);
        ptr.resize(n);
    }

    void add_edge(int v, int u, long long cap) {
        edges.emplace_back(v, u, cap);
        edges.emplace_back(u, v, 0);
        adj[v].push_back(m);
        adj[u].push_back(m + 1);
        m += 2;
    }

    bool bfs() {
        while (!q.empty()) {
            int v = q.front();
            q.pop();
            for (int id : adj[v]) {
                if (edges[id].cap - edges[id].flow < 1)
                    continue;
                if (level[edges[id].u] != -1)
                    continue;
                level[edges[id].u] = level[v] + 1;
                q.push(edges[id].u);
            }
        }
        return level[t] != -1;
    }

    long long dfs(int v, long long pushed) {
        if (pushed == 0)
            return 0;
        if (v == t)
            return pushed;
        for (int& cid = ptr[v]; cid < (int)adj[v].size(); cid++) {
            int id = adj[v][cid];
            int u = edges[id].u;
            if (level[v] + 1 != level[u] || edges[id].cap - edges[id].flow < 1)
                continue;
            long long tr = dfs(u, min(pushed, edges[id].cap - edges[id].flow));
            if (tr == 0)
                continue;
            edges[id].flow += tr;
            edges[id ^ 1].flow -= tr;
            return tr;
        }
        return 0;
    }

    long long flow() {
        long long f = 0;
        while (true) {
            fill(level.begin(), level.end(), -1);
            level[s] = 0;
            q.push(s);
            if (!bfs())
                break;
            fill(ptr.begin(), ptr.end(), 0);
            while (long long pushed = dfs(s, flow_inf)) {
                f += pushed;
            }
        }
        return f;
    }
};
```

3 Dynamic CHT

```cpp
const ll is_query = -(1LL<<62);
struct Line {
    ll m, b;
    mutable function<const Line*()> succ;
```

```
    bool operator<(const Line& rhs) const {
        if (rhs.b != is_query) return m < rhs.m;
        const Line* s = succ();
        if (!s) return 0;
        ll x = rhs.m;
        return b - s->b < (s->m - m) * x;
    }
};
struct HullDynamic : public multiset<Line> { // will
     maintain upper hull for maximum
    bool bad(iterator y) {
        auto z = next(y);
        if (y == begin()) {
            if (z == end()) return 0;
            return y->m == z->m && y->b <= z->b;
        }
        auto x = prev(y);
        if (z == end()) return y->m == x->m && y->b <= x->b;

        // **** May need long double typecasting here
        return (long double)(x->b - y->b)*(z->m - y->m) >= (
             long double)(y->b - z->b)*(y->m - x->m);
    }
    void insert_line(ll m, ll b) {
        auto y = insert({ m, b });
        y->succ = [=] { return next(y) == end() ? 0 : &*next(
             y); };
        if (bad(y)) { erase(y); return; }
        while (next(y) != end() && bad(next(y))) erase(next(y
             ));
        while (y != begin() && bad(prev(y))) erase(prev(y));
    }
    ll eval(ll x) {
        auto l = *lower_bound((Line) { x, is_query });
        return l.m * x + l.b;
    }
};
```

4 FFT

```
using cd = complex<double>;
const double PI = acos(-1);

void fft(vector<cd> & a, bool invert) {
    int n = a.size();
    if (n == 1)
        return;

    vector<cd> a0(n / 2), a1(n / 2);
    for (int i = 0; 2 * i < n; i++) {
        a0[i] = a[2*i];
        a1[i] = a[2*i+1];
    }
    fft(a0, invert);
    fft(a1, invert);

    double ang = 2 * PI / n * (invert ? -1 : 1);
    cd w(1), wn(cos(ang), sin(ang));
    for (int i = 0; 2 * i < n; i++) {
        a[i] = a0[i] + w * a1[i];
        a[i + n/2] = a0[i] - w * a1[i];
        if (invert) {
            a[i] /= 2;
            a[i + n/2] /= 2;
        }
        w *= wn;
    }
}

vector<int> multiply(vector<int> const& a, vector<int> const
     & b) {
    vector<cd> fa(a.begin(), a.end()), fb(b.begin(), b.end())
         ;
    int n = 1;
    while (n < a.size() + b.size())
        n <<= 1;
    fa.resize(n);
    fb.resize(n);

    fft(fa, false);
    fft(fb, false);
    for (int i = 0; i < n; i++)
        fa[i] *= fb[i];
    fft(fa, true);

    vector<int> result(n);
    for (int i = 0; i < n; i++)
        result[i] = round(fa[i].real());
    return result;
}
```

5 HLD

