

- 1 動態規劃
 - 1.1 背包問題
 - 1.2 Bitmask DP
 - 1.3 硬幣
 - 1.4 編輯距離
 - 1.5 LCS
 - 1.6 LIS
 - 1.7 Projects
 - 1.8 Removal Game

4.12 Prim	8
4.13 SCC 跟拓撲 DP	8
4.14 狀態 Dijkstra	9
4.15 Vis Dijkstra	9
數學	9
5.1 質因數分解	9
5.2 盧卡斯定理	10
Queries	10
6.1 BIT	10
6.2 Increasing Array Queries	10
6.3 線段樹	11
6.4 懶標線段樹	11
6.5 莫隊	12
6.6 Treap	12
搜尋與貪心	13
7.1 二分搜	13
7.2 Concert Ticket	13
7.3 Restaurant Customers	13
字串演算法	13
8.1 KMP	13
樹論	14
9.1 LCA	14
9.2 子樹 DP	14
9.3 樹重心	14
9.4 節點距離總和	14
9.5 無權樹直徑	14
9.6 有權樹直徑	15

1.1 背包問題

```
#include <bits/stdc++.h>
using namespace std;
int dp[1005][100005];
vector<int> Page(1005, 0);
vector<int> Price(1005, 0);
int main(){
    int n, bud;
    cin >> n >> bud;
    for(int i = 1; i <= n; i++){
        int tmp; cin >> tmp;
        Price[i] = tmp;
    }
    for(int i = 1; i <= n; i++){
        int tmp; cin >> tmp;
        Page[i] = tmp;
    }
    for(int i = 1; i <= n; i++){
        for(int j = 1; j <= bud; j++){
            if(j >= Price[i]){
                dp[i][j] = max(dp[i-1][j], dp[i-1][j-Price[i]]+Page[i]);
            }
            else {
                dp[i][j] = dp[i-1][j];
            }
        }
    }
    cout << dp[n][bud] << endl;
}
```

```
#include <bits/stdc++.h>
using namespace std;
// Bit_Mask_DP, Travel Exactly Once
int dp[(1 << 20) - 1][20];
vector<int> rev_adj[20];
int n, m;
const int mod = 1e9 + 7;
void solve(){
    cin >> n >> m;
    for(int i = 0; i < m; i++){
        int u, v; cin >> u >> v;
        rev_adj[-v].push_back(-u);
    }
    dp[1][0] = 1;
    for(int road = 0; road < (1 << n); road++){
        // Not include 1
        if(road & 1 == 0) continue;
        // include n but not all walked
```

```

if(road & (1 << (n
    - 1))) && road != ((1 << n) - 1)) continue;
// DP
for (int end = 0; end < n; end++) {
    // Not include end
    if ((road & (1 << end)) == 0) continue;
    // exclude end point is last road
    int pre_road = road - (1 << end);
    for (int pre_road_end : rev_adj[end]) {
        // pre_road_end is prev's end
        if ((road & (1 << pre_road_end))) {
            dp[road][end] += dp[pre_road][pre_road_end];
            dp[road][end] %= mod;
        }
    }
}
}
}
cout << dp[(1 << n) - 1][n - 1];
// elevator rides
// for(int i = 1; i < 1 << n; i++){
//     used[i] = dp[i] = inf;
//     for(int j = 0; j < n; j++){
//         if(i & (1 << j)){ // 有j
//             int last = i ^ (1 << j);
//             if(used[last] + s[j] <= x){
//                 if(dp[last] < dp[i] || dp[
last] == dp[i] && used[last] + s[j] < used[i]){
//                     used[i] = used[last] + s[j];
//                     dp[i] = dp[last];
//                 }
//             }
//         }
//     }
// }
// else {
//     if(dp[last] + 1 < dp[
i] || dp[last] + 1 == dp[i] && s[j] < used[i]){
//         used[i] = s[j];
//         dp[i] = dp[last] + 1;
//     }
// }
// }
// }
// }
// cout << dp[(1 << n) - 1];
}

```

```
#include <bits/stdc++.h>
using namespace std;
// combine
// arrange: nested loop exchange
int dp[2][1000001];
const int mod = 1e9 + 7;
void solve(){
    int n, x; cin >> n >> x;
    vector<int> coin(n + 1);
    for(int i = 1; i <= n; i++){
        cin >> coin[i];
    }
    dp[0][0] = 1;
    for(int i = 1; i <= n; i++){
        for(int j = 0; j <= x; j++){
            dp[i & 1][j] = dp[!(i & 1)][j];
            if(j >= coin[i]){
                (dp[i & 1][j]
                 += dp[i & 1][j - coin[i]]) %= mod;
            }
        }
    }
    cout << dp[n & 1][x];
}
// Minimize coins nums
void solve(){
    int n, x; cin >> n >> x;
    vector<int> coin(n);
    for(int i = 0; i < n; i++){
        cin >> coin[i];
    }
    int dp[x+1]; // init(dp, 0);
    dp[0] = 0;
    for(int i = 1; i <= x; i++){
        dp[i] = 2e18;
        for(auto &j : coin){
            if(j <= i){
                dp[i] = min(dp[i], dp[i - j] + 1);
            }
        }
    }
}
```

```

    }
    cout << (dp[x] == 2e18 ? -1 : dp[x]);
}

```

1.4 編輯距離

```

#include <bits/stdc++.h>
using namespace std;
int dp[1005][1005];
void solve(){
    string s1, s2; cin >> s1 >> s2;
    int size1 = s1.size(), size2 = s2.size();
    s1 = "0" + s1, s2 = "0" + s2;
    for(int i = 1; i <= size2; i++){
        dp[0][i] = i; // s2 = {}, s1 = ...;
    }
    for(int i = 1; i <= size1; i++){
        dp[i][0] = i; // s1 = {}, s2 = ...;
    }
    for(int i = 1; i <= size1; i++){
        for(int j = 1; j <= size2; j++){
            if(s1[i] == s2[j]){
                dp[i][j] = dp[i-1][j-1];
            }
            else {
                dp[i][j] = min(min(dp[i-1][j], dp[i][j-1]) + 1,
                                // modify
                                // s1 del / s2 add
                                // s1 add s2 del
                                dp[i-1][j] + 1, dp[i][j-1] + 1);
            }
        }
    }
    cout << dp[size1][size2];
}

```

1.5 LCS

```

#include <bits/stdc++.h>
using namespace std;
int main(){
    int m, n; cin >> m >> n;
    string s1, s2;
    cin >> s1 >> s2;
    s1.insert(s1.begin(), '1');
    s2.insert(s2.begin(), '1');
    int L = 0;
    vector<vector<int>> dp(m + 1, vector<int>(n + 1, 0));

    for(int i = 1; i <= m; i++){
        for(int j = 1; j <= n; j++){
            if(s1[i] == s2[j]){
                dp[i][j] = dp[i-1][j-1] + 1;
            }
            else {
                dp[i][j] = max(dp[i-1][j], dp[i][j-1]);
            }
        }
    }
    int length = dp[m][n];
    cout << length << "\n";
    vector<char> s(length);
    // along to dp to trace back
    while(m >= 1 && n >= 1){
        if(s1[m] == s2[n]){
            s[length - 1] = s1[m];
            m--;
            n--;
            length--;
        }
        else {
            if(dp[m-1][n] > dp[m][n-1]){
                m--;
            }
            else n--;
        }
    }
    for(auto c : s){
        cout << c;
    }
}

```

1.6 LIS

```

#include <bits/stdc++.h>
using namespace std;
// Rec Sequence LIS

```

```

void solve(){
    int n; cin >> n;
    vector<int> v(n);
    for(int i = 0; i < n; i++){
        cin >> v[i];
    }
    int dp[n]; vector<int> mono;
    mono.push_back(v[0]);
    dp[0] = 1; int L = 1;
    for(int i = 1; i < n; i++){
        if(v[i] > mono.back()){
            mono.push_back(v[i]);
            dp[i] = ++L;
        }
        else {
            auto it = lower_bound(
                mono.begin(), mono.end(), v[i]);
            *it = v[i];
            dp[i] = it - mono.begin() + 1;
        }
    }
    vector<int> ans;
    cout << L << endl;
    for(int i = n - 1; i >= 0; i--){
        if(dp[i] == L){
            ans.push_back(v[i]);
            L--;
        }
    }
    reverse(ans.begin(), ans.end());
    for(auto i : ans){
        cout << i << " ";
    }
}

