Topic
Dp, BIT, DSU, Computational geometry, Monotonic stack, LCA, FFT

Template

```cpp
#include <iostream>
#include <cmath>
#include <algorithm>
#include <vector>
#include <map>
#include <stack>
#include <set>
#include <queue>
#include <list>
#include <string.h>
#include <complex>
#include <sstream>
using namespace std;
#define INITIO()    ios_base::sync_with_stdio(false);cin.tie(NULL);
#define FILE()      freopen("in.txt","r",stdin);//freopen("out.txt","w",stdout);
#define endl        '\n'
#define F           first
#define S           second
#define pb          push_back
#define pf          push_front
#define all(a)      a.begin(),a.end()
#define rall(a)     a.rbegin(),a.rend()
#define sz(a)       (int)a.size()
#define FOR(i,a,b)  for(int i = (a); i < (b); i++)
using vii = vector<vector<int>>;
using vi = vector<int>;
using mii = map<int,int>;
using cd = complex<double>;
typedef long long ll;
typedef unsigned long long ull;
typedef pair<int,int> pii;
typedef pair<long,long> pll;
const double PI   = acos(-1);
const ll Mod     = 1e9+7;
const int  Inf     = 1e9+7;
const int N       = 2e5+9;
const int dy[4] = {1, 0, -1,  0};
const int dx[4] = {0, 1,  0, -1};
const int dx8[8] = {-1, -1, -1, 0, 1, 1, 1, 0};
const int dy8[8] = {-1, 0, 1, 1, 1, 0, -1, -1};
ll gcd(ll a, ll b){if( b==0 )return a;return gcd( b, a%b );}
ll lcm(ll a, ll b){return a * b / gcd(a, b);}
ll mul(ll a, ll b) {return (1LL * a * b) % Mod;}
ll add(ll a, ll b) {a += b;if (a >= Mod) a -= Mod;if (a < 0) a += Mod;return a;}
ll sub(ll a, ll b) {return (a - b) % Mod + ((a >= b) ? 0 : Mod);}
//
--------------------------------------------------------------------------------------------------

void solve(){

}

int main(){
   INITIO()
   //FILE()
   //int t;cin>>t;
   //while(t--){
       solve();
```

```
   //}
   return 0;
}
```

Binary Index Tree struct

```
BIT{
vector<ll>v;
void modify(ll x,ll val){for(;x<v.size();x+=(x&-x))v[x]+=val;}
ll query(ll x,ll y){
ll t=0,k=0;
for(;x;x-=(x&-x))t+=v[x];
for(;y;y-=(y&-y))k+=v[y];
return k-t;
}
};
```

Segment Tree

```
struct SegmentTree{
    vector<ll> tree,tag;
    void checklz(int l,int r,int v){
        tree[v*2]+=tag[v];
        tag[v*2]+=tag[v];
        tree[v*2+1]+=tag[v];
        tag[v*2+1]+=tag[v];
        tag[v]=0;
    }

    void build(int l,int r,int v=1){
        if(r==l){
            tree[v]=a[l];
            return;
        }
        int mid=(l+r)/2;
        build(l,mid,2*v);
        build(mid+1,r,2*v+1);
        tree[v]=min(tree[2*v],tree[2*v+1]);
    }

    ll query(int L,int R,int l,int r,int v=1){
        if (L > r || R < l || l > r)return 1e18;
        if (L <= l && r <= R)return tree[v];
        checklz(l, r, v);
        int mid=(l+r)/2;
        return min(query(L,R,l,mid,2*v), query(L,R,mid+1,r,2*v+1));
    }

    void update(int L,int R,int l,int r,ll val,int v=1){
        if (L > r || R < l || l > r)return;
        if(l>=L && r<=R){
            tag[v]+=val;
            tree[v]+=val;
            return;
        }
        int mid=(l+r)/2;
```

```
        checklz(l, r, v);
        update(L,R,l,mid,val,2*v);
        update(L,R,mid+1,r,val,2*v+1);
        tree[v] = min(tree[2*v], tree[2*v+1]);
    }
};
```

Binary search(Binary Lifting)

```
int ans=0, step= (1<<30);
while(step > 0){
   if(check(ans+step)) ans += step;
   step=(step>>1);
}
```

Binary Exponentiation

```
ll binpow(ll a, ll b) {
   if (b == 0) return 1;
   ll res = binpow(a, b / 2);
   if (b % 2) return res * res * a;
   else return res * res;
}
```

Modular Inverse

```
ll modInverse(ll a, ll m){
   ll m0 = m;
   ll y = 0, x = 1;
   if (m == 1) return 0;
   while (a > 1) {
       ll q = a / m;
       ll t = m;
       m = a % m, a = t;
       t = y;
       x = t;
   }
   if (x < 0)
   x += m0;
   return x;
}
```

Dijkstra

```
struct Edge{
   int v,w;
   bool operator < (const Edge &cmp) const { return cmp.w < w; }
};
vector<Edge> Graph[N]; int dis[N];
void dijkstra(int s){
   memset(dis,-1,sizeof(dis));
   priority_queue<Edge> pq;
   pq.push({s,0});
       while(pq.size()){
       auto node = pq.top(); pq.pop();
       if(dis[node.v] != -1) continue;
       dis[node.v] = node.w;
       for (auto k : Graph[node.v]){
           if(dis[k.v] == -1){
               pq.push({k.v, node.w + k.w});
           }
       }
   }
```

```
}
```

