```cpp
// xor tree
// Problem: E. Swap and Maximum Block
// Contest: Codeforces - Educational Codeforces Round 133 (Rated for Div. 2)

#include "bits/stdc++.h"

struct node{
    int pre, suf, sum, ans;
};

node merge(const node &a, const node &b){

    node ret;
    ret.sum = a.sum + b.sum;
    ret.pre = max(a.pre, a.sum + b.pre);
    ret.suf = max(b.suf, a.suf + b.sum);
    ret.ans = max({a.ans, b.ans, a.suf+b.pre});

    return ret;
}

int arr[NMAX];
vector<node> tree[NMAX << 2];

void build(int nd, int l, int r, int b){

    if(l==r){
        int tt = max(arr[l], 0ll);
        tree[nd].push_back((node){tt, tt, arr[l], tt});
        return;
    }

    int mid = (l+r)/2;
    build(nd*2, l, mid, b-1);
    build(nd*2+1, mid+1, r, b-1);

    for(int x = 0; x < (1<<(b-1)); x++){
        node a = tree[nd*2][x];
        node b = tree[nd*2+1][x];
        tree[nd].push_back(merge(a, b));
    }

    for(int x = 0; x < (1<<(b-1)); x++){
        node a = tree[nd*2][x];
        node b = tree[nd*2+1][x];
        tree[nd].push_back(merge(b, a));
    }
}

int32_t main(){

    int n;
    cin >> n;

    for(int i = 0; i < (1<<n); i++){
        cin >> arr[i];
    }

    build(1, 0, (1<<n)-1, n);

    int x = 0;
    int q; cin >> q; while(q--){

        int cx;
        cin >> cx;
        x ^= 1<<cx;

        cout << tree[1][x].ans << endl;
    }
}
```

```cpp
// xor segment tree v2 (queries with bs)
// Problem: F. Minimal String Xoration

// By AmmarDab3an

const int  MAX = 2e5 + 10;
const int NMAX = (1<<18) + 10;
const int MMAX = 2e5 + 10;
const int LOG_MAX = ceil(log2(double(NMAX)));
const int BLOCK = ceil(sqrt(double(NMAX)));

int n;
string str;

array<int, 2> p, m;
vector<array<int, 2>> pow_m;

bool is_prime(int x){
    for(ll i = 2; i*i <= x; i++) if(x%i==0){
        return false;
    }
    return true;
}

void init_hash(){

    p = {rand(1e4, 5e4), rand(6e4, 9e4)};
    m = {53, 79};

    while(!is_prime(p[0])) p[0]++;
    while(!is_prime(p[1])) p[1]++;

    pow_m.resize(NMAX);

    pow_m[0][0] = pow_m[0][1] = 1;

    for(int i = 1; i < NMAX; i++)
    for(int j = 0; j < 2; j++){
        pow_m[i][j] = (pow_m[i-1][j] * m[j])%p[j];
    }
}

vector<array<int, 2>> tree[NMAX << 2];

void build(int nd, int l, int r, int p){

    if(l==r){
        array<int, 2> cur;
        cur[0] = cur[1] = str[l]-'a'+1;
        tree[nd].push_back(cur);
        return;
    }

    int mid = (l+r)/2;
    build(nd*2, l, mid, p-1);
    build(nd*2+1, mid+1, r, p-1);

    for(int i = 0; i < (1<<(p-1)); i++){

        array<int, 2> a = tree[nd*2][i];
        array<int, 2> b = tree[nd*2+1][i];

        array<int, 2> cur;
        for(int j = 0; j < 2; j++){
            cur[j] = (a[j] * pow_m[1<<(p-1)][j] + b[j]) % ::p[j];
        }
        tree[nd].push_back(cur);
    }

    for(int i = 0; i < (1<<(p-1)); i++){