```

1.7 Projects

```

#include <bits/stdc++.h>
using namespace std;
#define int long long
#define pii pair<int, int>
const int maxn = 2e5 + 5;
typedef struct {
    int u, v, w;
} project;
void compress(vector<int> &sorted, vector<project> &projects, vector<vector<pii>> &EndProjects){
    sort(sorted.begin(), sorted.end());
    sorted.erase(unique(sorted.begin(), sorted.end()), sorted.end());
    for(int i = 0; i < projects.size(); i++){
        EndProjects[lower_bound(sorted.begin(), sorted.end(), projects[i].v) - sorted.begin() + 1].push_back({lower_bound(sorted.begin(), sorted.end(), projects[i].u) - sorted.begin() + 1, projects[i].w});
    }
}
signed main(){
    int n; cin >> n;
    vector<project> projects(n);
    vector<vector<pii>> EndProjects(2 * n + 1);
    vector<int> nums;
    for(int i = 0; i < n; i++){
        cin >> projects[i].u >> projects[i].v >> projects[i].w;
        nums.push_back(projects[i].u);
        nums.push_back(projects[i].v);
    }
    compress(nums, projects, EndProjects);
    vector<int> dp(nums.size() + 1, 0);
    for(int end = 1; end <= nums.size(); end++){
        dp[end] = dp[end - 1];
        for(auto [from, gain] : EndProjects[end]){
            dp[end] = max(dp[end], dp[from - 1] + gain);
        }
    }
    cout << dp[nums.size()];
}
// Monotonic DP in campus contest, use monotonic stack
// first is lowest mountain, second is pref in stack

```

1.8 Removal Game

```

#include <bits/stdc++.h>

```

```
using namespace std;
int dp[5005][5005];
void solve(){
    int n; cin >> n;
    int pref = 0;
    vector<int> v(n+1);
    for(int i = 1; i <= n; i++){
        cin >> v[i];
        pref += v[i];
    }
    // dp[i][j] = max_diff(i to j);
    for(int i = n; i > 0; i--){
        for(int j = 1; j <= n; j++){
            if(i > j) continue;
            else if(i == j){
                dp[i][j] = v[i];
            }
            else {
                dp[i][j] = max(v[
                    i] - dp[i+1][j], v[j] - dp[i][j-1])
                ; // i+1, j-1, care dp's order
            }
        }
    }
    // x + y = sum, dp[1][n] = x - y;
    cout << (pref + dp[1][n]) / 2;
}
```

2 最大流

2.1 Dinic

```
#include <bits/stdc++.h>
using namespace std;
bool vis[505];
int lev[505], n, m, ans;
typedef struct {
    int to, w, rev_ind;
} edge;
vector<edge> adj[505];
bool label_level(){ //
    Tag the depth, if can't reach end => return false
    memset(lev, -1, sizeof(lev));
    lev[1] = 0;
    queue<int> q; q.push(1);
    while(!q.empty()){
        int u = q.front(); q.pop();
        for(auto i : adj[u]){
            if(i.w > 0 && lev[i.to] == -1){
                q.push(i.to);
                lev[i.to] = lev[u] + 1;
            }
        }
    }
    return (lev[n] == -1 ? false : true);
}
int dfs(int u, int flow){
    if(u == n) return flow;
    for(auto &i : adj[u]){
        if(lev[i.to]
            ] == lev[u] + 1 && !vis[i.to] && i.w > 0) {
            vis[i.to] = true;
            int ret = dfs(i.to, min(flow, i.w));
            if(ret > 0) {
                i.w -= ret;
                adj[i.to][i.rev_ind].w += ret;
                return ret;
            }
        }
    }
    return 0; // if can't reach end => return 0
}
void dinic(){
    while(label_level()){
        while(1){
            init(vis, 0);
            int tmp = dfs(1, inf);
            if(tmp == 0) break;
            ans += tmp;
        }
    }
}
void build(){
    for(int i = 1; i <= m; i++) {
        int u, v, w; cin >> u >> v >> w;
        adj[u].push_back({v, w,
            (int)adj[v].sz}); // inverse flow's index
    }
}
```

```
adj[v].push_back({u, 0, (int)adj
    [u].sz-1}); // have pushed one, need to -1
}
// Police Chase, need to open
adj to Augment && ori to determine what pb give
// Dinic `dfs2, then use reach as u, if the edge pb has
given && w == 0 && v is not in reach, is the ans
void dfs2(int now, unordered_set<int> &reach){
    if(!vis[now]){
        vis[now] = 1;
        reach.insert(now);
        for(auto i : adj[now]){
            if(i.w > 0){
                dfs2(i.to, reach);
            }
        }
    }
}
// two two pair // School Dance
// Dinic
, then w == 0's edge, which pb has given is the ans

// Distinct Route
// edge set valid var, if we need to argument
pos road, the reverse edge set true valid;
// if we need argument the argumented
edge's both set false. Last, from v dfs ans times
bool get_road
(int now, vector<int> &ans, vector<bool> &vis){
    if(now == 1) return true;
    for(auto &v : adj[now]){
        if(v.arg_valid && !vis[v.to]){
            ans.push_back(v.to);
            vis[v.to] = true;
            bool flag = get_road(v.to, ans, vis);
            if(flag){
                v.arg_valid = false;
                return true;
            }
        }
        ans.pop_back();
    }
    return false;
}
```

2.2 MCMF

```
// Ceiled MinCostMaxFlow, if not, use dinic
typedef struct {
    int from, to, w, cost;
} edge;
int n, m, parcel;
vector<edge> adj; // set num to each edge
vector<int> p[505]; // p[u] has edge's num
int now_edge = 0;
void add_edge(int u, int v, int w, int cost){
    adj.push_back({u, v, w, cost});
    p[u].push_back(now_edge);
    now_edge++;
    adj.push_back({
        v, u, 0, -cost}); // argumenting path use -
    p[v].push_back(now_edge);
    now_edge++;
}
ll Bellman_Ford(){
    vector<ll> dis(n+1, inf); dis[1] = 0;
    vector<int> par(m);
    vector<int> flow_rec(n+1, 0); flow_rec[1] = 1e9;
    for(int i = 1; i < n; i++){
        bool flag = 1;
        int size = adj.sz;
        for(int i = 0; i < size; i++){
            auto &[from, to, w, cost] = adj[i];
            if(w > 0 && dis[to] > dis[from] + cost){
                flag = 0;
                dis[to] = dis[from] + cost;
                par[to] = i; // record num
                flow_rec[to] = min(flow_rec[from], w);
            }
        }
        if(flag) break;
    }
    if(dis[n] == 1e9) return 0;
    int mn_flow = flow_rec[n];
    int v = n;
```

```

while(v != 1){
    int u = adj[par[v]].from;
    adj[par[v]].w -= mn_flow;
    adj[par[v] ^ 1].w += mn_flow;
    v = u;
}
mn_flow = min(mn_flow, parcel);
parcel -= mn_flow;
return mn_flow * dis[n];
}

void solve(){
    cin >> n >> m >> parcel;
    ll ans = 0;
    for(int i = 1; i <= m; i++){
        int u, v, w, cost; cin >> u >> v >> w >> cost;
        add_edge(u, v, w, cost);
    }
    while(parcel > 0){
        int tmp = Bellman_Ford();
        if(tmp == 0) break;
        ans += tmp;
    }
    cout << (parcel > 0 ? -1 : ans);
}

```

3 向量

3.1 Cross Product

```

const double EPS = 1e-9;
struct point{
    double x, y;
    point operator * (ll a){return {a * x, a * y};}
    point operator
        + (point b){return {x + b.x, y + b.y};}
    point operator
        - (point b){return {x - b.x, y - b.y};}
    double
        operator * (point b){return x * b.x + y * b.y;}
    double
        operator ^ (point b){return x * b.y - y * b.x;}
    bool operator <
        (point b){return x == b.x ? y < b.y : x < b.x;}
};
// len
double
    abs(point a){return sqrt(a.x * a.x + a.y * a.y);}
int sign(double a){
    if(abs(a) < EPS)
        return 0;
    else
        return (a > 0 ? 1 : -1);
}
//cross product
int ori(point a, point b, point c){
    return sign((b - a) ^ (c - a));
}
bool colinear(point a, point b, point c){
    return sign((b - a) ^ (c - a)) == 0;
}
bool between
    (point a, point b, point c){ // c between a and b
    if(!colinear(a, b, c))
        return false;
    return sign((a - c) * (b - c)) <= 0;
}
bool intersect(point
    a, point b, point c, point d){ // line(a,b) line(c,d)
    int abc = ori(a, b, c);
    int abd = ori(a, b, d);
    int cda = ori(c, d, a);
    int cdb = ori(c, d, b);
    if(abc == 0 || abd == 0)
        return between(a, b, c) || between(
            a, b, d) || between(c, d, a) || between(c, d, b);
    return abc * abd <= 0 && cda * cdb <= 0;
}
int main(){
    int n;
    cin >> n;
    point p[1010];
    cin >> p[0].x >> p[0].y;
    ll ans = 0;
    for(int i = 1; i < n; i++){
        cin >> p[i].x >> p[i].y;
        ans += (p[i] ^ p[i - 1]);
    }
}