BellmanFord

```
struct Edge{
   int u,v,w;
};
vector<Edge> edgelist;
int n,dis[N];
void BellmanFord(int s){
   for(int i=0;i<N;i++)dis[i]=1e9+5;
   dis[s]=0;
   for(int i=0;i<n-1;i++){
       for(auto e : edgelist){
           dis[e.v] = min(dis[e.v], dis[e.u] + e.w);
       }
   }
}
```

Floyd Warshall

```
int dist[105][105] = {};
for(int k=0;k<100;k++)
   for(int i=0;i<100;i++)
           for(int j=0;j<100;j++)
               dist[i][j] = min(dist[i][j], dist[i][k] + dist[j][j]);
```

FFT

```
void fft(vector<cd> &a, bool invert){
   int n = sz(a);
   if(n == 1)return;
   vector<cd> a0(n/2),a1(n/2);
   for(int i=0;i*2<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;i*2<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;
   }
}

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

LCA

```
vector<int> e[N];
int n,m,dep[N]={},siz[N]={},p[20][N]={};

void dfs(int x){
    siz[x] = 1;
    for(auto it:e[x]){
        if(p[0][x]!=it){
            p[0][it] = x;
            dep[it] = dep[x] + 1;
            dfs(it);
            siz[x] += siz[it];
        }
    }
}

int lca(int a,int b){
    if(dep[a] > dep[b])swap(a,b);
    int u=a,v=b;
    if(dep[a] != dep[b]){
        int dif = dep[b] - dep[a];
        for(int i=0;i<20;i++){
            if(dif&1)b = p[i][b];
            dif>>=1;
        }
    }
    if(a==b)return dep[v]-dep[u];
    for(int i=19;i>=0;i--){
        if(p[i][a] != p[i][b]){
            a = p[i][a];
            b = p[i][b];
        }
    }
    return dep[u]+dep[v]-2*dep[p[0][a]];
}

void solve(){
    cin>>n>>m;
    FOR(i,0,n-1){
        int u,v;
        cin>>u>>v;
        e[u].pb(v);
        e[v].pb(u);
    }
    p[0][1]=1;
    dfs(1);
    for(int i=1;i<20;i++){
        for(int j=1;j<=n;j++){
            p[i][j] = p[i-1][p[i-1][j]];
        }
    }
    while(m--){
        int u,v;cin>>u>>v;
        ll x = 1ll*lca(u,v)+1;
        cout<<x*(x+1)/2 + n - x<<endl;
    }
}
```

Tree Centroid

```
#include <iostream> #include <vector>
using namespace std; const int maxn = 100005; int N;
vector <int> g[maxn]; int cost[maxn]; //慘度
int dfs(int now, int pre){ //tot: 以now為root的子樹size int tot = 1, ret = 0;
// now下方的子樹
for (auto nxt: g[now]){
if (nxt != pre){
ret = dfs(nxt, now);
tot += ret;
cost[now] = max(cost[now], ret);
} }
// now頭上的子樹
cost[now] = max(cost[now], N - tot); return tot;
}
int main() { ios_base::sync_with_stdio(0); cin.tie(0);
int T, a, b;
cin >> T;
while (T--){
cin >> N;
for (int i = 0; i < N; i++){
cost[i] = 0;
g[i].clear(); }
for (int i = 0; i < N-1; i++){ cin >> a >> b;
g[a].push_back(b);
g[b].push_back(a); }
dfs(0, -1); //將0當作root int mn = 0x7FFFFFFF;
int ans = -1;
for (int i = 0; i < N; i++){
if (cost[i] < mn){ mn = cost[i];
ans = i; }
}
cout << ans << "\n"; }
return 0; }
```

Number Theory Formula
Catalan number:
f(n) = f(n-1)*f(2) + f(3)*f(n-2) + ... + f(n-1)*f(2)
This formula can be used when counting the way of triangulation of a convex.

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

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

bool judge(int a,int b,int c){
    for(int i=0;i<n;i++){
        if(i==a || i==b || i==c)continue;
        double s = area(a,b,i) + area(a,c,i) + area(b,c,i) - area(a,b,c);
        if(fabs(s) < 0.01)return false;
    }
    return true;
}

double d[55][55];

double dp(int i,int j){
    if(i+1 >= j)return d[i][j]=0;
```

```
    if(d[i][j] != -1.0)return d[i][j];
    double ans=1e9*1.0;
    for(int k=i+1;k<j;k++){
        if(judge(i,j,k))ans = min(ans, max(dp(i,k),max(dp(k,j), area(i,k,j))));
    }
    return d[i][j] = ans;
}

int main(){
    ios_base::sync_with_stdio(false);cin.tie(NULL);
    //freopen("a.in","r",stdin);freopen("out.txt","w",stdout);
    int t;cin>>t;
    while(t--){
        cin>>n;
        a.resize(n);FOR(i,0,n)cin>>a[i].F>>a[i].S;
        FOR(i,0,n)FOR(j,0,n)d[i][j] = -1.0;
        printf("%.1f\n", dp(0,n-1));

    }
    return 0;
}
```