```
#include "bits/stdc++.h"
using namespace std;

const int N = 2e5+5;
const int D = 19;
const int S = (1<<D);

int n, q, v[N];
vector<int> adj[N];

int sz[N], p[N], dep[N];
int st[S], id[N], tp[N];

void update(int idx, int val) {
 st[idx += n] = val;
 for (idx /= 2; idx; idx /= 2)
  st[idx] = max(st[2 * idx], st[2 * idx + 1]);
}

int query(int lo, int hi) {
 int ra = 0, rb = 0;
 for (lo += n, hi += n + 1; lo < hi; lo /= 2, hi /= 2) {
  if (lo & 1)
   ra = max(ra, st[lo++]);
  if (hi & 1)
   rb = max(rb, st[--hi]);
 }
 return max(ra, rb);
}

int dfs_sz(int cur, int par) {
 sz[cur] = 1;
 p[cur] = par;
 for(int chi : adj[cur]) {
  if(chi == par) continue;
  dep[chi] = dep[cur] + 1;
  p[chi] = cur;
  sz[cur] += dfs_sz(chi, cur);
 }
 return sz[cur];
}

int ct = 1;

void dfs_hld(int cur, int par, int top) {
 id[cur] = ct++;
 tp[cur] = top;
 update(id[cur], v[cur]);
 int h_chi = -1, h_sz = -1;
 for(int chi : adj[cur]) {
  if(chi == par) continue;
  if(sz[chi] > h_sz) {
   h_sz = sz[chi];
```

```
   h_chi = chi;
  }
 }
 if(h_chi == -1) return;
 dfs_hld(h_chi, cur, top);
 for(int chi : adj[cur]) {
  if(chi == par || chi == h_chi) continue;
  dfs_hld(chi, cur, chi);
 }
}

int path(int x, int y){
 int ret = 0;
 while(tp[x] != tp[y]){
  if(dep[tp[x]] < dep[tp[y]])swap(x,y);
  ret = max(ret, query(id[tp[x]],id[x]));
  x = p[tp[x]];
 }
 if(dep[x] > dep[y])swap(x,y);
 ret = max(ret, query(id[x],id[y]));
 return ret;
}

// Tiap edge punya value.
// Query 1: ubah value suatu node
// Query 2: cari max value di path a ke b

int main() {
 scanf("%d%d", &n, &q);
 for(int i=1; i<=n; i++) scanf("%d", &v[i]);
 for(int i=2; i<=n; i++) {
  int a, b;
  scanf("%d%d", &a, &b);
  adj[a].push_back(b);
  adj[b].push_back(a);
 }
 dfs_sz(1, 1);
 dfs_hld(1, 1, 1);
 while(q--) {
  int t;
  scanf("%d", &t);
  if(t == 1) {
   int s, x;
   scanf("%d%d", &s, &x);
   v[s] = x;
   update(id[s], v[s]);
  } else {
   int a, b;
   scanf("%d%d", &a, &b);
   int res = path(a,b);
   printf("%d ", res);
  }
 }
}
```

6 Miller Rabin

```
using u64 = uint64_t;
using u128 = __uint128_t;

u64 binpower(u64 base, u64 e, u64 mod) {
    u64 result = 1;
    base %= mod;
    while (e) {
        if (e & 1)
            result = (u128)result * base % mod;
        base = (u128)base * base % mod;
        e >>= 1;
    }
    return result;
}

bool check_composite(u64 n, u64 a, u64 d, int s) {
    u64 x = binpower(a, d, n);
    if (x == 1 || x == n - 1)
        return false;
    for (int r = 1; r < s; r++) {
        x = (u128)x * x % n;
        if (x == n - 1)
            return false;
    }
    return true;
};

bool MillerRabin(u64 n, int iter=5) { // returns true if n
     is probably prime, else returns false.
    if (n < 4)
        return n == 2 || n == 3;

    int s = 0;
    u64 d = n - 1;
    while ((d & 1) == 0) {
        d >>= 1;
        s++;
    }

    for (int i = 0; i < iter; i++) {
        int a = 2 + rand() % (n - 3);
        if (check_composite(n, a, d, s))
            return false;
    }
    return true;
}
```

7 NTT

```
// TEMPLATE FFT/NTT AWOKWOK
const int mod = 998244353;

ll pang(ll x,ll y){
 if(x==0)return 0;
 if(y==0)return 1;
 if(y==1)return x;
 ll z=pang(x,y/2);
 return z*z%mod*pang(x,y%2)%mod;
}

const int root = pang(3,119);
const int root_1 = pang(root,mod-2);
const int root_pw = 1 << 23;

ll inv[300005],fact[300005],ifact[300005];

void fft(vector<ll> & a, bool invert) {
    int n = a.size();

    for (int i = 1, j = 0; i < n; i++) {
        int bit = n >> 1;
        for (; j & bit; bit >>= 1)
            j ^= bit;
        j ^= bit;

        if (i < j)
            swap(a[i], a[j]);
    }

    for (int len = 2; len <= n; len <<= 1) {
        int wlen = invert ? root_1 : root;
        for (int i = len; i < root_pw; i <<= 1)
            wlen = (int)(1LL * wlen * wlen % mod);

        for (int i = 0; i < n; i += len) {
            int w = 1;
            for (int j = 0; j < len / 2; j++) {
                int u = a[i+j], v = (int)(1LL * a[i+j+len/2] *
                      w % mod);
```