```

```

}
ans += (p[0] ^ p[n - 1]);
cout << abs(ans) << '\n';

return 0;
}

```

3.2 Convex Hull

```

vector<pii> P, L, U;
ll cross(pii o, pii a, pii b){ // OA OB >0 counterclock
    return (a.first - o.first) * (b.second - o.second)
        - (a.second - o.second) * (b.first - o.first);
}
ll Andrew_monotone_chain(ll n){
    sort(P.begin(), P.end());
    ll l = 0, u = 0; // upper and lower hull
    for (ll i=0; i<n; ++i){
        while (l
            >= 2 && cross(L[l-2], L[l-1], P[i]) <= 0){
            l--;
            L.pop_back();
        }
        while (u
            >= 2 && cross(U[u-2], U[u-1], P[i]) >= 0){
            u--;
            U.pop_back();
        }
        l++;
        u++;
        L.push_back(P[i]);
        U.push_back(P[i]);
    }
    cout << l << ' ' << u << '\n';
    return l + u;
}
int main(){
    ll n, x, y;
    cin >> n;
    for(ll i = 0; i < n; i++){
        cin >> x >> y;
        P.push_back({x, y});
    }
    ll ans = Andrew_monotone_chain(n) - 2;
    cout << ans << "\n";
    return 0;
}

```

4 圖論

4.1 2-SAT

```

#include <bits/stdc++.h>
using namespace std;
// +(-) u or +(-) v
const int maxn = 1e5 + 5;
vector<int> adj[2 * maxn], rev_adj[2 * maxn];
vector<int> order;
int cat[2 * maxn];
int k = 1;
bool vis[2 * maxn];
void dfs(int now){
    if (!vis[now]){
        vis[now] = 1;
        for (auto v : adj[now]){
            dfs(v);
        }
        order.push_back(now);
    }
}
void rev_dfs(int now){
    if (!vis[now]){
        cat[now] = k;
        vis[now] = 1;
        for (auto v : rev_adj[now]){
            rev_dfs(v);
        }
    }
}
void solve(){
    int n, m;
    cin >> m >> n;
    for(int i = 1; i <= m; i++){
        int u, v;
        char a, b;
        cin >> a >> u >> b >> v;
    }
}

```

```

    if (a == '-') {
        u = 2 * n - u + 1; // reverse
    }
    if (b == '-') {
        v = 2 * n - v + 1; // reverse
    }
    adj[2 * n - u + 1].push_back(v); // from -u to v; // if -u, then v
    adj[2 * n - v + 1].push_back(u); // from -v to u; // if -v, then u
    rev_adj[v].push_back(2 * n - u + 1);
    rev_adj[u].push_back(2 * n - v + 1);
}
for(int i = 1; i <= 2 * n; i++){
    if (!vis[i]){
        dfs(i);
    }
}
memset(vis, 0, sizeof(vis));
reverse(order.begin(), order.end());
for (auto i : order){
    if (!vis[i]){
        rev_dfs(i);
        k++;
    }
}
char ans[2 * n + 1];
for(int i = 1; i <= n; i++){
    if (cat[i] == cat[2 * n - i + 1]){
        cout << "IMPOSSIBLE";
        return;
    }
    if (cat[i] > cat[2 * n - i + 1]){
        ans[i] = '+';
    }
    else ans[i] = '-';
}
for(int i = 1; i <= n; i++){
    cout << ans[i] << " ";
}
}

```

4.2 DFS跟BFS

```

#include <bits/stdc++.h>
using namespace std;
const int maxn = 1e6;
vector<int> adj[maxn];
bool vis[maxn];
void DFS(int s){
    if(vis[s]) return;
    vis[s] = true;
    for(auto u: adj[s]){
        DFS(u);
    }
}
queue<int> q;
int dis[maxn];
void BFS(int x){
    vis[x] = true;
    dis[x] = 0;
    q.push(x);
    while(!q.empty()){
        int now = q.front(); q.pop();
        for(auto nxt : adj[now]){
            if(vis[nxt]) continue;
            vis[nxt] = true;
            dis[nxt] = dis[now] + 1;
            q.push(nxt);
        }
    }
}
}

```

4.3 DSU

```

#include <bits/stdc++.h>
using namespace std;
// After each day, print the number of components
// and the size of the largest component
const int maxn = 2e5 + 5;
int ans, mx_sz = 1;
int boss[maxn];
int set_sz[maxn];
int find_boss(int x){
    if(boss[x] == x) return x;
    return boss[x] = find_boss(boss[x]);
}

```

```

}
void dsu(int x, int y){
    int boss_x = find_boss(x);
    int boss_y = find_boss(y);
    if(boss_x != boss_y){
        ans--;
        if(set_sz[boss_x] < set_sz[boss_y]){
            swap(boss_x, boss_y);
        }
        boss[boss_y] = boss_x;
        set_sz[boss_x] += set_sz[boss_y];
        mx_sz = max(mx_sz, set_sz[boss_x]);
    }
    cout << ans << " " << mx_sz << endl;
}
void solve(){
    int n, q; cin >> n >> q;
    ans = n;
    for(int i = 1; i <= n; i++){
        boss[i] = i;
        set_sz[i] = 1;
    }
    for(int i = 1; i <= q; i++){
        int x, y;
        cin >> x >> y;
        dsu(x, y);
    }
}

```

4.4 EulerRoad

```

#include <bits/stdc++.h>
using namespace std;
// Undirected: check adj[i].sz == odd => IMPOSSIBLE, road.sz != m+1 => IMPOSSIBLE
// Directed: minimize to 1 -> 2, so check in_degree == out_degree
int n, m;
const int maxn = 1e5 + 5;
set<int> adj[maxn]; // rev_adj[maxn];
int in[maxn];
void dfs(int now, vector<int> &road){
    while(!adj[now].empty()){
        int nxt = *adj[now].begin();
        adj[now].erase(nxt);
        dfs(nxt, road);
    }
    road.push_back(now);
}
void solve(){
    cin >> n >> m;
    memset(in, sizeof(in), 0);
    for(int i = 1; i <= m; i++){
        int u, v; cin >> u >> v;
        adj[u].insert(v);
        in[v]++;
    }
    in[1]++;
    in[n]--;
    for(int i = 1; i <= n; i++){
        if(adj[i].size() != in[i]){
            cout << "IMPOSSIBLE";
            return;
        }
    }
    vector<int> road;
    dfs(1, road);
    if(road.size() != m+1){
        cout << "IMPOSSIBLE";
        return;
    }
    reverse(road.begin(), road.end());
    for(auto i : road) cout << i << " ";
}

```

4.5 FloydWarshall

```

#include <bits/stdc++.h>
using namespace std;
const int maxn = 505;
int graph[maxn][maxn];
int dis[maxn][maxn];
int n, m, q; int a, b, c;
const int INF = 1e18;
int main(){
    cin >> n >> m >> q;
}

```

```

for(int i = 0; i <= n; i++) {
    for(int j = 0; j <= n; j++) {
        graph[i][j] = INF;
    }
}
for(int i = 0; i < m; i++){
    cin >> a >> b >> c;
    graph[a][b] = min(graph[a][b], c);
    graph[b][a] = min(graph[b][a], c);
}
for(int i = 0; i <= n; i++) {
    for(int j = 0; j <= n; j++) {
        dis[i][j] = graph[i][j];
    }
}
for(int i = 0; i <= n; i++) // self to self is 0
    dis[i][i] = 0;

for(int k = 1; k <= n; k++){
    for(int i = 1; i <= n; i++){
        for(int j = 1; j <= n; j++){
            dis[i][j] = min
                (dis[i][j], dis[i][k] + dis[k][j]);
        }
    }
}
for(int i = 0; i < q; i++){
    cin >> a >> b;
    cout << (
        dis[a][b] >= INF ? -1 : dis[a][b]) << "\n";
}
}

