Team notebook

CNH.yehale

November 8, 2025

Contents

1 Bay

1.1 fastInput

```
const int BUFFER_SIZE = 1 << 20;
static char input_buffer[BUFFER_SIZE];
static int input_pos = 0, input_len = 0;

inline char nextChar() {
    if (input_pos == input_len) {
        input_pos = 0;
        input_len = (int)fread(input_buffer, 1, BUFFER_SIZE,
              stdin);
        if (input_len == 0) return EOF;
    }
    return input_buffer[input_pos++];
}

inline void fast (int &x) {
    x = 0;
    char c;
    bool neg = false;

    do {
        c = nextChar();
    } while (c != '-' && (c < '0' || c > '9'));

    if (c == '-') {
        neg = true;
        c = nextChar();
    }
    while (c >= '0' && c <= '9') {
        x = x * 10 + (c - '0');
        c = nextChar();
    }

    if (neg) x = -x;
}

inline void fast (string &s) {
    s.clear();
    char c;
    do {
        c = nextChar();
    } while (c <= ' ' && c != EOF);

    while (c > ' ' && c != EOF) {
        s.push_back(c);
        c = nextChar();
    }
}
```

1.2 gen

```
#include <bits/stdc++.h>

#define task    "BriantheCrab"

#define int   long long
```

```cpp
#define pii   pair <int, int>
#define fi    first
#define se    second
#define szf   sizeof
#define sz(s)  (int)((s).size())
#define all(v)  (v).begin(), (v).end()

using namespace std;

mt19937_64 rd (chrono :: steady_clock :: now
      ().time_since_epoch ().count ());

long long Rand (long long l, long long r)
{
    return uniform_int_distribution <long long> (l, r) (rd);
}

void gen ()
{
    freopen ("bai1.inp", "w", stdout);
    int n = Rand(5, 5e3);
    cout << n << '\n';
    for (int i = 1; i <= n; i ++) {
        cout << Rand (1, 2e2) << ' ';
    }
}

signed main ()
{
    gen ();
    return 0;
}
```

1.3 huongdan

```
--- Ubuntu Shortcuts ---
Ctrl + Alt + T       :  M  terminal
Ctrl + Shift + T     :  M  tab terminal  mi
Ctrl + C             :  Dng   lnh   ang   chy
Ctrl + D             :   ng    xut  terminal / thot shell
Ctrl + L             :  Xa   mn   hnh  terminal
Ctrl + Shift + C     : Copy trong terminal
Ctrl + Shift + V     : Paste trong terminal
Alt + Tab            :  Chuyn   gia   cc   ng   dng
Alt + F4             :   ng    ca    s    hin    ti
Ctrl + Alt + L       :  Kha   mn   hnh
Super (Windows key)  :  M  dash / tm   kim    ng   dng
Ctrl + Alt + arrow key : Chuyn workspace
Ctrl + Shift + N     :  To  folder  mi  trong file manager
Ctrl + H             :  Hin / n   file   n  trong file manager

--- Compile C++ trn Linux ---
g++ filename.cpp -o outputfile   : Compile C++ ra executable
chmod +x outputfile               : Cho  php   chy  file
      executable
./outputfile                      :  Chy  file executable
g++ -O2 filename.cpp -o outputfile : Compile vi  ti   u   ha
g++ -Wall filename.cpp -o outputfile : Hin  th   tt    c  warning
g++ -std=c++17 filename.cpp -o outputfile : Compile theo chun
      C++17

--- Compile C trn Linux ---
gcc filename.c -o outputfile      : Compile C ra executable
chmod +x outputfile               : Cho  php   chy  file
      executable
./outputfile                      :  Chy  file executable

--- Notes ---
-  Trn  Linux, "exe"  thng    l  file  nh   phn   chy   trc
      tip , khng   cn    ui   .exe
-  Lnh  chmod +x  l   bt    buc    nu    mun    chy   file sau khi
      compile
```

1.4 Multithreads

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

// ================= TEMPLATE MULTITHREAD =================
// Constants
const int MAX_THREADS = 4; // number of threads

// Worker function example: compute sum of a segment
void worker(const vector<int> &a, int l, int r, long long &res)
      {
    long long sum = 0;
    for (int i = l; i < r; i++)
        sum += a[i];
    res = sum;
}

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

    // Input example
    int n;
    cin >> n;
    vector<int> a(n);
    for (int i = 0; i < n; i++)
        cin >> a[i];

    // ===== Multithread execution =====
    vector<thread> threads;
    vector<long long> results(MAX_THREADS, 0);

    int chunk = (n + MAX_THREADS - 1) / MAX_THREADS;
    for (int i = 0; i < MAX_THREADS; i++) {
        int l = i * chunk;
        int r = min(n, l + chunk);
        if (l < r)
            threads.emplace_back(worker, cref(a), l, r,
                  ref(results[i]));
    }

    for (auto &t : threads) t.join(); // wait all threads

    // Merge results
    long long total = 0;
    for (auto x : results) total += x;

    cout << total << "\n";
    return 0;
}

/* ================= COMMAND CHEAT SHEET =================
#include <thread>      // include threading library
thread t(func,args...) // create thread running func with
      arguments
t.join()             // wait for thread t to finish
ref(x)               // pass variable by reference to thread
cref(x)              // pass variable by const reference
t.detach()           // run thread independently
thread::hardware_concurrency() // get number of CPU threads
======================================================= */
```

1.5 pragma

```cpp
#pragma GCC target("avx")
#pragma GCC optimize(3)
#pragma GCC optimize("Ofast")
#pragma GCC optimize("inline")
#pragma GCC optimize("-fgcse")
#pragma GCC optimize("-fgcse-lm")
#pragma GCC optimize("-fipa-sra")
#pragma GCC optimize("-ftree-pre")
#pragma GCC optimize("-ftree-vrp")
#pragma GCC optimize("-fpeephole2")
#pragma GCC optimize("-ffast-math")
#pragma GCC optimize("-fsched-spec")
#pragma GCC optimize("unroll-loops")
#pragma GCC optimize("-falign-jumps")
#pragma GCC optimize("-falign-loops")
#pragma GCC optimize("-falign-labels")
#pragma GCC optimize("-fdevirtualize")
#pragma GCC optimize("-fcaller-saves")
#pragma GCC optimize("-fcrossjumping")
#pragma GCC optimize("-fthread-jumps")
#pragma GCC optimize("-funroll-loops")
#pragma GCC optimize("-fwhole-program")
#pragma GCC optimize("-freorder-blocks")
#pragma GCC optimize("-fschedule-insns")
#pragma GCC optimize("inline-functions")
#pragma GCC optimize("-ftree-tail-merge")
#pragma GCC optimize("-fschedule-insns2")
#pragma GCC optimize("-fstrict-aliasing")
#pragma GCC optimize("-fstrict-overflow")
#pragma GCC optimize("-falign-functions")
#pragma GCC optimize("-fcse-skip-blocks")
#pragma GCC optimize("-fcse-follow-jumps")
```

```
#pragma GCC optimize("-fsched-interblock")
#pragma GCC optimize("-fpartial-inlining")
#pragma GCC optimize("no-stack-protector")
#pragma GCC optimize("-freorder-functions")
#pragma GCC optimize("-findirect-inlining")
#pragma GCC optimize("-fhoist-adjacent-loads")
#pragma GCC optimize("-frerun-cse-after-loop")
#pragma GCC optimize("inline-small-functions")
#pragma GCC optimize("-finline-small-functions")
#pragma GCC optimize("-ftree-switch-conversion")
#pragma GCC optimize("-foptimize-sibling-calls")
#pragma GCC optimize("-fexpensive-optimizations")
#pragma GCC optimize("-funsafe-loop-optimizations")
#pragma GCC optimize("inline-functions-called-once")
#pragma GCC optimize("-fdelete-null-pointer-checks")
#pragma GCC optimize("O3")
#pragma GCC optimization("Ofast,unroll-loops")
#pragma GCC target("avx2,bmi,bmi2,lzcnt,popcnt")
#pragma GCC
      optimize("O3,no-stack-protector,fast-math,unroll-loops,tree-vectorize")
#pragma GCC optimize("conserve-stack")
#pragma GCC target("sse4.2,popcnt,lzcnt,abm,mmx,fma,bmi,bmi2")
#pragma GCC
      target("avx2,popcnt,lzcnt,abm,bmi,bmi2,fma,tune=native")
#pragma GCC target("avx2,popcnt,lzcnt,abm,bmi,bmi2,fma")
```

1.6 pragma1

```
#pragma GCC optimize ("O3, unroll-loops")
#pragma GCC target ("avx2 ,bmi, bmi2, lzcnt, popcnt")
```

1.7 RandCanLeMinhHieu

```
mt19937_64 rd (chrono :: steady_clock :: now
      ().time_since_epoch ().count ());
int Rand (int l, int r) {return uniform_int_distribution <int>
      (l, r) (rd);}
using namespace std::chrono;
steady_clock::time_point start_time;
const long long TIME_LIMIT_MS = 950;
unsigned long long check_counter = 0;
inline void maybe_timeout_check(){
    if((++check_counter & 1023) == 0){
        auto now = steady_clock::now();
        auto ms = duration_cast<milliseconds>(now -
              start_time).count();
        if(ms >= TIME_LIMIT_MS){
            //cout << max1 << "\n";
            cout.flush();
            exit(0);
        }
    }
}
// __start_time = steady_clock::now();
// __maybe_timeout_check();
```

1.8 runme

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

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

        while (true) { // for (int t = 1; t <= ...; t++)
                system("gen.exe");
                system("bai1.exe");
                system("bai1_bf.exe");
                if (system("fc bai1.out bai12.out") == 1) {
                        cout << "Difference is found";
                        break;
                }
        }

        return 0;
}
```

1.9 TEMP

```
#include <bits/stdc++.h>

#define task  "htgdtctn"

#define int   long long
#define pii   pair <int, int>
#define fi    first
#define se    second
#define szf   sizeof
#define sz(s) (int)((s).size())
#define all(v) (v).begin(), (v).end()

using namespace std;

template <class T> void minimize (T &t, T f) {if (t > f) t = f;}
template <class T> void maximize (T &t, T f) {if (t < f) t = f;}

const int maxN = 2e5 + 5;
const int inf = 1e18 + 7;
const int mod = 1e9 + 7;

// mt19937_64 rd (chrono :: steady_clock :: now
      ().time_since_epoch ().count ());
// int Rand (int l, int r) {return uniform_int_distribution
      <int> (l, r) (rd);}
inline void Solve () {

    return;
}

signed main () {
    cin.tie (nullptr) -> sync_with_stdio (false);
    if (fopen (task".inp", "r")) {
        freopen (task".inp", "r", stdin);
        freopen (task".out", "w", stdout);
    }
    int t = 1;
    //cin >> t;
    while (t --) {
        Solve ();
    }
    return 0;
}
// wake me up when September ends
```

2 Data Structure

2.1 fenwick2D

```
struct Fenwick2D {
    int m, n;
    vector <vector <int>> bit[4];

    inline void init (int _m, int _n) {
        m = _m;
        n = _n;
        for (int i = 0; i < 4; i ++) {
            bit[i].resize (m + 4, vector <int> (n + 4, 0));
        }
    }

    inline void upd (int x, int y, int v) {
        for (int i = x; i <= m; i += i & (-i)) {
            for (int j = y; j <= n; j += j & (-j)) {
                bit[0][i][j] += v;
                bit[1][i][j] += x * v;
                bit[2][i][j] += y * v;
                bit[3][i][j] += x * y * v;
            }
        }
    }

    inline int get (int x, int y) {
        int res = 0;
        for (int i = x; i; i -= i & (-i)) {
            for (int j = y; j; j -= j & (-j)) {
                res += (x + 1) * (y + 1) * bit[0][i][j] - (y +
                      1) * bit[1][i][j]
                - (x + 1) * bit[2][i][j] + bit[3][i][j];
            }
        }
        return res;
    }

    inline void updR (int x1, int y1, int x2, int y2, int v) {
        upd (x1, y1, v);
        upd (x1, y2 + 1, -v);
        upd (x2 + 1, y1, -v);
        upd (x2 + 1, y2 + 1, v);
    }

    inline int getR (int x1, int y1, int x2, int y2) {
        return get (x2, y2) - get (x1 - 1, y2) - get (x2, y1 -
            1) + get (x1 - 1, y1 - 1);
    }
} T;
```

2.2 LiChao

```
struct Line {
    int a, b;

    Line () : a (0), b (-inf) {}
    Line (int a, int b) : a (a), b (b) {}

    inline int get (int x) {
        return a * x + b;
    }
};

struct Node {
    Line val;
    Node *l, *r;

    Node () {
        val = Line ();
        l = r = nullptr;
    }
};

struct LiChao {
    int L, R;
    Node *root;

    void init (int l, int r) {
        L = l;
        R = r;
        root = new Node ();
    }

    void upd (Node *&cur, int l, int r, Line nw) {
        if (!cur) {
            cur = new Node ();
        }
        int mid = (l + r) >> 1;
        bool left = nw.get (l) > cur -> val.get (l);
        bool m = nw.get (mid) > cur -> val.get (mid);
        if (m) {
            swap (cur -> val, nw);
        }
        if (l == r) {
            return;
        }
        if (left != m) {
            upd (cur -> l, l, mid, nw);
        }
        else {
            upd (cur -> r, mid + 1, r, nw);
        }
    }

    int get (Node *cur, int l, int r, int x) {
        if (!cur) {
            return -inf;
        }
        int res = cur -> val.get (x);
        if (l == r) {
            return res;
        }
        int mid = (l + r) >> 1;
        if (x <= mid) {
            return max (res, get (cur -> l, l, mid, x));
        }
        return max (res, get (cur -> r, mid + 1, r, x));
    }

    void addLine (Line nw) {
        upd (root, L, R, nw);
    }

    int query (int x) {
        return get (root, L, R, x);
    }
};

int n;
int a[maxN];
LiChao T1, T2;
int s[maxN], sl[maxN], sr[maxN];

void solve () {
    cin >> n;
    for (int i = 1; i <= n; i ++) {
        cin >> a[i];
        s[i] = s[i - 1] + a[i] * i;
        sl[i] = sl[i - 1] + a[i] * (i - 1);
        sr[i] = sr[i - 1] + a[i] * (i + 1);
    }
    T1.init (-1e9, 1e9);
    T2.init (-1e9, 1e9);
    int res = s[n];
    for (int i = 1; i <= n; i ++) {
        int p3 = s[n] - s[i] + sl[i];
        int p12 = T1.query (i);
        if (p12 != -inf) {
            maximize (res, p3 + p12);
        }
        T1.addLine ({a[i], s[i - 1] - sl[i]});
    }
    for (int i = n; i >= 1; i --) {
        int p3 = s[n] + s[i - 1] - sr[i - 1];
        int p12 = T2.query (i);
        if (p12 != -inf) {
            maximize (res, p3 + p12);
        }
        T2.addLine ({a[i], sr[i - 1] - s[i]});
    }
    cout << res;
    return;
}
```

