Beijing Normal University

Template

Pray Never

2025 年 4 月 19 日

目录

1 data structure

1.1 01tries.cpp

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N = 1e5 + 10;
vector<pair<int, int> > v[N];
int cnt;
int node[5 * N][2];
bool isval[5 * N];
void insert(int val){
    int p = 0;
    stack<int> s;
    for(int i = 0; i < 31; ++i){
      s.push(val % 2);
      val /= 2;
    }
    for(int i = 0; i < 31; i++){
        int c = s.top();
        s.pop();
        if(!node[p][c]) node[p][c] = ++cnt;
        p = node[p][c];
    }
    isval[p]++;
}
int a[N];
void dfs(int pos, int far){
  for(auto[u, w]: v[pos]){
    if(u == far);
      continue;
    a[u] = a[pos] ^ w;
    dfs(u, pos);
  }
}
int query(int val){

}
int main(){
  int n;
  cin >> n;
  for(int i = 1; i < n; ++i){
    int x, y, z;
    cin >> x >> y >> z;
    v[x].emplace_back(y, z);
    v[y].emplace_back(x, z);
  }
  dfs(1, 0);
  int maxy = 0;
  for(int i = 1; i <= n; ++i){
    maxy = max(maxy, query(a[i]));
  }
}
```

1.2 persistant-segement-tree.cpp

```
#ifdef IGNORE_THIS_FILE
struct PTR{
    int n, tot;
    vector<int> ls, rs, sum, root;
    PTR(int _n){
        n = _n;
        tot = 0;
        ls = vector<int>(n << 6), rs = vector<int>(n << 6), sum = vector<int>(n << 6);
        root = vector<int>(n + 1);
    }
    int update(int q, int l, int r, int rt){
        int dir = ++tot;
        ls[dir] = ls[rt], rs[dir] = rs[rt];
        if(l == r){
          sum[dir] = sum[rt] + 1;
          return dir;
        }
        int mid = (l + r) >> 1;
        if(q <= mid)
          ls[dir] = update(q, l, mid, ls[dir]);
        else
          rs[dir] = update(q, mid + 1, r, rs[dir]);
        sum[dir] = sum[ls[dir]] + sum[rs[dir]];
        return dir;
    };
    int query(int q, int l, int r, int tl, int tr){
        if(l == r)
          return l;
        int mid = (l + r) >> 1;
        int sum1 = sum[ls[tr]] - sum[ls[tl]];
        if(q <= sum1)
          return query(q, l, mid, ls[tl], ls[tr]);
        else
          return query(q - sum1, mid + 1, r, rs[tl], rs[tr]);
      };
};

#endif


```

1.3 trie.cpp

```
#ifdef IGNORE_THIS_FILE
struct Trie {
  static const int N = 5000100; // 预设最大节点数
  int cnt = 0;
  vector<vector<int>> node{N, vector<int>(26)};
  vector<int> isval{N};

  void insert(string& s, int len) {
    int p = 0;
    for(int i = 0; i < len; ++i){
      int c = s[i] - 'a';
      if(!node[p][c]){
        node[p][c] = ++cnt;
      }
      p = node[p][c];
    }
    ++isval[p];
  }

  int find(string& s, int len) {
    int p = 0;
    for(int i = 0; i < len; ++i){
      int c = s[i] - 'a';
      if(!node[p][c]) return 0;
      p = node[p][c];
    }
    return isval[p];
  }
};
#endif
```

2 geometry

2.1 convex.cpp

```cpp
#ifdef IGNORE_THIS_FILE
  struct Point {
     ll x,y;
  };
  auto andrew = [](vector<Point>& p) -> vector<Point> {    // 传入下标从零开始的点数组，返回凸包数组
    auto cmp = [](Point &a, Point &b) -> bool {
      if(a.x != b.x) return a.x < b.x;
      return a.y < b.y;
    };
    auto cross = [](Point &u, Point &v, Point &w) -> bool {
      ll x1 = u.x - v.x, y1 = u.y - v.y;
      ll x2 = w.x - v.x, y2 = w.y - v.y;
      return x1 * y2 - x2 * y1 > 0; //如果不希望在凸包的边上有输入点。把 > 改成 >=
    };
    sort(p.begin(), p.end(), cmp);
    int n = p.size(), m = 0;
    vector<Point> res(n + 1);
    for(int i = 0; i < n; ++i){
      while(m > 1 && !cross(res[m - 1],res[m - 2], p[i])) --m;
      res[m++] = p[i];
    }
    int kk = m;
    for(int i = n - 2; i >= 0; i--){
      while(m > kk && !cross(res[m - 1], res[m - 2], p[i])) --m;
      res[m++] = p[i];
    }
    if(n > 1) --m;//凸包有 m 个顶点
    res.erase(res.begin() + m, res.end());
    return res;
  };
#endif
```

3 graph

3.1 Dinic.cpp

```cpp
#ifdef IGNORE_THIS_FILE
constexpr int N = 500;
constexpr ll INF = 0x3fffffffffffff;
struct edge {
    int from, to;
    ll can_flow;
    edge(int f, int t, ll can_f) : from(f), to(t), can_flow(can_f){};
};
struct Dinic {
    vector<edge> e;
    vector<int> G[N];
    int dep[N], cur[N];
    int n, m;
    void init(int n) {
        this->n = n;
        for(int i = 0; i <= n; ++i) G[i].clear();
        e.clear();
    }

    void addedge(int from, int to, ll cap) {
        e.emplace_back(from ,to, cap);
        e.emplace_back(to, from, 0);
        m = e.size();
        G[from].push_back(m - 2);
        G[to].push_back(m - 1);
    }

    bool bfs(int S, int T) {
        queue<int> q;
        memset(dep, 0, sizeof(int) * (n + 1));

        dep[S] = 1;
        q.push(S);
        while (q.size()) {
            int u = q.front();
            q.pop();
            for(int i = 0; i < G[u].size(); ++i){
                int id = G[u][i];
                const auto&[from, to, can_flow] = e[id];
                if ((!dep[to]) && can_flow) {
                    dep[to] = dep[u] + 1;
                    q.push(to);
                }
            }
        }
        return dep[T];
    }