```

4.6 用 Bellman 找負環

```

#include <bits/stdc++.h>
using namespace std;
typedef struct{
    int from; int to;
    int weight;
} edge;
// NegCyc_Finding_Road
vector<edge> graph;
int main(){
    int src = 0;
    int n, m;    cin >> n >> m;
    vector<int> par(n + 1), dis(n + 1);
    for(int i = 0; i < m; i++){
        int a, b, w; cin >> a >> b >> w;
        graph.push_back({a, b, w});
    }
    for(int i = 1; i <= n; i++){
        dis[i] = 1e9 + 5;
    }
    dis[1] = 0;
    for(int i = 0; i <= n; i++){
        src = 0;
        for(auto [a, b, w] : graph){
            if(dis[b] > dis[a] + w){
                dis[b] = dis[a] + w;
                par[b] = a;
                src = b;
            }
        }
    }
    if(src){
        vector<int> ans;
        cout << "YES" << endl;
        for(int i = 0; i <= n; i++) src = par[src];
        ans.push_back(src);
        for(int i = par[src]; i != src; i = par[i]){
            ans.push_back(i);
        }
        ans.push_back(src);
        reverse(ans.begin(), ans.end());
        for (auto i : ans){
            cout << i << " ";
        }
    }
    else {
        cout << "NO" << endl;
    }
}

```

4.7 最大距離

```

#include <bits/stdc++.h>
using namespace std;
// Max_Dis, Use Topo, Use queue
// If 1 can
// 't reach n, still may be relaxed, Should dis[n] < 0
// Only Directed Graph
void print_ans(int n, vector<int> &par){
    deque<int> ans;
    int now = n;
    while(now != 1){
        ans.push_front(now);
        now = par[now];
    }
    ans.push_front(1);
    cout << ans.size() << endl;
    for(auto i : ans){
        cout << i << " ";
    }
}
void solve(){
    int n, m;
    cin >> n >> m;
    vector<int> dis(n + 1, -1e9); dis[1] = 0;
    vector<int> graph[n+1];
    vector<bool> vis(n+1, 0);
    vector<int> par(n+1);
    vector<int> in(n+1, 0);
    queue<int> q;
    for(int i = 1; i <= m; i++){
        int u, v; cin >> u >> v;
        graph[u].push_back(v);
        in[v]++;
    }
    for(int i = 1; i <= n; i++){
        if(in[i] == 0) q.push(i);
    }
    while(!q.empty()){
        int u = q.front(); q.pop();
        for(auto nxt : graph[u]){
            if(dis[nxt] < dis[u] + 1){
                dis[nxt] = dis[u] + 1;
                par[nxt] = u;
            }
            in[nxt]--; if(in[nxt] == 0) q.push(nxt);
        }
        vis[u] = 1;
    }
    if(dis[n] < 0){
        cout << "IMPOSSIBLE";
    }
    else print_ans(n, par);
}

```

4.8 負權最大距離

```

#include <bits/stdc++.h>
using namespace std;
const int maxn = 2505;
typedef struct {
    int u, v, w;
} edge;
int m, n;
vector<edge> graph;
vector<pair<int, int>> adj[maxn];
vector<int> rev_adj[maxn];
int dis[maxn];
bool vis[maxn] = {0};
bool nvis[maxn] = {0};
void dfs(int par, int now){
    if (vis[now] == 1) return;
    vis[now] = 1;
    for (auto [i, w] : adj[now]){
        if (i != par){
            dfs(now, i);
        }
    }
}
void rev_dfs(int par, int now){
    if (nvis[now] == 1) return;
    nvis[now] = 1;
    for (auto i : rev_adj[now]){
        if (i != par){
            rev_dfs(now, i);
        }
    }
}

```

```

void solve(){
    cin >> n >> m;
    for(int i = 1; i <= m; i++){
        int u, v, w;
        cin >> u >> v >> w;
        graph.push_back({u, v, w});
        adj[u].push_back({v, w});
        rev_adj[v].push_back(u);
    }
    for(int i = 1; i <= n; i++) dis[i] = -1e9;
    dis[1] = 0;
    for(int i = 1; i <= n; i++){
        for (auto [u, v, w] : graph){
            if (dis[u] + w > dis[v]){
                dis[v] = dis[u] + w;
            }
        }
    }
    dfs(0, 1);
    rev_dfs(0, n);
    for (auto [u, v, w] : graph){
        if (dis[u] + w > dis[v]
            && nvis[u] && nvis[v] && vis[u] && vis[v]){
            cout << -1;
            return;
        }
    }
    cout << dis[n];
}

```

4.9 Planet Queries II

```

#include <bits/stdc++.h>
using namespace std;
// now
    on a and want to reach b, the min steps, directed
int n, q;
const int maxn = 2e5 + 5;
int dp[30][maxn];
vector<vector<int>> cycles;
int no[maxn]; // Order & Can be in cycle, or out
int cycle_idx[maxn];
bool vis[maxn];
void set_out_of_cycle_no
    (int now, unordered_set<int> &done){
    if (done.find(now) != done.end()){
        return;
    }
    set_out_of_cycle_no(dp[0][now], done);
    done.insert(now);
    no[now] = no[dp[0][now]] - 1;
}
int wiint_go_to
    (int u, int k){ // return the node when walk k
    for(int i = 0; i <= 18; i++){
        if (k & (1 << i)){
            u = dp[i][u];
        }
    }
    return u;
}
void find_cycle(int now){
    unordered_set<int> appear;
    vector<int> vec;
    bool flag = true;
    while (appear.find(now) == appear.end()){
        appear.insert(now);
        vec.push_back(now);
        if (vis[now]){ // Didn't Find Cycle
            flag = false;
            break;
        }
    }
    now = dp[0][now];
    for (auto i : vec) vis[i] = true;
    if (!flag) return;
    int z = find(vec.begin(), vec.end(), now
        ) - vec.begin(); // start pushing from last now
    int m = vec.size();
    vector<int> cycle;
    for (int i = z; i < m; i++){
        cycle.push_back(vec[i]);
    }
    cycles.push_back(cycle);
}
void solve(){
    cin >> n >> q;

```

```

    for(int u = 1; u <= n; u++){
        cin >> dp[0][u];
    }
    for(int i = 1; i <= 18; i++){ // Make Chart
        for(int u = 1; u <= n; u++){
            dp[i][u] = dp[i - 1][dp[i - 1][u]];
        }
    }
    for(int i = 1; i <= n; i++){
        if (!vis[i]) find_cycle(i);
    }
    int idx = 0;
    memset(no, -1, sizeof(no));
    memset(cycle_idx, -1, sizeof(cycle_idx));
    unordered_set<int> done;
    for (auto &i : cycles){
        int c = 0;
        for (auto &j : i){
            no[j] = c++;
            cycle_idx[j] = idx;
            done.insert(j);
        }
        idx++;
    }
    for(int i = 1; i <= n; i++) set_out_of_cycle_no(i, done);
    for(int i = 1; i <= q; i++){
        int u, v; cin >> u >> v;
        // Same Cycle
        if (cycle_idx[u] == cycle_idx[v] &&
            cycle_idx[u] != -1 && cycle_idx[v] != -1){
            int cyc_size = cycles[cycle_idx[u]].size();
            cout << (no[v]
                - no[u] + cyc_size) % cyc_size << "\n";
        }
        else if (cycle_idx[u] == -1 && cycle_idx
            [v] == -1){ // Both are not in a Cycle
            if (no[u] > no[v]){
                cout << -1 << "\n";
                continue;
            }
            int jump = no[v] - no[u];
            if (wiint_go_to(u, jump) == v){
                cout << jump << "\n";
            }
            else cout << -1 << "\n";
        }
        else if
            (cycle_idx[u] == -1 && cycle_idx[v] != -1)
        { // v is in cycle, Smainter Binary Search
            int l = -1, r = n;
            while (l <= r){
                int m = (l + r) / 2;
                if (cycle_idx[wiint_go_to
                    (u, m)] == cycle_idx[v]){
                    r = m - 1;
                }
                else
                    l = m + 1;
            }
            if (l != -1 && l <= n){
                int in_cycle_of_u = wiint_go_to(u, l);
                int cycle_size
                    = cycles[cycle_idx[v]].size();
                cout << l + (no[v] - no[in_cycle_of_u]
                    + cycle_size) % cycle_size << "\n";
            }
            else cout << -1 << "\n";
        }
        else { // u is death in the cycle, can't reach
            cout << -1 << "\n";
        }
    }
}

```

4.10 Planets Cycles

```

#include <bits/stdc++.h>
using namespace std;
vector<int> dis, v;
vector<bool> vis;
int step;
queue<int> path;
void dfs(int x){
    path.push(x);
    if (vis[x]){