2.3 LiChao2

```
struct Line {
    int a, b;

    Line () : a (0), b (inf) {}
    Line (int a, int b) : a (a), b (b) {}

    int calc (int x) {
        return a * x + b;
    }
};

struct LiChao {
    vector <Line> lc;

    void init (int n) {
        lc.resize (n * 4, Line ());
    }

    void upd (int id, int l, int r, Line L) {
        int mid = (l + r) >> 1;
        bool left = L.calc (l) < lc[id].calc (l);
        bool m = L.calc (mid) < lc[id].calc (mid);
        if (m) {
            swap (lc[id], L);
        }
        if (l == r) {
            return;
        }
        if (left != m) {
            upd (id * 2, l, mid, L);
        }
        else {
            upd (id * 2 + 1, mid + 1, r, L);
        }
    }

    int get (int id, int l, int r, int x) {
        if (l == r) {
            return lc[id].calc (x);
        }
        int mid = (l + r) >> 1;
        if (x <= mid) {
```

```cpp
            return min (lc[id].calc (x), get (id * 2, l, mid,
                x));
        }
        return min (lc[id].calc (x), get (id * 2 + 1, mid + 1,
            r, x));
    }
};

LiChao T;
int h[maxN], w[maxN];
int dp[maxN];
int pfs[maxN];

int p1 (int x) {
    return - 2 * h[x];
}

int p2 (int x) {
    return dp[x] + h[x] * h[x] - w[x];
}

int p3 (int x) {
    return h[x] * h[x] + w[x - 1];
}

void solve () {
    int n;
    cin >> n;
    for (int i = 1; i <= n; i ++) {
        cin >> h[i];
    }
    for (int i = 1; i <= n; i ++) {
        cin >> w[i];
        w[i] += w[i - 1];
    }
    T.init (maxN + 1);
    dp[1] = 0;
    T.upd (1, 0, maxN, {p1 (1), p2 (1)});
    for (int i = 2; i <= n; i ++) {
        dp[i] = T.get (1, 0, maxN, h[i]) + p3 (i);
        T.upd (1, 0, maxN, {p1 (i), p2 (i)});
    }
    cout << dp[n];
    return;
}
```

2.4 persistent segment tree

```cpp
struct PST {
    struct Node {
        int l, r, val;
    };

    vector<Node> seg;
    vector<int> root;

    PST(int n = 0) {
        seg.reserve(40 * n);
    }

    int build(int l, int r) {
        int id = seg.size();
        seg.push_back({-1, -1, 0});
        if (l == r) return id;
        int m = (l + r) / 2;
        seg[id].l = build(l, m);
        seg[id].r = build(m + 1, r);
        return id;
    }

    int update(int prv, int l, int r, int pos, int val) {
        int id = seg.size();
        seg.push_back(seg[prv]);
        if (l == r) {
            seg[id].val += val;
            return id;
        }
        int m = (l + r) / 2;
        if (pos <= m) seg[id].l = update(seg[prv].l, l, m, pos,
             val);
        else seg[id].r = update(seg[prv].r, m + 1, r, pos, val);
        seg[id].val = seg[seg[id].l].val + seg[seg[id].r].val;
        return id;
    }

    int query(int id, int l, int r, int x, int y) {
        if (x > r || y < l) return 0;
        if (x <= l && r <= y) return seg[id].val;
        int m = (l + r) / 2;
        return query(seg[id].l, l, m, x, y) + query(seg[id].r,
             m + 1, r, x, y);
    }
};
```

2.5 Treap

```cpp
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
#define ordered_set tree<int, null_type,less<int>,
     rb_tree_tag,tree_order_statistics_node_update>
#define int long long
using namespace std;
using namespace __gnu_pbds;

struct Treap{ /// hash = 96814
    int len;
    const int ADD = 1000010;
    const int MAXVAL = 1e15;
    unordered_map <long long, int> mp; /// Change to int if
         only int in treap
    tree<long long, null_type, less<long long>, rb_tree_tag,
         tree_order_statistics_node_update> T;

    Treap(){
        len = 0;
        T.clear(), mp.clear();
    }

    inline void clear(){
        len = 0;
        T.clear(), mp.clear();
    }

    inline void insert(long long x){
        len++, x += MAXVAL;
        int c = mp[x]++;
        T.insert((x * ADD) + c);
    }

    inline void erase(long long x){
        x += MAXVAL;
        int c = mp[x];
        if (c){
            c--, mp[x]--, len--;
            T.erase((x * ADD) + c);
        }
    }

    // tra ve so thu k
    inline long long kth(int k){
        if (k < 1 || k > len) return -1;
        auto it = T.find_by_order(--k);
        return ((*it) / ADD) - MAXVAL;
    }

    // sl so < k
    inline int count(long long x){
        x += MAXVAL;
        int c = mp[--x];
        return (T.order_of_key((x * ADD) + c));
    }

     // size
    inline int size(){
        return len;
    }
};
```

3 DP

3.1 CHT

```cpp
struct CHT {
        ll getval(pll X, ll x) {
                return X.a * x + X.b;
        }
        bool bad(pll x, pll y, pll z) {
                return (y.b - x.b) * (x.a - z.a) >= (z.b - x.b)
                     * (x.a - y.a);
        }
        vector<pll> s;
        void add(ll a, ll b) {
                int m = sze(s);
                while(m >= 2 && bad(s[m - 2], s[m - 1],
                      make_pair(a, b))) {
                        s.pop_back();
                        --m;
                }
                s.pb(make_pair(a, b));
        }
        ll get(ll x) {
                if (s.empty()) return oo;
                int l = 0, r = sze(s) - 2;
                ll ans = getval(s[l], x);
                while(l <= r) {
                        int mid = (l + r) >> 1;
                        ll X = getval(s[mid], x);
                        ll Y = getval(s[mid + 1], x);
                        if (X > Y) {
                                l = mid + 1;
                        } else {
                                r = mid - 1;
                        }
                        ans = min(ans, min(X, Y));
                }
                return ans;
        }
}
```

3.2 dpDNC

```
#include <bits/stdc++.h>

#define task  "BriantheCrab"

#define pii   pair <int, int>
#define fi    first
#define se    second
#define szf   sizeof
#define sz(s) (int)((s).size())

using namespace std;

template <class T> void mini (T &t, T f) {if (t > f) t = f;}
template <class T> void maxi (T &t, T f) {if (t < f) t = f;}

const int maxN = 1e5 + 5;
const long long inf = 1e18 + 7;
const int mod = 1e9 + 7;

int n, k;
int a[maxN],trace[maxN][205];
long long pfs[maxN], dp[maxN][5];

void dnc (int x, int l, int r, int optL, int optR) {
    if (l > r) {
        return;
    }
    int m = (l + r) >> 1;
    long long best = -inf;
    int opt = -1;
    for (int i = optL; i <= min (m, optR); i ++) {
        if (best < dp[i - 1][(x & 1) ^ 1] + (pfs[m] - pfs[i -
             1]) * (pfs[n] - pfs[m])) {
            best = dp[i - 1][(x & 1) ^ 1] + (pfs[m] - pfs[i -
                 1]) * (pfs[n] - pfs[m]);
            opt = i;
        }
    }
    dp[m][(x & 1)] = best;
    //cout << m << ' ' << x << ' ' << dp[m][x & 1] << ' ' <<
         opt << '\n';
    trace[m][x] = opt - 1;
    dnc (x, l, m - 1, optL, opt);
    dnc (x, m + 1, r, opt, optR);
}

void solve () {
    cin >> n >> k;
    for (int i = 1; i <= n; i ++) {
        cin >> a[i];
        pfs[i] = pfs[i - 1] + a[i];
    }
    memset (trace, -1, szf (trace));
    for (int i = 1; i <= n; i ++) {
        dp[i][1] = pfs[i] * (pfs[n] - pfs[i]);
    }
    for (int i = 1; i <= k + 1; i ++) {
        dnc (i, i, n, i, n);
    }
    cout << max (0LL, dp[n][(k + 1) & 1]) << '\n';
    int curM = n, curX = k + 1;
    vector <int> all;
    all.push_back (curM - 1);
    while (trace[curM][curX] != -1) {
        all.push_back (trace[curM][curX]);
        curM = trace[curM][curX];
        curX --;
    }
    reverse (all.begin (), all.end ());
    for (int i = 1; i <= k; i ++) {
        cout << all[i] << ' ';
    }
    return;
}

signed main () {
    cin.tie (nullptr) -> sync_with_stdio (false);
    if (fopen (task".inp", "r")) {
        freopen (task".inp", "r", stdin);
        freopen (task".out", "w", stdout);
    }
    int t = 1;
    //cin >> t;
    while (t --) {
        solve ();
    }
    return 0;
}
// thfdgb
```

3.3 dpSOS

```
#include <bits/stdc++.h>

#define task   "BriantheCrab"

#define int   long long
#define pii   pair <int, int>
#define fi    first
#define se    second
#define szf   sizeof
#define sz(s) (int)((s).size())
#define all(v) (v).begin(), (v).end()

typedef long long ll;
typedef unsigned long long ull;
typedef long double ld;

using namespace std;

template <class T> void minimize (T &t, T f) {if (t > f) t = f;}
template <class T> void maximize (T &t, T f) {if (t < f) t = f;}

const int maxN = 2e6 + 5;
const int inf = 1e18 + 7;
const int mod = 1e9 + 7;

int a[maxN];
int f[(1 << 20) + 1];

void solve () {
    int n;
    cin >> n;
    for (int i = 0; i < (1 << n); i ++) {
        cin >> a[i];
    }
    for (int i = 0; i < n; i ++) {
        for (int mask = 0; mask < (1 << n); mask ++) {
            if ((mask >> i) & 1) {
                a[mask] += a[mask ^ (1 << i)];
            }
        }
    }
    for (int i = 0; i < (1 << n); i ++) {
        cout << a[i] << ' ';
    }
    return;
}

signed main () {
    cin.tie (nullptr) -> sync_with_stdio (false);
    if (fopen (task".inp", "r")) {
        freopen (task".inp", "r", stdin);
        freopen (task".out", "w", stdout);
    }
    int t = 1;
    //cin >> t;
    while (t --) {
        solve ();
    }
    return 0;
}
// thfv
```

3.4 LineContainer

```
ll divi(ll a, ll b) {
        ll res = (a / b);
        if (a < 0 && b > 0) --res;
        if (a > 0 && b < 0) --res;
        return res;
}

struct LC {
        struct line {
                ll a, b;
                mutable ll p;
                line(ll a_, ll b_, ll p_) : a(a_), b(b_), p(p_)
                    {}
                bool operator<(const line &other) const {
                        if (other.a == oo && other.b == oo)
                                return p < other.p;
                        return a < other.a;
                }
        };
        multiset<line> mylc;
        bool isect(multiset<line>::iterator x,
              multiset<line>::iterator y) {
                if (y == mylc.end()) {
                        x->p = oo;
                        return false;
                }
                if (x->a == y->a) {
                        if (x->b > y->b) {
                                x->p = oo;
                        } else {
                                x->p = -oo;
                        }
                } else {
                        x->p = divi((y->b - x->b), (x->a - y->a));
                }
                return x->p >= y->p;
        }
        void add(ll a, ll b) {
                multiset<line>::iterator x = mylc.insert(line(a,
                    b, 0)), y = next(x);
                while(isect(x, y)) y = mylc.erase(y);

                y = x;
                if (x != mylc.begin()) {
                        y = prev(y);
                        if (isect(y, x)) isect(y, mylc.erase(x));
                }
                while(y != mylc.begin()) {
                        x = prev(y);
                        if (x->p >= y->p) {
                                isect(x, mylc.erase(y));
                                y = x;
                        } else break;
                }
        }
        ll get(const ll x) {
                multiset<line>::iterator it =
                      mylc.lower_bound(line(oo, oo, x));
                return it->a * x + it-> b;
        }
} lc;
```

4 Geometry

4.1 AngleBisector

```
// bisector vector of <abc
PT angle_bisector(PT &a, PT &b, PT &c){
  PT p = a - b, q = c - b;
  return p + q * sqrt(dot(p, p) / dot(q, q));
}
```

4.2 Centroid

```
// centroid of a (possibly non-convex) polygon,
// assuming that the coordinates are listed in a clockwise or
// counterclockwise fashion. Note that the centroid is often
     known as
// the "center of gravity" or "center of mass".
PT centroid(vector<PT> &p) {
    int n = p.size(); PT c(0, 0);
    double sum = 0;
    for (int i = 0; i < n; i++) sum += cross(p[i], p[(i + 1) %
         n]);
    double scale = 3.0 * sum;
    for (int i = 0; i < n; i++) {
        int j = (i + 1) % n;
        c = c + (p[i] + p[j]) * cross(p[i], p[j]);
    }
    return c / scale;
}
```