Optimal Matrix Chain Multiplication

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

int p[105];

int dp(int i, int j){
    if(j <= i+1) return 0;
    int mn = 10000000;
    for(int k=i+1;k<j;k++){
        int cost = dp(i,k) + dp(k,j) + p[i]*p[k]*p[j];
        mn = min(cost, mn);
    }
    return mn;
}

int main(){
    int n;
    cin>>n;
    for(int i=0;i<n;i++)cin>>p[i];
    cout<<dp(0,n-1)<<endl;
    return 0;
}
```

Cutting Stick

```
vector<int> a;
int n,len,f[55][55];

int dp(int l, int r){
    if(l+1 >= r)return f[l][r] = 0;
    if(f[l][r] != -1)return f[l][r];
    int mn=1e9;
    for(int k=l+1;k<r;k++){
        mn = min(mn, dp(l,k) + dp(k,r) + a[r]-a[l]);
    }
    return f[l][r] = mn;
}
```

Check if point belongs to the convex polygon (logN for each query)

```
struct pt {
```

```
    long long x, y;
    pt() {}
    pt(long long _x, long long _y) : x(_x), y(_y) {}
    pt operator+(const pt &p) const { return pt(x + p.x, y + p.y); }
    pt operator-(const pt &p) const { return pt(x - p.x, y - p.y); }
    long long cross(const pt &p) const { return x * p.y - y * p.x; }
    long long dot(const pt &p) const { return x * p.x + y * p.y; }
    long long cross(const pt &a, const pt &b) const { return (a - *this).cross(b - *this); }
    long long dot(const pt &a, const pt &b) const { return (a - *this).dot(b - *this); }
    long long sqrLen() const { return this->dot(*this); }
};

bool lexComp(const pt &l, const pt &r) {
    return l.x < r.x || (l.x == r.x && l.y < r.y);
}

int sgn(long long val) { return val > 0 ? 1 : (val == 0 ? 0 : -1); }

vector<pt> seq;
pt translation;
int n;

bool pointInTriangle(pt a, pt b, pt c, pt point) {
    long long s1 = abs(a.cross(b, c));
    long long s2 = abs(point.cross(a, b)) + abs(point.cross(b, c)) + abs(point.cross(c, a));
    return s1 == s2;
}

void prepare(vector<pt> &points) {
    n = points.size();
    int pos = 0;
    for (int i = 1; i < n; i++) {
        if (lexComp(points[i], points[pos]))
            pos = i;
    }
    rotate(points.begin(), points.begin() + pos, points.end());

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

bool pointInConvexPolygon(pt point) {
    point = point - translation;
    if (seq[0].cross(point) != 1 &&
            sgn(seq[0].cross(point)) != sgn(seq[0].cross(seq[n - 1])))
        return false;
    if (seq[n - 1].cross(point) != 0 &&
            sgn(seq[n - 1].cross(point)) != sgn(seq[n - 1].cross(seq[0])))
        return false;

    if (seq[0].cross(point) == 0)
        return seq[0].sqrLen() >= point.sqrLen();

    int l = 0, r = n - 1;
    while (r - l > 1) {
        int mid = (l + r) / 2;
        int pos = mid;
        if (seq[pos].cross(point) >= 0)
            l = mid;
        else
```

```
            r = mid;
    }
    int pos = l;
    return pointInTriangle(seq[pos], seq[pos + 1], pt(0, 0), point);
}
```

Finite Knapsack

```
const int N = 100, W = 100000;
int cost[N], weight[N], number[N];
int c[W + 1];

void knapsack(int n, int w)
{
    for (int i = 0; i < n; ++i)
    {
        int num = min(number[i], w / weight[i]);
        for (int k = 1; num > 0; k *= 2)
        {
            if (k > num) k = num;
            num -= k;
            for (int j = w; j >= weight[i] * k; --j)
                c[j] = max(c[j], c[j - weight[i] * k] + cost[i] * k);
        }
    }
    cout << "最高的價值為" << c[w];
}
```

Infinite Knapsack

```
const int N = 100, W = 100000;
int cost[N], weight[N];
int c[W + 1];

void knapsack(int n, int w)
{
    memset(c, 0, sizeof(c));

    for (int i=0; i<n; ++i)
        for (int j = weight[i]; j <= w; ++j)
            c[j] = max(c[j], c[j - weight[i]] + cost[i]);

    cout << "最高的價值為" << c[w];
}
```

HomeWork

Bungee Builder (monotonic stack)

```
void solve(){
   int n;cin>>n;
   vector<int> a(n);FOR(i,0,n)cin>>a[i];
   stack<pii> s;
   int ans=0;
   FOR(i,0,n){
       int bot=a[i];
       while(sz(s)){
           pii x = s.top();s.pop();
           bot = min(bot, x.S);
           if(x.F > a[i]){
               s.push({x.F,bot});
               ans=max(ans,a[i]-bot);
```

```
                break;
            }
            ans=max(ans,x.first-bot);
        }
        s.push({a[i],a[i]});
    }
    cout<<ans<<endl;
}
```

Arachnophobia (dijkstra + binary search)