    ll dfs(int u, int T, ll last_flow) {
        if (u == T || !last_flow) return last_flow;
        ll flow = 0;
        ll f;
        for (int& i = cur[u]; i < G[u].size(); ++i) {
            int id = G[u][i];
            const auto&[from, to, can_flow] = e[id];
            if (dep[u] + 1 == dep[to] && (f = dfs(to, T, min(last_flow, can_flow))) > 0) {
                e[id].can_flow -= f;
                e[id ^ 1].can_flow += f;
                flow += f;
                last_flow -= f;
                if (!last_flow) break;
            }
        }
        return flow;
      }

    ll dinic(int S, int T) {
        ll maxflow = 0;
        while (bfs(S, T)) {
            memset(cur, 0, sizeof(cur));
            maxflow += dfs(S, T, INF);
        }
        return maxflow;
    }
};
#endif
```

3.2 EK.cpp

```cpp
#ifdef IGNORE_THIS_FILE
constexpr int N = 250;
constexpr ll INF = 0x3fffffffffffff;
struct Edge {
    int from, to;
    ll can_flow;
    Edge(int u, int v, ll can_f) : from(u), to(v), can_flow(can_f) {}
};
struct EK {
    int n, m;              // n: 点数, m: 边数
    vector<Edge> e;    // e: 所有边的集合
    vector<int> G[N];  // G: 点 x -> x 的所有边在 e 中的下标
    ll a[N];      // a: 点 x -> BFS 过程中最近接近点 x 的边给它的最大流
    int p[N];     // p: 点 x -> BFS 过程中最近接近点 x 的边

    void init(int n) {
        for (int i = 0; i <= n; i++) G[i].clear();
        e.clear();
    }

    void AddEdge(int from, int to, ll cap) {
        e.emplace_back(from, to, cap);
        e.emplace_back(to, from, 0);
        m = e.size();
        G[from].push_back(m - 2);
        G[to].push_back(m - 1);
    }

    ll Maxflow(int s, int t) {
        ll max_flow = 0;
        while(true) {
            memset(a, 0, sizeof(a));
            queue<int> qu;
            qu.push(s);
            a[s] = INF;
            while (!qu.empty()) {
                int u = qu.front();
                qu.pop();
                for (int i: G[u]) {  // 遍历以 u 作为起点的边
                    const auto& [from, to, can_flow] = e[i];
                    if (!a[to] && can_flow) {
                        p[to] = i;  // e[i] 是最近接近点 to 的边
                        a[to] = min(a[u], can_flow);  // 最近接近点 to 的边赋给它的流
                        qu.push(to);
                    }
                }
                if (a[t]) break;  // 如果汇点接受到了流，就退出 BFS
            }
            if (!a[t])
                break;  // 如果汇点没有接受到流，说明源点和汇点不在同一个连通分量上
            for (int u = t; u != s; u = e[p[u]].from) {  // 通过 u 追寻 BFS 过程中 s -> t 的路径
                e[p[u]].can_flow -= a[t];      // 减少路径上边的 can_flow 值
                e[p[u] ^ 1].can_flow += a[t];  // 增加反向路径的 can_flow 值
            }
            max_flow += a[t];
        }
        return max_flow;
    }
};
#endif
```

3.3 ISAP.cpp

```
#ifdef IGNORE_THIS_FILE
const int N = 1000;
const ll INF = 0x3fffffffffff;
struct Edge {
    int from, to;
    ll can_flow;
    Edge(int f, int t, ll cap) : from(f), to(t), can_flow(cap){};
};
struct ISAP {
    int n, m;
    int s, t;
    vector<int> G[N];
    vector<Edge> e;
    int dep[N], gap[N], cur[N];

    void init(int _n, int _s, int _t) {
        this->n = _n, this->s = _s, this->t = _t;
        for(int i = 0; i <= n; ++i){
            G[i].clear();
        }
    }

    void add_edge(int u, int v, ll cap) {
        e.emplace_back(u, v, cap);
        e.emplace_back(v, u, 0);
        m = e.size();
        G[u].push_back(m - 2);
        G[v].push_back(m - 1);
    }

    void bfs() {
        memset(dep, 0, sizeof(dep));
        queue<int> q;
        q.push(t);
        dep[t] = 1;
        gap[1] = 1;
        while (!q.empty()) {
            int u = q.front();
            q.pop();
            for (int i : G[u]) {
                int v = e[i].to;
                if (!dep[v]) {
                    dep[v] = dep[u] + 1;
                    gap[dep[v]]++;
                    q.push(v);
                }
            }
        }
    }

    ll dfs(int u, ll flow) {
        if (u == t || !flow) return flow;
        ll used = 0;
        for (int &i = cur[u]; i < G[u].size(); ++i) {
            int id = G[u][i];
            auto&[from, to, can_flow] = e[id];
            if (can_flow && dep[u] == dep[to] + 1) {
                ll tmp = dfs(to, min(flow - used, can_flow));
                if (tmp) {
                    e[id].can_flow -= tmp;
                    e[id ^ 1].can_flow += tmp;
                    used += tmp;
                }
                if (used == flow) return used;
```