```



```

        step += dis[x];
        return;
    }
    vis[x] = true;
    step++;
    dfs(v[x]);
}
// count path_dis to rep
int main(){
    int n; cin >> n;
    v.assign(n + 1, 0);
    dis.assign(n + 1, 0);
    vis.assign(n + 1, false);
    for (int i = 1; i <= n; i++){
        cin >> v[i];
    }
    for (int i = 1; i <= n; i++){
        step = 0;
        int is_outof_cycle = 1;
        dfs(i);
        while (!path.empty()){
            if (path.front() == path.back()){
                is_outof_cycle = 0;
            }
            dis[path.front()] = step;
            step -= is_outof_cycle;
            path.pop();
        }
    }
    for (int i = 1; i <= n; i++){
        cout << dis[i] << ' ';
    }
    cout << '\n';
}

```

4.11 找環

```

#include <bits/stdc++.h>
using namespace std;
const int maxn = 1e5 + 5;
vector<int> graph[maxn];
int color[maxn], parent[maxn];
bool vis[maxn];
int n, m;
void print_ans(int ori){
    int now = parent[ori];
    deque<int> ans;
    ans.push_front(ori);
    while (now != ori){
        ans.push_front(now);
        now = parent[now];
    }
    ans.push_front(ori);
    cout << ans.size() << endl;
    for (auto i : ans){
        cout << i << " ";
    }
    exit(0);
}
void dfs(int now){
    color[now] = 1;
    vis[now] = 1;
    for (auto nxt : graph[now]){
        parent[nxt] = now;
        if (color[nxt] == 1){
            print_ans(nxt);
        }
        else if (color[nxt] == 0){
            dfs(nxt);
        }
    }
    color[now] = 2;
}
void solve(){
    cin >> n >> m;
    for (int i = 1; i <= m; i++){
        int u, v; cin >> u >> v;
        graph[u].push_back(v);
    }
    for (int i = 1; i <= n; i++){
        if (!vis[i])
            dfs(i);
    }
    cout << "IMPOSSIBLE";
}

```

4.12 Prim

```

#include <bits/stdc++.h>
using namespace std;
#define pii pair<int, int>
int n, m;
int ans = 0;
const int maxn = 2e5 + 5;
vector<pair<int, int>> adj[maxn];
bool Prim(){
    int node_sz = 0;
    priority_queue<pii, vector<pii>, greater<pii>> pq;
    pq.push({0, 1});
    bool vis[maxn] = {false};
    while (!pq.empty()){
        auto [cost, u] = pq.top(); pq.pop();
        if (vis[u]) continue;
        vis[u] = true;
        ans += cost;
        node_sz++;
        for (auto [v, cost] : adj[u]){
            if (!vis[v])
                pq.push({cost, v});
        }
    }
    if (node_sz == n) return true;
    return false;
}
void solve(){
    cin >> n >> m;
    for (int i = 1; i <= m; i++){
        int u, v, cost; cin >> u >> v >> cost;
        adj[u].push_back({v, cost});
        adj[v].push_back({u, cost});
    }
    if (Prim()) cout << ans;
    else cout << "IMPOSSIBLE";
}

```

4.13 SCC 跟拓撲 DP

```

#include <bits/stdc++.h>
using namespace std;
// Fing Aint SCC and Build a DAG, then Topo DP
const int maxn = 1e5 + 5;
vector<int> v, adj[maxn], rev_adj[maxn], DAG[maxn];
int order[maxn], coin[maxn], in[maxn];
int n, m, k = 0;
bool vis[maxn];
void dfs(int now){
    if (!vis[now]){
        vis[now] = 1;
        for (auto i : adj[now]){
            dfs(i);
        }
        v.push_back(now);
    }
}
void rev_dfs(int now){
    if (!vis[now]){
        vis[now] = 1;
        order[now] = k;
        for (auto i : rev_adj[now]){
            rev_dfs(i);
        }
    }
}
void solve(){
    cin >> n >> m;
    for (int i = 1; i <= n; i++){
        cin >> coin[i];
    }
    for (int i = 1; i <= m; i++){
        int u, v; cin >> u >> v;
        adj[u].push_back(v);
        rev_adj[v].push_back(u);
    }
    for (int i = 1; i <= n; i++){
        if (!vis[i])
            dfs(i);
    }
    reverse(v.begin(), v.end());
    memset(vis, sizeof(vis), false);
    for (auto i : v){
        if (!vis[i]){

```



```

        k++;
        rev_dfs(i);
    }
}
// Categorized SCC
int sum_coin[k + 1], dp_coin[k + 1];
init(sum_coin, 0); init(dp_coin, 0);
int ans = -1e9;
for(int i = 1; i <= n; i++){
    sum_coin[order[i]] += coin[i]; // Now team(k) += coin;
    for(auto j : adj[i]){
        if(order[i] != order[j]){
            DAG[order[i]].push_back(order[j]);
            in[order[j]]++;
        }
    }
}
// Topo DP
queue<int> q;
for(int i = 1; i <= k; i++){
    if(in[i] == 0){
        q.push(i);
    }
}
while(!q.empty()){
    int now = q.front(); q.pop();
    dp_coin[now] += sum_coin[now];
    ans = max(ans, dp_coin[now]);
    for(auto v : DAG[now]){
        in[v]--;
        dp_coin[v] = max(dp_coin[v], dp_coin[now]);
        if(in[v] == 0) q.push(v);
    }
}
cout << ans;
}

```

4.14 狀態 Dijkstra

```

#include <bits/stdc++.h>
using namespace std;
#define pii pair<int, int>
// Flight Discount
int n, m;
const int maxn = 2e5 + 5;
vector<pii> graph[maxn];
int dis[maxn][2]; // 0 for not used
void dijkstra(){
    priority_queue<vector<int>, vector<vector<int>>, greater<vector<int>>> pq; // 0 for w, 1 for u, 2 for discount
    for(int i = 1; i <= n; i++){
        dis[i][0] = dis[i][1] = 1e9;
    }
    dis[1][0] = dis[1][1] = 0;
    pq.push({0, 1, 0});
    while(!pq.empty()){
        auto nxt = pq.top(); pq.pop();
        int dist = nxt[0], u = nxt[1]; bool us = nxt[2];
        if(dis[u][us] < dist) continue; // is out of time, pass
        if(us){
            for(auto [v, w] : graph[u]){
                if(dis[u][1] + w < dis[v][1]){
                    dis[v][1] = dis[u][1] + w;
                    pq.push({dis[v][1], v, 1});
                }
            }
        }
        else {
            for(auto [v, w] : graph[u]){
                if(dis[u][0] + w < dis[v][0]){
                    dis[v][0] = dis[u][0] + w;
                    pq.push({dis[v][0], v, 0});
                }
                if(dis[u][0] + w / 2 < dis[v][1]){
                    dis[v][1] = dis[u][0] + w / 2;
                    pq.push({dis[v][1], v, 1});
                }
            }
        }
    }
}
cout << min(dis[n][0], dis[n][1]);
}

```

```

void solve(){
    cin >> n >> m;
    for(int i = 1; i <= m; i++){
        int u, v, w;
        cin >> u >> v >> w;
        graph[u].push_back({v, w});
    }
    dijkstra();
}

```

4.15 Vis Dijkstra

```

#include <bits/stdc++.h>
using namespace std;
void solve(){
    int n, m, noon, night;
    cin >> n >> m >> noon >> night;
    int dis[n + 1];
    vector<int> graph[n + 1];
    bool vis[n + 1];
    for(int i = 1; i <= m; i++){
        int u, v, w; cin >> u >> v >> w;
        graph[u].push_back({v, w});
        graph[v].push_back({u, w});
    }
    priority_queue<vector<int>, vector<vector<int>>, greater<vector<int>>> pq;
    // noon is -
    for(int i = 1; i <= n; i++){
        dis[i] = 1e9; vis[i] = 0;
    }
    pq.push({0, -noon, 1});
    dis[1] = 0;
    while(!pq.empty()){
        vector<int> now = pq.top(); pq.pop();
        int now_noon = -now[1], u = now[2];
        if(vis[u]) continue;
        for(auto [nxt, w] : graph[u]){
            if(noon < w) continue; // never pass
            int tmp = dis[u] + (now_noon >= w ? w : now_noon + night + w);
            if(tmp < dis[nxt]){
                dis[nxt] = tmp;
                pq.push({dis[nxt], -(now_noon >= w ? now_noon - w : noon - w), nxt});
            }
        }
        vis[u] = true;
    }
    if(dis[n] == 1e9) cout << -1 << endl;
    else cout << dis[n] << endl;
}
// Investigation
void Investigation(){
    vector<vector<int>> graph[n + 1];
    for(auto [v, w] : graph[u]){
        if(dis[u] + w < dis[v]){
            dis[v] = dis[u] + w;
            pq.push({dis[v], v});
            min_price_nums[v] = min_price_nums[u];
            max_dis_min_price[v] = max_dis_min_price[u] + 1;
            min_dis_min_price[v] = min_dis_min_price[u] + 1;
        }
        else if(dis[u] + w == dis[v]){
            min_price_nums[v] = (min_price_nums[u] + min_price_nums[v]) % mod;
            max_dis_min_price[v] = max(max_dis_min_price[u] + 1, max_dis_min_price[v]);
            min_dis_min_price[v] = min(min_dis_min_price[u] + 1, min_dis_min_price[v]);
        }
    }
}
int main(){
    solve();
    Investigation();
}