```
int n,m,t,st,ed,k;
struct Edge{
    int v;ll w;
    bool operator<(const Edge &cmp) const {
        return cmp.w < w;
    }
};

vector<Edge> g[100005];
ll dis[100005],spiderdis[100005];

bool check(ll val){
    if(spiderdis[st] < val)return 0;
    if(spiderdis[ed] < val)return 0;
    FOR(i,0,n)dis[i]=Inf;
    priority_queue<Edge> pq;
    pq.push({st,0});
    while(pq.size()){
        auto node = pq.top(); pq.pop();
        if(dis[node.v] <= node.w)continue;
        dis[node.v] = node.w;
        for(auto it:g[node.v]){
            if(spiderdis[it.v] >= val && dis[it.v] > it.w + node.w){
                pq.push({it.v, it.w + node.w});
            }
        }
    }
    //FOR(i,0,n)cout<<dis[i]<<endl;
    return dis[ed] <= t;
}

void solve(){
    cin>>n>>m>>t;
    while(m--){
        int u,v,w;
        cin>>u>>v>>w;
        g[u].pb({v,w});
        g[v].pb({u,w});
    }
    cin>>st>>ed;
    cin>>k;
    priority_queue<Edge> pq;
    while(k--){
        int x;cin>>x;
        pq.push({x,0});
    }
    FOR(i,0,n)spiderdis[i]=Inf;
    while(pq.size()){
        auto node = pq.top(); pq.pop();
        if(spiderdis[node.v] <= node.w)continue;
        spiderdis[node.v] = node.w;
        for(auto it:g[node.v]){
```

```
            if(spiderdis[it.v] > it.w + node.w){
                pq.push({it.v,it.w + node.w});
            }
        }
    }
    ll ans=0,step= 1;
    FOR(i,0,62)step*=2;
    while(step > 0){
        if(check(ans+step)) ans += step;
        step=(step>>1);
    }
    cout<<ans<<endl;
}
```

Ascending Photo (Do some transition and DP)

```
void solve(){
    int n,cnt=0;cin>>n;
    vector<int> h;
    set<int> st;
    map<int,int> mp;
    FOR(i,0,n){
        int x;cin>>x;
        st.insert(x);
        if(h.empty() || h.back()!=x)h.pb(x);
    }
    for(auto it:st){
        if(!mp.count(it))mp[it] = cnt++;
    }
    n=sz(h);
    vector<vector<int>> pos(cnt);
    FOR(i,0,n){
        h[i] = mp[h[i]];
        pos[h[i]].pb(i);
    }
    pii best[2] = {{0,n}, {0,n}};
    FOR(i,0,cnt-1){
        pii nbest[2] = {best[0], best[1]};
        FOR(j,0,sz(pos[i])){
            int p = pos[i][j];
            if(p == n-1 || h[p]+1!=h[p+1])continue;
            pii s(0,n);
            if(p != best[0].second) s = best[0];
            else s = best[1];
            s.first++;
            s.second = p+1;
            if(pos[i+1].size() == 1) s.second = n;
            if(s > nbest[0]){
                nbest[1] = nbest[0];
                nbest[0] = s;
            }
            else if(s > nbest[1]) nbest[1] = s;
        }
        best[0] = nbest[0];
        best[1] = nbest[1];
    }
    cout<<n-1-best[0].first<<endl;
}
```

Increasing Subsequence (LIS + dp path tracking)

```
int a[205]={},dp[205]={},ps[205]={};
int n;
```

```
void solve(){
    for(int i=1;i<=n;i++){cin>>a[i];dp[i]=1;ps[i]=i;}
    for(int i=1;i<=n;i++){
        for(int j=i+1;j<=n;j++){
            if(a[j] > a[i]){
                if(dp[i]+1 > dp[j]){
                    dp[j] = dp[i]+1;
                    ps[j] = i;
                }
                else if(dp[i]+1 == dp[j] && a[i] < a[ps[j]]){
                    dp[j] = dp[i]+1;
                    ps[j] = i;
                }
            }
        }
    }
    vector<int> ans;
    for(int i=1;i<=n;i++){
        int idx=i;
        vector<int> v;
        while(1){
            v.pb(a[idx]);
            if(ps[idx]==idx)break;
            idx = ps[idx];
        }
        sort(all(v));
        if(v.size() > ans.size()){
            ans = v;
        }
        else if(v.size() == ans.size()){
            for(int j=0;j<sz(ans);j++){
                if(v[j] < ans[j]){
                    ans=v;
                    break;
                }
            }
        }
    }
    cout<<ans.size()<<' ';
    for(auto it:ans)cout<<it<<' ';
    cout<<endl;
}
```

Convex Hull

```
int n;
int cross(pii o,pii a,pii b){
    return (a.first - o.first)*(b.second - o.second) - (a.second - o.second)*(b.first - o.first);
}
void solve(){
    vector<pii> a,b;
    set<pii> s;
    FOR(i,0,n){
        int x,y;cin>>x>>y;
        if(s.count({x,y}))continue;
        a.pb({x,y});
        s.insert({x,y});
    }
    b=a;
    n=sz(a);
    sort(all(a));
    vector<pii> ans;
    FOR(i,0,n){
```

```
        while(sz(ans)>=2 && cross(ans[sz(ans)-1], ans[sz(ans)-2], a[i]) <= 0)ans.pop_back();
        ans.pb(a[i]);
    }
    for(int i=n-2, t=sz(ans)+1;i>=0;i--){
        while(sz(ans) >= t && cross(ans[sz(ans)-1], ans[sz(ans)-2], a[i]) <= 0)ans.pop_back();
        ans.pb(a[i]);
    }
    if(sz(ans)>1)ans.pop_back();
    cout<<ans.size()<<endl;
    reverse(all(ans));
    for(auto it:ans){
        cout<<it.F<<' '<<it.S<<endl;
    }
}
```

Largest Triangle (convex + monotonic stack)