```cpp
            }
        }

        --gap[dep[u]];
        if (!gap[dep[u]]) dep[s] = n + 1;
        ++dep[u];
        ++gap[dep[u]];
        return used;
    }

    ll isap() {
        ll max_flow = 0;
        bfs();
        while (dep[s] <= n) {
            memset(cur, 0, sizeof(cur));
            max_flow += dfs(s, INF);
        }
        return max_flow;
    }
};
#endif
```

3.4 Johnson.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;
using ll = long long;
const ll N = 3e3 + 10;
ll h[N];
struct edge{
  ll u;
  ll w;
  ll t;
};
vector<edge> v;
struct node{
  ll u;
  ll t;
  bool operator<(const node& other) const{
    return t < other.t;
  }
  bool operator>(const node& other) const{
    return t > other.t;
  }
};
priority_queue<node, vector<node>, greater<node>> q;
vector<node> s[N];
bool bellmanford(ll n){
  bool flag = false;
  for(ll i = 1; i <= n; ++i){
    flag = false;
    for(auto j: v){
      if(h[j.w] > h[j.u] + j.t){
        h[j.w] = h[j.u] + j.t;
        flag = true;
      }
    }
    if(!flag){
      break;
    }
  }
  return flag;
}
ll w[N][N];
void  Dijskal(ll st, ll n){
  vector<ll> isval(n + 10);
  w[st][st] = 0;
  priority_queue<node, vector<node>, greater<node>> q;
  q.emplace(st, 0);
  while(!q.empty()){
    auto u = q.top();
    q.pop();
    if(isval[u.u])
      continue;
    isval[u.u] = 1;
    for(auto i: s[u.u]){
      if(w[st][i.u] > w[st][u.u] + i.t){
        w[st][i.u] = w[st][u.u] + i.t;
        q.emplace(i.u, w[st][i.u]);
      }
    }
  }
  for(int i = 1; i <= n; ++i){
    w[st][i] = w[st][i] - h[st] + h[i];
  }
}
int main(){
  ll n, m;
  cin >> n >> m;
  for(ll i = 0; i < m; ++i){
    ll u, w, t;
    cin >> u >> w >> t;
    v.emplace_back(u, w, t);
  }
  for(ll i = 1; i <= n; ++i){
    v.emplace_back(0, i, 0);
  }
  if(bellmanford(n)){
    cout << -1;
    return 0;
  }
  for(auto& i: v){
    auto&[u, w, t] = i;
    t += h[u] - h[w];
  }
  for(ll i = 0; i < m; ++i){
    auto&[u, w, t] = v[i];
    s[u].emplace_back(w, t);
  }
  memset(w, 0x3f, sizeof(w));
  for(ll i = 1; i <= n; ++i){
    Dijskal(i, n);
  }
  for(ll i = 1; i <= n; ++i){
    ll sum = 0;
    for(ll j = 1; j <= n; ++j){
      if(w[i][j] > ll(1e9)){
        sum += j * ll(1e9);
      }
      else
        sum += w[i][j] * j;
    }
    cout << sum << '\n';
  }
}
```

3.5 LCA.cpp

```cpp
#ifdef IGNORE_THIS_FILE
  vector<vector<int> > a(n + 1, vector<int>(20)), v(n + 1);
  vector<int> dep(n + 1);
  auto build = [&](int u, int fa, auto&& self) -> void {
    dep[u] = dep[fa] + 1, a[u][0] = fa;
    for(int i = 1; i <= 19; ++i)
      a[u][i] = a[a[u][i - 1]][i - 1];
    for(int i: v[u]){
      if(i == fa) continue;
      self(i, u, self);
    }
  };
  auto lca = [&](int x, int y) -> int {
    if(dep[y] > dep[x]) swap(x, y);
    for(int i = 19; i >= 0; --i){
      if(dep[a[x][i]] >= dep[y])
        x = a[x][i];
    }
    if(x == y) return x;
    for(int i = 19; i >= 0; --i){
      if(a[x][i] != a[y][i])
        x = a[x][i], y = a[y][i];
    }
    return a[x][0];
  };
#endif



```

3.6 lcd.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const ll N = 1e6 + 10;
vector<ll> v[N];
ll fa[N], dep[N], son[N], sz[N], top[N];
ll dfn[N], rid[N];   //dfn 序列, dfn-> 标号
ll bot[N]; //维护子树 dfn 序结束编号
ll vl[N];
ll n, m, r, mod;
ll cnt = 0;

void dfs1(ll u, ll far){
  fa[u] = far, dep[u] = dep[far] + 1, sz[u] = 1;
  for(auto i: v[u]){
    if(i == far)
      continue;
    dfs1(i, u);
    sz[u] += sz[i];
    if(sz[son[u]] < sz[i])
      son[u] = i;
  }
}
void dfs2(ll u, ll head){
  top[u] = head;
  dfn[u] = ++cnt;
  rid[cnt] = u;
  if(!son[u]){
    bot[u] = cnt;
    return;
  }
  dfs2(son[u], head);
  for(auto i: v[u]){
    if(i == fa[u] || son[u] == i)
      continue;
    dfs2(i, i);
  }
  bot[u] = cnt;
}
ll sm[4 * N];
ll lz[4 * N];
void pushup(ll node){
  sm[node] = sm[node << 1] + sm[node << 1 | 1];
  sm[node] %= mod;
}

void pushdown(ll node, ll l, ll r, ll mid){
  if(lz[node]){
    lz[node << 1] += lz[node];
    lz[node << 1] %= mod;
    lz[node << 1 | 1] += lz[node];
    lz[node << 1 | 1] %= mod;
    sm[node << 1] += lz[node] * (mid - l + 1ll);
    sm[node << 1] %= mod;
    sm[node << 1 | 1] += lz[node] * (r - mid);
    sm[node << 1 | 1] %= mod;
    lz[node] = 0;
  }
}
void build(ll node, ll l, ll r){
  if(l == r){
    sm[node] = vl[rid[l]];
    sm[node] %= mod;
    return;
  }
  ll mid = (l + r) >> 1;
  build(node << 1, l, mid);
  build(node << 1 | 1, mid + 1, r);
  pushup(node);
}

void update(ll node, ll l, ll r, ll ql, ll qr, ll val){
  if(ql <= l && r <= qr){
    sm[node] += val * (r - l + 1ll);
    sm[node] %= mod;
    lz[node] += val;
    lz[node] %= mod;
    return;
  }
  ll mid = (l + r) >> 1;
  pushdown(node, l, r, mid);
  if(ql <= mid)
    update(node << 1, l, mid, ql, qr, val);
  if(qr > mid)
    update(node << 1 | 1, mid + 1, r, ql, qr, val);
  pushup(node);
}

ll query(ll node, ll l, ll r, ll ql, ll qr){
  if(ql <= l && r <= qr){
    return sm[node] % mod;
  }
  ll mid = (l + r) >> 1;
  pushdown(node, l, r, mid);
  ll sum = 0;
  if(ql <= mid)
    sum += query(node << 1, l, mid, ql, qr);
  if(qr > mid)
    sum += query(node << 1 | 1, mid + 1, r, ql, qr);
  return sum % mod;
}

void add_path(ll u, ll v, ll val){
  while(top[u] != top[v]){
    if(dep[top[u]] < dep[top[v]])
```