        array<int, 2> a = tree[nd*2][i];
        array<int, 2> b = tree[nd*2+1][i];

        array<int, 2> cur;
        for(int j = 0; j < 2; j++){
            cur[j] = (b[j] * pow_m[1<<(p-1)][j] + a[j]) % ::p[j];
        }
        tree[nd].push_back(cur);
    }
}

array<int, 2> query(int nd, int l, int r, int p, int x, int q_l, int q_r){


    if(r < q_l || q_r < l){
        return (array<int, 2>){0, 0};
    }

    if(q_l <= l && r <= q_r){

        array<int, 2> cur = tree[nd][x];

        for(int j = 0; j < 2; j++){
            cur[j] = (cur[j] * pow_m[q_r-r][j]) % ::p[j];
        }

        return cur;
    }

    int mid = (l+r)/2;
    array<int, 2> st_path, nd_path;

    if((x>>(p-1))&1){
        x ^= (1<<(p-1));
        st_path = query(nd*2+1, l, mid, p-1, x, q_l, q_r);
        nd_path = query(nd*2, mid+1, r, p-1, x, q_l, q_r);
    }
    else{
        st_path = query(nd*2, l, mid, p-1, x, q_l, q_r);
        nd_path = query(nd*2+1, mid+1, r, p-1, x, q_l, q_r);
    }

    array<int, 2> cur;
    for(int j = 0; j < 2; j++){
        cur[j] = (st_path[j] + nd_path[j]) % ::p[j];
    }

    return cur;
}

int query_bs(int nd_a, int nd_b, int l, int r, int a, int b, int p){

    if(l==r){
        if(tree[nd_a][0]!=tree[nd_b][0]){
            return l;
        }
        else{
            return l+1;
        }
    }

    int mid = (l+r)/2;
    int nd_lf_a = nd_a*2;
    int nd_lf_b = nd_b*2;

    if((a>>(p-1))&1) a ^= 1<<(p-1), nd_lf_a ^= 1;
    if((b>>(p-1))&1) b ^= 1<<(p-1), nd_lf_b ^= 1;

    if(tree[nd_lf_a][a] != tree[nd_lf_b][b]){
        return query_bs(nd_lf_a, nd_lf_b, l, mid, a, b, p-1);
    }
    else{
        return query_bs(nd_lf_a^1, nd_lf_b^1, mid+1, r, a, b, p-1);
    }
}

bool comp(int a, int b){

    int l = 0;
    int r = (1<<n)-1;

    // int ans = -1;

    // while(l <= r){
//
        // int mid = (l+r)/2;
//
        // array<int, 2> hash_a = query(1, 0, (1<<n)-1, n, a, 0, mid);
        // array<int, 2> hash_b = query(1, 0, (1<<n)-1, n, b, 0, mid);
//
        // if(hash_a==hash_b){
            // ans = mid;
            // l = mid+1;
        // }
        // else{
            // r = mid-1;
        // }
//
    // }

    int ans = query_bs(1, 1, 0, (1<<n)-1, a, b, n) -1;

    if(ans+1==(1<<n)){
        return false;
    }

    char ch_a = str[(ans+1)^a];
    char ch_b = str[(ans+1)^b];

    return ch_a < ch_b;
}

int32_t main(){

    cin >> n;
    cin >> str;

    init_hash();

    build(1, 0, (1<<n)-1, n);

    int ans = 0;
    for(int i = 1; i < (1<<n); i++){
        if(comp(i, ans)){
            ans = i;
        }
    }

    string ans_str(1<<n, '.');
    for(int i = 0; i < (1<<n); i++){
        ans_str[i] = str[i^ans];
    }

    cout << ans_str << endl;
}

```

```cpp
// Problem: H. Codeforces Scoreboard
// Contest: Codeforces - TypeDB Forces 2023 (Div. 1 + Div. 2, Rated, Prizes!)