```
            a[i+j] = u + v < mod ? u + v : u + v - mod;
            a[i+j+len/2] = u - v >= 0 ? u - v : u - v +
                 mod;
            w = (int)(1LL * w * wlen % mod);
        }
    }
}

if (invert) {
    int n_1 = inv[n];
    for (ll & x : a)
        x = (int)(1LL * x * n_1 % mod);
}
}
```

8 Push Relabel

```
const int inf = 1000000000;

int n;
vector<vector<int>> capacity, flow;
vector<int> height, excess;

void push(int u, int v)
{
    int d = min(excess[u], capacity[u][v] - flow[u][v]);
    flow[u][v] += d;
    flow[v][u] -= d;
    excess[u] -= d;
    excess[v] += d;
}

void relabel(int u)
{
    int d = inf;
    for (int i = 0; i < n; i++) {
        if (capacity[u][i] - flow[u][i] > 0)
            d = min(d, height[i]);
    }
    if (d < inf)
        height[u] = d + 1;
}

vector<int> find_max_height_vertices(int s, int t) {
    vector<int> max_height;
    for (int i = 0; i < n; i++) {
        if (i != s && i != t && excess[i] > 0) {
            if (!max_height.empty() && height[i] > height[
                 max_height[0]])
                max_height.clear();
            if (max_height.empty() || height[i] == height[
                 max_height[0]])
                max_height.push_back(i);
        }
    }
    return max_height;
}

int max_flow(int s, int t)
{
    height.assign(n, 0);
    height[s] = n;
    flow.assign(n, vector<int>(n, 0));
    excess.assign(n, 0);
    excess[s] = inf;
    for (int i = 0; i < n; i++) {
        if (i != s)
            push(s, i);
    }

    vector<int> current;
    while (!(current = find_max_height_vertices(s, t)).empty
         ()) {
        for (int i : current) {
            bool pushed = false;
            for (int j = 0; j < n && excess[i]; j++) {
                if (capacity[i][j] - flow[i][j] > 0 && height[
                     i] == height[j] + 1) {
                    push(i, j);
                    pushed = true;
                }
            }
            if (!pushed) {
                relabel(i);
                break;
            }
        }
    }

    int max_flow = 0;
    for (int i = 0; i < n; i++)
        max_flow += flow[i][t];
    return max_flow;
}
```

9 String Automaton

```
struct state {
    int len, link;
    map<char, int> next;
};

const int MAXLEN = 100000;
state st[MAXLEN * 2];
int sz, last;

void sa_init() {
    st[0].len = 0;
    st[0].link = -1;
    sz++;
    last = 0;
}

void sa_extend(char c) {
    int cur = sz++;
    st[cur].len = st[last].len + 1;
    int p = last;
    while (p != -1 && !st[p].next.count(c)) {
        st[p].next[c] = cur;
        p = st[p].link;
    }
    if (p == -1) {
        st[cur].link = 0;
    } else {
        int q = st[p].next[c];
        if (st[p].len + 1 == st[q].len) {
            st[cur].link = q;
        } else {
            int clone = sz++;
            st[clone].len = st[p].len + 1;
            st[clone].next = st[q].next;
            st[clone].link = st[q].link;
            while (p != -1 && st[p].next[c] == q) {
                st[p].next[c] = clone;
                p = st[p].link;
            }
            st[q].link = st[cur].link = clone;
        }
    }
    last = cur;
}