```
      swap(u, v);
    update(1, 1, n, dfn[top[u]], dfn[u], val);
    u = fa[top[u]];
  }
  if(dep[u] < dep[v]) swap(u, v);
  update(1, 1, n, dfn[v], dfn[u], val);
}

ll get_path(ll u, ll v){
  ll sum = 0;
  while(top[u] != top[v]){
    if(dep[top[u]] < dep[top[v]])
      swap(u, v);
    // sum += query(1, 1, n, dfn[top[u]], dfn[u]);
    // sum %= mod;
    u = fa[top[u]];
  }
  if(dep[u] < dep[v]) swap(u, v);
  sum += query(1, 1, n, dfn[v], dfn[u]);
  return sum % mod;
}

void update_root(ll x, ll val){
  update(1, 1, n, dfn[x], bot[x], val);
}
ll get_root(ll x){
  return query(1, 1, n, dfn[x], bot[x]);
}
int main(){
  cin >> n >> m >> r >> mod;
  for(ll i = 1; i <= n; ++i){
    cin >> vl[i];
  }
  for(ll i = 1; i < n; ++i){
    ll x, y;
    cin >> x >> y;
    v[x].push_back(y);
    v[y].push_back(x);
  }
  dfs1(r, 0);
  dfs2(r, r);
  build(1, 1, n);
  for(ll i = 1; i <= m; ++i){
    ll op, x, y, z;
    cin >> op;
    if(op == 1){
      cin >> x >> y >> z;
      add_path(x, y, z);
    }
    else if(op == 2){
      cin >> x >> y;
      cout << get_path(x, y) << '\n';
    }
    else if(op == 3){
      cin >> x >> z;
      update_root(x, z);
    }
    else{
      cin >> x;
      cout << get_root(x) << '\n';
    }
  }
}
```

3.7 lcd2.cpp

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const ll N = 2e5 + 10;
ll fa[N], ht[N], top[N];
pair<ll, ll> son[N];
ll len[N];
vector<pair<ll, ll> > v[N];


void dfs1(ll u, ll far){
  fa[u] = far;
  for(auto [i, w]: v[u]){
    if(i == far)
      continue;
    dfs1(i, u);
    ht[u] = max(ht[u], ht[i] + w);
    if(ht[son[u].first] + son[u].second < ht[i] + w)
      son[u] = {i, w};
  }
}

ll ans[N];
void dfs2(ll u, ll head){
  top[u] = head;
  if(!son[u].first){
    return;
  }
  len[son[u].first] = len[u] + son[u].second;
  dfs2(son[u].first, head);
  for(auto [i, w]: v[u]){
    if(i == fa[u] || son[u].first == i)
      continue;
    len[i] = w;
    dfs2(i, i);
  }
}
int main(){
  ll n;
  cin >> n;
```

```cpp
  for(ll i = 1; i < n; ++i){
    ll x, y, z;
    cin >> x >> y >> z;
    v[x].emplace_back(y, z);
    v[y].emplace_back(x, z);
  }
  dfs1(1, 0);
  dfs2(1, 1);
  for(ll i = 1; i <= n; ++i){
    ans[top[i]] = max(ans[top[i]], len[i]);
  }
  ll sum = 0;
  sort(ans + 1, ans + 1 + n, [](const auto x, const auto y){return x > y;});
  for(ll i = 1; i <= n; ++i){
    sum += ans[i] * 2ll;
    cout << sum << ' ';
  }

}
```

3.8 tarjan.cpp

```cpp
#ifdef IGNORE_THIS_FILE
  int dfn_cnt = 0, scc_cnt = 0;
  vector<int> dfn(n + 1), low(n + 1), scc_id(n + 1);
  stack<int> s;
  vector<bool> in_stack(n + 1);
  vector<vector<int> > v(n + 1), scc(n + 1);
  // scc_id 是每个节点所属于的 scc 编号, scc 是这个编号下的所有节点
  auto tarjan = [&](int u, auto&& self) -> void {
    dfn[u] = low[u] = ++dfn_cnt;
    s.push(u), in_stack[u] = true;
    for(int i: v[u]){
      if(!dfn[i]){
        self(i, self);
        low[u] = min(low[u], low[i]);
      }
      else if(in_stack[i])
        low[u] = min(low[u], dfn[i]);
    }
    if(dfn[u] == low[u]){
      int tp;
      ++scc_cnt;
      do{
        tp = s.top();
        scc_id[tp] = scc_cnt;
        scc[scc_cnt].push_back(tp);
        in_stack[tp] = false;
        s.pop();
      } while(tp != u);
    }
  };
```

```
#endif
```

3.9 tree divide and conquer.cpp

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N = 1e4 + 10;
vector<pair<int, int>> v[N];
int n, m;
bool vis[N];
int siz[N], dep[N];
int st[1000 * N];
int wt[N];
int ans[101];
int get(int u, int far, int& num) { // 求子树的重心
    siz[u] = 1;
    wt[u] = 0;
    ++num;
    int root = -1;
    for(auto&[i, w]: v[u]){
        if(i == far || vis[i]) continue;
        root = max(root, get(i, u, num));
        siz[u] += siz[i];
        wt[u] = max(wt[u], siz[i]);
    }
    wt[u] = max(wt[u], num - siz[u]);
    if(wt[u] <= num / 2){
        return u;
    }
    return root;
}

void cal(int u, int far, int w, vector<int> &lst) { // 向下暴力递归计算
    dep[u] = dep[far] + w;
    lst.push_back(u);
    for (auto &[i, w] : v[u]) {
        if (vis[i] || i == far) continue;
        cal(i, u, w, lst);
    }
}

void dfs2(int u, vector<int>& cur) {
    int num = 0;
    int root = get(u, 0, num);
    vis[root] = true;
    st[0] = 1;
    dep[root] = 0;
    vector<vector<int> > lst;
    for (auto &[i, w] : v[root]) {
        if (vis[i]) continue;
        vector<int> tmp;
```