4.3 Circle

```
struct circle {
  PT p;
  double r;
  circle() {}
  circle(PT _p, double _r) : p(_p), r(_r){};
  // center (x, y) and radius r
  circle(double x, double y, double _r) : p(PT(x, y)), r(_r){};
  // circumcircle of a triangle
  // the three points must be unique
  circle(PT a, PT b, PT c) {
    b = (a + b) * 0.5;
    c = (a + c) * 0.5;
    line_line_intersection(b, b + rotatecw90(a - b), c, c +
         rotatecw90(a - c), p);
    r = dist(a, p);
  }
  // inscribed circle of a triangle
  circle(PT a, PT b, PT c, bool t) {
    line u, v;
    double m = atan2(b.y - a.y, b.x - a.x), n = atan2(c.y -
         a.y, c.x - a.x);
    u.a = a;
    u.b = u.a + (PT(cos((n + m) / 2.0), sin((n + m) / 2.0)));
    v.a = b;
    m = atan2(a.y - b.y, a.x - b.x), n = atan2(c.y - b.y, c.x -
         b.x);
    v.b = v.a + (PT(cos((n + m) / 2.0), sin((n + m) / 2.0)));
    line_line_intersection(u.a, u.b, v.a, v.b, p);
    r = dist_from_point_to_seg(a, b, p);
  }
  bool operator==(circle v) { return p == v.p && sign(r - v.r)
       == 0; }
  double area() { return PI * r * r; }
  double circumference() { return 2.0 * PI * r; }
};
// 0 if outside, 1 if on circumference, 2 if inside circle
int circle_point_relation(PT p, double r, PT b) {
  double d = dist(p, b);
  if (sign(d - r) < 0) return 2;
  if (sign(d - r) == 0) return 1;
  return 0;
}
// 0 if outside, 1 if on circumference, 2 if inside circle
int circle_line_relation(PT p, double r, PT a, PT b) {
  double d = dist_from_point_to_line(a, b, p);
  if (sign(d - r) < 0) return 2;
  if (sign(d - r) == 0) return 1;
  return 0;
}
// compute intersection of line through points a and b with
// circle centered at c with radius r > 0
vector<PT> circle_line_intersection(PT c, double r, PT a, PT b)
     {
  vector<PT> ret;
  b = b - a;
  a = a - c;
  double A = dot(b, b), B = dot(a, b);
  double C = dot(a, a) - r * r, D = B * B - A * C;
  if (D < -eps) return ret;
  ret.push_back(c + a + b * (-B + sqrt(D + eps)) / A);
  if (D > eps) ret.push_back(c + a + b * (-B - sqrt(D)) / A);
  return ret;
}
// 5 - outside and do not intersect
// 4 - intersect outside in one point
// 3 - intersect in 2 points
// 2 - intersect inside in one point
// 1 - inside and do not intersect
int circle_circle_relation(PT a, double r, PT b, double R) {
  double d = dist(a, b);
  if (sign(d - r - R) > 0) return 5;
  if (sign(d - r - R) == 0) return 4;
  double l = fabs(r - R);
  if (sign(d - r - R) < 0 && sign(d - l) > 0) return 3;
  if (sign(d - l) == 0) return 2;
  if (sign(d - l) < 0) return 1;
  assert(0);
  return -1;
}
vector<PT> circle_circle_intersection(PT a, double r, PT b,
      double R) {
  if (a == b && sign(r - R) == 0) return {PT(1e18, 1e18)};
  vector<PT> ret;
  double d = sqrt(dist2(a, b));
  if (d > r + R || d + min(r, R) < max(r, R)) return ret;
  double x = (d * d - R * R + r * r) / (2 * d);
  double y = sqrt(r * r - x * x);
  PT v = (b - a) / d;
  ret.push_back(a + v * x + rotateccw90(v) * y);
  if (y > 0) ret.push_back(a + v * x - rotateccw90(v) * y);
  return ret;
}
// returns two circle c1, c2 through points a, b and of radius r
// 0 if there is no such circle, 1 if one circle, 2 if two
      circle
int get_circle(PT a, PT b, double r, circle &c1, circle &c2) {
  vector<PT> v = circle_circle_intersection(a, r, b, r);
  int t = v.size();
  if (!t) return 0;
  c1.p = v[0], c1.r = r;
  if (t == 2) c2.p = v[1], c2.r = r;
  return t;
}
// returns two circle c1, c2 which is tangent to line u, goes
      through
// point q and has radius r1; 0 for no circle, 1 if c1 = c2 , 2
      if c1 != c2
int get_circle(line u, PT q, double r1, circle &c1, circle &c2)
      {
  double d = dist_from_point_to_line(u.a, u.b, q);
  if (sign(d - r1 * 2.0) > 0) return 0;
  if (sign(d) == 0) {
    cout << u.v.x << ' ' << u.v.y << '\n';
    c1.p = q + rotateccw90(u.v).truncate(r1);
    c2.p = q + rotatecw90(u.v).truncate(r1);
    c1.r = c2.r = r1;
    return 2;
  }
  line u1 = line(u.a + rotateccw90(u.v).truncate(r1), u.b +
      rotateccw90(u.v).truncate(r1));
  line u2 = line(u.a + rotatecw90(u.v).truncate(r1), u.b +
      rotatecw90(u.v).truncate(r1));
  circle cc = circle(q, r1);
  PT p1, p2;
  vector<PT> v;
  v = circle_line_intersection(q, r1, u1.a, u1.b);
  if (!v.size()) v = circle_line_intersection(q, r1, u2.a,
      u2.b);
  v.push_back(v[0]);
  p1 = v[0], p2 = v[1];
  c1 = circle(p1, r1);
  if (p1 == p2) {
    c2 = c1;
    return 1;
  }
  c2 = circle(p2, r1);
  return 2;
}
// returns area of intersection between two circles
double circle_circle_area(PT a, double r1, PT b, double r2) {
  double d = (a - b).norm();
  if (r1 + r2 < d + eps) return 0;
  if (r1 + d < r2 + eps) return PI * r1 * r1;
  if (r2 + d < r1 + eps) return PI * r2 * r2;
  double theta_1 = acos((r1 * r1 + d * d - r2 * r2) / (2 * r1 *
      d)),
       theta_2 = acos((r2 * r2 + d * d - r1 * r1) / (2 * r2 *
          d));
  return r1 * r1 * (theta_1 - sin(2 * theta_1)) / 2.) + r2 * r2
      * (theta_2 - sin(2 * theta_2) / 2.);
}
// tangent lines from point q to the circle
int tangent_lines_from_point(PT p, double r, PT q, line &u,
     line &v) {
  int x = sign(dist2(p, q) - r * r);
  if (x < 0) return 0; // point in cricle
  if (x == 0) {       // point on circle
    u = line(q, q + rotateccw90(q - p));
    v = u;
    return 1;
  }
  double d = dist(p, q);
  double l = r * r / d;
  double h = sqrt(r * r - l * l);
  u = line(q, p + ((q - p).truncate(l) + (rotateccw90(q -
      p).truncate(h))));
  v = line(q, p + ((q - p).truncate(l) + (rotatecw90(q -
      p).truncate(h))));
  return 2;
}
// returns outer tangents line of two circles
// if inner == 1 it returns inner tangent lines
int tangents_lines_from_circle(PT c1, double r1, PT c2, double
     r2, bool inner, line &u, line &v) {
  if (inner) r2 = -r2;
  PT d = c2 - c1;
  double dr = r1 - r2, d2 = d.norm2(), h2 = d2 - dr * dr;
  if (d2 == 0 || h2 < 0) {
    assert(h2 != 0);
    return 0;
  }
  vector<pair<PT, PT>> out;
  for (int tmp : {-1, 1}) {
    PT v = (d * dr + rotateccw90(d) * sqrt(h2) * tmp) / d2;
    out.push_back({c1 + v * r1, c2 + v * r2});
  }
  u = line(out[0].first, out[0].second);
  if (out.size() == 2) v = line(out[1].first, out[1].second);
  return 1 + (h2 > 0);
}
// O(n^2 log n)
struct CircleUnion {
  int n;
  double x[2020], y[2020], r[2020];
  int covered[2020];
  vector<pair<double, double>> seg, cover;
  double arc, pol;
  inline int sign(double x) { return x < -eps ? -1 : x > eps; }
  inline int sign(double x, double y) { return sign(x - y); }
  inline double SQ(const double x) { return x * x; }
  inline double dist(double x1, double y1, double x2, double
      y2) {
    return sqrt(SQ(x1 - x2) + SQ(y1 - y2));
  }
  inline double angle(double A, double B, double C) {
    double val = (SQ(A) + SQ(B) - SQ(C)) / (2 * A * B);
    if (val < -1) val = -1;
    if (val > +1) val = +1;
    return acos(val);
  }
  CircleUnion() {
    n = 0;
    seg.clear(), cover.clear();
    arc = pol = 0;
  }
  void init() {
    n = 0;
    seg.clear(), cover.clear();
    arc = pol = 0;
  }
  void add(double xx, double yy, double rr) {
    x[n] = xx, y[n] = yy, r[n] = rr, covered[n] = 0, n++;
  }
  void getarea(int i, double lef, double rig) {
    arc += 0.5 * r[i] * r[i] * (rig - lef - sin(rig - lef));
    double x1 = x[i] + r[i] * cos(lef), y1 = y[i] + r[i] *
        sin(lef);
    double x2 = x[i] + r[i] * cos(rig), y2 = y[i] + r[i] *
        sin(rig);
    pol += x1 * y2 - x2 * y1;
  }
  double solve() {
    for (int i = 0; i < n; i++) {
      for (int j = 0; j < i; j++) {
        if (!sign(x[i] - x[j]) && !sign(y[i] - y[j]) &&
            !sign(r[i] - r[j])) {
          r[i] = 0.0;
          break;
    }
   }
  }
  for (int i = 0; i < n; i++) {
   for (int j = 0; j < n; j++) {
    if (i != j && sign(r[j] - r[i]) >= 0 &&
      sign(dist(x[i], y[i], x[j], y[j]) - (r[j] - r[i]))
         <= 0) {
     covered[i] = 1;
     break;
    }
   }
  }
  for (int i = 0; i < n; i++) {
   if (sign(r[i]) && !covered[i]) {
    seg.clear();
    for (int j = 0; j < n; j++) {
     if (i != j) {
      double d = dist(x[i], y[i], x[j], y[j]);
      if (sign(d - (r[j] + r[i])) >= 0 || sign(d -
         abs(r[j] - r[i])) <= 0) {
       continue;
      }
      double alpha = atan2(y[j] - y[i], x[j] - x[i]);
      double beta = angle(r[i], d, r[j]);
      pair<double, double> tmp(alpha - beta, alpha + beta);
      if (sign(tmp.first) <= 0 && sign(tmp.second) <= 0) {
       seg.push_back(pair<double, double>(2 * PI +
          tmp.first, 2 * PI + tmp.second));
      } else if (sign(tmp.first) < 0) {
       seg.push_back(pair<double, double>(2 * PI +
          tmp.first, 2 * PI));
       seg.push_back(pair<double, double>(0, tmp.second));
      } else {
       seg.push_back(tmp);
      }
     }
    }
    sort(seg.begin(), seg.end());
    double rig = 0;
    for (vector<pair<double, double>>::iterator iter =
       seg.begin(); iter != seg.end(); iter++) {
     if (sign(rig - iter->first) >= 0) {
      rig = max(rig, iter->second);
     } else {
      getarea(i, rig, iter->first);
      rig = iter->second;
     }
    }
    if (!sign(rig)) {
     arc += r[i] * r[i] * PI;
    } else {
     getarea(i, rig, 2 * PI);
    }
   }
  }
  return pol / 2.0 + arc;
 }
} CU;
```

4.4 GeometryTemplate

```
const long double PI = acos(-1);
struct Vector {
 using type = long long;
 type x, y;
 Vector operator-(const Vector &other) const {
  return {x - other.x, y - other.y};
 }
 type operator*(const Vector &other) const {
  return x * other.y - other.x * y;
 }
 type operator%(const Vector &other) const {
  return x * other.x + y * other.y;
 }
 bool operator==(const Vector &other) const {
  return x == other.x and y == other.y;
 }
 bool operator!=(const Vector &other) const { return !(*this
    == other); }
 friend type cross(const Vector &A, const Vector &B, const
    Vector &C) {
  return (B - A) * (C - A);
 }
 friend type dist(Vector A) { return A.x * A.x + A.y * A.y; }
 friend type dot(const Vector &A, const Vector &B, const
    Vector &C) {
  Vector u = (B - A), v = (C - A);
  return u % v;
 }
 friend istream &operator>>(istream &is, Vector &V) {
  is >> V.x >> V.y;
  return is;
 }
 friend ostream &operator<<(ostream &os, Vector &V) {
  os << V.x << ' ' << V.y;
  return os;
 }
 friend double angle(const Vector &A, const Vector &B, const
    Vector &C) {
  double x = dot(B, A, C) / sqrt(dist(A - B) * dist(C - B));
  return acos(min(1.0, max(-1.0, x))) * 180.0 / PI;
 }
};
using Point = Vector;
const Point origin = {0, 0};

long double area(Point A, Point B, Point C) {
 long double res =
   cross(origin, A, B) + cross(origin, B, C) + cross(origin,
      C, A);
 return abs(res) / 2.0;
}
```