// OP STRING ALGO AMORGOS
```

10 Treap

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <functional>

using namespace std;

typedef long long ll;
const ll LLINF = 2e16, LLBOUND = 2e15;

struct Node {
 ll val, mx, mn, mdiff;
 int size, priority;
 Node *l, *r;
 Node(ll _val) : val(_val), mx(_val), mn(_val), mdiff(LLINF)
      , size(1) {
  priority = rand();
 }
};

int size(Node *p) { return p == NULL ? 0 : p->size; }
ll getmax(Node *p) { return p == NULL ? -LLINF : p->mx; }
ll getmin(Node *p) { return p == NULL ? LLINF : p->mn; }
ll getmdiff(Node *p) { return p == NULL ? LLINF : p->mdiff;
    }
void update(Node *p) {
 if (p == NULL) return;
 p->size = 1 + size(p->l) + size(p->r);
 p->mx = max(p->val, max(getmax(p->l), getmax(p->r)));
 p->mn = min(p->val, min(getmin(p->l), getmin(p->r)));
 p->mdiff = LLINF;
 if (p->l != NULL)
  p->mdiff = min(p->mdiff, min(getmdiff(p->l), p->val -
       getmax(p->l)));
 if (p->r != NULL)
  p->mdiff = min(p->mdiff, min(getmdiff(p->r), getmin(p->r)
       - p->val));
}
void merge(Node *&t, Node *l, Node *r) {
 if (l == NULL)    { t = r; }
 else if (r == NULL) { t = l; }
 else if (l->priority > r->priority) {
  merge(l->r, l->r, r);
  t = l;
 } else {
  merge(r->l, l, r->l);
  t = r;
 }
 update(t);
}
void splitat(Node *t, Node *&l, Node *&r, int at) {
 if (t == NULL) { l = r = NULL; return; }
 int id = size(t->l);
 if (id > at) {
  splitat(t->l, l, t->l, at);
  r = t;
 } else {
  splitat(t->r, t->r, r, at - id - 1);
  l = t;
 }
 update(t);
}
ll Nquery(Node *t, int i, int j) {
 Node *l, *r;
 splitat(t, l, t, i - 1);
 splitat(t, t, r, j - i);
 ll ret = getmdiff(t);
 merge(t, l, t);
 merge(t, t, r);
 return (ret <= 0 || ret > LLBOUND ? -1 : ret);
}
ll Xquery(Node *t, int i, int j) {
 Node *l, *r;
 splitat(t, l, t, i - 1);
 splitat(t, t, r, j - i);
 ll ret = getmax(t) - getmin(t);
 merge(t, l, t);
 merge(t, t, r);
 return (ret <= 0 || ret > LLBOUND ? -1 : ret);
}
void split(Node *t, Node *&l, Node *&r, ll val) {
 if (t == NULL) { l = r = NULL; return; }
 if (t->val >= val) {
  split(t->l, l, t->l, val);
  r = t;
 } else {
  split(t->r, t->r, r, val);
  l = t;
 }
 update(t);
}
void insert(Node *&t, ll val) {
 Node *n = new Node(val), *l, *r;
 split(t, l, t, val);
 split(t, t, r, val + 1);
 merge(t, l, n);
 merge(t, t, r);
}
void erase(Node *&t, ll val, bool del = true) {
 Node *L, *rm;
 split(t, t, L, val);
 split(L, rm, L, val + 1);
 merge(t, t, L);
 if (del && rm != NULL) delete rm;
}
void inorder(Node *p) {
 if (p == NULL) return;
 inorder(p->l);
 cout << p->val << ' ';
 inorder(p->r);
}
void cleanup(Node *p) {
 if (p == NULL) return;
 cleanup(p->l); cleanup(p->r);
 delete p;
}

int main() {
 ios::sync_with_stdio(false);
 cin.tie(NULL);

 Node *tree = NULL;

 srand(time(NULL));

 int Q;
 cin >> Q;
 for (int q = 1; q <= Q; ++q) {
  char c;
  cin >> c;
  switch (c) {
   case 'I':
    ll k;
    cin >> k;
    insert(tree, k);
    break;
   case 'D':
    ll kd;
    cin >> kd;
    erase(tree, kd);
    break;
   case 'X':
    int l, r;
    cin >> l >> r;
    if (r - l < 1) cout << -1 << '\n';
    else cout << Xquery(tree, l, r) << '\n';
    break;
   case 'N':
```

```
    int ll, rr;
    cin >> ll >> rr;
    if (rr - ll < 1) cout << -1 << '\n';
    else cout << Nquery(tree, ll, rr) << '\n';
    break;
  }
//  cout << " ";
//  inorder(tree); cout << endl;
 }
 cout << flush;
 cleanup(tree);

 return 0;
}
```

11 bridgearticulation

```
int time;

void dfs(int u, int parent) {
    disc[u] = low[u] = time++;
    for (int v: adj[u]) {
        if (disc[v] == -1) {
            ++child[u];
            dfs(v, u);
            if (low[v] > disc[u]) {
                // (u, v) adalah bridge
            }
            if (low[v] >= disc[u]) {
                // u adalah articulation point
            }
            low[u] = min(low[u], low[v]);
        }
        else if (v != parent) {
            low[u] = min(low[u], disc[v]);
        }
    }
}

dfs(root, -1);
// Special case
if (child[root] < 2) {
    // root bukan articulation point
} else {
    // root adalah articulation point
}
```

12 centroid

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
#define ff first
#define ss second
#define pb push_back
const ll nax = 2e5 + 5;
const ll inf = 1e10;

ll n, m;
ll par[nax], removed[nax], sub[nax];
vector<vector<ll>> v(nax);

// Centroid

void get_sz(ll idx, ll bfr){
 sub[idx] = 1;
 for(auto y : v[idx]){
  if(y != bfr && !removed[y]){
   get_sz(y, idx);
   sub[idx] += sub[y];
  }
 }
}

ll find_centroid(ll idx){
 get_sz(idx, -1);
 ll tree = sub[idx];

 ll cek = 0;
 while(!cek){
  cek = 1;
  for(auto y : v[idx]){
   if(removed[y] || sub[y] > sub[idx]) continue;
   if(sub[y] > tree / 2){
    cek = 0;
    idx = y;
    break;
   }
  }
 }
 return idx;
}

void solve(ll idx){
  // Do smth here
}

ll built_centroid(ll idx){
 idx = find_centroid(idx);
  // Do smth here
  solve(idx);
 removed[idx] = 1;

 for(auto y : v[idx]){
  if(!removed[y]){
   ll nxt = built_centroid(y);
   par[nxt] = idx;
  }
 }
 return idx;
}

