Std Code Library(Qinhuangdao)

tingyx

Nanjing University of Science and Technology

October 9, 2023

Contents

一切的开始

Codeforces/XCPC

- 需要 C++17/C++20

```cpp
#include <bits/stdc++.h>
#define endl '\n'
#define pll pair<ll, ll>
#define tll tuple<ll, ll, ll>
#define vi vector<int>
#define vl vector<ll>
#define x first
#define y second
#define rep(i, j, k) for(int i = (j); i <= (k); i++)
#define per(i, j ,k) for(int i = (j); i >= (k); i--)
#define ios ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
typedef long long ll;
const ll maxn = 2e5 + 10;
const ll mod = 998244353;
const ll inf32 = 1e9;
const ll inf64 = 1e18;


void solve(){

}

int main(){
    ios;
    //freopen("sample.txt", "r", stdin);
    //freopen("resout.txt", "w", stdout);
    int t = 1;
    //cin >> t;
    while(t--){
        solve();
    }
    return 0;
}
// ----------------------------------------------------------------------------
```

int128

- 不要使用 cin/cout，记得关同步流

```cpp
typedef __int128 i128;

i128 read()
{
   i128 x = 0; bool f = 0;
   char c = getchar();
   while (c < '0' || c > '9')
   {
       if (c == '-')
           f = 1;
       c = getchar();
   }
   while (c >= '0' && c <= '9')
   {
       x = (x << 1) + (x << 3) + (c ^ 48);
       c = getchar();
   }
   return f ? -x : x;
}

inline void write(i128 x)
{
   if (x < 0)
       putchar('-'), x = -x;
   if (x > 9)
```

```
        write(x / 10);
    putchar(x % 10 + '0');
}
```

数据结构

二维数点

- 逆序对

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const int maxn = 500010;
ll m;
ll a[maxn], b[maxn], c[maxn];
int lowbit(int x){return x & (-x);}
void add(int x, ll y){
    for (int i = x; i <= m; i += lowbit(i)) c[i] += y;
}
ll sum(int x){
    ll res = 0;
    for (int i = x; i; i -= lowbit(i)) res += c[i];
    return res;
}
int main(){
    int n;
    cin >> n;
    for (int i = 1; i <= n; ++i){
        cin >> a[i];
        b[i] = a[i];
    }
    sort(b + 1, b + n + 1);
    m = unique(b + 1, b + n + 1) - b - 1;
    ll ans = 0;
    for (int i = n; i; i--){
        int k = lower_bound(b + 1, b + m + 1, a[i]) - b;
        ans += sum(k - 1);
        add(k, 1);
    }
    cout << ans;
    return 0;
}
```

- 园丁的烦恼 (矩阵内点的个数)

```
#include <bits/stdc++.h>
#define endl '\n'
#define pll pair<ll, ll>
#define tll tuple<ll, ll, ll>
#define pii pair<int, int>
#define vi vector<int>
#define vl vector<ll>
#define rep(i, j, k) for(int i = (j); i <= (k); i++)
#define per(i, j ,k) for(int i = (j); i >= (k); i--)
#define ios ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
typedef long long ll;
const ll maxn = 1e7 + 10;
const ll mod = 998244353;
const ll inf = 0x3f3f3f3f;

struct BIT{
    int tr[maxn];
    int lowbit(int x){return x & -x;}
    void add(int p, int x){
        for (; p < maxn; p += lowbit(p)) tr[p] += x;
    }
    ll query(int p){
        ll sum = 0;
```

```cpp
        for (; p > 0; p -= lowbit(p))
            sum += tr[p];
        return sum;
    }
}Tr;

void solve(){
    int n, m;
    cin >> n >> m;
    vector<pii> pos;
    vector<tuple<int, int, int, int>> q;
    vector<ll> ans(m + 1);
    rep(i, 1, n){
        int tx, ty;
        cin >> tx >> ty;
        tx++, ty++;
        pos.push_back({tx, ty});
    }
    sort(pos.begin(), pos.end());
    rep(i, 1, m){
        int x1, y1, x2, y2;
        cin >> x1 >> y1 >> x2 >> y2;
        x1++, y1++, x2++, y2++;
        q.push_back({x1 - 1, y1 - 1, 1, i});
        q.push_back({x1 - 1, y2, -1, i});
        q.push_back({x2, y1 - 1, -1, i});
        q.push_back({x2, y2, 1, i});
    }
    sort(q.begin(), q.end());
    int cur = 0;
    for (auto [x, y, c, id] : q){
        while (cur < n && pos[cur].first <= x) Tr.add(pos[cur++].second, 1);
        ans[id] += c * Tr.query(y);
    }
    rep(i, 1, m) cout << ans[i] << endl;
}