#include "bits/stdc++.h"

using namespace std;

struct node{

    int size, dp, lazy;
    node *lf, *ri;

    ~node(){
        delete lf;
        delete ri;
    }

    bool is_leaf(){
        assert((lf==0) == (size==1));
        return size==1;
    }

    void push(){
        if(!lazy) return;
        dp += lazy;
        if(lf) lf->lazy += lazy;
        if(ri) ri->lazy += lazy;
        lazy = 0;
    }

    void merge(){
        assert(!is_leaf());
        assert(lazy==0);
        size = lf->size + ri->size;
        dp = ri->dp;
    }

    bool is_balanced()  {
        assert(!is_leaf());
        int l = lf->size;
        int r = ri->size;
        return (l <= 2*r+5) && (r <= 2*l+5);
    }

    void balance(){

        assert(!is_leaf());
        if(is_balanced()) return;

        vi elements;
        dfs(elements);
        build(elements);
    }

    void build(vi elements){

        lazy = 0;
        size = elements.size();
        dp = elements.back();

        delete lf;
        delete ri;

        assert(!elements.empty());

        if(elements.size()==1){
            lf = ri = nullptr;
            return;
        }

        lf = new node();
        ri = new node();
        auto mid = elements.begin() + elements.size()/2;
        lf->build(vi(elements.begin(), mid));
        ri->build(vi(mid, elements.end()));
    }

    bool wants(int i, int k, int b){
        push();
        int tt = b-(i+size-1)*k;
        return tt > dp;
    }

    void update(int i, int k, int b){

        push();

        if(is_leaf()){
            build({b-i*k, dp-k});
            return;
        }

        if(lf->wants(i, k, b)){
            lf->update(i, k, b);
            ri->lazy -= k;
            ri->push();
        }
        else{
            ri->update(i+lf->size, k, b);
        }

        merge();
        balance();
    }

    void dfs(vi &elements){

        push();

        if(is_leaf()){
            elements.push_back(dp);
        }
        else{
            lf->dfs(elements);
            ri->dfs(elements);
        }
    }
};

int32_t main(){

    fastIO;

    int t; cin >> t; while(t--){

        int n;
        cin >> n;

        int sm_a = 0;
        vector<pii> vec(n);
```

```cpp

        for(auto &[k, b] : vec){
            cin >> k >> b;
            int ai;
            cin >> ai;
            sm_a += ai;
            b -= ai;
        }

        sort(vec.rbegin(), vec.rend());

        node rt = (node){1, -INFLL, 0, nullptr, nullptr};

        for(auto [k, b] : vec){
            rt.update(1, k, b);
        }

        vi elements;
        rt.dfs(elements);

        int ans = sm_a;
        for(auto e : elements){
            if(e >= 0){
                ans += e;
            }
        }

        cout << ans << endl;
    }
}

```

```cpp
// sqrt cmp fst
// Problem: D. ! Divisible
// Contest: Codeforces - ACPC 2022

const int AMAX = 1e6 + 1e4;

const int NMAX = 2e5 + 10;
const int BLOCK = 650;
const int LOG_MAX = ceil(log2(double(NMAX)));

const int PMAX = 1e5 + 10;
const int PBLOCK = 300;
const int PMX = PMAX/PBLOCK + 1;

inline int64_t hilbertOrder(int x, int y, int pow, int rotate) {

    if (pow == 0) {
        return 0;
    }
    int hpow = 1 << (pow-1);
    int seg = (x < hpow) ? (
        (y < hpow) ? 0 : 3
    ) : (
        (y < hpow) ? 1 : 2
    );
    seg = (seg + rotate) & 3;
    const int rotateDelta[4] = {3, 0, 0, 1};
    int nx = x & (x ^ hpow), ny = y & (y ^ hpow);
    int nrot = (rotate + rotateDelta[seg]) & 3;
    int64_t subSquareSize = int64_t(1) << (2*pow - 2);
    int64_t ans = seg * subSquareSize;
    int64_t add = hilbertOrder(nx, ny, pow-1, nrot);
    ans += (seg == 1 || seg == 2) ? add : (subSquareSize - add - 1);
    return ans;
}

struct query{

    int order;
    int l, r, fix, idx;

    bool operator < (const query &other){
        return order < other.order;
    }
};

vi adj[NMAX];
int val[NMAX];
vi primes;
bool not_prime[AMAX];
int prime_pos[AMAX];
int tin[NMAX], tout[NMAX], tim;
int arr[2*NMAX];

bool vis[NMAX];
int pcnt[PMAX];
int blk_sz[PMX];
int blk_osz[PMX];

int depth[NMAX];
int par[NMAX][LOG_MAX];

query queries[NMAX];
int ans[NMAX];

```

```cpp
void init_sieve(){

    for(ll i = 2; i < AMAX; i++) if(!not_prime[i]){