// Centroid

// Full Code Prob : CF 342E

int main(){
 ios_base::sync_with_stdio(false); cout.tie(NULL); cin.tie(
     NULL);

 cin >> n >> m;
 for(ll i = 1; i < n; i++){
  ll x, y;
  cin >> x >> y;
  v[x].pb(y);
  v[y].pb(x);
 }
 built_centroid(1);
}
```

13 closest$_{p}air$

```
long long ClosestPair(vector<pair<int, int>> pts) {
    int n = pts.size();
    sort(pts.begin(), pts.end());
    set<pair<int, int>> s;

    long long best_dist = 1e18;
    int j = 0;
    for (int i = 0; i < n; ++i) {
        int d = ceil(sqrt(best_dist));
        while (pts[i].first - pts[j].first >= d) {
            s.erase({pts[j].second, pts[j].first});
            j += 1;
        }
```

```
        auto it1 = s.lower_bound({pts[i].second - d, pts[i].
            first});
        auto it2 = s.upper_bound({pts[i].second + d, pts[i].
            first});

        for (auto it = it1; it != it2; ++it) {
            int dx = pts[i].first - it->second;
            int dy = pts[i].second - it->first;
            best_dist = min(best_dist, 1LL * dx * dx + 1LL *
                dy * dy);
        }
        s.insert({pts[i].second, pts[i].first});
    }
    return best_dist;
}
```

14 directed MST

```
/**
 * Author: chilli, Takanori MAEHARA, Benq, Simon Lindholm
 * Date: 2019-05-10
 * License: CC0
 * Source: https://github.com/spaghetti-source/algorithm/
     blob/master/graph/arborescence.cc
 * and https://github.com/bqi343/USACO/blob/42
     d177dfb9d6ce350389583cfa71484eb8ae614c/Implementations/
     content/graphs%20(12)/Advanced/DirectedMST.h for the
     reconstruction
 * Description: Finds a minimum spanning
 * tree/arborescence of a directed graph, given a root node.
      If no MST exists, returns -1.
 * Time: O(E \log V)
 * Status: Stress-tested, also tested on NWERC 2018
     fastestspeedrun
 */
#pragma once

#include "../data-structures/UnionFindRollback.h"

struct Edge { int a, b; ll w; };
struct Node { /// lazy skew heap node
 Edge key;
 Node *l, *r;
 ll delta;
 void prop() {
  key.w += delta;
  if (l) l->delta += delta;
  if (r) r->delta += delta;
  delta = 0;
 }
 Edge top() { prop(); return key; }
};
Node *merge(Node *a, Node *b) {
 if (!a || !b) return a ?: b;
 a->prop(), b->prop();
 if (a->key.w > b->key.w) swap(a, b);
 swap(a->l, (a->r = merge(b, a->r)));
 return a;
}
void pop(Node*& a) { a->prop(); a = merge(a->l, a->r); }

pair<ll, vi> dmst(int n, int r, vector<Edge>& g) {
 RollbackUF uf(n);
 vector<Node*> heap(n);
 for (Edge e : g) heap[e.b] = merge(heap[e.b], new Node{e});
 ll res = 0;
 vi seen(n, -1), path(n), par(n);
 seen[r] = r;
 vector<Edge> Q(n), in(n, {-1,-1}), comp;
 deque<tuple<int, int, vector<Edge>>> cycs;
 rep(s,0,n) {
  int u = s, qi = 0, w;
  while (seen[u] < 0) {
   if (!heap[u]) return {-1,{}};
   Edge e = heap[u]->top();
   heap[u]->delta -= e.w, pop(heap[u]);
   Q[qi] = e, path[qi++] = u, seen[u] = s;
   res += e.w, u = uf.find(e.a);
   if (seen[u] == s) { /// found cycle, contract
    Node* cyc = 0;
    int end = qi, time = uf.time();
    do cyc = merge(cyc, heap[w = path[--qi]]);
    while (uf.join(u, w));
    u = uf.find(u), heap[u] = cyc, seen[u] = -1;
    cycs.push_front({u, time, {&Q[qi], &Q[end]}});
   }
  }
  rep(i,0,qi) in[uf.find(Q[i].b)] = Q[i];
 }

 for (auto& [u,t,comp] : cycs) { // restore sol (optional)
  uf.rollback(t);
  Edge inEdge = in[u];
  for (auto& e : comp) in[uf.find(e.b)] = e;
  in[uf.find(inEdge.b)] = inEdge;
 }
 rep(i,0,n) par[i] = in[i].a;
 return {res, par};
}
```

15 fordfulkerson

```
LL bneck,adj[5005][5005],source,sink,ans=0,n;
bool visited[5005];

void dfs(LL node,LL bottleneck){
 if(node==sink){
  ans+=bottleneck;
  sudah=true;
  bneck=bottleneck;
  return;
 }
 if(!visited[node]){
  visited[node]=true;
  for(LL i=1;i<=n;i++){
   if(adj[node][i]>0){
    dfs(i,min(adj[node][i],bottleneck));
    if(sudah){
     adj[node][i]-=bneck;
     adj[i][node]+=bneck;
     return;
    }
   }
  }
 }
}

int main(){
 source=1,sink=n;
 sudah=true;
 while(sudah){
  memset(visited,false,sizeof(visited));
  sudah=false;
  dfs(source,1e18);
 }
 cout << ans << endl;
}
```

16 graham scan