```
ll cross(pll a,pll b,pll c){
    return (a.F-b.F)*(a.S-c.S) - (a.S-b.S)*(a.F-c.F);
}

void solve(){
    int n;
    scanf("%d", &n);
    vector<pll> a;
    set<pii> s;
    FOR(i,0,n){
        int x,y;
        scanf("%d %d", &x, &y);
        if(s.count({x,y}))continue;
        a.pb({x,y});
        s.insert({x,y});
    }
    n=sz(a);
    sort(all(a));
    vector<pll> ans;
    FOR(i,0,n){
        while(sz(ans)>=2 && cross(ans[sz(ans)-1], ans[sz(ans)-2], a[i]) <= 0)ans.pop_back();
        ans.pb(a[i]);
    }
    for(int i=n-2, t=sz(ans)+1;i>=0;i--){
        while(sz(ans) >= t && cross(ans[sz(ans)-1], ans[sz(ans)-2], a[i]) <= 0)ans.pop_back();
        ans.pb(a[i]);
    }
    if(sz(ans)>1)ans.pop_back();
    a = ans;
    n=sz(a);
    ll best=0;
    //FOR(i,0,n)cout<<a[i].F<<' '<<a[i].S<<endl;
    for(int i=0;i<n-2;i++){
        int k=i+1;
        for(int j=i+2;j<n;j++){
            ll area = abs(cross(a[i],a[k],a[j]));
            while(1){
                k++;
                ll newarea = abs(cross(a[i],a[k],a[j]));
                if(newarea <= area || k >= j)break;
                area = newarea;
            }
            k = max(i+1, k-1);
            best = max(area, best);
        }
    }
```

```
    printf("%.5f\n",0.5*best);
}
```

Pokemongogo (TSP)

```
int n,ans,cnt=0 , g[22][22]={}, dp[1<<22][22];
map<string,int> mp;
vector<int> pok[22];
vector<pii> p, v;

int dfs(int i,int j){
    if(dp[i][j] != -1)return dp[i][j];
    if(i == (1 << (n+1))-1 && j==0)return dp[i][j]=0;
    int res = 1e9+5;
    for(int k=0;k<=n;k++){
        if(!((i >> k) & 1)){
            res= min(res, dfs((i | (1 << k)), k) + g[j][k]);
        }
    }
    /*
     int b = i;
     while(b){
         cout<<(b&1);
         b/=2;
     }
     cout<<' '<<j<<' ';
     cout<<res<<endl;
    */

    return dp[i][j] = res;
}

void f(vector<int> a, int id){
    if(a.size() == n+1){

        for(int i=0;i<=n;i++){
            if(a[i])continue;
            a[i] = 1;

            int cn=0;
            for(int i=0, k=1;i<sz(a);i++, k*=2)
                cn += a[i] * k;
            if(dp[cn][i]!=-1){
                ans = min(ans, dp[cn][i] + g[0][i]);
                //for(int i=0;i<sz(a);i++)cout<<a[i];
                //cout<<' '<<i<<' ';
                //cout<<dp[cn][i]<<' ';
                //cout<<endl;
            }
            a[i]=0;
        }
        //cout<<endl;

    }
    int k = sz(a);
    FOR(i,0,sz(pok[id]))a.pb(1);
    for(int i=0;i < pok[id].size();i++){
        a[i+k]=0;
        f(a, id+1);
        a[i+k]=1;
    }
}
```

```
void solve(){
    cin>>n;
    p.resize(n+1);
    v.resize(n);
    FOR(i,0,n){
        int r,c;cin>>r>>c;
        string s;cin>>s;
        if(mp.count(s)==0){
            mp[s] = cnt++;
        }
        pok[mp[s]].pb(i);
        v[i] = {r,c};
    }
    p[0] = {0,0};
    for(int i=0,k=1;i<cnt;i++){
        for(int j=0;j<pok[i].size();j++){
            p[k++] = v[pok[i][j]];
        }
    }
    FOR(i,0,n+1){
        FOR(j,0,n+1){
            g[i][j] = abs(p[i].F-p[j].F) + abs(p[i].S-p[j].S);
            //cout<<g[i][j]<<' ';
        }
        //cout<<endl;
    }
    memset(dp, -1, sizeof(dp));
    dfs(0,0);
    ans=1e9+5;
    f({0}, 0);
    cout<<ans<<endl;
}
```

British Menu (Dp + SCC) Copy from others

```
#include <cmath>
#include <cstdio>
#include <cstring>
#include <iostream>
#include <algorithm>
#include <vector>
#define MAX_V 200005
using namespace std;
int n,ans,ans1[MAX_V],ans2[MAX_V],c[MAX_V][6],cnum[MAX_V],id[MAX_V],dis[MAX_V][6],m;
vector<int> G[MAX_V];
vector<int> rG[MAX_V];
vector<int> vs;
bool used[MAX_V];
int cmp[MAX_V];
void add_edge(int from,int to)
{
    G[from].push_back(to);
    rG[to].push_back(from);
}
void dfs(int v)
{
    used[v]=1;
    for(int i=0;i<G[v].size();++i)
    {
        if(!used[G[v][i]])
            dfs(G[v][i]);
```