```

5 數學

5.1 質因數分解

```
// a^(m-1) ≡ 1 (mod m)
// a^(m-2) ≡ 1/a (mod m)
// EXP2
: cout << fast_exp(x, fast_exp(y, p, MOD - 1), MOD)
// Filter +
DP; DP save min factor, recur, factor decomposition
// FacNums = (x+1)(y+1)(z+1)...
// FacSum = (a^0+a^1...+a^x)(b^0+...+b^y)
// FacMul = N(x+1)(y+1)(z+1)/2
int Is_Prime[1000005][2];
int main(){
    for(int
        i = 1; i <= 1000000; i++) Is_Prime[i][0] = 1;
    Is_Prime[1][0] = 0;
    for(int i = 2; i <= 1000; i++){
        if(Is_Prime[i][0]){
            for(int j = i + i; j <= 1000000; j += i){
                Is_Prime[j][0] = 0;
                Is_Prime[j][1] = i;
            }
        }
    }
    int ans = 1;
    int q; cin >> q;
    if(q == 1) {
        cout << 1 << endl;
        return;
    }
    map<int, int> mp;
    while(!Is_Prime[q][0]){
        mp[Is_Prime[q][1]]++;
        q /= Is_Prime[q][1];
    }
    mp[q]++;

    for(auto [a, b] : mp){
        ans *= b + 1;
    }
    cout << ans << endl;
}
```

5.2 盧卡斯定理

```
struct nCr {
    int mod;
    nCr(int mod) : mod(mod){};
    int inverse(int num){
        if(num == 1) return 1;
        return (mod - ((mod
            / num) * inverse(mod % num)) % mod) % mod;
    }
    int fast_exp(int x, int p){
        int ans = 1;
        while(p > 0){
            if(p & 1) ans = (ans * x) % mod;
            x = x * x % mod;
            p >>= 1;
        }
        return ans;
    }
    vector<int> fac;
    void BuildLucas(int n){
        fac.resize(n + 1);
        fac[0] = 1;
        for(int i = 1; i <= n; i++){
            fac[i] = fac[i - 1] * i % mod;
        }
    }
    int C(int m, int n){
        return m < n ? 0 : fac[m] * inverse
            (fac[n]) % mod * inverse(fac[m - n]) % mod;
    }
    int Lucas(int m, int n){
        return n == 0 ? 1 % mod : Lucas(m /
            mod, n / mod) * C(m % mod, n % mod) % mod;
    }
};
```

6 Queries

6.1 BIT

```
#include <bits/stdc++.h>
using namespace std;
struct BIT {
    int n;
```

```
vector<int> bit;
BIT(int n) {
    this->n = n;
    bit.resize(n + 1, 0);
}
void modify(int i, int val) {
    for (; i <= n; i += i & -i) {
        bit[i] += val;
    }
}
int query(int r) {
    int ans = 0;
    for (; r; r -= r & -r) ans += bit[r];
    return ans;
}
};

struct TwoDimensionBIT {
    int nx, ny;
    vector<vector<int>> bit;
    TwoDimensionBIT(int x, int y) {
        nx = x; ny = y;
        bit.resize(x + 1, vector<int>(y + 1, 0));
    }
    void modify(int x, int y, int mod){
        for(; x <= nx; x += x & -x){
            for(int
                tmp = y; tmp <= ny; tmp += tmp & -tmp){
                bit[x][tmp] += mod;
            }
        }
    }
    int query(int r1, int r2){
        int ans = 0;
        for(; r1; r1 -= r1 & -r1){
            for(int tmp = r2; tmp; tmp -= tmp & -tmp){
                ans += bit[r1][tmp];
            }
        }
        return ans;
    }
};
```

6.2 Increasing Array Queries

```
#include <bits/stdc++.h>
using namespace std;
const int maxn = 2e5+5;
int n, q;
int nums[maxn],
    prefix[maxn], ans[maxn], BIT[maxn], contrib[maxn];
vector<pair<int, int>> queries[maxn];
void update(int pos, int val) {
    for (; pos <= n; pos += pos & -pos) BIT[pos] += val;
}
int query(int a, int b) {
    int ans = 0;
    for (; b; b -= b & -b) ans += BIT[b];
    for (a--; a; a -= a & -a) ans -= BIT[a];
    return ans;
}
void solve(){
    cin >> n >> q;
    for(int i = 1; i <= n; i++){
        cin >> nums[i];
        prefix[i] = prefix[i-1] + nums[i];
    }
    nums[n + 1] = 1e9;
    prefix[n + 1] = 2e18;
    for(int i = 1; i <= q; i++){
        int a, b; cin >> a >> b;
        queries[a].push_back({b, i});
    }
    deque<int> mono; mono.push_front(n+1);
    for(int i = n; i > 0;
        i--){ // question from start at n to start at 1
        while (nums[i] >= nums[mono.front()]) {
            update(mono.front(), -contrib[mono.front
                ()]); // mono.front's contrib become 0
            mono.pop_front();
        }
        contrib[i] = (mono.front() - 1 - i) * nums
            [i] - (prefix[mono.front() - 1] - prefix[i]);
        update(i, contrib[i]);
        mono.push_front(i);
        for (auto j : queries[
            i]) { // pos is the index in mono <= end's
```

```

int pos = upper_bound(mono.begin
(), mono.end(), j.first) - mono.begin() - 1;
ans[j.second] = (pos ? query(i, mono
[pos - 1]) : 0) // smainter than y's mono
// mono
// to y caculate directly
+ (j.first
- mono[pos]) * nums[mono[pos]]
- (prefix[j.first
] - prefix[mono[pos]]);
}
}
for(int i = 1; i <= q; i++){
cout << ans[i] << endl;
}
}

```

6.3 線段樹

```

#include <bits/stdc++.h>
using namespace std;
template <class Node, class Lazy>
struct Seg {
int n;
vector<Node> tree;
template <typename T>
Seg (vector<T> init_){
n = init_.size() - 1;
tree.resize(4 * n);
function <void(int, int,
int)> build = [&](int now, int l, int r) {
if (l == r) {
tree[now] = init_[l];
return;
}
int m = (l + r) / 2;
build(now << 1, l, m);
build((now << 1) + 1, m + 1, r);
pull(now);
};
build(1, 1, n);
}
Node query(int l, int r, int ql, int qr, int now){
int m = (l + r) >> 1;
if(qr < l || ql > r){
// -----out of
// range, return what-----
return {0};
//
}
if(ql <= l && r <= qr){
return tree[now];
}
return query(l, m, ql, qr, now <<
1) + query(m + 1, r, ql, qr, (now << 1) + 1);
}
Node query
(int l, int r) { return query(1, n, l, r, 1); }
void pull(int now){
tree[now
] = tree[now << 1] + tree[(now << 1) + 1];
}
void modify
(int l, int r, int idx, int now, int add){
if(l == r){
// -----
// how to modify ?-----
tree[now].sum = add;
//
}
return;
}
int m = (l + r) >> 1;
if(idx <= m){
modify(l, m, idx, now << 1, add);
}
else {
modify(m + 1, r, idx, (now << 1) + 1, add);
}
pull(now);
}
}

```

```

void modify(int
idx, int add) { modify(1, n, idx, 1, add); }
};
// -----define
// structure and info plus-----
struct Node {
int sum;
};
Node operator+(const Node &a, const Node &b) {
return {{a.sum + b.sum}};
// use lc `rc to undate now
// tree[now].sum = tree[lc].sum + tree[rc].sum;
// tree[now].prefix = max(
tree[lc].sum+tree[rc].prefix, tree[lc].prefix);
// tree[now].suffix = max(
tree[lc].suffix+tree[rc].sum, tree[rc].suffix);
// tree[now].middle_max
= max(max(tree[lc].middle_max, tree[rc].
middle_max), tree[lc].suffix+tree[rc].prefix);
// tree
[now].middle_max = max(max(tree[now].middle_max
, tree[now].prefix), tree[now].suffix);
}
//
// -----
// pizza_queries
// Left(s < t): dis_l = (pizza[s] - s) + t;
// Right(t < s): dis_r = (pizza[s] + s) - t;
// List Removals
// Use seg_tree to maintain
// how many nums have been selected in the range
// Use binary_Search
// to find "mod" nums have been selected before ans
// if ans - mod == pos
// , nums[ans] is the answer, and we modify tree[pos]
// polynomial queries
// Lazy_seg, set under and distance