        prime_pos[i] = primes.size();
        primes.push_back(i);
        for(ll j = i*i; j < AMAX; j+=i){
            not_prime[j] = true;
        }
    }

    for(int i = 0; i < primes.size(); i++){
        blk_osz[i/PBLOCK]++;
    }
}

void dfs(int u, int p){

    tin[u] = tim;
    arr[tim] = u;
    tim++;

    for(auto v : adj[u]) if(v != p){

        depth[v] = depth[u]+1;

        par[v][0] = u;
        for(int i = 1; i < LOG_MAX; i++){
            par[v][i] = par[par[v][i-1]][i-1];
        }

        dfs(v, u);
    }

    tout[u] = tim;
    arr[tim] = u;
    tim++;
}

int lca(int u, int v){

    if(depth[u] < depth[v]){
        swap(u, v);
    }

    int dif = depth[u] - depth[v];

    for(int i = 0; i < LOG_MAX; i++) if((dif>>i)&1){
        u = par[u][i];
    }

    if(u==v) return u;

    for(int i = LOG_MAX-1; i >= 0; i--) if(par[u][i] != par[v][i]){
        u = par[u][i];
        v = par[v][i];
    }

    return par[u][0];
}

void update(int f, int d){

    pcnt[f] += d;

    if(pcnt[f]==1){
        blk_sz[f/PBLOCK] += 1;
    }
    else if(pcnt[f]==0){
        blk_sz[f/PBLOCK] -= 1;
    }
}

void add(int i){

    int u = arr[i];
    if(val[u]==-1) return;

    vis[u] ^= 1;
    update(val[u], vis[u] ? +1 : -1);
}

void rem(int i){
    add(i);
}

int get_ans(){

    for(int i = 0; i < PMX; i++) if(blk_sz[i] != blk_osz[i]){

        assert(blk_sz[i] < blk_osz[i]);

        int lo = i*PBLOCK;
        int hi = (i+1)*PBLOCK;

        for(int j = lo; j < hi; j++){

            assert(j < PMAX);

            if(pcnt[j]==0){
                return primes[j];
            }
        }
    }

    assert(false);
}

int32_t main(){

    fastIO;

    init_sieve();

    int t; cin >> t; while(t--){

        int n;
        cin >> n;

        {
            tim = 0;
            for(int i = 0; i < n; i++){
                adj[i].clear();
            }
        }

        for(int i = 0; i < n; i++){
            int x;
            cin >> x;
            val[i] = !not_prime[x] ? prime_pos[x] : -1;
        }

        for(int i = 1; i < n; i++){

            int u, v;
            cin >> u >> v;
            u--, v--;

            adj[u].push_back(v);
            adj[v].push_back(u);
        }

        dfs(0, -1);

        int q;
        cin >> q;

        for(int i = 0; i < q; i++){

            int u, v;
            cin >> u >> v;
            u--, v--;

            if(depth[u] > depth[v]){
                swap(u, v);
            }

            int p = lca(u, v);

            if(u==p){
                queries[i] = {0, tin[u], tin[v], -1, i};
            }
            else{
                queries[i] = {0, tout[u], tin[v], tin[p], i};
            }
        }

        for(int i = 0; i < q; i++){
            int l = queries[i].l;
            int r = queries[i].r;
            queries[i].order = hilbertOrder(l, r, 20, 0);
        }

        sort(queries, queries+q);

        int l = 1, r = 0;

        for(int i = 0; i < q; i++){

            auto [co, cl, cr, fix, idx] = queries[i];

            while(cl < l) add(--l);
            while(r < cr) add(++r);
            while(l < cl) rem(l++);
            while(cr < r) rem(r--);

            if(fix != -1){
                add(fix);
            }

            ans[idx] = get_ans();

            if(fix != -1){
                rem(fix);
            }
        }
```

```cpp

        while(r >= l){
            rem(r--);
        }

        for(int i = 0; i < q; i++){
            cout << ans[i] << endl;
        }
    }
}

```

```cpp
// crt
// Problem: C. Counting Trees
// Contest: Codeforces - NUS CS3233 Final Team Contest 2023 Mirror

#include "bits/stdc++.h"

using namespace std;

const int MOD = 3000301; // 1e9 + 7;
const int NMAX = MOD;

int fac[NMAX], ifac[NMAX];
int mod = 10000003233;
vi mod_factors = {3, 11, 101, 3000301};
int mem[3][222][222];

void init(){

    fac[0] = 1;
    for(int i = 1; i < NMAX; i++){
        fac[i] = mul(fac[i-1], i);
    }

    ifac[NMAX-1] = inv(fac[NMAX-1]);
    for(int i = NMAX-2; i >= 0; i--){
        ifac[i] = mul(ifac[i+1], i+1);
    }

    memset(mem, -1, sizeof mem);
}

int cho(int n, int c, int i){

    if(n == c) return 1;
    if(n == 0) return 0;