```
    }
    vs.push_back(v);
}
void rdfs(int v,int k)
{
    used[v]=1;
    cmp[v]=k;
    for(int i=0;i<rG[v].size();++i)
    {
        if(!used[rG[v][i]])
        rdfs(rG[v][i],k);
    }
}
int scc()
{
    memset(used,0,sizeof used);
    vs.clear();
    for(int v=1;v<=n;v++)
    {
        if(!used[v]) dfs(v);
    }
    memset(used,0,sizeof used);
    int k=1;
    for(int i=vs.size()-1;i>=0;--i)
    {
        if(!used[vs[i]])
        rdfs(vs[i],k++);
    }
    return k;
}
void cal(int now,int nowid,int f,int nowdis)
{
    used[now]=1;
    dis[now][nowid]=max(dis[now][nowid],nowdis);
    for(int i=0;i<G[now].size();++i)
    if(cmp[G[now][i]]==f&&!used[G[now][i]])
    {
        cal(G[now][i],nowid,f,nowdis+1);
    }
    used[now]=0;
    return;
}
int get2(int now);
int get1(int now)
{
    if(ans1[now]!=-1) return ans1[now];
    int nowans=1;
    for(int i=0;i<rG[now].size();i++)
    if(cmp[rG[now][i]]!=cmp[now])
    {
        nowans=max(nowans,get2(rG[now][i])+1);
    }
    return ans1[now]=nowans;
}
int get2(int now)
{
    if(ans2[now]!=-1) return ans2[now];
    int nowans=-1;
    for(int i=1;i<=cnum[cmp[now]];++i)
    {
        nowans=max(nowans,get1(c[cmp[now]][i])+dis[now][i]);
    }
    return ans2[now]=nowans;
```

```
}
int main()
{
    scanf("%d%d",&n,&m);
    int u,v;
    for(int i=1;i<=m;++i)
    {
        scanf("%d%d",&u,&v);
        add_edge(u,v);
    }
    scc();
    memset(used,0,sizeof used);
    for(int i=1;i<=n;++i)
    {
        ans1[i]=ans2[i]=-1;
        c[cmp[i]][++cnum[cmp[i]]]=i;
        id[i]=cnum[cmp[i]];
        cal(i,id[i],cmp[i],0);
    }
    for(int i=1;i<=n;++i)
    {
        ans=max(ans,get2(i));
    }
    cout<<ans;
    return 0;
}
```

Flight (Tree diameter/radius)

```
int n,vis[2505],dis[2505], siz[2505], h1[2505], h2[2505],c1[2505],c2[2505],p[2505];
vector<int> edge[2505];

pii treedia(int x){
    for(int i=1;i<=n;i++){
        vis[i]=0;
        dis[i]=0;
    }
    queue<int> q;
    q.push(x);
    while(q.size()){
        int top = q.front();
        q.pop();
        vis[top]=1;
        for(auto it:edge[top]){
            if(!vis[it]){
                dis[it] = dis[top]+1;
                q.push(it);
            }
        }
    }
    for(int i=1;i<=n;i++){
        if(dis[x] < dis[i])x = i;
    }
    for(int i=1;i<=n;i++){
        vis[i]=0;
        dis[i]=0;
    }
    q.push(x);
    while(q.size()){
        int top = q.front();
        q.pop();
        vis[top]=1;
```

```
        for(auto it:edge[top]){
            if(!vis[it]){
                dis[it] = dis[top]+1;
                q.push(it);
            }
        }
    }
    int id=0;
    for(int i=1;i<=n;i++){
        if(dis[id] < dis[i])id=i;
    }
    int ans=0;
    for(int i=1;i<=n;i++){
        if(dis[i] == dis[id]/2 + (dis[id]%2?1:0))ans = i;
    }
    if(dis[id] == 0)ans = x;
    return {ans, id};
}

void record(int x, int height, int child)
{
    if (height > h1[x])
    {
        h2[x] = h1[x]; c2[x] = c1[x];
        h1[x] = height; c1[x] = child;
    }
    else if (height > h2[x])
    {
        h2[x] = height; c2[x] = child;
    }
}

void dfs1(int x){
    h1[x] = h2[x] = 0;
    for(auto it:edge[x]){
        if(p[x] != it){
            p[it] = x;
            dfs1(it);
            record(x,h1[it]+1,it);
        }
    }
}

void dfs2(int x){
    if(p[x] != x){
        int y = p[x];
        if(c1[y] == x)record(x,h2[y]+1,y);
        else record(x,h1[y]+1,y);
    }
    for(auto it:edge[x]){
        if(it != p[x]){
            dfs2(it);
        }
    }
}

int treecentroid(int x){
    int ans=x;
    FOR(i,1,n+1) h1[i] = h2[i] = c1[i] = c2[i] = p[i] = 0;
    p[x] = x;
    dfs1(x);
    dfs2(x);
    for(int i=1;i<=n;i++){
```