4.5 Line

```
struct line {
  PT a, b; // goes through points a and b
  PT v; double c; //line form: direction vec [cross] (x, y) =
     c
  line() {}
  //direction vector v and offset c
  line(PT v, double c) : v(v), c(c) {
    auto p = get_points();
    a = p.first; b = p.second;
  }
  // equation ax + by + c = 0
  line(double _a, double _b, double _c) : v({_b, -_a}),
     c(-_c) {
    auto p = get_points();
    a = p.first; b = p.second;
  }
  // goes through points p and q
  line(PT p, PT q) : v(q - p), c(cross(v, p)), a(p), b(q) {}
    pair<PT, PT> get_points() { //extract any two points
       from this line
    PT p, q; double a = -v.y, b = v.x; // ax + by = c
    if (sign(a) == 0) {
      p = PT(0, c / b);
      q = PT(1, c / b);
    }
    else if (sign(b) == 0) {
      p = PT(c / a, 0);
      q = PT(c / a, 1);
    }
    else {
      p = PT(0, c / b);
      q = PT(1, (c - a) / b);
    }
    return {p, q};
    }
  //ax + by + c = 0
  array<double, 3> get_abc() {
    double a = -v.y, b = v.x;
    return {a, b, c};
  }
  // 1 if on the left, -1 if on the right, 0 if on the line
  int side(PT p) { return sign(cross(v, p) - c); }
  // line that is perpendicular to this and goes through
     point p
  line perpendicular_through(PT p) { return {p, p + perp(v)};
     }
  // translate the line by vector t i.e. shifting it by
     vector t
  line translate(PT t) { return {v, c + cross(v, t)}; }
  // compare two points by their orthogonal projection on
     this line
  // a projection point comes before another if it comes
     first according to vector v
  bool cmp_by_projection(PT p, PT q) { return dot(v, p) <
     dot(v, q); }
  line shift_left(double d) {
    PT z = v.perp().truncate(d);
    return line(a + z, b + z);
  }
};
```

4.6 Utilities

```cpp
double perimeter(vector<PT> &p) {
    double ans=0; int n = p.size();
    for (int i = 0; i < n; i++) ans += dist(p[i], p[(i + 1) %
         n]);
    return ans;
}
double area(vector<PT> &p) {
    double ans = 0; int n = p.size();
    for (int i = 0; i < n; i++) ans += cross(p[i], p[(i + 1) %
         n]);
    return fabs(ans) * 0.5;
}
double area_of_triangle(PT a, PT b, PT c) {
    return fabs(cross(b - a, c - a) * 0.5);
}
// 0 if cw, 1 if ccw
bool get_direction(vector<PT> &p) {
    double ans = 0; int n = p.size();
    for (int i = 0; i < n; i++) ans += cross(p[i], p[(i + 1) %
         n]);
    if (sign(ans) > 0) return 1;
    return 0;
}
// find a point from a through b with distance d
PT point_along_line(PT a, PT b, double d) {
    assert(a != b);
    return a + (((b - a) / (b - a).norm()) * d);
}
// projection point c onto line through a and b assuming a != b
PT project_from_point_to_line(PT a, PT b, PT c) {
    return a + (b - a) * dot(c - a, b - a) / (b - a).norm2();
}
// reflection point c onto line through a and b assuming a != b
PT reflection_from_point_to_line(PT a, PT b, PT c) {
    PT p = project_from_point_to_line(a,b,c);
    return p + p - c;
}
// minimum distance from point c to line through a and b
double dist_from_point_to_line(PT a, PT b, PT c) {
    return fabs(cross(b - a, c - a) / (b - a).norm());
}
// 0 if not parallel, 1 if parallel, 2 if collinear
int is_parallel(PT a, PT b, PT c, PT d) {
    double k = fabs(cross(b - a, d - c));
    if (k < eps){
        if (fabs(cross(a - b, a - c)) < eps && fabs(cross(c -
             d, c - a)) < eps) return 2;
        else return 1;
    }
    else return 0;
}
// check if two lines are same
bool are_lines_same(PT a, PT b, PT c, PT d) {
    if (fabs(cross(a - c, c - d)) < eps && fabs(cross(b - c, c
         - d)) < eps) return true;
    return false;
}
```

```cpp
// 1 if point is ccw to the line, 2 if point is cw to the line,
     3 if point is on the line
int point_line_relation(PT a, PT b, PT p) {
    int c = sign(cross(p - a, b - a));
    if (c < 0) return 1;
    if (c > 0) return 2;
    return 3;
}
```

5 Graph

5.1 AuxTree

```cpp
#include <bits/stdc++.h>

#define task "BriantheCrab"

#define int  long long
#define pii  pair <int, int>
#define fi   first
#define se   second
#define szf  sizeof
#define sz(s) (int)((s).size())

using namespace std;

template <class T> void mini (T &t, T f) {if (t > f) t = f;}
template <class T> void maxi (T &t, T f) {if (t < f) t = f;}

const int maxN = 2e5 + 5;
const int inf = 1e18 + 7;
const int mod = 1e9 + 7;
const int LOG = 20;

vector <int> adj[maxN], aux[maxN * 2];
int dfsTime = 0;
int tIn[maxN], tOut[maxN], h[maxN];
int up[maxN][LOG];

void dfs (int u, int p) {
    tIn[u] = ++ dfsTime;
    up[u][0] = p;
    for (int j = 1; j < LOG; j ++) {
        up[u][j] = ((up[u][j - 1] == -1) ? -1 : up[up[u][j -
             1]][j - 1]);
    }
    for (auto v : adj[u]) {
        if (v != p) {
            h[v] = h[u] + 1;
            dfs (v, u);
        }
    }
    tOut[u] = dfsTime;
}

bool isAnces (int u, int v) {
    return (tIn[u] <= tIn[v] && tIn[v] <= tOut[u]);
}

int lca (int u, int v) {
    if (isAnces (u, v)) {
        return u;
    }
    if (isAnces (v, u)) {
        return v;
    }
    for (int j = LOG - 1; j >= 0; j --) {
        if (up[u][j] != -1 && !isAnces (up[u][j], v)) {
            u = up[u][j];
        }
    }
    u = up[u][0];
    return u;
}

bool cmp (int u, int v) {
    return tIn[u] < tIn[v];
}

int buildAuxiliary (vector <int> &all) {
    int sz = (int) all.size ();
    sort (all.begin (), all.end (), cmp);
    for (int i = 0; i < sz - 1; i ++) {
        int newVer = lca (all[i], all[i + 1]);
        all.push_back (newVer);
    }
    sort (all.begin (), all.end (), cmp);
    all.erase (unique (all.begin (), all.end ()), all.end ());
    stack <int> st;
    int auxRoot = all[0];
    st.push (auxRoot);
    for (int i = 1; i < all.size (); i ++) {
        int u = all[i];
        while (!st.empty () && !isAnces (st.top (), u)) {
            st.pop ();
        }
        int last = st.top ();
        aux[last].push_back (u);
        //cout << last << ' ' << u << '\n';
        st.push (u);
    }
    return auxRoot;
}

void solve () {
    int n;
    cin >> n;
    for (int i = 1; i < n; i ++) {
        int u, v;
        cin >> u >> v;
        adj[u].push_back (v);
        adj[v].push_back (u);
    }
    dfs (1, -1);
    int q;
    cin >> q;
    while (q --) {
        int k;
        cin >> k;
        vector <int> all (k);
        for (int i = 0; i < k; i ++) {
            cin >> all[i];
        }
        int auxRoot = buildAuxiliary (all);
        bool passable = true;
        for (int u : all) {
            if (aux[u].size() > 1 + (u == auxRoot)) {
                passable = false;
                break;
            }
        }
        cout << (passable ? "YES\n" : "NO\n");
        for (int u : all) {
            aux[u].clear ();
        }
    }
}

signed main () {
    cin.tie (nullptr) -> sync_with_stdio (false);
    if (fopen (task".inp", "r")) {
        freopen (task".inp", "r", stdin);
        freopen (task".out", "w", stdout);
    }
    int t = 1;
    //cin >> t;
    while (t --) {
        solve ();
    }
    return 0;
}
// thfdgb
```

5.2 capGhepCucDai

```
#include <bits/stdc++.h>

#define task    "BriantheCrab"

#define int   long long
#define pii   pair <int, int>
#define fi    first
#define se    second
#define szf   sizeof
#define sz(s) (int)((s).size())
#define all(v) (v).begin(), (v).end()

typedef long long ll;
typedef unsigned long long ull;
typedef long double ld;

using namespace std;

template <class T> void minimize (T &t, T f) {if (t > f) t = f;}
template <class T> void maximize (T &t, T f) {if (t < f) t = f;}

const int maxN = 2e5 + 5;
const int inf = 1e18 + 7;
const int mod = 1e9 + 7;

int a[maxN];
vector <int> adj[maxN];
int V1, V2, E;
int match_v[maxN];
bool visited[maxN];

bool dfs_augment (int u) {
    for (int v : adj[u]) {
        if (!visited[v]) {
            visited[v] = true;
            if (match_v[v] == 0 || dfs_augment (match_v[v])) {
                match_v[v] = u;
                return true;
            }
        }
    }
    return false;
}

void solve () {
    cin >> V1 >> V2 >> E;
    for (int i = 0; i < E; ++ i) {
        int u, v;
        cin >> u >> v;
        adj[u].push_back (v);
    }

    int matching_size = 0;
    for (int u = 1; u <= V1; ++ u) {
        for (int i = 1; i <= V2; ++ i) {
            visited[i] = false;
        }
        if (dfs_augment (u)) {
            matching_size ++;
        }
    }

    cout << matching_size << '\n';

    return;
}

signed main () {
    cin.tie (nullptr) -> sync_with_stdio (false);
    if (fopen (task".inp", "r")) {
        freopen (task".inp", "r", stdin);
        freopen (task".out", "w", stdout);
    }
    int t = 1;
    //cin >> t;
    while (t --) {
        solve ();
    }
    return 0;
}
// thfv
```

5.3 CauKhop

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

const int N = 1e5 + 5;

int n, m;

bool joint[N];
int timeDfs = 0, brigde = 0;
int low[N], num[N];
vector <int> adj[N];

void dfs (int u, int pre) {
    int child = 0;
    num[u] = low[u] = ++timeDfs;
    for (auto v : adj[u]) {
        if (v == pre) {
            continue;
        }
        if (!num[v]) {
            dfs (v, u);
            low[u] = min (low[v], low[u]);
            if (low[v] == num[v]) {
                brigde ++;
            }
            child ++;
            if (u == pre) {
                if (child > 1) {
                    joint[u] = true;
                }
            }
            else if (low[v] >= num[u]) {
                joint[u] = true;
            }
        }
        else {
            low[u] = min (low[u], num[v]);
        }
    }
}
int main () {
    cin >> n >> m;
    for (int i = 1; i <= m; i ++) {
        int u, v;
        cin >> u >> v;
        adj[u].push_back (v);
        adj[v].push_back (u);
    }
    for (int i = 1; i <= n; i ++) {
        if (!num[i]) {
            dfs (i, i);
        }
    }
    int cntjoint = 0;
    for (int i = 1; i <= n; i ++) {
        cntjoint += (joint[i]);
    }
    cout << cntjoint << '\n';
    for (int i = 1; i <= n; i ++) {
        if (joint[i]) cout << i << " ";
    }
}
```

5.4 centroidDecomp

```
#include <bits/stdc++.h>

#define task "BriantheCrab"

#define int  long long
#define pii  pair <int, int>
#define fi   first
#define se   second
#define szf  sizeof
#define sz(s) (int)((s).size())

using namespace std;

template <class T> void mini (T &t, T f) {if (t > f) t = f;}
template <class T> void maxi (T &t, T f) {if (t < f) t = f;}

const int maxN = 2e5 + 5;
const int inf = 1e18 + 7;
const int mod = 1e9 + 7;

int n, k;
int sz[maxN], cnt[maxN], L[maxN];
bool del[maxN];
vector <int> adj[maxN], tmp;

void szCal (int u, int p) {
    sz[u] = 1;
    for (auto v : adj[u]) {
        if (v == p || del[v]) {
            continue;
        }
        szCal (v, u);
        sz[u] += sz[v];
    }
}

int findCen (int u, int p, int curSz) {
    for (auto v : adj[u]) {
        if (v == p || del[v]) {
            continue;
        }
        if (sz[v] > curSz / 2) {
            return findCen (v, u, curSz);
        }
    }
    return u;
}

void calDepth (int u, int p, int curD) {
    if (curD > k && k != -1) {
        return;
    }
    L[u] = curD;
    tmp.push_back (curD);
    for (int v : adj[u]) {
        if (v != p && !del[v]) {
            calDepth (v, u, curD + 1);
        }
    }
}

int cal (int u) {
    szCal (u, 0);
    int rt = findCen (u, 0, sz[u]);
    for (int i = 0; i <= sz[rt]; ++i) {
        cnt[i] = 0;
    }
    cnt[0] = 1;
    int res = 0;
    for (int v : adj[rt]) {
        if (!del[v]) {
            tmp.clear ();
            calDepth (v, rt, 1);
            for (int it : tmp) {
                if (k - it >= 0 && (k - it) <= sz[rt]) {
                    res += cnt[k - it];
                }
            }
            for (int it : tmp) {
                if (it <= sz[rt]) {
                    cnt[it] ++;
                }
            }
        }
    }
    del[rt] = true;
    for (int v : adj[rt]) {
        if (!del[v]) {
            res += cal(v);
        }
    }
    return res;
}

void solve () {
    cin >> n >> k;
    for (int i = 1; i < n; i ++) {
        int u, v;
        cin >> u >> v;
        adj[u].push_back (v);
        adj[v].push_back (u);
    }
    cout << cal (1);
    return;
}

signed main () {
    cin.tie (nullptr) -> sync_with_stdio (false);
    if (fopen (task".inp", "r")) {
        freopen (task".inp", "r", stdin);
        freopen (task".out", "w", stdout);
    }
    int t = 1;
    //cin >> t;
    while (t --) {
        solve ();
    }
    return 0;
}
// thfdgb
```

5.5 centroidTree

```
#include <bits/stdc++.h>

#define task "htgdtctn"