```cpp
        cal(i, root, w, tmp);
        lst.push_back(tmp);
        for (auto &node : tmp) {
            for(int k = 0; k < cur.size(); ++k){
                if (cur[k] >= dep[node]) {
                    ans[k] += st[cur[k] - dep[node]];
                }
            }
        }

        for (auto &node : tmp) {
            if(dep[node] <= 1e7){
                ++st[dep[node]];
            }
        }
    }
    for(auto& i: lst){
        for(auto& j: i){
            if(dep[j] <= 1e7)
                st[dep[j]] = 0;
        }
    }
    st[0] = 0;
    for (auto &[i, w] : v[root]) {
        if (vis[i]) continue;
        dfs2(i, cur);
    }
}

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    cin >> n >> m;
    for (int i = 1; i < n; ++i) {
        int u, v_, w;
        cin >> u >> v_ >> w;
        v[u].emplace_back(v_, w);
        v[v_].emplace_back(u, w);
    }
    vector<int> cur;
    for(int i = 0; i < m; ++i){
        int x;
        cin >> x;
        cur.push_back(x);
    }
    dfs2(1, cur);
    for(int i = 0; i < m; ++i){
        cout << (ans[i] ? "AYE\n" : "NAY\n");
    }
    return 0;
}
```

4 heading

4.1 debug.h

```
#include <bits/stdc++.h>
#define typet typename T
#define typeu typename U
#define types typename... Ts
#define tempt template <typet>
#define tempu template <typeu>
#define temps template <types>
#define tandu template <typet, typeu>

tandu std::ostream& operator<<(std::ostream& os, const std::pair<T, U>& p) {
return os << '<' << p.ff << ',' << p.ss << '>';
}
template <
typet, typename = decltype(std::begin(std::declval<T>())),
typename = std::enable_if_t<!std::is_same_v<T, std::string>>>
std::ostream& operator<<(std::ostream& os, const T& c) {
auto it = std::begin(c);
if (it == std::end(c)) return os << "{}";
for (os << '{' << *it; ++it != std::end(c); os << ',' << *it);
return os << '}';
}
#define debug(arg...) \
do { \
 std::cerr << "[" #arg "] :"; \
 dbg(arg); \
}while(false)

temps void dbg(Ts... args) {
(..., (std::cerr << ' ' << args));
std::cerr << '\n';
}

```

4.2 duipai.cpp

```
#ifdef IGNORE_THIS_FILE
  system("g++ -std=c++2a wa.cpp -o/wa");
  system("g++ -std=c++2a ac.cpp -o/ac");
  system("g++ -std=c++2a gen.cpp -o/gen");
  for(int i = 1; i <= 50; i++){
    std::cerr << "Test" << i << " : ";
    system("./gen > gen.in");
    system("./ac < gen.in > ac.out");
    system("./wa < gen.in > wa.out");
    if (system("diff ac.out wa.out")) {
      std::cerr << "ERR\n";
      return 0;
    }
    std::cerr << "AC\n";
  }
#endif
```

4.3 heading.cpp

```
#include <bits/stdc++.h>
using namespace std;
using ll = long long;
using i128 = __int128;
#define ff first
#define ss second
#include "debug.h"
constexpr int mod = 998244353;
constexpr ll INF = 1e18;
constexpr double pi = 3.141592653589793;
constexpr double eps = 1e-6;


```

5 math

5.1 Eratost.cpp

```cpp
#ifdef IGNORE_THIS_FILE
  vector<int> sieve(int n){
    vector<bool> is_prime(n + 1);
    vector<int> prime;
    for(int i = 2; i <= n; ++i){
      is_prime[i] = true;
    }
    for(int i = 2; i * i <= n; ++i){
      if(is_prime[i]){
        for(int j = i * i; j <= n; j += i)
          is_prime[j] = false;
      }
    }
    for(int i = 2; i <= n; ++i){
      if(is_prime[i])
        prime.push_back(i);
    }
    return prime;
  }
#endif
```

5.2 Euler.cpp

```cpp
#ifdef IGNORE_THIS_FILE
  // vector<int> fac(n + 1);
  vector<int> sieve(int n){
    vector<int> prime;
    vector<bool> no_prime(n + 1);
    for(int i = 2; i <= n; ++i){
      if(!no_prime[i]){
        prime.push_back(i);
        // fac[i] = i;
      }
      for(int j: prime){
        if(j * i > n) break;
        no_prime[j * i] = true;
        // fac[j * i] = j;
        if(i % j == 0) break;
      }
    }
    return prime;
  }
#endif
```

5.3 FFT.cpp

```
#ifdef IGNODE_THIS_FILE
/*
 * 做 FFT
 * len 必须是 2^k 形式
 * on == 1 时是 DFT, on == -1 时是 IDFT
 */
void fft(Complex y[], int len, int on) {
    // 位逆序置换
    change(y, len);
    // 模拟合并过程，一开始，从长度为一合并到长度为二，一直合并到长度为 len。
    for (int h = 2; h <= len; h <<= 1) {
      // wn: 当前单位复根的间隔: w^1_h
      Complex wn(cos(2 * PI / h), sin(on * 2 * PI / h));
      // 合并，共 len / h 次。
      for (int j = 0; j < len; j += h) {
        // 计算当前单位复根，一开始是 1 = w^0_n, 之后是以 wn 为间隔递增: w^1_n
        // ...
        Complex w(1, 0);
        for (int k = j; k < j + h / 2; k++) {
          // 左侧部分和右侧是子问题的解
          Complex u = y[k];
          Complex t = w * y[k + h / 2];
          // 这就是把两部分分治的结果加起来
          y[k] = u + t;
          y[k + h / 2] = u - t;
          // 后半个 「step」中的  一定和 「前半个」中的成相反数
          // 「红圈」上的点转一整圈「转回来」，转半圈正好转成相反数
          // 一个数相反数的平方与这个数自身的平方相等
          w = w * wn;
        }
      }
    }
    // 如果是 IDFT, 它的逆矩阵的每一个元素不只是原元素取倒数，还要除以长度 len。
    if (on == -1) {
      for (int i = 0; i < len; i++) {
        y[i].x /= len;
        y[i].y /= len;
      }
    }
  }