int main(){
    ios;
    //freopen("sample.txt", "r", stdin);
    //freopen("resout.txt", "w", stdout);
    int t = 1;
    //cin >> t;
    while(t--){
        solve();
    }
    return 0;
}
```

- HH 的项链（区间元素种类）照常把 x 所在一维降掉后，发现 y 轴并没有明显的偏序关系。可以这样考虑，我们只计每个元素第一次在区间中出现时有贡献，设 pre[i] 表示位置 i 的元素前一次出现的位置，在整个序列中第一次出现时记为 0

```cpp
#include <bits/stdc++.h>
#define endl '\n'
#define pll pair<ll, ll>
#define tll tuple<ll, ll, ll>
#define vi vector<int>
#define vl vector<ll>
#define rep(i, j, k) for(int i = (j); i <= (k); i++)
#define per(i, j ,k) for(int i = (j); i >= (k); i--)
#define ios ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
typedef long long ll;
const ll maxn = 1e6 + 10;
const ll mod = 998244353;
const ll inf = 0x3f3f3f3f;

struct BIT{
    ll tr[maxn];
    int lowbit(int x){return x & -x;}
    void add(int p, ll x){
        for (; p < maxn; p += lowbit(p)) tr[p] += x;
    }
    ll query(int p){
        ll sum = 0;
        for (; p > 0; p -= lowbit(p))
            sum += tr[p];
        return sum;
    }
}Tr;

ll pre[maxn], ans[maxn];
void solve(){
    int n, m;
    cin >> n;
    vector<pll> pos;
    vector<tuple<int, int, int, int>> q;
    for (int i = 3; i <= n + 2; ++i){
        int a;
        cin >> a;
        pos.push_back({i, pre[a] ? pre[a] : 2}), pre[a] = i;
    }
    sort(pos.begin(), pos.end());
    cin >> m;
    for (int i = 1; i <= m; ++i){
        int l, r;
        cin >> l >> r;
        l += 2, r += 2;
        q.push_back({l - 1, 1, 1, i});
        q.push_back({l - 1, l - 1, -1, i});
        q.push_back({r, 1, -1, i});
        q.push_back({r, l - 1, 1, i});
    }
    sort(q.begin(), q.end());
    int cur = 0;
    for (auto [x, y, c, id] : q)
    {
        while (cur < n && pos[cur].first <= x)
            Tr.add(pos[cur++].second, 1);
        ans[id] += c * Tr.query(y);
    }
    for (int i = 1; i <= m; i++) cout << ans[i] << endl;
}

int main(){
    ios;
    //freopen("sample.txt", "r", stdin);
    //freopen("resout.txt", "w", stdout);
    int t = 1;
    //cin >> t;
    while(t--){
        solve();
    }
    return 0;
}
```

- 矩阵内权值之和

```cpp
#include <bits/stdc++.h>
#define endl '\n'
#define pll pair<ll, ll>
#define tll tuple<ll, ll, ll>
#define vi vector<int>
#define vl vector<ll>
#define rep(i, j, k) for(int i = (j); i <= (k); i++)
#define per(i, j ,k) for(int i = (j); i >= (k); i--)
#define ios ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
typedef long long ll;
const ll maxn = 3e5 + 10;
const ll mod = 998244353;
const ll inf = 0x3f3f3f3f;

```

```cpp
struct BIT{
    ll tr[maxn];
    int lowbit(int x){return x & -x;}
    void add(int p, ll x){
        for (; p < maxn; p += lowbit(p)) tr[p] += x;
    }
    ll query(int p){
        ll sum = 0;
        for (; p > 0; p -= lowbit(p))
            sum += tr[p];
        return sum;
    }
}Tr;