#define int  long long
#define pii  pair <int, int>
#define fi   first
#define se   second
#define szf  sizeof
#define sz(s) (int)((s).size())
#define all(v) (v).begin(), (v).end()

using namespace std;

template <class T> void minimize (T &t, T f) {if (t > f) t = f;}
template <class T> void maximize (T &t, T f) {if (t < f) t = f;}

const int maxN = 1e5 + 5;
const int inf = 1e18 + 7;
const int mod = 1e9 + 7;

struct Fenwick {
    int n;
    vector <int> bit;

    inline void init (int _n) {
        n = _n;
        bit.assign (n + 2, 0);
    }

    inline void upd (int id, int val) {
        for (; id <= n; id += id & (-id)) {
            bit[id] += val;
        }
    }
    inline int get (int id) {
        int res = 0;
        for (; id; id -= id & (-id)) {
            res += bit[id];
        }
        return res;
    }
};

struct Fire {
    int t, u, c, s;
};

struct Query {
    int type, time, node, id;
};

bool cmp (Query A, Query B) {
    if (A.time != B.time) {
        return A.time < B.time;
    }
    return A.type < B.type;
}

int n, m, q;
Fire a[maxN];
pii b[maxN];
vector <Query> qry;
vector <int> adj[maxN];
Fenwick T1[maxN], T2[maxN];
int sz[maxN], par[maxN], del[maxN];
int h[maxN];
int res[maxN];
int rtCen;

int dfsSz (int u, int p) {
    sz[u] = 1;
    for (auto v : adj[u]) {
        if (v == p || del[v]) {
            continue;
        }
        sz[u] += dfsSz (v, u);
    }
    return sz[u];
}

int findCen (int u, int p, int curSz) {
    for (auto v : adj[u]) {
        if (v == p || del[v]) {
            continue;
        }
        if (sz[v] * 2 > curSz) {
            return findCen (v, u, curSz);
        }
    }
    return u;
}

vector <int> comp;
```

```
void collect (int u, int p) {
    comp.push_back (u);
    for (auto v : adj[u]) {
        if (v == p || del[v]) {
            continue;
        }
        collect (v, u);
    }
}

vector <int> chainC[maxN];

void build (int u, int d, int lstC) {
    int curSz = dfsSz (u, u);
    int c = findCen (u, 0, curSz);
    if (lstC != 0) {
        par[c] = lstC;
    }
    else {
        par[c] = 0;
        rtCen = c;
    }
    comp.clear ();
    collect (c, 0);
    for (auto x : comp) {
        chainC[x].push_back (c);
    }
    del[c] = 1;
    for (auto v : adj[c]) {
        if (del[v]) {
            continue;
        }
        build (v, d + 1, c);
    }
}

int up[maxN][18];

void pre (int u, int p) {
    up[u][0] = p;
    for (int j = 1; j < 18; j ++) {
        up[u][j] = up[up[u][j - 1]][j - 1];
    }
    for (auto v : adj[u]) {
        if (v == p) {
            continue;
        }
        h[v] = h[u] + 1;
        pre (v, u);
    }
}

inline int lca (int u, int v) {
    if (h[u] < h[v]) {
        swap (u, v);
    }
    int k = h[u] - h[v];
    for (int i = 17; i >= 0; i --) {
        if ((k >> i) & 1) {
            u = up[u][i];
```

```
        }
    }
    if (u == v) {
        return u;
    }
    for (int i = 17; i >= 0; i --) {
        if (up[u][i] != up[v][i]) {
            u = up[u][i];
            v = up[v][i];
        }
    }
    return up[u][0];
}

inline int dist (int u, int v) {
    return h[u] + h[v] - 2 * h[lca (u, v)];
}

vector <int> zip1[maxN], zip2[maxN];

inline void FjumpUpd (int u, int curDis, int val) {
    for (int k = 0; k < sz (chainC[u]); k ++) {
        int p = chainC[u][k];
        int child = (k ? chainC[u][k - 1] : 0);
        int d1 = dist (p, u);
        if (curDis - d1 >= 0) {
            zip1[p].push_back (curDis - d1);
            if (child != 0) {
                zip2[child].push_back (curDis - d1);
            }
        }
    }
}

inline void FjumpGet (int u) {
    for (int k = 0; k < sz (chainC[u]); k ++) {
        int p = chainC[u][k];
        int child = (k ? chainC[u][k - 1] : 0);
        int d1 = dist (p, u);
        zip1[p].push_back (d1);
        if (child != 0) {
            zip2[child].push_back (d1);
        }
    }
}

inline void jumpUpd (int u, int curDis, int val) {
    for (int k = 0; k < sz (chainC[u]); k ++) {
        int p = chainC[u][k];
        int child = (k ? chainC[u][k - 1] : 0);
        int d1 = dist (p, u);
        if (curDis - d1 >= 0) {
            int x1 = lower_bound (all (zip1[p]), curDis - d1) -
                zip1[p].begin () + 1;
            T1[p].upd (x1, val);
            if (child != 0) {
                int x2 = lower_bound (all (zip2[child]), curDis
                    - d1) - zip2[child].begin () + 1;
                T2[child].upd (x2, val);
            }
```

```cpp
        }
    }
}

inline int jumpGet (int u) {
    int ans = 0;
    for (int k = 0; k < sz (chainC[u]); k ++) {
        int p = chainC[u][k];
        int child = (k ? chainC[u][k - 1] : 0);
        int d1 = dist (p, u);
        int x1 = lower_bound (all (zip1[p]), d1) -
            zip1[p].begin () + 1;
        ans += T1[p].get (T1[p].n) - T1[p].get (x1 - 1);
        if (child != 0) {
            int x2 = lower_bound (all (zip2[child]), d1) -
                zip2[child].begin () + 1;
            ans -= T2[child].get (T2[child].n) - T2[child].get
                (x2 - 1);
        }
    }
    return ans;
}

inline void Solve () {
    cin >> n >> m >> q;
    for (int i = 1; i <= n - 1; i ++) {
        int u, v;
        cin >> u >> v;
        adj[u].push_back (v);
        adj[v].push_back (u);
    }
    for (int i = 1; i <= m; i ++) {
        int t, u, c, s;
        cin >> t >> u >> c >> s;
        a[i] = {t, u, c, s};
        qry.push_back ({0, t, u, i});
    }
    for (int i = 1; i <= q; i ++) {
        cin >> b[i].fi >> b[i].se;
        qry.push_back ({1, b[i].fi, b[i].se, i});
    }
    build (1, 0, 0);
    pre (1, 1);
    for (int u = 1; u <= n; u ++) {
        reverse (all (chainC[u]));
    }
    sort (all (qry), cmp);
    for (int i = 0; i < sz (qry); i ++) {
        auto [type, time, node, id] = qry[i];
        if (type == 0) {
            FjumpUpd (node, a[id].s, a[id].c);
        }
        else {
            FjumpGet (node);
        }
    }
    for (int i = 1; i <= n; i ++) {
        sort (all (zip1[i]));
        zip1[i].erase (unique (all (zip1[i])), zip1[i].end ());
        sort (all (zip2[i]));
        zip2[i].erase (unique (all (zip2[i])), zip2[i].end ());
        T1[i].init (sz (zip1[i]));
        T2[i].init (sz (zip2[i]));
    }
    for (int i = 0; i < sz (qry); i ++) {
        auto [type, time, node, id] = qry[i];
        if (type == 0) {
            jumpUpd (node, a[id].s, a[id].c);
        }
        else {
            res[id] = jumpGet (node);
        }
    }
    for (int i = 1; i <= q; i ++) {
        cout << res[i] << '\n';
    }
    return;
}

signed main () {
    cin.tie (nullptr) -> sync_with_stdio (false);
    if (fopen (task".inp", "r")) {
        freopen (task".inp", "r", stdin);
        freopen (task".out", "w", stdout);
    }
    int t = 1;
    while (t --) {
        Solve ();
    }
    return 0;
}
// wake me up when September ends
```

5.6 chonViecFlow

```cpp
#include <bits/stdc++.h>

#define task "BriantheCrab"

#define int   long long
#define pii   pair <int, int>
#define fi    first
#define se    second
#define szf   sizeof
#define sz(s) (int)((s).size())

using namespace std;

template <class T> void mini (T &t, T f) {if (t > f) t = f;}
template <class T> void maxi (T &t, T f) {if (t < f) t = f;}

const int maxN = 1e3 + 5;
const int inf = 1e18 + 7;
const int mod = 1e9 + 7;

int n, m, s, t;
int maxFlow;
int c[maxN][maxN], f[maxN][maxN], trace[maxN];
vector <int> adj[maxN];

void bfs () {
    fill (trace, trace + n + 4, 0);
    trace[s] = -1;
    queue <int> q;
    q.push (s);
    while (!q.empty ()) {
        int u = q.front ();
        q.pop ();
        for (auto v : adj[u]) {
            if (trace[v]) {
                continue;
            }
            if (f[u][v] - c[u][v] == 0) {
                continue;
            }
            trace[v] = u;
            q.push (v);
        }
    }
}

void incFlow () {
    int delta = inf;
    int v = t;
    while (v != s) {
        int u = trace[v];
        mini (delta, c[u][v] - f[u][v]);
        v = u;
    }
    maxFlow += delta;
    v = t;
    while (v != s) {
        int u = trace[v];
        f[u][v] += delta;
        f[v][u] -= delta;
        v = u;
    }
}

void solve () {
    cin >> n;
    s = n + 1, t = n + 2;
    int add = 0;
    for (int i = 1; i <= n; i ++) {
        int x;
        cin >> x;
        if (x > 0) {
            adj[s].push_back (i);
            adj[i].push_back (s);
            c[s][i] = x;
            add += x;
            //cout << s << ' ' << i << ' ' << x << '\n';
        }
        else {
            adj[i].push_back (t);
            adj[t].push_back (i);
            c[i][t] = -x;
            //cout << i << ' ' << t << ' ' << -x << '\n';
        }
    }
    cin >> m;
    for (int i = 1; i <= m; i ++) {
        int u, v;
        cin >> u >> v;
        c[u][v] = add + 1;
        adj[u].push_back (v);
        adj[v].push_back (u);
        //cout << u << ' ' << v << ' ' << inf << '\n';
    }
    maxFlow = 0;
    do {
        bfs ();
        //cout << trace[t] << '\n';
        if (trace[t]) {
            incFlow ();
        }
    } while (trace[t]);
    cout << add - maxFlow;
    return;
}

signed main () {
    cin.tie (nullptr) -> sync_with_stdio (false);
    if (fopen (task".inp", "r")) {
        freopen (task".inp", "r", stdin);
        freopen (task".out", "w", stdout);
    }
    int t = 1;
    //cin >> t;
    while (t --) {
        solve ();
    }
    return 0;
}
// thfdgb
```

5.7 DynamicConectivity

```
#include <bits/stdc++.h>

#define task   "BriantheCrab"

#define int   long long
#define pii   pair <int, int>
#define fi    first
#define se    second
#define szf   sizeof
#define sz(s) (int)((s).size())
#define all(v) (v).begin(), (v).end()

typedef long long ll;
typedef unsigned long long ull;
typedef long double ld;

using namespace std;

template <class T> void minimize (T &t, T f) {if (t > f) t = f;}
template <class T> void maximize (T &t, T f) {if (t < f) t = f;}

const int maxN = 5e5 + 5;
const int inf = 1e18 + 7;
const int mod = 1e9 + 7;

int n;
map <pii, int> mp;
int res = 0;
int sad[maxN * 4];
struct Edge {
    int u, v;
};

vector <Edge> Ed[maxN];

struct DsuRollBack {
    vector <int> par, sz;
    stack <pii> rollback;
    vector <vector <pii>> ed;
    int sum = 0;

    void init (int _n, int _q) {
        par.resize (_n + 2);
        sz.resize (_n + 2);
        ed.clear ();
        ed.resize (_q * 4 + 2);
        sum = 0;
        for (int i = 1; i <= _n; i ++) {
            sz[i] = 1;
            par[i] = i;
        }
        while (!rollback.empty ()) rollback.pop ();
    }

    inline int getRoot (int u) {
        while (par[u] != u) {
            u = par[u];
        }
        return u;
    }

    void upd (int id, int l, int r, int L, int R, pii curEd) {
        if (l > R || r < L) {
            return;
        }
        if (l >= L && r <= R) {
            ed[id].push_back (curEd);
            return;
        }
        int mid = (l + r) >> 1;
        upd (id * 2, l, mid, L, R, curEd);
        upd (id * 2 + 1, mid + 1, r, L, R, curEd);
    }

    void get (int id, int l, int r) {
        int cnt = 0;
        for (auto [u, v] : ed[id]) {
            int rootU = getRoot (u);
            int rootV = getRoot (v);
            if (rootU == rootV) {
                continue;
            }
            if (sz[rootU] < sz[rootV]) {
                swap (rootU, rootV);
            }
            rollback.push ({rootV, sz[rootV]});
            rollback.push ({rootU, sz[rootU]});
            cnt += 2;
            par[rootV] = rootU;
            sz[rootU] += sz[rootV];
        }

        if (l == r) {
            //cout << sad[l] << ' ' << l << '\n';
            //cout << sad[l] << '\n';
            for (auto [u, v] : Ed[sad[l]]) {
                //cout << u << ' ' << v << '\n';
                u = getRoot (u);
                v = getRoot (v);
                //cout << u << ' ' << v << '\n';
                //cout << sz[u] << ' ' << sz[v] << '\n';
                res += sz[u] * sz[v];
            }
        }
        else {
            int mid = (l + r) >> 1;
            get (id * 2, l, mid);
            get (id * 2 + 1, mid + 1, r);
        }
        while (cnt > 0) {
            auto [u, s1] = rollback.top (); rollback.pop ();
            auto [v, s2] = rollback.top (); rollback.pop ();
            cnt -= 2;
            par[u] = u;
            par[v] = v;
            sz[u] = s1;
            sz[v] = s2;
        }
    }
} dsu;

struct query {
    int u, v, type;
};

vector <query> qry;

void solve () {
    cin >> n;
    for (int i = 1; i <= n - 1; i ++) {
        int u, v, w;
        cin >> u >> v >> w;
        if (u > v) {
            swap (u, v);
        }
        Ed[w].push_back ({u, v});
    }
    for (int i = 1; i <= n; i ++) {
        for (auto [u, v] : Ed[i]) {
            //mp[{u, v}] = i;
            qry.push_back ({u, v, 1});
        }
    }
    for (int i = 1; i <= n; i ++) {
        if (sz (Ed[i]) == 0) {
            continue;
        }
        for (auto [u, v] : Ed[i]) {
            qry.push_back ({u, v, 2});
        }
        qry.push_back ({i, i, 3});
        for (auto [u, v] : Ed[i]) {
            qry.push_back ({u, v, 1});
        }
    }
    int q = sz (qry);
    dsu.init (n, q);
    for (int i = 1; i <= sz (qry); i ++) {
        auto [u, v, t] = qry[i - 1];
        if (t == 1) {
            mp[{u, v}] = i;
        }
        else if (t == 2) {
            auto it = mp.find ({u, v});
            if (it != mp.end ()) {
                dsu.upd (1, 1, q, it -> se, i - 1, {u, v});
                mp.erase (it);
            }
        }
        else {
            //cout << i << ' ' << sad[i] << '\n';
            sad[i] = u;
        }
    }
    for (auto it : mp) {
        int u = it.first.first;
        int v = it.first.second;
        int L = it.second;
        dsu.upd(1, 1, q, L, q, {u, v});
    }
    mp.clear ();
    dsu.get (1, 1, q);
    cout << res;
}

signed main () {
    cin.tie (nullptr) -> sync_with_stdio (false);
    if (fopen (task".inp", "r")) {
        freopen (task".inp", "r", stdin);
        freopen (task".out", "w", stdout);
    }
    int t = 1;
    //cin >> t;
    while (t --) {
        solve ();
    }
    return 0;
}
// thfv
```

5.8 euler