#endif
```

5.4 Gause$_X$or.cpp

```
#ifdef IGNODE_THIS_FILE

std::bitset<1010> matrix[2010];  // matrix[1~n]: 增广矩阵，0 位置为常数
std::vector<bool> GaussElimination(int n, int m) {
  // n 为未知数个数, m 为方程个数, 返回方程组的 (多解 / 无解返回一个空的 vector)
  for (int i = 1; i <= n; i++) {
    int cur = i;
    while (cur <= m && !matrix[cur].test(i)) cur++;
    if (cur > m) return std::vector<bool>(0);
    if (cur != i) swap(matrix[cur], matrix[i]);
    for (int j = 1; j <= m; j++)
      if (i != j && matrix[j].test(i)) matrix[j] ^= matrix[i];
  }
  std::vector<bool> ans(n + 1);
  for (int i = 1; i <= n; i++) ans[i] = matrix[i].test(0);
  return ans;
}
#endif
```

5.5 NTT.cpp

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

int read() {
  int x = 0, f = 1;
  char ch = getchar();
  while (ch < '0' || ch > '9') {
    if (ch == '-') f = -1;
    ch = getchar();
  }
  while (ch <= '9' && ch >= '0') {
    x = 10 * x + ch - '0';
    ch = getchar();
  }
  return x * f;
}

void print(int x) {
  if (x < 0) putchar('-'), x = -x;
  if (x >= 10) print(x / 10);
  putchar(x % 10 + '0');
}

constexpr int N = 300100, P = 998244353;

int qpow(int x, int y) {
  int res(1);
  while (y) {
    if (y & 1) res = 1ll * res * x % P;
    x = 1ll * x * x % P;
    y >>= 1;
  }
  return res;
}

```

```cpp
int r[N];

void ntt(int *x, int lim, int opt) {
  int i, j, k, m, gn, g, tmp;
  for (i = 0; i < lim; ++i)
    if (r[i] < i) swap(x[i], x[r[i]]);
  for (m = 2; m <= lim; m <<= 1) {
    k = m >> 1;
    gn = qpow(3, (P - 1) / m);
    for (i = 0; i < lim; i += m) {
      g = 1;
      for (j = 0; j < k; ++j, g = 1ll * g * gn % P) {
        tmp = 1ll * x[i + j + k] * g % P;
        x[i + j + k] = (x[i + j] - tmp + P) % P;
        x[i + j] = (x[i + j] + tmp) % P;
      }
    }
  }
  if (opt == -1) {
    reverse(x + 1, x + lim);
    int inv = qpow(lim, P - 2);
    for (i = 0; i < lim; ++i) x[i] = 1ll * x[i] * inv % P;
  }
}

int A[N], B[N], C[N];

char a[N], b[N];

int main() {
  int i, lim(1), n;
  scanf("%s", a);
  n = strlen(a);
  for (i = 0; i < n; ++i) A[i] = a[n - i - 1] - '0';
  while (lim < (n << 1)) lim <<= 1;
  scanf("%s", b);
  n = strlen(b);
  for (i = 0; i < n; ++i) B[i] = b[n - i - 1] - '0';
  while (lim < (n << 1)) lim <<= 1;
  for (i = 0; i < lim; ++i) r[i] = (i & 1) * (lim >> 1) + (r[i >> 1] >> 1);
  ntt(A, lim, 1);
  ntt(B, lim, 1);
  for (i = 0; i < lim; ++i) C[i] = 1ll * A[i] * B[i] % P;
  ntt(C, lim, -1);
  int len(0);
  for (i = 0; i < lim; ++i) {
    if (C[i] >= 10) len = i + 1, C[i + 1] += C[i] / 10, C[i] %= 10;
    if (C[i]) len = max(len, i);
  }
  while (C[len] >= 10) C[len + 1] += C[len] / 10, C[len] %= 10, len++;
  for (i = len; ~i; --i) putchar(C[i] + '0');
  puts("");
  return 0;
}
```