    int &ret = mem[i][n][c];
    if(ret+1) return ret;

    int st_path = cho(n-1, c-1, i);
    int nd_path = cho(n-1, c, i);
    int ans = (st_path + nd_path) % mod_factors[i];

    return ret = ans;
}

int choose(int n, int c, int i){

    if(n < c) return 0;

    if(i < 3){
        return cho(n, c, i);
    }
    else{
        return mul(fac[n], mul(ifac[c], ifac[n-c]));
    }
}

struct Congruence {
    long long a, m;
};

long long chinese_remainder_theorem(vector<Congruence> const& congruences) {

    long long M = 1;
    for (auto const& congruence : congruences) {
        M *= congruence.m;
    }

    long long solution = 0;
    for (auto const& congruence : congruences) {
        long long a_i = congruence.a;
        long long M_i = M / congruence.m;
        long long N_i = pow_exp(M_i, congruence.m-2, congruence.m);
        solution = (solution + a_i * M_i % M * N_i) % M;
    }

    return solution;
}

int32_t main(){

    init();

    int t; cin >> t; while(t--){

        int n, k, c;
        cin >> n >> k >> c;

        if(c > n){
            cout << 0 << endl;
            continue;
        }

        vector<Congruence> vec(4);

        for(int i = 0; i < 4; i++){

            int cm = mod_factors[i];

            int cho = 1;
            int a = n-1;
            int b = c-1;

            while(a > 0){
                int ccho = choose(a%cm, b%cm, i);
                cho = (cho * ccho)%cm;
                a /= cm, b /= cm;
            }

            vec[i] = {cho, cm};
        }

        int ans = chinese_remainder_theorem(vec);
        ans = (ans * 2) % mod;

        cout << ans << endl;
    }
}
```

```
// x_factor int
// I. Investors
// https://qoj.ac/contest/1103/problem/5507

// By AmmarDab3an

const int  MAX = 2e5 + 10;
const int NMAX = 2e5 + 10;
const int MMAX = 2e5 + 10;
const int LOG_MAX = ceil(log2(double(NMAX)));
const int BLOCK = ceil(sqrt(double(NMAX)));

struct FenwickTree {
    vector<int> bit;  // binary indexed tree
    int n;

    FenwickTree(int n) {
        this->n = n;
        bit.assign(n, 0);
    }

    FenwickTree(vector<int> a) : FenwickTree(a.size()) {
        for (size_t i = 0; i < a.size(); i++)
            add(i, a[i]);
    }

    int sum(int r) {
        int ret = 0;
        for (; r >= 0; r = (r & (r + 1)) - 1)
            ret += bit[r];
        return ret;
    }

    int sum(int l, int r) {
        return sum(r) - sum(l - 1);
    }

    void add(int idx, int delta) {
        for (; idx < n; idx = idx | (idx + 1))
            bit[idx] += delta;
    }
};

int pre[6060][6060];

int32_t main(){

    fastIO;

    int t; cin >> t; while(t--){

        int n, k;
        cin >> n >> k;

        k++;

        vi vec(n);
        for(auto &i : vec) cin >> i;

        if(n==0){
            srand(0);
            n = 6000;
            vec = vi(n);
            iota(vec.begin(), vec.end(), 0);
            random_shuffle(vec.begin(), vec.end());
            k = 5 + 1;
        }

        vi tmp = vec;
        sort(tmp.begin(), tmp.end());
        tmp.erase(unique(tmp.begin(), tmp.end()), tmp.end());

        for(auto &e : vec){
            e = lower_bound(tmp.begin(), tmp.end(), e) - tmp.begin();
        }

        for(int i = 0; i < n; i++){
            FenwickTree bit(n);
            int cnt = 0;
            for(int j = i; j >= 0; j--){
                cnt += bit.sum(0, vec[j]-1);
                bit.add(vec[j], 1);
                pre[i][j] = cnt;
            }
        }

        auto calc = [&](int x_factor)->pii{

            vpii dp(n);

            for(int i = 0; i < n; i++){

                pii cans = {INFLL, 0};

                for(int j = i; j >= 0; j--){

                    pii nxt = j ? dp[j-1] : (pii){0, 0};
                    nxt.first += pre[i][j] + x_factor;
                    nxt.second++;

                    cans = min(cans, nxt);
                }

                dp[i] = cans;
            }

            return dp[n-1];
        };

        int l = 0;
        int r = INF;

        int bs_ans = -1;

        while(l <= r){

            int mid = (l+r)/2;

            pii cans = calc(mid);

            if(cans.second <= k){
                bs_ans = mid;
                r = mid-1;