```
/* Quick Note :
 * Jangan Mikir Lama - lama, sampahin dulu aja kalo OI
```

```
 * Always Try to reset
*/
#include <bits/stdc++.h>
using namespace std;
#define ff first
#define ss second
#define pb push_back
#define debug(val) cerr << "The value of " << #val << " is =
      " << val << '\n';
typedef long double ld;
typedef long long ll;
typedef unsigned long long ull;
const ll mod = 1e9 + 7;
const ll inf = 922337203685477;
const ll nax = 0;

struct point{
 ll x, y;
};

ll t, n;
vector<point> a;

ll cross(point p, point q, point r){
 ll val = (q.y - p.y) * (r.x - q.x) - (q.x - p.x) * (r.y - q
      .y);
 if(val == 0){
  return 0;
 }
 else if(val > 0){
  return 1;
 }
 else{
  return -1;
 }
}

ll dist(point p, point q){
 ll dx = p.x - q.x, dy = p.y - q.y;
 return dx * dx + dy * dy;
}

bool cmp(point p, point q){
 ll order = cross(a[0], p, q);
 if(order == 0){
  return dist(a[0], p) < dist(a[0], q);
 }
 else{
  return (order == -1);
 }
}

// Problem : 681 - Convex Hull Finding - UVA

int main(){
    ios_base::sync_with_stdio(false); cin.tie(NULL); cout.tie
         (NULL);
    //freopen("test.in", "r", stdin);
    //freopen("test.out", "w", stdout);

    cin >> t;
    cout << t << '\n';
    while(t--){
  a.clear();
  cin >> n;
  ll mini = 0;
  for(ll i = 0; i < n; i++){
   ll x, y;
   cin >> x >> y;
   a.pb({x, y});
   if(y < a[mini].y){
    mini = i;
   }
  }
  if(t){
   ll gbg;
   cin >> gbg;
  }
  // Jadiin satu titik sebagai titik acuan / pivot, titik
       yang dipakai adalah titik yang paling bawah
  swap(a[0], a[mini]);
  // Sort by polar angel
  sort(a.begin() + 1, a.end(), cmp);
  vector<point> v;
  for(ll i = 0; i < n; i++){
   if(v.size() < 2){
    v.pb(a[i]);
   }
   else{
    // Kalau Cross product nya tidak Counter Clockwise
         pop_back();
    while(v.size() >= 2 && cross(v[v.size()-2], v[v.size()
         -1], a[i]) != -1){
     v.pop_back();
    }
    v.pb(a[i]);
   }
  }
  cout << v.size() + 1 << '\n';
  for(auto p : v){
   cout << p.x << " " << p.y << '\n';
  }
  cout << a[0].x << " " << a[0].y << '\n';
  if(t){
   cout << "-1\n";
  }
 }
}
```

17 hungarian

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef long double ld;
#define pb push_back
#define ff first
#define ss second
const ld PI = 4 * atan((ld)1);
const ll nax = 25;
const ll inf = 1e16;

ll n;
ll ans;

ll dist(pair<ll,ll> x, pair<ll,ll> y){
 return abs(x.ff - y.ff) + abs(x.ss - y.ss);
}

ll hungarian(vector<pair<ll,ll>>&a, vector<pair<ll,ll>>&b){
  // pairing a ke b
 vector<ll> u(n + 1), v(n + 1), p(n + 1), way(n + 1);
 for(ll i = 1; i <= n; i++){
  p[0] = i;
  ll curM = 0;
  vector<ll> minv(n + 1, inf);
  vector<bool> used(n + 1, 0);
  while(p[curM] != 0){
   used[curM] = 1;
   ll curN = p[curM], delta = inf;
   ll nexM;
   for(int j = 1; j <= n; ++j){
    if(!used[j]){
     int cur = dist(a[curN-1], b[j-1]) - u[curN] - v[j];
     if(cur < minv[j]){
      minv[j] = cur, way[j] = curM;
     }
     if(minv[j] < delta){
```

```
      delta = minv[j], nexM = j;
     }
    }
   }
   for(int j = 0; j <= n; j++){
    if(used[j]){
     u[p[j]] += delta, v[j] -= delta;
    }
    else{
     minv[j] -= delta;
    }
   }
   curM = nexM;
  }
  do{
   ll nexM = way[curM];
   p[curM] = p[nexM];
   curM = nexM;
  }while(curM != 0);
 }
 return (-v[0]);
}

void make_diagonal(vector<pair<ll,ll>>&a){
 vector<pair<ll,ll>> b;
 for(ll i = 1; i <= n; i++){
  b.pb({i, i});
 }
 ans = min(ans, hungarian(a, b));
 b.clear();
 ll cnt = 1;
 for(ll i = n; i >= 1; i--){
  b.pb({cnt, i});
  cnt++;
 }
 ans = min(ans, hungarian(a, b));
}

void make_horizontal(vector<pair<ll,ll>> &a){
 vector<pair<ll,ll>> b;
 for(ll i = 1; i <= n; i++){
  for(ll j = 1; j <= n; j++){
   b.pb({i, j});
  }
  ans = min(ans, hungarian(a, b));
  b.clear();
 }
}

void make_vertical(vector<pair<ll,ll>> &a){
 vector<pair<ll,ll>> b;
 for(ll i = 1; i <= n; i++){
  for(ll j = 1; j <= n; j++){
   b.pb({j, i});
  }
  ans = min(ans, hungarian(a, b));
  b.clear();
 }
}

int main(){
 ios_base::sync_with_stdio(false); cin.tie(NULL); cout.tie(
     NULL);

 ll ct = 0;
 while(1){
  cin >> n;
  ans = inf;
  if(n == 0){
   break;
  }
  vector<pair<ll,ll>> a;
  for(ll i = 1; i <= n; i++){
   ll x, y;
   cin >> x >> y;
   a.pb({x, y});
  }
  make_diagonal(a);
  make_horizontal(a);
  make_vertical(a);
  cout << "Board " << ++ct << ": " << ans << " moves
      required." << "\n\n";
 }

}
```