```
        //cout<<h1[i]<<' ';
        if(h1[i] && h1[i] < h1[ans]){
            ans = i;
        }
    }
    //cout<<endl;
    return ans;
}

void solve(){
    cin>>n;
    vector<pii> ed;
    FOR(i,0,n-1){
        int u,v;cin>>u>>v;
        edge[u].pb(v);
        edge[v].pb(u);
        ed.pb({u,v});
    }
    int ans=1e9+5, a[4]={};
    for(auto [u,v]:ed){
        for(auto it = edge[u].begin();it!=edge[u].end();it++){
            if(*it == v){
                edge[u].erase(it);
                break;
            }
        }
        for(auto it = edge[v].begin();it!=edge[v].end();it++){
            if(*it == u){
                edge[v].erase(it);
                break;
            }
        }
        int k = treecentroid(u);
        int j = treecentroid(v);
        //cout<<k<<' '<<j<<endl;
        edge[k].pb(j);
        edge[j].pb(k);
        int y = treedia(u).second;
        if(ans > dis[y]){
            ans = dis[y];
            a[0] = u;
            a[1] = v;
            a[2] = k;
            a[3] = j;
        }
        for(auto it = edge[k].begin();it!=edge[k].end();it++){
            if(*it == j){
                edge[k].erase(it);
                break;
            }
        }
        for(auto it = edge[j].begin();it!=edge[j].end();it++){
            if(*it == k){
                edge[j].erase(it);
                break;
            }
        }
        edge[u].pb(v);
        edge[v].pb(u);
    }
    cout<<ans<<endl;
    cout<<a[0]<<' '<<a[1]<<endl;
    cout<<a[2]<<' '<<a[3]<<endl;
```

```
}
```

Rooted Subtree (Combinatorics + LCA)

```
vector<int> e[N];
int n,m,dep[N]={},siz[N]={},p[20][N]={};

void dfs(int x){
    siz[x] = 1;
    for(auto it:e[x]){
        if(p[0][x]!=it){
            p[0][it] = x;
            dep[it] = dep[x] + 1;
            dfs(it);
            siz[x] += siz[it];
        }
    }
}

int lca(int a,int b){
    if(dep[a] > dep[b])swap(a,b);
    int u=a,v=b;
    if(dep[a] != dep[b]){
        int dif = dep[b] - dep[a];
        for(int i=0;i<20;i++){
            if(dif&1)b = p[i][b];
            dif>>=1;
        }
    }
    if(a==b)return dep[v]-dep[u];
    for(int i=19;i>=0;i--){
        if(p[i][a] != p[i][b]){
            a = p[i][a];
            b = p[i][b];
        }
    }
    return dep[u]+dep[v]-2*dep[p[0][a]];
}

void solve(){
    cin>>n>>m;
    FOR(i,0,n-1){
        int u,v;
        cin>>u>>v;
        e[u].pb(v);
        e[v].pb(u);
    }
    p[0][1]=1;
    dfs(1);
    for(int i=1;i<20;i++){
        for(int j=1;j<=n;j++){
            p[i][j] = p[i-1][p[i-1][j]];
        }
    }
    while(m--){
        int u,v;cin>>u>>v;
        ll x = 1ll*lca(u,v)+1;
        cout<<x*(x+1)/2 + n - x<<endl;
    }
}
```

Stogovi (LCA + DSU)

```cpp
vector<int> e[N];
int n,m,dep[N]={},siz[N]={},p[20][N]={},f[N]={};

void dfs(int x){
    siz[x] = 1;
    for(auto it:e[x]){
        dep[it] = dep[x] + 1;
        dfs(it);
        siz[x] += siz[it];
    }
}

int lca(int a,int b){
    if(dep[a] > dep[b])swap(a,b);
    if(dep[a] != dep[b]){
        int dif = dep[b] - dep[a];
        for(int i=0;i<20;i++){
            if(dif&1)b = p[i][b];
            dif>>=1;
        }
    }
    if(a==b)return a;
    for(int i=19;i>=0;i--){
        if(p[i][a] != p[i][b]){
            a = p[i][a];
            b = p[i][b];
        }
    }
    return p[0][a];
}

void prelca(){
    dfs(0);
    for(int i=1;i<20;i++){
        for(int j=1;j<=n;j++){
            p[i][j] = p[i-1][p[i-1][j]];
        }
    }
}

int find(int x){
    if(x == f[x])return x;
    return f[x] = find(f[x]);
}

void solve(){
    cin>>n;
    FOR(i,1,n+1)f[i]=i;
    vector<pii> ans;
    FOR(i,1,n+1){
        char c;
        cin>>c;
        int v,w;
        if(c=='a'){
            cin>>v;
            v = find(v);
            p[0][i] = v;
            e[v].pb(i);
        }
        else if(c=='b'){
            cin>>v;
            v = find(v);
            f[i] = p[0][v];
```

```
            ans.pb({v,-1});
        }
        else {
            cin>>v>>w;
            v = find(v);
            w = find(w);
            f[i] = v;
            ans.pb({v,w});
        }
    }
    prelca();
    for(auto [v,w]:ans){
        if(w == -1){
            cout<<v<<endl;
        }
        else{
            int x = lca(v,w);
            cout<<dep[x]<<endl;
        }
    }
}
```

Midterm pC (Monotonic stack)