```
int n, m;
bool visited[maxN];
vector <pii> adj[maxN];
vector <pii> ed;
int deg[maxN];
vector <vector <int>> res;
map <pii, int> mp;

list <pii> euler (int u) {
    list <pii> ans;
    while (!adj[u].empty()) {
        auto [v, id] = adj[u].back ();
        adj[u].pop_back ();
        if (visited[id]) {
            continue;
        }
        visited[id] = true;
        auto t = euler (v);
        ans.splice (ans.end (), t);
        ans.push_back ({v, id > m});
    }
    return ans;
}

inline void Solve () {
    cin >> n >> m;
    for (int i = 1; i <= m; i ++) {
        int u, v;
        cin >> u >> v;
        adj[u].push_back ({v, i});
        adj[v].push_back ({u, i});
        mp[{u, v}] ++;
        deg[u] ++;
        deg[v] ++;
    }
    vector <int> odd;
    for (int i = 1; i <= n; i ++) {
        if (deg[i] & 1) {
            odd.push_back (i);
        }
    }
    int cnt = m;
    for (int i = 0; i + 1 < sz (odd); i += 2) {
        int u = odd[i];
        int v = odd[i + 1];
        cnt ++;
        adj[u].push_back ({v, cnt});
        adj[v].push_back ({u, cnt});
        //cout << u << ' ' << v << '\n';
    }
    for (int i = 0; i + 1 < sz (odd); i += 2) {
        auto cur = euler (odd[i]);
        if (sz (cur) <= 1) {
            continue;
        }
        vector <int> ok;
        for (auto [it, t] : cur) {
            ok.push_back (it);
            if (t == 1) {
                res.push_back (ok);
                ok.clear ();
            }
        }
        if (sz (ok) != 0) {
            res.push_back (ok);
        }
    }
    cout << sz (res) << '\n';
    for (auto cur : res) {
        vector <int> v;
        for (auto it : cur) {
            v.push_back (it);
        }
        cout << sz (v) << ' ';
        for (int i = 0; i < sz (v); i ++) {
            cout << v[i] << ' ';
        }
        cout << '\n';
    }
    return;
}
```

5.9 flow

```
#include <bits/stdc++.h>

#define task "BriantheCrab"

#define int  long long
#define pii  pair <int, int>
#define fi   first
#define se   second
#define szf  sizeof
#define sz(s) (int)((s).size())

using namespace std;

template <class T> void mini (T &t, T f) {if (t > f) t = f;}
template <class T> void maxi (T &t, T f) {if (t < f) t = f;}

const int maxN = 1e3 + 5;
const int inf = 1e18 + 7;
const int mod = 1e9 + 7;

int n, m, s, t;
int maxFlow;
int c[maxN][maxN], f[maxN][maxN], trace[maxN];
vector <int> adj[maxN];
```

```
void bfs () {
    fill (trace, trace + n + 1, 0);
    trace[s] = -1;
    queue <int> q;
    q.push (s);
    while (!q.empty ()) {
        int u = q.front ();
        q.pop ();
        for (auto v : adj[u]) {
            if (trace[v]) {
                continue;
            }
            if (f[u][v] - c[u][v] == 0) {
                continue;
            }
            trace[v] = u;
            q.push (v);
        }
    }
}

void incFlow () {
    int delta = inf;
    int v = t;
    while (v != s) {
        int u = trace[v];
        mini (delta, c[u][v] - f[u][v]);
        v = u;
    }
    maxFlow += delta;
    v = t;
    while (v != s) {
        int u = trace[v];
        f[u][v] += delta;
        f[v][u] -= delta;
        v = u;
    }
}

void solve () {
    cin >> n >> m >> s >> t;
    for (int i = 1; i <= m; i ++) {
        int u, v;
        cin >> u >> v;
        cin >> c[u][v];
        adj[u].push_back (v);
        adj[v].push_back (u);
    }
    maxFlow = 0;
    do {
        bfs ();
        if (trace[t]) {
            incFlow ();
        }
    } while (trace[t]);
    cout << maxFlow;
    return;
}
```

```
signed main () {
    cin.tie (nullptr) -> sync_with_stdio (false);
    if (fopen (task".inp", "r")) {
        freopen (task".inp", "r", stdin);
        freopen (task".out", "w", stdout);
    }
    int t = 1;
    //cin >> t;
    while (t --) {
        solve ();
    }
    return 0;
}
// thfdgb
```

5.10 heavyLightLiftingLubenica

```
#include <bits/stdc++.h>

#define task "BriantheCrab"

#define ll   long long
#define pii  pair <int, int>
#define fi   first
#define se   second
#define szf  sizeof
#define sz(s) (int)((s).size())

using namespace std;

template <class T> void mini (T &t, T f) {if (t > f) t = f;}
template <class T> void maxi (T &t, T f) {if (t < f) t = f;}

const int maxN = 1e5 + 5;
const int inf = 1e9 + 7;
const int mod = 1e9 + 7;

struct ST {
    pii st[maxN * 4];

    ST () {
        for (int i = 0; i < maxN * 4; i ++) {
            st[i] = {-inf, inf};
        }
    }

    void Upd (int id, int l, int r, int pos, int val) {
        if (pos < l || pos > r) {
            return;
        }
        if (l == r) {
            st[id] = {val, val};
            return;
        }
        int mid = (l + r) >> 1;
        Upd (id * 2, l, mid, pos, val);
        Upd (id * 2 + 1, mid + 1, r, pos, val);
```

```
        int tMax = max (st[id * 2].fi, st[id * 2 + 1].fi);
        int tMin = min (st[id * 2].se, st[id * 2 + 1].se);
        st[id] = {tMax, tMin};
        return;
    }

    pii Get (int id, int l, int r, int u, int v) {
        if (u > r || v < l) {
            return {-inf, inf};
        }
        if (u <= l && r <= v) {
            return st[id];
        }
        int mid = (l + r) >> 1;
        auto tL = Get (id * 2, l, mid, u, v);
        auto tR = Get (id * 2 + 1, mid + 1, r, u, v);
        return {max (tL.fi, tR.fi), min (tL.se, tR.se)};
    }
};

struct edge {
    int u, v, w;
};

int n, a[maxN];
int sz[maxN], par[maxN], head[maxN], h[maxN];
int nodeId[maxN], pos[maxN], cnt = 0;
vector <int> adj[maxN];
vector <edge> all;
ST T;

void dfs (int u, int p) {
    sz[u] = 1;
    par[u] = p;
    for (auto v : adj[u]) {
        if (v == p) {
            continue;
        }
        h[v] = h[u] + 1;
        dfs (v, u);
        sz[u] += sz[v];
    }
}

void HLD (int u, int h, int p) {
    head[u] = h;
    pos[u] = ++cnt;
    nodeId[cnt] = u;
    if (adj[u].size () == 1 && u != 1) {
        return;
    }
    int nxt = 0, curMax = 0;
    for (auto v : adj[u]) {
        if (v == p) {
            continue;
        }
        if (sz[v] > curMax) {
            curMax = sz[v];
            nxt = v;
        }
    }
    HLD (nxt, h, u);
    for (auto v : adj[u]) {
        if (v != nxt && v != p) {
            HLD (v, v, u);
        }
    }
}

int LCA (int u, int v) {
    while (head[u] != head[v]) {
        if (sz[head[u]] < sz[head[v]]) {
            u = par[head[u]];
        }
        else {
            v = par[head[v]];
        }
    }
    return ((sz[u] > sz[v]) ? u : v);
}

pii query (int u, int v) {
    int res1 = 0, res2 = inf;
    int lca = LCA (u, v);
    while (head[u] != head[v]) {
        if (h[head[u]] < h[head[v]]) {
            swap (u, v);
        }
        pii tmp = T.Get (1, 1, n, pos[head[u]] + (head[u] ==
             lca), pos[u]);
        maxi (res1, tmp.fi);
        mini (res2, tmp.se);
        u = par[head[u]];
    }
    if (h[u] > h[v]) {
        swap (u, v);
    }
    pii tmp = T.Get (1, 1, n, pos[u] + (u == lca), pos[v]);
    maxi (res1, tmp.fi);
    mini (res2, tmp.se);
    return {res1, res2};
}

void Solve () {
    cin >> n;
    for (int i = 1; i <= n - 1; i ++) {
        int u, v, w;
        cin >> u >> v >> w;
        all.push_back ({u, v, w});
        adj[u].push_back (v);
        adj[v].push_back (u);
    }
    dfs (1, 1);
    HLD (1, 1, 1);
    for (auto [u, v, w] : all) {
        if (pos[u] < pos[v]) {
            swap (u, v);
        }
        T.Upd (1, 1, n, pos[u], w);
    }
    int q;
    cin >> q;
    //cout << q << '\n';
    while (q --) {
        int u, v;
        cin >> u >> v;
        auto [mx, mi] = query (u, v);
        cout << mi << ' ' << mx << '\n';
    }
    return;
}

signed main () {
    cin.tie (nullptr) -> sync_with_stdio (false);
    if (fopen (task".inp", "r")) {
        freopen (task".inp", "r", stdin);
        freopen (task".out", "w", stdout);
    }
    int t = 1;
    //cin >> t;
    while (t --) {
        Solve ();
    }
    return 0;
}
// Belligerent :)
```

5.11 heavyLightLiftingPointUpdateMaxRange

```
#include <bits/stdc++.h>

#define task "BriantheCrab"

#define ll   long long
#define pii  pair <int, int>
#define fi   first
#define se   second
#define szf  sizeof
#define sz(s) (int)((s).size())

using namespace std;

template <class T> void mini (T &t, T f) {if (t > f) t = f;}
template <class T> void maxi (T &t, T f) {if (t < f) t = f;}

const int maxN = 1e5 + 5;
const int inf = 1e9 + 7;
const int mod = 1e9 + 7;

struct ST {
    int st[maxN * 4];

    ST () {
        memset (st, 0, szf (st));
    }

    void Upd (int id, int l, int r, int pos, int val) {
        if (pos < l || pos > r) {
            return;
        }
        if (l == r) {
            st[id] = val;
            return;
        }
        int mid = (l + r) >> 1;
        Upd (id * 2, l, mid, pos, val);
        Upd (id * 2 + 1, mid + 1, r, pos, val);
        st[id] = max (st[id * 2], st[id * 2 + 1]);
        return;
    }

    int Get (int id, int l, int r, int u, int v) {
        if (u > r || v < l) {
            return -inf;
        }
        if (u <= l && r <= v) {
            return st[id];
        }
        int mid = (l + r) >> 1;
        int tL = Get (id * 2, l, mid, u, v);
        int tR = Get (id * 2 + 1, mid + 1, r, u, v);
        return max (tL, tR);
    }
};

int n, a[maxN];
int sz[maxN], par[maxN], head[maxN], h[maxN];
int nodeId[maxN], pos[maxN], cnt = 0;
vector <int> adj[maxN];
ST T;

void dfs (int u, int p) {
    sz[u] = 1;
    par[u] = p;
    for (auto v : adj[u]) {
        if (v == p) {
            continue;
        }
        h[v] = h[u] + 1;
        dfs (v, u);
        sz[u] += sz[v];
    }
}

void HLD (int u, int h, int p) {
    head[u] = h;
    pos[u] = ++cnt;
    nodeId[cnt] = u;
    if (adj[u].size () == 1 && u != 1) {
        return;
    }
    int nxt = 0, curMax = 0;
    for (auto v : adj[u]) {
        if (v == p) {
            continue;
        }
        if (sz[v] > curMax) {
            curMax = sz[v];
            nxt = v;
        }
    }
    HLD (nxt, h, u);
    for (auto v : adj[u]) {
        if (v != nxt && v != p) {
            HLD (v, v, u);
        }
    }
}

int query (int u, int v) {
    int res = 0;
    while (head[u] != head[v]) {
        if (h[head[u]] < h[head[v]]) {
            swap (u, v);
        }
        maxi (res, T.Get (1, 1, n, pos[head[u]], pos[u]));
        u = par[head[u]];
    }
    if (h[u] > h[v]) {
        swap (u, v);
    }
    maxi (res, T.Get (1, 1, n, pos[u], pos[v]));
    return res;
}

void Solve () {
    int q;
    cin >> n >> q;
    for (int i = 1; i <= n; i ++) {
        cin >> a[i];
    }
    for (int i = 1; i <= n - 1; i ++) {
        int u, v;
        cin >> u >> v;
        adj[u].push_back (v);
        adj[v].push_back (u);
    }
    dfs (1, 1);
    HLD (1, 1, 1);
    for (int i = 1; i <= n; i ++) {
        T.Upd (1, 1, n, pos[i], a[i]);
    }
    while (q --) {
        int t;
        cin >> t;
        if (t == 1) {
            int s, x;
            cin >> s >> x;
            T.Upd (1, 1, n, pos[s], x);
        }
        else {
            int u, v;
            cin >> u >> v;
            cout << query (u, v) << '\n';
        }
    }
    return;
}

signed main () {
    cin.tie (nullptr) -> sync_with_stdio (false);
    if (fopen (task".inp", "r")) {
        freopen (task".inp", "r", stdin);
        freopen (task".out", "w", stdout);
    }
    int t = 1;
    //cin >> t;
    while (t --) {
        Solve ();
    }
    return 0;
}
// Belligerent :)
```

5.12 heavyLightLiftingRangeUpdateGetSum