            }
            else{
                l = mid+1;
            }
        }

        pii ans = calc(bs_ans);
        cout << ans.first - bs_ans*k << endl;
    }
}

```

```cpp
// x_factor double
// Problem: E. Gosha is hunting

#include "bits/stdc++.h"

pair<double, int> merge(const pair<double, int> &a, const pair<double, int> &b){
    if(abs(a.first-b.first) < 1e-6){
        return a.second < b.second ? a : b;
    }
    else{
        return a.first > b.first ? a : b;
    }
}

int32_t main(){

    int n, a, b;
    cin >> n >> a >> b;

    vector<vector<double>> vec(2, vector<double>(n));
    for(auto &v : vec) for(auto &i : v) cin >> i;

    auto calc = [&](double x_factor){

        vector<pair<double, int>> dp(a+1);

        for(int i = 0; i < n; i++){

            vector<pair<double, int>> ndp(a+1, {-1e18, INF});

            double p = vec[0][i];
            double q = vec[1][i];
            double pq = 1.0 - (1.0-p)*(1.0-q);

            for(int j = 0; j <= a; j++){

                // dp[i][a] = max(
                    // dp[i-1][a],
                    // dp[i-1][a-1] + vec[0][i]
                    // dp[i-1][a] + vec[1][i] - x_factor
                    // dp[i-1][a-1] + (1 - (1-vec[0][i])*(1-vec[1][i])) - x_factor
                // )

                auto &cans = ndp[j];

                auto st_path = dp[j];
                auto nd_path = dp[j];
                nd_path.first += q - x_factor;
                nd_path.second += 1;

                cans = merge(cans, merge(st_path, nd_path));

                if(j){
                    auto rd_path = dp[j-1];
                    rd_path.first += p;
                    auto th_path = dp[j-1];
                    th_path.first += pq - x_factor;
                    th_path.second += 1;
                    cans = merge(cans, merge(rd_path, th_path));
                }
            }

            dp = ndp;
        }

```

```cpp
            return dp.back();
        };

        double l = 0;
        double r = 1e9;

        double bs_ans = -1;

        int cnt = 100;
        while(cnt--){

            double mid = (l+r)/2;
            auto cans = calc(mid);

            if(cans.second <= b){
                bs_ans = mid;
                r = mid;
            }
            else{
                l = mid;
            }
        }

        double ans = calc(bs_ans).first + b*bs_ans;

        cout << fixed << setprecision(6) << ans << endl;
    }

```

```cpp
// dp_schoelace
// B. Bars
// https://qoj.ac/problem/5500

// By AmmarDab3an

#include "bits/stdc++.h"

int32_t main(){

    int t; cin >> t; while(t--){

        int n;
        cin >> n;

        vi vec(n);
        for(auto &i : vec) cin >> i;

        vpii tmp;
        tmp.push_back({0, 0});

        // dp[i] = max(dp[j] + (pi+pj)*(i-j))
        // dp[i] = max(dp[j] + pi*i + pj*j - pi*j - pj*j)
        // dp[i] = pi*i + max((dp[j]-pj*j) + pj*i - pi*j )

        // ans = sum((pi+pj) * (i-j))
        // Shoelace Formula

        auto calc = [&](const pii &a, const pii &b){
            return a.first*b.second - b.first*a.second;
        };

        for(int i = 0; i <= n; i++){

            pii cur = i < n ? (pii){vec[i], i} : (pii){0, n-1};

            while(tmp.size() >= 2){

                pii a = tmp[tmp.size()-2];
                pii b = tmp[tmp.size()-1];
                pii c = cur;

                if(calc(a, b) + calc(b, c) <= calc(a, c)){
                    tmp.pop_back();
                }
                else{
                    break;
                }
            }

            tmp.push_back(cur);
        }

        int ans = 0;
        for(int i = 1; i < tmp.size(); i++){
            ans += calc(tmp[i-1], tmp[i]);
        }

        cout << ans << endl;
    }
}
```

```cpp
// fft any mod

#include<bits/stdc++.h>
using namespace std;
#define LL long long
using namespace std;
using cd = complex < long double >;
long double PI = acos ( - 1 );
long long mod = 1e9 + 7;
namespace fft{
    struct num{
        double x,y;
        num() {x=y=0;}
        num(double x,double y):x(x),y(y){}
    };
    inline num operator+(num a,num b) {return num(a.x+b.x,a.y+b.y);}
    inline num operator-(num a,num b) {return num(a.x-b.x,a.y-b.y);}