```

6.4 懶標線段樹

```

#include <bits/stdc++.h>
using namespace std;
template <class Node, class Lazy>
struct LazySeg {
int n;
vector<Node> tree;
vector<Lazy> lazy;
template <typename T>
LazySeg (vector<T> init_){
n = init_.size() - 1;
tree.resize(4 * n);
lazy.resize(4 * n);
function <void(int, int,
int)> build = [&](int now, int l, int r) {
if (l == r) {
tree[now] = init_[l];
return;
}
int m = (l + r) / 2;
build(now << 1, l, m);
build((now << 1) + 1, m + 1, r);
pull(now);
};
build(1, 1, n);
}
Node query(int l, int r, int ql, int qr, int now){
int m = (l + r) >> 1;
if(qr < l || ql > r){
// -----out of
// range, return what-----
return Node();
//
}
push(now, l, r);
if(ql <= l && r <= qr){
return tree[now];
}
return query(l, m, ql, qr, now <<
1) + query(m + 1, r, ql, qr, (now << 1) + 1);
}
}

```

```

}
Node query
(int l, int r) { return query(1, n, l, r, 1); }
void pull(int now){
    tree[now] = tree[now << 1] + tree[(now << 1) + 1];
}
void modify_add(int
    l, int r, int ql, int qr, int now, int add){
    int m = (l + r) >> 1;
    if(qr < l || ql > r){
        return;
    }
    if(ql <= l && r <= qr){
// -----
// how to modify ?-----
//
// -----
        lazy[now].add += add;
// -----
// -----
        return;
    }
    push(now, l, r);
    modify_add(l, m, ql, qr, now << 1, add);
    modify_add
        (m + 1, r, ql, qr, (now << 1) + 1, add);
    push(now << 1, l, m);
    push((now << 1) + 1, m + 1, r);
    pull(now);
}
void modify_add(int l, int
    r, int add) { modify_add(1, n, l, r, 1, add); }
void modify_set(int
    l, int r, int ql, int qr, int now, int val){
    int m = (l + r) >> 1;
    if(qr < l || ql > r){
        return;
    }
    if(ql <= l && r <= qr){
// -----
// how to modify ?-----
//
// -----
        lazy[now].set_val = val;
        lazy[now].add = 0;
// -----
// -----
        return;
    }
    push(now, l, r);
    modify_set(l, m, ql, qr, now << 1, val);
    modify_set
        (m + 1, r, ql, qr, (now << 1) + 1, val);
    push(now << 1, l, m);
    push((now << 1) + 1, m + 1, r);
    pull(now);
}
void modify_set(int l, int
    r, int val) { modify_set(1, n, l, r, 1, val); }
void push(int now, int l, int r){
    apply(now, l, r);
// -----how to
// push down ?-----
// -----
    if(l != r){
        if(lazy[now].set_val){
            lazy[now]
                << 1).set_val = lazy[now].set_val;
            lazy[(now << 1)
                + 1].set_val = lazy[now].set_val;
            lazy[now << 1].add = lazy[now].add;
            lazy[(
                now << 1) + 1].add = lazy[now].add;
        }
        else {
            lazy[now << 1].add += lazy[now].add;
            lazy[(now
                << 1) + 1].add += lazy[now].add;
        }
    }
// -----
// -----
    lazy[now] = Lazy();
}

```

```

void apply(int now, int l, int r){
    if(lazy[now].set_val){
        tree[now].
            sum = (r - l + 1) * lazy[now].set_val;
    }
    tree[now].sum += (r - l + 1) * lazy[now].add;
}
};
// -----define
// structure and info plus-----
struct Node {
    int sum;
};
struct Lazy {
    int set_val; int add;
};
Node operator+(const Node &a, const Node &b) {
    return {{a.sum + b.sum}};
}
// -----
// -----

```

6.5 莫隊

```

#include <bits/stdc++.h>
using namespace std;
struct query {
    int l, r, id;
} typedef query;
void MO(int n, vector<query> &queries){
    int block = sqrt(n);
    function <bool
        (query, query)> cmp = [&](query a, query b) {
        int block_a = a.l / block;
        int block_b = b.l / block;
        if(block_a
            != block_b) return block_a < block_b;
        return a.r < b.r;
    };
    sort(queries.begin(), queries.end(), cmp);
}
void compress(vector<int> &nums){
    vector<int> sorted = nums;
    sort(sorted.begin(), sorted.end());
    sorted.erase(unique
        (sorted.begin(), sorted.end()), sorted.end());
    for(int i = 0; i < nums.size(); i++){
        nums[i] = lower_bound(sorted.begin(), sorted
            .end(), nums[i]) - sorted.begin() + 1;
    }
}

```

6.6 Treap

```

#include <bits/stdc++.h>
using namespace std;
struct Treap {
    Treap *l, *r;
    int pri, subsize; char val; bool rev_valid;
    Treap(int val){
        this->val = val;
        pri = rand();
        l = r = nullptr;
        subsize = 1; rev_valid = 0;
    }
    void pull
        (){ // update subsize or other information
        subsize = 1;
        for(auto i : {l, r}){
            if(i) subsize += i->subsize;
        }
    }
};
int size(Treap *treap) {
    if (treap == NULL) return 0;
    return treap->subsize;
}
// lazy
void push(Treap *t){
    if(!t) return;
    if(t->rev_valid){
        swap(t->l, t->r);
        if(t->l) t->l->rev_valid ^= 1;
        if(t->r) t->r->rev_valid ^= 1;
    }
}

```

```

    }
    t->rev_valid = false;
}
Treap *merge(Treap *a, Treap *b){
    if(!a || !b) return a ? a : b;
    // push(a); push(b); // lazy
    if(a->pri > b->pri){
        a->r = merge(a->
            r, b); // a->r = new, inorder, make sense
        a->pull();
        return a;
    }
    else {
        b->l = merge(a,
            b->l); // new->l = a, inorder, make sense
        b->pull();
        return b;
    }
}
pair<Treap*,
    Treap*> split(Treap *root, int k) { // find 1~k
    if (root == nullptr) return {nullptr, nullptr};
    // push(root); // lazy
    if (size(root->l) < k) {
        auto
            [a, b] = split(root->r, k - size(root->l) - 1);
        root->r = a;
        root->pull();
        return {root, b};
    }
    else {
        auto [a, b] = split(root->l, k);
        root->l = b;
        root->pull();
        return {a, root};
    }
}
void Print(Treap *t){
    if(t){
        // push(t); // lazy
        Print(t->l);
        cout << t->val;
        Print(t->r);
    }
}
void substring_rev(){
    int n, m; cin >> n >> m;
    Treap *root = nullptr;
    string str; cin >> str;
    for(auto c : str){
        root = merge(root, new Treap(c));
    }
    for(int i = 1; i <= m; i++){
        int x, y; cin >> x >> y;
        auto [a,
            b] = split(root, x-1); // a: 1~x-1, b: x~n
        auto [
            c, d] = split(b, y-x+1); // Use b to split
        // c->rev_valid ^= true;
        // push(c);
        b = merge(a, d); // Notice the order
        root = merge(b, c);
    }
    Print(root);
}
}

```

7 搜尋與貪心

7.1 二分搜

```

int main(){
    int l = 1, r = 10;
    // 1 to tar, find tar
    while(l <= r){
        int m = (l + r) / 2;
        if(check(m)) l = m + 1;
        else r = m - 1;
    }
    cout << r;
    // tar to end
    while(l <= r){
        int m = (l + r) / 2;
        if(check(m)) r = m - 1;
        else l = m + 1;
    }
}

```

```

    cout << l;
}

```

7.2 Concert Ticket

```

// Better than Binary Search
int main(){
    int n, m; cin >> n >> m;
    multiset<int> tik;
    for(int i = 0; i < n; i++){
        int tmp; cin >> tmp;
        tik.insert(tmp);
    }
    while(m--){
        int x; cin >> x;
        auto it = tik.upper_bound(x);
        if(it == tik.begin()){
            cout << -1 << " ";
            continue;
        }
        it--;
        cout << *it << " ";
        tik.erase(it);
    }
}