void solve(){
    int n, m;
    cin >> n >> m;
    vector<tuple<int, int, int>> pos;
    vector<tuple<int, int, int, int>> q;
    vector<ll> ans(m + 1);
    vector<int> yy;
    rep(i, 1, n){
        int x, y, p;
        cin >> x >> y >> p;
        yy.push_back(y);
        pos.push_back({x, y, p});
    }
    sort(pos.begin(), pos.end());
    rep(i, 1, m){
        int x1, y1, x2, y2;
        cin >> x1 >> y1 >> x2 >> y2;
        yy.push_back(y1 - 1), yy.push_back(y2);
        q.push_back({x1 - 1, y1 - 1, 1, i});
        q.push_back({x2, y1 - 1, -1, i});
        q.push_back({x1 - 1, y2, -1, i});
        q.push_back({x2, y2, 1, i});
    }
    sort(q.begin(), q.end());
    sort(yy.begin(), yy.end());
    yy.erase(unique(yy.begin(), yy.end()), yy.end());
    int cur = 0;
    for (auto [x, y, c, id] : q){
        y = lower_bound(yy.begin(), yy.end(), y) - yy.begin() + 1;
        while (cur < n){
            auto [_x, _y, p] = pos[cur];
            if (_x > x) break;
            _y = lower_bound(yy.begin(), yy.end(), _y) - yy.begin() + 1;
            Tr.add(_y, p), ++cur;
        }
        ans[id] += c * Tr.query(y);
    }
    for (int i = 1; i <= m; ++i) cout << ans[i] << endl;
}

int main(){
    ios;
    //freopen("sample.txt", "r", stdin);
    //freopen("resout.txt", "w", stdout);
    int t = 1;
    //cin >> t;
    while(t--){
        solve();
    }
    return 0;
}
```

可持续化线段树

- 区间第 k 小

前缀和思想

```cpp
#include <bits/stdc++.h>
#define endl '\n'
#define pll pair<ll, ll>
#define tll tuple<ll, ll, ll>
#define vi vector<int>
#define vl vector<ll>
#define x first
#define y second
#define rep(i, j, k) for(int i = (j); i <= (k); i++)
#define per(i, j ,k) for(int i = (j); i >= (k); i--)
#define ios ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
typedef long long ll;
#define int ll
const ll maxn = 2e5 + 10;
const ll mod = 998244353;
const ll inf = 0x3f3f3f3f;

struct node {
    int ls, rs;
    int cnt;
} tr[maxn << 5];
int idx = 0, rt[maxn << 5];

void push_up(int u) {
    tr[u].cnt = tr[tr[u].ls].cnt + tr[tr[u].rs].cnt;
}

int build(int l, int r) {
    int u = idx++;
    if (l == r) {
        tr[u].cnt = 0;
        return u;
    }
    int mid = l + r >> 1;
    tr[u].ls = build(l, mid);
    tr[u].rs = build(mid + 1, r);
    push_up(u);
    return u;
}

int update(int old, int l, int r, int pos, int val) {
    int u = idx++;
    tr[u] = tr[old];
    if (l == pos && r == pos) {
        tr[u].cnt += val;
        return u;
    }
    int mid = l + r >> 1;
    if (pos <= mid) tr[u].ls = update(tr[old].ls, l, mid, pos, val);
    else tr[u].rs = update(tr[old].rs, mid + 1, r, pos, val);
    push_up(u);
    return u;
}

int query(int l, int r, int o, int v, int kth) {
    if (l == r) return l;
    int mid = (l + r) >> 1;
    int res = tr[tr[v].ls].cnt - tr[tr[o].ls].cnt;
    if (kth <= res) return query(l, mid, tr[o].ls, tr[v].ls, kth);
    else return query(mid + 1, r, tr[o].rs, tr[v].rs, kth - res);
}

int b[maxn], stb[maxn];
void solve() {
    int n, m;
    cin >> n >> m;
    for (int i = 1; i <= n; ++i) {
        cin >> b[i], stb[i] = b[i];
    }
    sort(stb + 1, stb + 1 + n);
    int cnt = 1;
    for (int i = 2; i <= n; ++i) {
        if (stb[i] != stb[cnt]) stb[++cnt] = stb[i];
    }
    rt[0] = build(1, cnt);
    for (int i = 1; i <= n; ++i) {
        int p = lower_bound(stb + 1, stb + cnt + 1, b[i]) - stb;
        rt[i] = update(rt[i - 1], 1, cnt, p, 1);
    }
    for (int i = 1; i <= m; ++i) {
        int l, r, k;
        cin >> l >> r >> k;
        int idx = query(1, cnt, rt[l - 1], rt[r], k);
        cout << stb[idx] << endl;
    }
}