18 segment tree lazy

```
/* Quick Note :
 * Jangan Mikir Lama - lama, sampahin dulu aja kalo OI
 * Always Try to reset
*/
#include <bits/stdc++.h>
using namespace std;
#define ff first
#define ss second
#define pb push_back
#define debug(val) cerr << "The value of " << #val << " is =
      " << val << '\n';
typedef long double ld;
typedef int ll;
typedef unsigned long long ull;
const ld PI = 4*atan((ld)1);
const ll mod = 1e9 + 7;
const ll inf = 1e9;
const ll nax = 1e6 + 5;

struct info{
 ll four, sev, inc, dec;
};

ll n, m;
ll prop[4*nax];
info seg[4*nax];
string s;

info merge(info x, info y){
 info ret;
 ret.four = x.four + y.four;
 ret.sev = x.sev + y.sev;
 ret.inc = max({x.four + y.four, x.sev + y.sev, x.four + y.
      inc, x.inc + y.sev});
 ret.dec = max({x.four + y.four, x.sev + y.sev, x.sev + y.
      dec, x.dec + y.four});
 return ret;
}

void rev(ll x){
 swap(seg[x].four, seg[x].sev);
 swap(seg[x].inc, seg[x].dec);
}

void lazy(ll x){
 if(prop[x]){
  rev(2*x), rev(2*x+1);
  prop[2*x] ^= 1, prop[2*x+1] ^= 1;
  prop[x] = 0;
 }
}

void built(ll l, ll r, ll pos){
 if(l == r){
  seg[pos] = {s[l-1] == '4', s[l-1] == '7', 1, 1};
 }
 else{
  ll mid = (l + r) / 2;
  built(l, mid, 2*pos);
```

```
  built(mid + 1, r, 2*pos+1);
  seg[pos] = merge(seg[2*pos], seg[2*pos+1]);
 }
}

void upd(ll l, ll r, ll pos, ll fl, ll fr){
 if(fl <= l && fr >= r){
  rev(pos);
  prop[pos] ^= 1;
 }
 else if(fl > r || fr < l){
  return;
 }
 else{
  lazy(pos);
  ll mid = (l + r) / 2;
  upd(l, mid, 2*pos, fl, fr);
  upd(mid + 1, r, 2*pos+1, fl, fr);
  seg[pos] = merge(seg[2*pos], seg[2*pos+1]);
 }
}
```

19 sos

```
//DP SOS (Sum over submask)
for(int i=0;i<m;i++){
 for(int mask=(1<<m)-1;mask>=0;mask--){
  if(mask & (1<<i))dp[mask]+=dp[mask^(1<<i)];
 }
}
```