```

7.3 Restaurant Customers

```

int main(){
    vector<pair<int, int>> times;
    int n; cin >> n;
    for(int i = 0; i < n; i++){
        int u, v; cin >> u >> v;
        times.push_back({u, 1});
        times.push_back({v, -1});
    }
    sort(times.begin(), times.end());
    int now_people = 0, ans = 0;
    for(auto [t, x] : times){
        ans = max(ans, (now_people += x));
    }
    cout << ans;
}

```

8 字串演算法

8.1 KMP

```

#include <bits/stdc++.h>
using namespace std;
struct KMP {
    string s;
    vector<int> failure;
    KMP(string s) {
        this->s = s;
        failure.resize(s.size(), -1);
    }
    void buildFailFunction() {
        for(int i = 1; i < s.size(); i++) {
            int now = failure[i - 1];
            while(now != -1 &&
                s[now + 1] != s[i]) now = failure[now];
            if (s[
                now + 1] == s[i]) failure[i] = now + 1;
        }
    }
    void KMPmatching(string &sub) {
        for(int i = 0, now = -1; i < s.size(); i++) {
            // now is the compare suceeded length -1
            while (s[i] != sub[now
                + 1] && now != -1) now = failure[now];
            // f stores
            if comparison fail, move to where
            if (s[i] == sub[now + 1]) now++;
            if (now + 1 == sub.size()) {
                cout << "found a match start
                    at position " << i - now << "\n";
                now = failure[now];
            }
        }
    }
};
int main(){
    string s = "BABA";
    string sub = "BA";
    KMP kmp(s);
}

```

```

    kmp.buildFailFunction();
    kmp.KMPmatching(sub);
}

```

9 樹論

9.1 LCA

```

#include <bits/stdc++.h> // LCA from 1
using namespace std;
const int maxn = 2e5+5;
int boss[maxn];
int height[maxn];
int arr[18][maxn];
vector<int> tree[maxn];
void Calculate_H(int now_node){
    for(auto nxt_node : tree[now_node]){
        height[nxt_node] = height[now_node] + 1;
        Calculate_H(nxt_node);
    }
}
int Find_Ancesor(int k, int h){
    for(int i = 0; i <= 17; i++){
        if(h & (1 << i)) k = arr[i][k];
    }
    return k;
}
int main(){
    memset(arr, 0, sizeof(arr));
    int n, q; cin >> n >> q;
    boss[1] = 1;
    for(int i = 2; i <= n; i++){
        int tmp; cin >> tmp; // tmp to i
        boss[i] = tmp;
        tree[tmp].push_back(i);
    }
    Calculate_H(1);
    for(int i = 2; i <= n; i++){
        arr[0][i] = boss[i];
    }
    for(int i = 1; i <= 17; i++){ // make chart
        for(int j = 1; j <= n; j++){
            arr[i][j] = arr[i-1][arr[i-1][j]];
        }
    }
    while(q--){
        int a, b; cin >> a >> b;
        if(height[a] < height[b]) swap(a, b);
        a = Find_Ancesor(a, height[a] - height[b]); // same depth from 1
        if(a == b){ // same point
            cout << a << "\n";
            continue;
        }
        for(int i = 17; i >= 0; i--){
            if(arr[i][a] != arr[i][b]){
                // if a, b up 2^i not the same point
                a = arr[i][a];
                b = arr[i][b];
            }
        }
        cout << arr[0][a] << "\n"; // more one
    }
}

```

9.2 子樹 DP

```

#include <bits/stdc++.h>
using namespace std;
const int maxn = 2e5+5;
vector<int> tree[maxn];
int sub[maxn];
void dfs(int now){
    for(auto nxt : tree[now]){
        dfs(nxt);
        sub[now] += sub[nxt] + 1;
    }
}
int main(){
    memset(sub, 0, sizeof(sub));
    int n; cin >> n;
    for(int i = 2; i <= n; i++){
        int b; cin >> b;
        tree[b].push_back(i);
    }
    dfs(1);
}

```

```

for(int i = 1; i <= n; i++){
    cout << sub[i] << " ";
}
}

```

9.3 樹重心

```

#include <bits/stdc++.h>
using namespace std;
const int maxn = 2e5+5;
vector<int> tree[maxn];
int cen = 0, n;
int dfs(int par, int now){
    bool flag = 1;
    int size = 0;
    for(auto nxt : tree[now]){
        if(par != nxt){
            int subsize = dfs(now, nxt);
            if(subsize > n / 2) flag = false;
            size += subsize;
        }
    }
    if(n - 1 - size > n / 2) flag = false;
    if(flag) cen = now;
    return size + 1;
}
int main(){
    cin >> n;
    for(int i = 1; i < n; i++){
        int u, v; cin >> u >> v;
        tree[u].push_back(v);
        tree[v].push_back(u);
    }
    for(int i = 1; i <= n; i++){
        dfs(i, 0);
        if(cen) break;
    }
}

```

9.4 節點距離總和

```

#include <bits/stdc++.h>
using namespace std;
const int maxn = 2e5+5;
vector<int> tree[maxn];
vector<int> subtree(maxn, 1);
long long ans[maxn];
int n;
void dfs(int par, int now, int depth){
    ans[1] += depth;
    for(auto nxt : tree[now]){
        if(par != nxt){
            dfs(now, nxt, depth + 1);
            subtree[now] += subtree[nxt];
        }
    }
}
void find_ans(int par, int now){
    // each sub's dis make - 1, non subnode + 1
    for(auto nxt : tree[now]){
        if(par != nxt){
            ans[nxt] = ans[now]
                + (n - subtree[nxt]) - subtree[nxt];
            find_ans(now, nxt);
        }
    }
}
int main(){
    cin >> n;
    for(int i = 1; i < n; i++){
        int u, v; cin >> u >> v;
        tree[u].push_back(v);
        tree[v].push_back(u);
    }
    dfs(0, 1, 0);
    find_ans(0, 1);
    for(int i = 1; i <= n; i++){
        cout << ans[i] << " ";
    }
}

```

9.5 無權樹直徑

```

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

```

```

const int maxn = 2e5+5;
int dep1[maxn], dep2[maxn];
vector<int> tree[maxn];
int start, maxdep = 1, End;
void dfs1(int par, int now){
    for(auto nxt : tree[now]){
        if(par != nxt){
            dep1[nxt] = dep1[now] + 1;
            if(dep1[nxt] > maxdep){
                maxdep = dep1[nxt];
                start = nxt;
            }
            dfs1(now, nxt);
        }
    }
}

void find_depth1(int par, int now){
    for(auto nxt : tree[now]){
        if(par != nxt){
            dep1[nxt] = dep1[now] + 1;
            if(dep1[nxt] > maxdep){
                maxdep = dep1[nxt];
                End = nxt;
            }
            find_depth1(now, nxt);
        }
    }
}

void find_depth2(int par, int now){
    for(auto nxt : tree[now]){
        if(par != nxt){
            dep2[nxt] = dep2[now] + 1;
            find_depth2(now, nxt);
        }
    }
}

int main(){
    int n; cin >> n;
    for(int i = 1; i < n; i++){
        int u, v; cin >> u >> v;
        tree[u].push_back(v);
        tree[v].push_back(u);
    }
    dep1[1] = 1;
    dfs1(0, 1);
    dep1[start] = 1;
    maxdep = 1;
    find_depth1(start, start);
    dep2[End] = 1;
    find_depth2(End, End);
    for(int i = 1; i <= n; i++){
        cout << max(dep1[i], dep2[i]) - 1 << " ";
    }
}

    tree[v].push_back({u, w});
}
DP(1, 0);
cout << (ans < 0 ? 0 : ans);
}

```

9.6 有權樹直徑

```

#include <bits/stdc++.h> // weighted tree centroid
using namespace std;
const int maxn = 1e5+5;
using ll = long long;
vector<pair<int, int>> tree[maxn];
ll dp[maxn];
ll ans = 0;
void DP(int now, int par){
    ll mx1 = 0; ll mx2 = 0;
    for(auto [nxt, w] : tree[now]){
        if(nxt == par) continue;
        DP(nxt, now);
        if(mx1 < w + dp[nxt]){ // mx2 = mx1, mx1 = new mx
            mx2 = mx1; mx1 = w + dp[nxt];
        }
        else if(mx2 < w + dp[nxt]){ // mx2 = new
            mx2 = w + dp[nxt];
        }
    }
    dp[now] = mx1;
    ans = max(ans, mx1 + mx2);
}

int main(){
    int n; cin >> n;
    memset(dp, 0, sizeof(dp));
    for(int i = 1; i < n; i++){
        int u, v, w; cin >> u >> v >> w;
        tree[u].push_back({v, w});
    }
}

```