    inline num operator*(num a,num b) {return num(a.x*b.x-a.y*b.y,a.x*b.y+a.y*b.x);}
    inline num conj(num a) {return num(a.x,-a.y);}
    int base=1;
    vector<num> roots={{0,0},{1,0}};
    vector<int> rev={0,1};
    const double PI=acosl(-1.0);
    void ensure_base(int nbase){
        if(nbase<=base) return;
        rev.resize(1<<nbase);
        for(int i=0;i<(1<<nbase);i++)
            rev[i]=(rev[i>>1]>>1)+((i&1)<<(nbase-1));
        roots.resize(1<<nbase);
        while(base<nbase){
            double angle=2*PI/(1<<(base+1));
            for(int i=1<<(base-1);i<(1<<base);i++){
                roots[i<<1]=roots[i];
                double angle_i=angle*(2*i+1-(1<<base));
                roots[(i<<1)+1]=num(cos(angle_i),sin(angle_i));
            }
            base++;
        }
    }

    void fft(vector<num> &a,int n=-1){
        if(n==-1) n=a.size();
        assert((n&(n-1))==0);
        int zeros=__builtin_ctz(n);
        ensure_base(zeros);
        int shift=base-zeros;
        for(int i=0;i<n;i++)
            if(i<(rev[i]>>shift))
                swap(a[i],a[rev[i]>>shift]);
        for(int k=1;k<n;k<<=1){
            for(int i=0;i<n;i+=2*k){
                for(int j=0;j<k;j++){
                    num z=a[i+j+k]*roots[j+k];
                    a[i+j+k]=a[i+j]-z;
                    a[i+j]=a[i+j]+z;
                }
            }
        }
    }
    vector<num> fa,fb;
    vector<int> multiply(vector<int> &a, vector<int> &b){
        int need=a.size()+b.size()-1;
        int nbase=0;
        while((1<<nbase)<need) nbase++;
        ensure_base(nbase);
        int sz=1<<nbase;
        if(sz>(int)fa.size()) fa.resize(sz);
        for(int i=0;i<sz;i++){
            int x=(i<(int)a.size()?a[i]:0);
            int y=(i<(int)b.size()?b[i]:0);
            fa[i]=num(x,y);
        }
        fft(fa,sz);
        num r(0,-0.25/sz);
        for(int i=0;i<=(sz>>1);i++){
            int j=(sz-i)&(sz-1);
            num z=(fa[j]*fa[j]-conj(fa[i]*fa[i]))*r;
            if(i!=j) fa[j]=(fa[i]*fa[i]-conj(fa[j]*fa[j]))*r;
            fa[i]=z;
        }
        fft(fa,sz);
        vector<int> res(need);
        for(int i=0;i<need;i++) res[i]=fa[i].x+0.5;
        return res;
    }

    vector<int> multiply_mod(vector<int> &a,vector<int> &b,int m,int eq=0){
        int need=a.size()+b.size()-1;
        int nbase=0;
        while((1<<nbase)<need) nbase++;
        ensure_base(nbase);
        int sz=1<<nbase;
        if(sz>(int)fa.size()) fa.resize(sz);
        for(int i=0;i<(int)a.size();i++){
            int x=(a[i]%m+m)%m;
            fa[i]=num(x&((1<<15)-1),x>>15);
        }
        fill(fa.begin()+a.size(),fa.begin()+sz,num{0,0});
        fft(fa,sz);
        if(sz>(int)fb.size()) fb.resize(sz);
        if(eq) copy(fa.begin(),fa.begin()+sz,fb.begin());
        else{
            for(int i=0;i<(int)b.size();i++){
                int x=(b[i]%m+m)%m;
                fb[i]=num(x&((1<<15)-1),x>>15);
            }
            fill(fb.begin()+b.size(),fb.begin()+sz,num{0,0});
            fft(fb,sz);
        }
        double ratio=0.25/sz;
        num r2(0,-1),r3(ratio,0),r4(0,-ratio),r5(0,1);
        for(int i=0;i<=(sz>>1);i++){
            int j=(sz-i)&(sz-1);
            num a1=(fa[i]+conj(fa[j]));
            num a2=(fa[i]-conj(fa[j]))*r2;
            num b1=(fb[i]+conj(fb[j]))*r3;
            num b2=(fb[i]-conj(fb[j]))*r4;
            if(i!=j){
                num c1=(fa[j]+conj(fa[i]));
                num c2=(fa[j]-conj(fa[i]))*r2;
                num d1=(fb[j]+conj(fb[i]))*r3;
                num d2=(fb[j]-conj(fb[i]))*r4;
                fa[i]=c1*d1+c2*d2*r5;
                fb[i]=c1*d2+c2*d1;
            }
            fa[j]=a1*b1+a2*b2*r5;
            fb[j]=a1*b2+a2*b1;
        }
        fft(fa,sz);fft(fb,sz);
        vector<int> res(need);
        for(int i=0;i<need;i++){
            LL aa=fa[i].x+0.5;
            LL bb=fb[i].x+0.5;
            LL cc=fa[i].y+0.5;
            res[i]=(aa+((bb%m)<<15)+((cc%m)<<30))%m;
        }
        return res;
    }
    vector<int> square_mod(vector<int> &a,int m){
        return multiply_mod(a,a,m,1);
    }
};
```

```cpp
// mob_baath
// Problem: I. Will you accept Basharo challenge?