20 suffix array

```
#include <bits/stdc++.h>
using namespace std;
#define ff first
#define ss second
#define pb push_back
#define debug(val) cerr << "The value of " << #val << " is =
      " << val << '\n';
typedef long double ld;
typedef long long ll;
typedef unsigned long long ull;
const ld PI = 4*atan((ld)1);
const ll mod = 1e9 + 7;
const ll inf = 922337203685477;
const ll nax = 5e5 + 5;

ll n;
ll sa[nax], ra[nax];
ll tempSA[nax], tempRA[nax];
ll freq_radix[nax];
string s;

void radixSort(ll k){
 ll maxi = max(300ll, n);
 memset(freq_radix, 0, sizeof(freq_radix));
 for(ll i = 0; i < n; i++){
  if(i + k < n){
   freq_radix[ra[i+k]]++;
  }
  else{
   freq_radix[0]++;
  }
 }
 ll sum = 0;
 for(ll i = 0; i < maxi; i++){
  ll temp = freq_radix[i];
  freq_radix[i] = sum;
  sum += temp;
 }
 for(ll i = 0; i < n; i++){
  ll temp = sa[i] + k;
  if(temp < n){
   tempSA[freq_radix[ra[temp]]++] = sa[i];
  }
  else{
   tempSA[freq_radix[0]++] = sa[i];
  }
 }
 for(ll i = 0; i < n; i++){
  sa[i] = tempSA[i];
 }
}

void builtSA(){
 for(ll i = 0; i < n; i++){
  ra[i] = s[i];
  sa[i] = i;
 }
 for(ll k = 1; k < n; k *= 2){
  radixSort(k);
  radixSort(0);
  tempRA[sa[0]] = 0;
  ll r = 0;
  for(ll i = 1; i < n; i++){
   if(ra[sa[i]] == ra[sa[i-1]] && ra[sa[i]+k] == ra[sa[i-1]+
        k]){
    tempRA[sa[i]] = r;
   }
   else{
    tempRA[sa[i]] = ++r;
   }
  }
  for(ll i = 0; i < n; i++){
   ra[i] = tempRA[i];
  }
  if (ra[sa[n-1]] == n-1) break; // nice optimization trick
 }
}

int main(){
    ios_base::sync_with_stdio(false); cin.tie(NULL); cout.tie
         (NULL);
    //freopen("test.in", "r", stdin);
    //freopen("test.out", "w", stdout);

    /*
 contoh input
 qwedasd

 contoh output
 asd: URUTAN KE 1
 d: URUTAN KE 2
 dasd: URUTAN KE 3
 edasd: URUTAN KE 4
 qwedasd: URUTAN KE 5
 sd: URUTAN KE 6
 wedasd: URUTAN KE 7
 */

    cin >> s;
    s += '$';
    n = s.size();
    builtSA();
    for(ll i = 1; i < n; i++){
  for(ll j = sa[i]; j < n - 1; j++){
   cout << s[j];
  }
  cout << ": URUTAN KE " << i << '\n';
 }
}
```

21 unionrectangle

```
struct Edge {
    bool open;
    int x, yMin, yMax;
    Edge(int x, int y1, int y2, bool op) {
        this->x = x;
        yMin = y1, yMax = y2;
        open = op;
    }
    bool operator < (const Edge &e) const {
        return (x < e.x);
    }
};

int m, h[maxN << 1];
int sum[maxN << 5], counter[maxN << 5];
vector<Edge> edges;

void update(int p, int l, int r, int yMin, int yMax, bool
     open) {
  if (h[r] < yMin || yMax < h[l]) return;
    int c = p << 1, mid = (l + r) >> 1;
    if (yMin <= h[l] && h[r] <= yMax) {     // ymin --- h[l]
         --- h[r] --- ymax
        counter[p] += open ? 1 : -1;
        if (counter[p]) sum[p] = h[r] - h[l]; //if there is a
             rectangle at that posn that is bw h[l] and h[r]
             we will add that to length
        else sum[p] = sum[c] + sum[c + 1]; // else we will
             just sumup of lengths above and beloew this
             region
        return;
    }
    if (l + 1 >= r) return;
    update(c, l, mid, yMin, yMax, open);
    update(c + 1, mid, r, yMin, yMax, open);
    if (counter[p]) sum[p] = h[r] - h[l];
    else sum[p] = sum[c] + sum[c + 1];
}

long long solve() {
                                // process height for horzntl.
                                    sweep line
    sort(h + 1, h + m + 1); //  Sorting the hieght according
         to the y coordinates
    int k = 1;
    for(int i=2;i<=m;i++) if (h[i] != h[k]) // Deleting the
         same horizontal sweeplines
       h[++k] = h[i];              //    as they are redundant
          m = k;

    for (int i = 0, lm = (int)edges.size() << 4; i < lm; i++)
         // This is the initialization step of segment tree
       sum[i] = 0, counter[i] = 0;

    long long area = 0LL; // Initializing the Area
    sort(edges.begin(),edges.end()); // Sorting according to
         x coordinates for ver. swp line
    update(1, 1, m, edges[0].yMin, edges[0].yMax, edges[0].
         open);
    for (int i = 1; i < edges.size(); i++) {
        area += sum[1] * (long long)(edges[i].x - edges[i -
             1].x);
        update(1, 1, m, edges[i].yMin, edges[i].yMax, edges[i
             ].open);
    }
    return area;
}

int main(){
    edges.pb(Edge(x1, y1, y2, true)); // Inserting the Left
         edge
    edges.pb(Edge(x2, y1, y2, false)); // Inserting the Right
          Edge
    /*(x1,y2)    (x2,y2)
                ---------
               |         |
   LeftEdge <- |         |-> Right Edge
               |_________|
            (x1,y1)    (x2,y1)
    */
    h[++m] = y1; // Inserting the Lower y Coordinate 1 based
         inddexiing
    h[++m] = y2; // Inserting the Upper y Coordinate
    solve();
}
```

22 xor 1 to n

```
int computeXOR(int n)
{

// If n is a multiple of 4
if (n % 4 == 0)
 return n;

// If n%4 gives remainder 1
if (n % 4 == 1)
 return 1;

// If n%4 gives remainder 2
if (n % 4 == 2)
 return n + 1;

// If n%4 gives remainder 3
return 0;
}
```