```
#include <bits/stdc++.h>

#define task "BriantheCrab"

#define int  long long
#define pii  pair <int, int>
#define fi   first
#define se   second
#define szf  sizeof
#define sz(s) (int)((s).size())

using namespace std;

template <class T> void mini (T &t, T f) {if (t > f) t = f;}
template <class T> void maxi (T &t, T f) {if (t < f) t = f;}

const int maxN = 2e5 + 5;
const int inf = 1e9 + 7;
const int mod = 1e9 + 7;

struct ST {
    int st[maxN * 4];

    ST () {
        memset (st, 0, szf (st));
    }

    void Upd (int id, int l, int r, int pos, int val) {
        if (pos < l || pos > r) {
            return;
        }
        if (l == r) {
            st[id] = val;
            return;
        }
        int mid = (l + r) >> 1;
        Upd (id * 2, l, mid, pos, val);
        Upd (id * 2 + 1, mid + 1, r, pos, val);
        st[id] = st[id * 2] + st[id * 2 + 1];
        return;
    }

    int Get (int id, int l, int r, int u, int v) {
        if (u > r || v < l) {
            return 0;
        }
        if (u <= l && r <= v) {
            return st[id];
        }
        int mid = (l + r) >> 1;
        int tL = Get (id * 2, l, mid, u, v);
        int tR = Get (id * 2 + 1, mid + 1, r, u, v);
        return (tL + tR);
    }
};

int n, a[maxN];
int sz[maxN], par[maxN], head[maxN], h[maxN];
int nodeId[maxN], pos[maxN], cnt = 0;
vector <int> adj[maxN];
ST T;

void dfs (int u, int p) {
    sz[u] = 1;
    par[u] = p;
    for (auto v : adj[u]) {
        if (v == p) {
            continue;
        }
        h[v] = h[u] + 1;
        dfs (v, u);
        sz[u] += sz[v];
    }
}

void HLD (int u, int h, int p) {
    head[u] = h;
    pos[u] = ++cnt;
    nodeId[cnt] = u;
    if (adj[u].size () == 1 && u != 1) {
        return;
    }
    int nxt = 0, curMax = 0;
    for (auto v : adj[u]) {
        if (v == p) {
            continue;
        }
        if (sz[v] > curMax) {
            curMax = sz[v];
            nxt = v;
        }
    }
    HLD (nxt, h, u);
    for (auto v : adj[u]) {
        if (v != nxt && v != p) {
            HLD (v, v, u);
        }
    }
}

int query (int u, int v) {
    int res = 0;
    while (head[u] != head[v]) {
        if (h[head[u]] < h[head[v]]) {
            swap (u, v);
        }
        res += T.Get (1, 1, n, pos[head[u]], pos[u]);
        u = par[head[u]];
    }
    if (h[u] > h[v]) {
        swap (u, v);
    }
    res += T.Get (1, 1, n, pos[u], pos[v]);
    return res;
}

void Solve () {
    int q;
    cin >> n >> q;
    for (int i = 1; i <= n; i ++) {
        cin >> a[i];
    }
    for (int i = 1; i <= n - 1; i ++) {
        int u, v;
        cin >> u >> v;
        adj[u].push_back (v);
        adj[v].push_back (u);
    }
    dfs (1, 1);
    HLD (1, 1, 1);
    for (int i = 1; i <= n; i ++) {
        T.Upd (1, 1, n, pos[i], a[i]);
    }
    while (q --) {
        int t;
        cin >> t;
        if (t == 1) {
            int s, x;
            cin >> s >> x;
            T.Upd (1, 1, n, pos[s], x);
        }
        else {
            int x;
            cin >> x;
            cout << query (1, x) << '\n';
        }
    }
    return;
}

signed main () {
    cin.tie (nullptr) -> sync_with_stdio (false);
    if (fopen (task".inp", "r")) {
        freopen (task".inp", "r", stdin);
        freopen (task".out", "w", stdout);
    }
    int t = 1;
    //cin >> t;
    while (t --) {
        Solve ();
    }
    return 0;
}
// Belligerent :)
```

5.13 lcaO1

```
#include <bits/stdc++.h>

#define task   "BriantheCrab"

#define int   long long
#define pii   pair <int, int>
#define fi    first
#define se    second
#define szf   sizeof
#define sz(s) (int)((s).size())
#define all(v) (v).begin(), (v).end()

typedef long long ll;
typedef unsigned long long ull;
typedef long double ld;

using namespace std;

template <class T> void minimize (T &t, T f) {if (t > f) t = f;}
template <class T> void maximize (T &t, T f) {if (t < f) t = f;}

const int maxN = 2e5 + 5;
const int inf = 1e18 + 7;
const int mod = 1e9 + 7;

int a[maxN];

vector <int> adj[maxN];
vector <int> euler_tour;
vector <int> dep;
vector <int> first_occ;
vector <vector <int>> spt;
vector <int> log_tab;

void dfs (int u, int p, int d) {
    euler_tour.push_back (u);
    dep.push_back (d);
    if (first_occ[u] == -1) {
        first_occ[u] = sz(euler_tour) - 1;
    }
    for (int v : adj[u]) {
        if (v == p) {
            continue;
        }
        dfs (v, u, d + 1);
        euler_tour.push_back (u);
        dep.push_back (d);
    }
}

int query_rmq (int l, int r) {
    int k = log_tab[r - l + 1];
    int u = spt[l][k];
    int v = spt[r - (1 << k) + 1][k];
    if (dep[u] < dep[v]) {
        return u;
    }
    return v;
}

int lca (int u, int v) {
    int idx_u = first_occ[u];
    int idx_v = first_occ[v];
    if (idx_u > idx_v) {
        swap (idx_u, idx_v);
    }
    int res_idx = query_rmq (idx_u, idx_v);
    return euler_tour[res_idx];
}

void solve () {
    int n, q;
    cin >> n >> q;

    for (int i = 0; i < n - 1; ++ i) {
        int u, v;
        cin >> u >> v;
        adj[u].push_back (v);
        adj[v].push_back (u);
    }

    first_occ.assign (n + 1, -1);
    dfs (1, 0, 0); // Start DFS from node 1 with depth 0

    int len_euler = sz(euler_tour);
    log_tab.assign (len_euler + 1, 0);
    for (int i = 2; i <= len_euler; ++ i) {
        log_tab[i] = log_tab[i / 2] + 1;
    }

    int max_log = log_tab[len_euler];
    spt.assign (len_euler, vector <int> (max_log + 1));

    for (int i = 0; i < len_euler; ++ i) {
        spt[i][0] = i;
    }

    for (int j = 1; j <= max_log; ++ j) {
        for (int i = 0; i + (1 << j) <= len_euler; ++ i) {
            int u_idx = spt[i][j - 1];
            int v_idx = spt[i + (1 << (j - 1))][j - 1];
            if (dep[u_idx] < dep[v_idx]) {
                spt[i][j] = u_idx;
            } else {
                spt[i][j] = v_idx;
            }
        }
    }
    while (q --) {
        int u, v;
        cin >> u >> v;
        cout << lca (u, v) << "\n";
    }

    return;
}

signed main () {
    cin.tie (nullptr) -> sync_with_stdio (false);
    if (fopen (task".inp", "r")) {
        freopen (task".inp", "r", stdin);
        freopen (task".out", "w", stdout);
    }
    int t = 1;
    //cin >> t;
    while (t --) {
        solve ();
    }
    return 0;
}
// thfv
```

5.14 reRootDistanceOnTree

```
#include <bits/stdc++.h>

using namespace std;

int dist[2][200000];
vector<int> adj[200000];

int dfs(int u, int p, int d, int i) {
        dist[i][u] = d;
        int opt = -1;
        for (int v : adj[u]) {
                if (v != p) {
                        int x = dfs(v, u, d + 1, i);
                        if (opt == -1 || dist[i][x] >
                            dist[i][opt]) opt = x;
                }
        }
        return opt == -1 ? u : opt;
}

int main() {
        int n;
        cin >> n;
        for (int i = 0; i < n - 1; i++) {
                int a, b;
                cin >> a >> b;
                --a;
                --b;
                adj[a].push_back(b);
                adj[b].push_back(a);
        }
        int mxNode = dfs(0, 0, 0, 0);
        int mxNode2 = dfs(mxNode, mxNode, 0, 0);
        dfs(mxNode2, mxNode2, 0, 1);
        for (int i = 0; i < n; i++) {
                cout << max(dist[0][i], dist[1][i]) << " \n"[i
                     == n - 1];
        }
        return 0;
}
```

5.15 rmqLubenica

```
#include <bits/stdc++.h>

#define task  "SHIP"

#define int  long long
#define pii  pair <int, int>
#define fi   first
#define se   second
#define szf  sizeof
#define sz(s) (int)((s).size())

using namespace std;

template <class T> void mini (T &t, T f) {if (t > f) t = f;}
template <class T> void maxi (T &t, T f) {if (t < f) t = f;}

const int maxN = 2e5 + 5;
const int LOG = 20;
const int inf = 1e18 + 7;
const int mod = 1e9 + 7;

int n;
int a[maxN], h[maxN];
int mx[maxN][LOG], par[maxN][LOG];
vector <int> adj[maxN];
int dp[maxN];

void dfs (int u, int p) {
    par[u][0] = p;
    mx[u][0] = a[u];
    for (int j = 1; j <= LOG - 1; j ++) {
        par[u][j] = par[par[u][j - 1]][j - 1];
        mx[u][j] = max (mx[u][j - 1], mx[par[u][j - 1]][j - 1]);
    }
    for (auto v : adj[u]) {
        if (v == p) {
            continue;
        }
        h[v] = h[u] + 1;
        dfs (v, u);
    }
}

int LCA (int u, int v) {
    if (h[u] < h[v]) {
        swap (u, v);
    }
    int k = h[u] - h[v];
    int res = 0;
    for (int j = LOG - 1; j >= 0; j --) {
        if ((k >> j) & 1) {
            res = max (res, mx[u][j]);
            u = par[u][j];
        }
    }
    if (u == v) {
        return max ({res, a[u], a[v]});
    }
    for (int j = LOG - 1; j >= 0; j --) {
        if (par[u][j] != par[v][j]) {
            res = max ({res, mx[u][j], mx[v][j]});
            u = par[u][j];
            v = par[v][j];
        }
    }
    return max ({res, a[par[u][0]], a[u], a[v]});
}


void solve () {
    cin >> n;
    for (int i = 1; i <= n; i ++) {
        cin >> a[i];
    }
    for (int i = 1; i <= n - 1; i ++) {
        int u, v;
        cin >> u >> v;
        adj[u].push_back (v);
        adj[v].push_back (u);
    }
    int k;
    cin >> k;
    dfs (1, 0);
    for (int i = 1; i <= n; i ++) {
        dp[i] = -inf;
    }
    dp[1] = 0;
    for (int i = 1; i <= k; i ++) {
        int u, v;
        cin >> u >> v;
        if (dp[u] == -inf) {
            continue;
        }
        maxi (dp[v], dp[u] + LCA (u, v));
    }
    cout << *max_element (dp + 1, dp + n + 1);
}

signed main () {
    cin.tie (nullptr) -> sync_with_stdio (false);
    if (fopen (task".inp", "r")) {
        freopen (task".inp", "r", stdin);
        freopen (task".out", "w", stdout);
    }
    int t = 1;
    //cin >> t;
    while (t --) {
        solve ();
    }
    return 0;
}
// thfdgb
```

5.16 scc

```
#include <bits/stdc++.h>

#define task   "BriantheCrab"

#define int  long long
#define pii  pair <int, int>
#define fi   first
#define se   second
#define szf  sizeof
#define sz(s) (int)((s).size())
#define all(v) (v).begin(), (v).end()

typedef long long ll;
typedef unsigned long long ull;
typedef long double ld;

using namespace std;

template <class T> void minimize (T &t, T f) {if (t > f) t = f;}
template <class T> void maximize (T &t, T f) {if (t < f) t = f;}

const int maxN = 2e5 + 5;
const int inf = 1e18 + 7;
const int mod = 1e9 + 7;

int a[maxN];
int V, E;
vector <int> adj[maxN];
int disc[maxN], low[maxN];
bool onStack[maxN];
stack <int> st;
int timer;
int scc_count;

void find_scc (int u) {
    disc[u] = low[u] = ++ timer;
    st.push (u);
    onStack[u] = true;

    for (int v : adj[u]) {
        if (disc[v] == -1) { // Not visited
            find_scc (v);
            low[u] = min (low[u], low[v]);
        } else if (onStack[v]) { // Back edge or cross edge to
            a node in current recursion stack
            low[u] = min (low[u], disc[v]);
        }
    }

    if (low[u] == disc[u]) {
        scc_count ++;
        cout << "SCC " << scc_count << ": ";
        int node;
        while (true) {
            node = st.top ();
            st.pop ();
            onStack[node] = false;
            cout << node << " ";
            if (node == u) {
                break;
            }
        }
        cout << '\n';
    }
}

void solve () {
    cin >> V >> E;
    for (int i = 0; i < E; ++ i) {
        int u, v;
        cin >> u >> v;
        adj[u].push_back (v);
    }

    for (int i = 1; i <= V; ++ i) {
        disc[i] = -1;
        low[i] = -1;
        onStack[i] = false;
    }
    timer = 0;
    scc_count = 0;

    for (int i = 1; i <= V; ++ i) {
        if (disc[i] == -1) {
            find_scc (i);
        }
    }

    cout << "Total SCCs: " << scc_count << '\n';

    return;
}

signed main () {
    cin.tie (nullptr) -> sync_with_stdio (false);
    if (fopen (task".inp", "r")) {
        freopen (task".inp", "r", stdin);
        freopen (task".out", "w", stdout);
    }
    int t = 1;
    //cin >> t;
    while (t --) {
        solve ();
    }
    return 0;
}
// thfv
```

5.17 spfa