// Contest: Codeforces - Al-Baath Collegiate Programming Contest 2023

#include "bits/stdc++.h"

const int NMAX = 5e4 + 10;
const int AMAX = 5e4 + 10;

int arr[NMAX];
vi adj[NMAX];
int ans[NMAX], cans;
int frq0[AMAX];
int frq1[AMAX];
vi factors[AMAX];
int sub[NMAX];
map<pii, int> edge_id;

vector<int> prime;
bool not_prime[AMAX];
int mob[AMAX];

void mobius(int n = AMAX){

    mob[1] = 1;

    for(int i = 2; i < n; i++){

        if(!not_prime[i]){
            prime.push_back(i);
            mob[i] = -1;
        }

        for (int j = 0; j < prime.size () && i * prime[j] < n; ++j) {

            not_prime[i * prime[j]] = true;

            if (i % prime[j] == 0){
                mob[i * prime[j]] = 0;
                break;
            }
            else{
                mob[i * prime[j]] = mob[i] * mob[prime[j]];
            }
        }
    }
}

void init_factors(){
    for(int i = 1; i < AMAX; i++)
    for(int j = i; j < AMAX; j+=i){
        factors[j].push_back(i);
    }
}

void dfs(int u, int p){
    sub[u] = 1;
    if(p != -1) adj[u].erase(find(adj[u].begin(), adj[u].end(), p));
    for(auto &v : adj[u]) if(v != p){
        dfs(v, u);
        sub[u] += sub[v];
        if(sub[v] > sub[adj[u][0]]){
            swap(v, adj[u][0]);
        }
    }
```

```cpp
}

void add(int u, int d){
    for(auto f : factors[arr[u]]){
        cans -= mob[f] * (frq0[f]-frq1[f]) * frq1[f];
        frq1[f] += d;
        cans += mob[f] * (frq0[f]-frq1[f]) * frq1[f];
    }
}

void add(int u, int p, int d){
    add(u, d);
    for(auto v : adj[u]) if(v != p){
        add(v, u, d);
    }
}

void calc(int u, int p, bool keep){

    for(auto v : adj[u]) if(v != p) if(v != adj[u][0]){
        calc(v, u, 0);
    }

    if(!adj[u].empty()){
        calc(adj[u][0], u, 1);
    }

    for(auto v : adj[u]) if(v != p) if(v != adj[u][0]){
        add(v, u, 1);
    }

    add(u, 1);
    if(p != -1) ans[edge_id[{u, p}]] = cans;

    if(!keep){
        add(u, p, -1);
    }
}

int32_t main(){

    fastIO;

    int n;
    cin >> n;

    mobius();
    init_factors();

    for(int i = 0; i < n; i++){
        cin >> arr[i];
        for(auto f : factors[arr[i]]){
            frq0[f]++;
        }
    }

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

    if(n==0){
        n = 5e4;
        // fill(arr, arr+n, 1);
        iota(arr, arr+n, 1);
        for(int i = 0; i < n; i++){
            for(auto f : factors[arr[i]]){
                frq0[f]++;
            }
        }
        rng = mt19937(0);
        for(int i = 1; i < n; i++){
            adj[i-1].push_back(i);
            adj[i].push_back(i-1);
            edge_id[{i-1, i}] = edge_id[{i, i-1}] = i;
        }
    }

    dfs(0, -1);
    calc(0, -1, 0);

    for(int i = 1; i < n; i++){
        cout << ans[i] << ' ';
    }
}
```