5.6 pollar-rho.cpp

```
#ifdef IGNORE_THIS_FILE
  using ll = long long;
  using ull = unsigned long long;
  bool is_prime(ull n) {
    if(n == 2) { return true; }
    if(n % 2 == 0) { return false; }
    auto internal_pow = [&](ull x, ull y) {
      ull r = 1;
      __uint128_t c = x;
      for(; y; y >>= 1, c = c * c % n) {
        if(y & 1) { r = __uint128_t(r) * c % n; }
      }
      return r;
    };
    auto MillerRabin = [&](ull a) {
      if(n <= a) { return true; }
      int e = __builtin_ctzll(n - 1);
      ull z = internal_pow(a, (n - 1) >> e);
      if(z == 1 || z == n - 1) { return true; }
      while(--e) {
        z = __uint128_t(z) * z % n;
        if(z == 1) { return false; }
        if(z == n - 1) { return true; }
      }
      return false;
    };
    vector<ull> cur;
    if(n < 4759123141) cur = vector<ull>{2, 7, 61};
    else cur = vector<ull>{2, 325, 9375, 28178, 450775, 9780504, 1795265022};
    return all_of(cur.begin(), cur.end(), [&](auto x) { return MillerRabin(x); });
  }

  struct Montgomery {
    ull mod, R;
    public:
    Montgomery(ull n): mod(n), R(n) {
      for(int i = 0; i < 5; i++) { R *= 2 - mod * R; }
    }
    ull fma(ull a, ull b, ull c) const {
      const __uint128_t d = __uint128_t(a) * b;
      const ull e = c + mod + (d >> 64);
      const ull f = ull(d) * R;
      const ull g = (__uint128_t(f) * mod) >> 64;
      return e - g;
    }
    ull mul(ull a, ull b) const { return fma(a, b, 0); }
  };
  ull PollardRho(ull n) {
```

```cpp
  if(n % 2 == 0) { return 2; }
  const Montgomery m(n);
  constexpr ull C1 = 1, C2 = 2, M = 512;
  ull Z1 = 1, Z2 = 2;
retry:
  ull z1 = Z1, z2 = Z2;
  for(unsigned k = M;; k <<= 1) {
    const ull x1 = z1 + n, x2 = z2 + n;
    for(unsigned j = 0; j < k; j += M) {
      const ull y1 = z1, y2 = z2;
      ull q1 = 1, q2 = 2;
      z1 = m.fma(z1, z1, C1), z2 = m.fma(z2, z2, C2);
      for(unsigned i = 0; i < M; i++) {
        const ull t1 = x1 - z1, t2 = x2 - z2;
        z1 = m.fma(z1, z1, C1), z2 = m.fma(z2, z2, C2);
        q1 = m.mul(q1, t1), q2 = m.mul(q2, t2);
      }
      q1 = m.mul(q1, x1 - z1), q2 = m.mul(q2, x2 - z2);
      const ull q3 = m.mul(q1, q2), g3 = gcd(n, q3);
      if(g3 == 1) { continue; }
      if(g3 != n) { return g3; }
      const ull g1 = gcd(n, q1), g2 = gcd(n, q2);
      const ull C = g1 != 1 ? C1 : C2, x = g1 != 1 ? x1 : x2;
      ull z = g1 != 1 ? y1 : y2, g = g1 != 1 ? g1 : g2;
      if(g == n) {
        do {
          z = m.fma(z, z, C);
          g = gcd(n, x - z);
        } while(g == 1);
      }
      if(g != n) { return g; }
      Z1 += 2, Z2 += 2;
      goto retry;
    }
  }
}
vector<ull> PrimeFactorize(ull n) {
  vector<ull> r;
  auto rec = [&](auto &&rec, ull n, vector<ull> &r) -> void {
    if(n <= 1) { return; }
    if(is_prime(n)) {
      r.emplace_back(n);
      return;
    }
    const ull p = PollardRho(n);
    rec(rec, p, r);
    rec(rec, n / p, r);
  };
  rec(rec, n, r);
  sort(r.begin(), r.end());
  return r;
}
vector<pair<ll, ll>> Prime(ll n) {
    auto ans = PrimeFactorize(n);
```

```
      vector<pair<ll, ll>> cur;
      for(ll i = 0; i < ans.size(); ++i){
          ll e = 1;
          while(i + 1 < ans.size() && ans[i + 1] == ans[i]) ++e, ++i;
          cur.push_back({ans[i], e});
      }
      return cur;
  }
  // auto get_tot = [](auto &&get_tot, vector<pair<ll, ll>>& prime, vector<ll>& tot, int pos, ll val){
  //   if(pos == prime.size()){
  //       tot.push_back(val);
  //       return;
  //   }
  //   for(ll j = 0, sum = 1; j <= prime[pos].second; ++j, sum *= prime[pos].first){
  //       get_tot(get_tot, prime, tot, pos + 1, val * sum);
  //   }
  // };
#endif
```

5.7 Xor$_b$*ase.cpp*

```
#ifdef IGNORE_THIS_FILE
auto insert = [](ll x, vector<ll>& a) -> void {
  for(int i = 63; i >= 0; --i){
    if(!((x >> i) & 1)) continue;
    if(a[i]) x ^= a[i];
    else{
      for(int j = 0; j < i; ++j){
        if((x >> j) & 1)
          x ^= a[j];
      }
      for(int j = i + 1; j <= 63; ++j){
        if((a[j] >> i) & 1)
          a[j] ^= x;
      }
      a[i] = x;
    }
  }
};
#endif
```

6 sort

6.1 merge-sort.cpp

```
#ifdef IGNORE_THIS_TILE
  void merge_sort(int l, int r, vector<int>& a) {
    if(l == r)
      return;
    int mid = (l + r) >> 1;
    merge_sort(l, mid, a);
    merge_sort(mid + 1, r, a);
    auto merge = [](int l, int r, int mid, vector<int>& a) -> void {
      vector<int> b;
      int lp = l;
      int rp = r + 1;
      while(lp <= mid && rp <= r){
        if(a[lp] < a[rp])
          b.push_back(a[lp++]);
        else
          b.push_back(a[rp++]);
      }
      while(lp <= mid)
        b.push_back(a[lp++]);
      while(rp <= r)
        b.push_back(a[rp++]);
      copy(b.begin(), b.end(), a.begin() + l);
    };
    merge(l, r, mid, a);
  };
#endif
```