```
#include <bits/stdc++.h>

#define task   "BriantheCrab"

#define int  long long
#define pii  pair <int, int>
#define fi   first
#define se   second
#define szf  sizeof
#define sz(s) (int)((s).size())
#define all(v) (v).begin(), (v).end()

typedef long long ll;
typedef unsigned long long ull;
typedef long double ld;

using namespace std;

template <class T> void minimize (T &t, T f) {if (t > f) t = f;}
template <class T> void maximize (T &t, T f) {if (t < f) t = f;}

const int maxN = 2e5 + 5;
const int inf = 1e18 + 7;
const int mod = 1e9 + 7;

int a[maxN];

int V, E, s;
vector <pii> adj[maxN];
int dist[maxN];
bool inQueue[maxN];
int cnt[maxN];

void solve () {
    cin >> V >> E >> s;
    for (int i = 0; i < E; ++ i) {
        int u, v, w;
        cin >> u >> v >> w;
        adj[u].push_back ({v, w});
    }

    for (int i = 1; i <= V; ++ i) {
        dist[i] = inf;
        inQueue[i] = false;
        cnt[i] = 0;
    }

    queue <int> q;
    q.push (s);
    dist[s] = 0;
    inQueue[s] = true;
    cnt[s] = 1;
    while (!q.empty ()) {
        int u = q.front ();
        q.pop ();
        inQueue[u] = false;

        for (auto &edge : adj[u]) {
            int v = edge.fi;
            int w = edge.se;

            if (dist[u] + w < dist[v]) {
                dist[v] = dist[u] + w;
                if (!inQueue[v]) {
                    q.push (v);
                    inQueue[v] = true;
                    cnt[v] ++;

                    if (cnt[v] >= V) {
                        cout << "Graph contains a negative
                              cycle." << '\n';
                        return;
                    }
                }
            }
        }
    }

    for (int i = 1; i <= V; ++ i) {
        if (dist[i] == inf) {
            cout << "INF" << '\n';
        } else {
            cout << dist[i] << '\n';
        }
    }

    return;
}

signed main () {
    cin.tie (nullptr) -> sync_with_stdio (false);
    if (fopen (task".inp", "r")) {
        freopen (task".inp", "r", stdin);
        freopen (task".out", "w", stdout);
    }
    int t = 1;
    //cin >> t;
    while (t --) {
        solve ();
    }
    return 0;
}
// thfv
```

5.18 virtual tree

```
function build_virtual_tree(U):
    // B1: thm   tt    c  LCA  vo  U
    sort U theo in[u]
    for i = 0    |U|-2:
        U.push( lca(U[i], U[i+1]) )

    // B2:  loi   b   trng   lp
    sort U theo in[u]
    U.erase_duplicates()

    // B3:  dng   cy   bng  stack
    create empty stack st
    clear adjacency list mini_adj

    st.push(U[0])
    for i = 1    |U|-1:
        while !is_ancestor(st.top(), U[i]):
            st.pop()
        mini_adj[st.top()].push_back(U[i])
        st.push(U[i])

    return U[0]  // root  ca   cy   o
```

6 Math

6.1 cKn

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

#define int long long

const int mod = 1e9 + 7;
const int N = 1e6 + 5;

int gt[N];

void prepare() {
    gt[0] = 1;
    for (int i = 1 ; i <= 1e6 ; i ++) {
        gt[i] = gt[i - 1] * i % mod;
    }
}

int mul (int a, int b) {
    if (b == 0) {
        return 0;
    }
    int t = mul (a, b / 2);
    t = (t + t) % mod;
    if (b % 2 == 1) {
        t = (t + a) % mod;
    }
    return t;
}

int power (int a, int b) {
    if (b == 0) {
        return 1;
    }
    int t = power (a, b / 2);
    t = (t * t) % mod;
    if (b % 2 == 1) {
        t = (t * a) % mod;
    }
    return t;
}

int inversion (int a) {
    return power (a, mod - 2);
}

int C (int n, int k) {
    if (k > n) return 0;
    return ((gt[n] * inversion(gt[k]) % mod) * inversion(gt[n -
         k])) % mod;
}

signed main() {
    ios_base::sync_with_stdio (0);
    cin.tie (0);
    prepare();
    int n, q, k;
    cin >> q;
    while (q --) {
        cin >> n >> k;
        cout << C (n, k) << '\n';
    }
}
```

6.2 dpCkn

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

const int MAXN = 1005; // thay  i  theo n  ti   a
const int MOD = 1e9 + 7; // hoc 998244353

int C[MAXN][MAXN];

void buildCKN() {
    for (int n = 0; n < MAXN; n++) {
        C[n][0] = 1; // C(n,0) = 1
        for (int k = 1; k <= n; k++) {
            C[n][k] = (C[n-1][k-1] + C[n-1][k]) % MOD; // C(n,k)
                 = C(n-1,k-1) + C(n-1,k)
        }
    }
}

int main() {
    buildCKN();
    //  v   d : C(5,2)
    cout << C[5][2] << "\n"; // Output: 10
}
```

6.3 PhiHamEuler

```cpp
#include <bits/stdc++.h>

#define task    "EulerPhi" //  i   tn  task cho ph  hp

#define int  long long // int mc   nh   l  long long
#define pii  pair <int, int>
#define fi   first
#define se   second
#define szf  sizeof
#define sz(s) (int)((s).size())
#define all(v) (v).begin(), (v).end()

typedef long long ll;
typedef unsigned long long ull;
typedef long double ld;

using namespace std;

template <class T> void minimize (T &t, T f) {if (t > f) t = f;}
template <class T> void maximize (T &t, T f) {if (t < f) t = f;}

const int maxN = 2e5 + 5; // C  th  khng  cn   nu  bi  ton
      ch  tnh  phi  hm
const int inf = 1e18 + 7;
const int mod = 1e9 + 7;

//  Hm  tnh  Phi Euler
ll euler_phi (ll n) {
    ll res = n;
    for (ll i = 2; i * i <= n; ++ i) {
        if (n % i == 0) {
            res = res / i * (i - 1); // p  dng  cng   thc  (1 -
                1/p_i)
            while (n % i == 0) {
                n /= i; // Chia n cho tt  c   cc  ly  tha
                       ca  i
            }
        }
    }
    //  Nu  sau  vng   lp  , n  vn   cn  > 1, ngha  l  n  l   mt
           s   nguyn   t   ln
    if (n > 1) {
        res = res / n * (n - 1);
    }
    return res;
}

void solve () {
    ll n;
    cin >> n;
    cout << euler_phi (n) << "\n";
    return;
}

signed main () {
    cin.tie (nullptr) -> sync_with_stdio (false);
    if (fopen (task".inp", "r")) {
        freopen (task".inp", "r", stdin);
        freopen (task".out", "w", stdout);
    }
    int t = 1;
    //cin >> t; // Uncomment if multiple test cases
    while (t --) {
        solve ();
    }
    return 0;
}
// thfv
```

6.4 SieveAndSieveSmallDiv

```cpp
#include <bits/stdc++.h>

#define task    "BriantheCrab"

#define int  long long
#define pii  pair <int, int>
#define fi   first
#define se   second
#define szf  sizeof
#define sz(s) (int)((s).size())
#define all(v) (v).begin(), (v).end()

typedef long long ll;
typedef unsigned long long ull;
typedef long double ld;

using namespace std;

template <class T> void minimize (T &t, T f) {if (t > f) t = f;}
template <class T> void maximize (T &t, T f) {if (t < f) t = f;}

const int maxN = 2e5 + 5;
const int inf = 1e18 + 7;
const int mod = 1e9 + 7;

int a[maxN];

namespace Sieve {
    vector <bool> check;
    vector <int> spf;

    void calPrime (int lim) {
        check.resize (lim + 1, true);
        check[0] = check[1] = 0;
        for (int i = 2; i * i <= lim; i ++) {
            if (check[i]) {
                for (int j = i * i; j <= lim; j += i) {
                    check[j] = 0;
                }
            }
        }
        return;
    }

    void calDiv (int lim) {
        spf.resize (lim + 1);
        for (int i = 1; i <= lim; i ++) {
            spf[i] = i;
        }
        for (int i = 1; i * i <= lim; i ++) {
            if (spf[i] == i) {
                for (int j = i * i; j <= lim; j += i) {
                    if (spf[j] == j) {
                        spf[j] = i;
                    }
                }
            }
        }
        return;
    }

    void calFactor (int x, vector <int> &f) {
        while (x > 1) {
            f.push_back (spf[x]);
            x /= spf[x];
        }
        return;
    }
}

void solve () {
    int n;
    cin >> n;
    Sieve :: calPrime (n);
    for (int i = 1; i <= n; i ++) {
        if (Sieve :: check[i]) {
            cout << i << ' ';
        }
    }
    return;
}

signed main () {
    cin.tie (nullptr) -> sync_with_stdio (false);
    if (fopen (task".inp", "r")) {
        freopen (task".inp", "r", stdin);
        freopen (task".out", "w", stdout);
    }
    int t = 1;
    //cin >> t;
    while (t --) {
        solve ();
    }
    return 0;
}
// thfv
```

6.5 XOR basis

```cpp
void insert(int mask) {
    for (int i = 1; i <= now; i++) {
        mask = min(mask, mask ^ basis[i]);
    }
    if (mask != 0) basis[++now] = mask;
}

// XOR basis: moi mask trong XOR basis la duy nhat, ko the bieu
      dien bang XOR cua nhung so con lai
// used for:
/*
find max XOR
int get_max_xor(int x = 0) {
    for (int b : basis)
        x = max(x, x ^ b);
    return x;
}

find min XOR
bool can_make(int x) {
    for (int b : basis)
        x = min(x, x ^ b);
    return x == 0;
}

count how many XOR value can make
have x numbers in basis -> 2^k value can use
*/
```

7 String

7.1 hash

```
typedef long long ll;

const int base = 31;
const ll MOD = 1000000003;
const ll maxn = 1000111;

using namespace std;

ll POW[maxn], hashT[maxn];


ll getHashT(int i,int j) {
    return (hashT[j] - hashT[i - 1] * POW[j - i + 1] + MOD *
          MOD) % MOD;
}


int main() {
    // Input
    string T, P;
    cin >> T >> P;

    // Initialize
    int lenT = T.size(), lenP = P.size();
    T = " " + T;
    P = " " + P;
    POW[0] = 1;

    // Precalculate base^i
    for (int i = 1; i <= lenT; i++)
        POW[i] = (POW[i - 1] * base) % MOD;

    // Calculate hash value of T[1..i]
    for (int i = 1; i <= lenT; i++)
        hashT[i] = (hashT[i - 1] * base + T[i] - 'a' + 1) % MOD;

    // Calculate hash value of P
    ll hashP=0;
    for (int i = 1; i <= lenP; i++)
        hashP = (hashP * base + P[i] - 'a' + 1) % MOD;

    // Finding substrings of T equal to string P
    for (int i = 1; i <= lenT - lenP + 1; i++)
        if (hashP == getHashT(i, i + lenP - 1))
            printf("%d ", i);
}
```

7.2 KMP

```
vector<int> prefix_function(string s) {
    int n = (int)s.length();            // do dai chuoi s
    vector<int> pi(n);                  // mang pi[i]: do dai
          tien to dai nhat cung la hau to cua s[0..i]
    for (int i = 1; i < n; i++) {       // duyet tung ky tu tu vi
          tri 1 -> n-1
        int j = pi[i - 1];              // j: do dai tien to
              trung khop truoc do
        while (j > 0 && s[i] != s[j])  // neu ky tu tiep theo
              khong khop
            j = pi[j - 1];              // rut ngan do dai tien
                  to ve gia tri truoc do (theo border truoc)
        if (s[i] == s[j])               // neu ky tu hien tai
              trung ky tu tiep theo cua tien to
            j++;                        // mo rong do dai tien to
                  trung khop them 1
        pi[i] = j;                      // luu lai do dai tien to
              hau to khop dai nhat tai vi tri i
    }
    return pi;                          // tra ve mang pi
          (prefix-function)
}

int main() {
    string pattern = "aba";
    string text = "abacaba";
    string s = pattern + "#" + text;
    vector<int> pi = prefix_function(s);

    int m = pattern.size();
    for (int i = 0; i < (int)s.size(); i++) {
        if (pi[i] == m) {
            cout << "Pattern found at position " << i - 2*m <<
                  "\n";
            // i - 2*m = v  tr  bt   u   ca  pattern trong
                  text
        }
    }
}
/*

Ung dung thuc te cua ham prefix_function

Ung dung: Tim chuoi con
Muc tieu: Tim vi tri xuat hien cua pattern trong text
Mo ta: Su dung trong thuat toan KMP search

Ung dung: Tim do lap cua chuoi
Muc tieu: Tim xem chuoi co cau truc lap nao khong
Mo ta: Dung pi de phat hien pattern lap lai, vi du "abcabcabc"

Ung dung: Chuoi con trung dau cuoi
Muc tieu: Tim tien to cung la hau to dai nhat
Mo ta: Dung nhieu trong bai toan xu ly chuoi vong, palindrome

Ung dung: Tien xu ly trong bai toan chuoi
Muc tieu: Xac dinh bien (border) hoac phan lap
Mo ta: Dung trong suffix automation, hashing, Z-function, vv
*/
```