```
void solve(){
    int n;
    cin>>n;
    vector<int> h(n),w(n);
    FOR(i,0,n){cin>>h[i];h[i]=h[i]*2/3;}
    FOR(i,0,n)cin>>w[i];
    w.pb(0);
    h.pb(0);
    ll ans=0,cnt=0;
    stack<pll> s;
    s.push({0,0});
    FOR(i,0,n+1){
        cnt+=w[i];
        ll r = s.top().second;
        while(h[i] < s.top().first){
            pll tmp = s.top();
            s.pop();
            ans = max(ans, (r-s.top().S)*tmp.first);
        }
        s.push({h[i], cnt});
    }
    cout<<ans<<endl;
}
```

Midterm pE (DSU)

```
int q,n,f[N], st[N];
ll cnt[N]={}, siz[N], tot=0, mx=0;
vector<pii> qey;
vector<ll> ans;

int find(int x){
    if(x == f[x]) return x;
    return f[x] = find(f[x]);
}

void unio(int x,int y){
    x = find(x);y = find(y);
```

```
    if(x != y){
        f[x] = y;
        cnt[y] += cnt[x];
        siz[y] += siz[x];
    }
}

void solve(){
    cin>>n>>q;
    FOR(i,0,n){
        cin>>cnt[i];
        st[i] = 1;
        siz[i] = 1;
        f[i] = i;
    }
    while(q--){
        string c;
        cin>>c;
        if(c == "D"){
            int v;cin>>v;
            st[v] = 0;
            qey.pb({0,v});
        }
        else if(c == "QM")qey.pb({1,0});
        else qey.pb({2,0});
    }

    ll sum=0, k=0;
    FOR(i,0,n){
        if(st[i]){
            sum += cnt[i];
            k++;
        }
        else {
            tot += sum * k;
            sum=0;
            k=0;
        }
    }
    tot += sum*k;

    FOR(i,1,n){
        if(st[i-1] && st[i])unio(i-1,i);
    }

    FOR(i,0,n){
        if(st[i])mx = max(mx, cnt[i] * siz[i]);
    }

    reverse(all(qey));
    vector<ll> ans;
    for(auto [ty, x]: qey){
        if(ty == 0){
            if(x+1 < n && st[x+1]){
                int g= find(x+1);
                tot -= cnt[g] * siz[g];
                unio(x+1, x);
            }
            if(x-1 >=0 && st[x-1]){
                int g= find(x-1);
                tot -= cnt[g] * siz[g];
                unio(x-1, x);
            }
```

```
            st[x]=1;
            int fr = find(x);
            tot += cnt[fr] * siz[fr];
            mx = max(mx ,cnt[fr] * siz[fr]);
        }
        else if(ty == 1){
            ans.pb(mx);
        }
        else {
            ans.pb(tot);
        }
    }
    reverse(all(ans));
    for(auto x:ans)cout<<x<<endl;
}
```

Boxes (LCA)

```
vector<int> e[N];
int n,dep[N]={},siz[N]={},p[20][N]={};

void dfs(int x,int d){
    siz[x] = 1;
    dep[x] = d+1;
    for(auto it:e[x]){
        dfs(it,d+1);
        siz[x] += siz[it];
    }
}

int lca(int a,int b){
    if(dep[a] > dep[b])swap(a,b);
    if(dep[a] != dep[b]){
        int dif = dep[b] - dep[a];
        for(int i=0;i<20;i++){
            if(dif&1)b = p[i][b];
            dif>>=1;
        }
    }
    if(a==b)return a;
    for(int i=19;i>=0;i--){
        if(p[i][a] != p[i][b]){
            a = p[i][a];
            b = p[i][b];
        }
    }
    return p[0][a];
}

void solve(){
    cin>>n;
    FOR(i,1,n+1){
        cin>>p[0][i];
        e[p[0][i]].pb(i);
    }
    dfs(0,0);
    for(int i=1;i<20;i++){
        for(int j=1;j<=n;j++){
            p[i][j] = p[i-1][p[i-1][j]];
        }
    }
    int m;cin>>m;
    while(m--){
```

```
        int t;cin>>t;
        vector<int> a,b(t,1);
        while(t--){
            int x;cin>>x;
            a.pb(x);
        }
        ll ans=0;
        for(int i=0;i<sz(a);i++){
            for(int j=0;j<sz(a);j++){
                if(i!=j && lca(a[i],a[j]) == a[j]){
                    b[i]=0;
                    break;
                }
            }
            if(b[i])ans+=1ll*siz[a[i]];
        }
        cout<<ans<<endl;
    }
}
```

Mega Inversion (BIT)

```
struct BIT{
    vector<ll> a;
    void modify(ll x,ll val){
        for(;x<a.size();x+=(x&-x))a[x]+=val;
    }
    ll query(ll x,ll y){
        ll cnt1=0,cnt2=0;
        for(;y;y-=(y&-y))cnt2+=a[y];
        for(;x;x-=(x&-x))cnt1+=a[x];
        return cnt2-cnt1;
    }
};

void solve(){
    int n;cin>>n;
    vector<int> a(n);
    FOR(i,0,n)cin>>a[i];
    BIT tr1,tr2;
    tr1.a.resize(n+1,0);
    tr2.a.resize(n+1,0);
    ll ans=0;
    for(int i=0;i<n;i++){
        ans+=tr2.query(a[i],n);
        tr2.modify(a[i],tr1.query(a[i],n));
        tr1.modify(a[i],1);
    }
    cout<<ans<<endl;
}
```