7 string

7.1 AC$_a$*utomation.cpp*

```cpp
#ifdef IGNORE_THIS_FILE
struct ACAutomaton {
    static const int N = 2e6 + 6;
    static const int maxn = 2e5 + 10;

    int tran[N][26] = {};
    vector<int> uid[maxn];
    int fail[N] = {};
    int ru_degree[maxn] = {};
    int dp[N] = {};
    int ans[maxn] = {};
    int tot = 0;

    void clear() {
        for(int i = 0; i < N; i++) {
            memset(tran[i], 0, sizeof(tran[i]));
            fail[i] = dp[i] = 0;
        }
        for(int i = 0; i < maxn; i++) uid[i].clear();
        memset(ru_degree, 0, sizeof(ru_degree));
        memset(ans, 0, sizeof(ans));
        tot = 0;
    }

    void insert(const string& s, int id) {
        int u = 0;
        for(char ch : s) {
            int c = ch - 'a';
            if(!tran[u][c]) tran[u][c] = ++tot;
            u = tran[u][c];
        }
        uid[u].push_back(id);
    }

    void build() {
        queue<int> q;
        for(int i = 0; i < 26; i++)
            if(tran[0][i]) q.push(tran[0][i]);

        while(!q.empty()) {
            int u = q.front();
            q.pop();
            for(int i = 0; i < 26; i++) {
                if(tran[u][i]) {
                    fail[tran[u][i]] = tran[fail[u]][i];
                    q.push(tran[u][i]);
                } else {
                    tran[u][i] = tran[fail[u]][i];
                }
            }
        }
    }

    void query(const string& t) {
        int u = 0;
        for(char ch : t) {
            u = tran[u][ch - 'a'];
            dp[u]++;
        }

        for(int i = 1; i <= tot; i++)
            ru_degree[fail[i]]++;

        queue<int> q;
        for(int i = 1; i <= tot; i++)
            if(!ru_degree[i]) q.push(i);

        while(!q.empty()) {
            int u = q.front();
            q.pop();
            dp[fail[u]] += dp[u];
            if(--ru_degree[fail[u]] == 0)
                q.push(fail[u]);
        }

        for(int i = 1; i <= tot; i++)
            for(int id : uid[i])
                ans[id] = dp[i];
    }
};

#endif
```

7.2 KMP.cpp

```cpp
#ifdef IGNORE_THIS_FILE


vector<int> prefix_function(string s) {
  int n = (int)s.length();
  vector<int> pi(n);
  for (int i = 1; i < n; i++) {
    int j = pi[i - 1];
    while (j > 0 && s[i] != s[j]) j = pi[j - 1];
    if (s[i] == s[j]) j++;
    pi[i] = j;
  }
  return pi;
}

```

```cpp

void compute_automaton(string s, vector<vector<int>>& aut) {
  s += '#';
  int n = s.size();
  vector<int> pi = prefix_function(s);
  aut.assign(n, vector<int>(26));
  for (int i = 0; i < n; i++) {
    for (int c = 0; c < 26; c++) {
      if (i > 0 && 'a' + c != s[i])
        aut[i][c] = aut[pi[i - 1]][c];
      else
        aut[i][c] = i + ('a' + c == s[i]);
    }
  }
}


#endif
```

7.3 Manacher.cpp

```cpp
#ifdef IGNORE_THIS_FILE

struct Manacher {
    string transformed;
    vector<int> d1;

    Manacher(const string& s) {
        // 预处理字符串，插入'#'
        transformed = "#";
        for (char c : s) {
            transformed += c;
            transformed += '#';
        }
        int n = transformed.size();
        d1.resize(n);
        // 计算每个中心点的最长回文半径
        int l = 0, r = -1;
        for (int i = 0; i < n; ++i) {
            int k = (i > r) ? 1 : min(d1[l + r - i], r - i + 1);
            while (i - k >= 0 && i + k < n && transformed[i - k] == transformed[i + k]) {
                k++;
            }
            d1[i] = k--;
            if (i + k > r) {
                l = i - k;
                r = i + k;
            }
        }
    }
};

```

```
#endif
```

7.4 PAM.cpp

```
#include <vector>
#include <string>
#include <array>
using namespace std;

struct Eertree {
    vector<array<int, 26>> tr;    // 转移数组
    vector<int> fail;             // 失配指针
    vector<int> len;              // 节点表示的回文串长度
    vector<int> ans;              // 存储每个位置的回文深度
    vector<int> depth;            // 节点深度
    int tot, last;                // 总节点数和当前最后节点

    Eertree() {
        tr.resize(2);
        tr[0].fill(0), tr[1].fill(0);
        fail.resize(2);
        len.resize(2);
        depth.resize(2);

        fail[0] = 1;  // 偶根失配指向奇根
        fail[1] = 0;  // 奇根失配指向偶根
        len[0] = 0;   // 偶根长度 0
        len[1] = -1;  // 奇根长度-1
        depth[0] = 0;
        depth[1] = 0;
        tot = 1;      // 已创建两个节点
        last = 1;     // 初始指向奇根
    }

    void insert(const string& s) {
        ans.resize(s.size());
        for (int i = 0; i < s.size(); i++) {
            int c = s[i] - 'a';
            int fa = get_fail(last, i, s);

            if (!tr[fa][c]) {
                create_node(fa, c, i, s);
            }
            last = tr[fa][c];
            ans[i] = depth[last];
        }
    }

    int get_fail(int pos, int idx, const string& s) {
        while (idx - len[pos] - 1 < 0 ||
               s[idx] != s[idx - len[pos] - 1]) {
            pos = fail[pos];
        }
        return pos;
    }

    void create_node(int fa, int c, int idx, const string& s) {
        int cur = ++tot;
        tr.resize(tot + 1);
        tr.back().fill(0);

        // 扩展关联数组
        fail.resize(tot + 1);
        len.resize(tot + 1);
        depth.resize(tot + 1);

        tr[fa][c] = cur;
        len[cur] = len[fa] + 2;

        // 设置失配指针
        int tmp = get_fail(fail[fa], idx, s);
        fail[cur] = (fa != 1) ? tr[tmp][c] : 0;

        // 计算节点深度
        depth[cur] = depth[fail[cur]] + 1;
    }
};
```

7.5 $z_function.cpp$

```
#ifdef IGNORE_THIS_FILE
vector<int> z_function(string s) {
    int n = (int)s.size();
    vector<int> z(n);
    for(int i = 1, l = 0, r = 0; i < n; ++i) {
        if(i <= r && z[i - l] < r - i + 1) {
            z[i] = z[i - l];
        }
        else {
            z[i] = max(0, r - i + 1);
            while(i + z[i] < n && s[z[i]] == s[i + z[i]]) ++z[i];
        }
        if(i + z[i] - 1 > r) l = i, r = i + z[i] - 1;
    }
}
#endif
```