signed main() {
    ios;
    //freopen("sample.txt", "r", stdin);
    //freopen("resout.txt", "w", stdout);
    int t = 1;
    //cin >> t;
    while (t--) {
        solve();
    }
    return 0;
}
```

- HH 的项链

求区间内不重复的数的个数。扫描数列建立可持久化线段树，第 i 个数若第一次出现，则在线段树中的位置 i 加 1；若不是第一次出现，将上次出现的位置减 1，在本次位置加 1。对于每个询问的区间 [L,R]，在第 R 个版本上的线段树只有前 R 个数，在线段树上查询位置 L，对经过的区间中的和进行累计即可。

```cpp
#include <bits/stdc++.h>
#define endl '\n'
#define pll pair<ll, ll>
#define tll tuple<ll, ll, ll>
#define vi vector<int>
#define vl vector<ll>
#define x first
#define y second
#define rep(i, j, k) for(int i = (j); i <= (k); i++)
#define per(i, j ,k) for(int i = (j); i >= (k); i--)
#define ios ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
typedef long long ll;
const ll maxn = 1e6 + 10;
const ll mod = 998244353;
const ll inf = 0x3f3f3f3f;

struct node{
    int ls, rs;
    int cnt;
}tr[maxn << 5];
int idx = 0, rt[maxn];

void push_up(int u){
    tr[u].cnt = tr[tr[u].ls].cnt + tr[tr[u].rs].cnt;
}

int build(int l, int r){
    int u = idx++;
    if (l == r){
        tr[u].cnt = 0;
        return u;
    }
    int mid = l + r >> 1;
    tr[u].ls = build(l, mid);
    tr[u].rs = build(mid + 1, r);
    push_up(u);
    return u;
}

int update(int old, int l, int r, int pos, int val){
    int u = idx++;
    tr[u] = tr[old];
    if (l == pos && r == pos){
        tr[u].cnt += val;
        return u;
    }
    int mid = l + r >> 1;
    if (pos <= mid) tr[u].ls = update(tr[old].ls, l, mid, pos, val);
    else tr[u].rs = update(tr[old].rs, mid + 1, r, pos, val);
    push_up(u);
    return u;
}

int query(int l, int r, int ver, int pos){
    if (l == r) return tr[ver].cnt;
    int mid = l + r >> 1;
    if (pos <= mid) return tr[tr[ver].rs].cnt + query(l, mid, tr[ver].ls, pos);
    else return query(mid + 1, r, tr[ver].rs, pos);
}

int b[maxn], sortb[maxn];
map<int, int> mp;
void solve(){
    int n, m;
    cin >> n;
    for (int i = 1; i <= n; ++i) cin >> b[i], sortb[i] = b[i];
    sort(sortb + 1, sortb + 1 + n);
    int cnt = 1;
    for (int i = 2; i <= n)
    rt[0] = build(1, n);

}

int main(){
    ios;
    //freopen("sample.txt", "r", stdin);
    //freopen("resout.txt", "w", stdout);
    int t = 1;
    //cin >> t;
    while(t--){
        solve();
    }
    return 0;
}
```

- 区间离散化，多少数字不一样

```
#include <bits/stdc++.h>
#define endl '\n'
#define pll pair<ll, ll>
#define tll tuple<ll, ll, ll>
#define vi vector<int>
#define vl vector<ll>
#define x first
#define y second
#define int ll
#define rep(i, j, k) for(int i = (j); i <= (k); i++)
#define per(i, j ,k) for(int i = (j); i >= (k); i--)
#define ios ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
typedef long long ll;
const ll maxn = 3e5 + 10;
const ll mod = 998244353;
const ll inf = 0x3f3f3f3f;

struct node{
    int ls, rs;
    int cnt, mex;
}tr[maxn << 5];
int idx = 0, rt[maxn];

void push_up(int u){
    tr[u].cnt = tr[tr[u].ls].cnt + tr[tr[u].rs].cnt;
    tr[u].mex = min(tr[tr[u].ls].mex, tr[tr[u].rs].mex);
}

int build(int l, int r){
    int u = idx++;
    if (l == r){
        tr[u].cnt = 0;
        return u;
    }
    int mid = l + r >> 1;
    tr[u].ls = build(l, mid);
    tr[u].rs = build(mid + 1, r);
    push_up(u);
    return u;
}

int update(int old, int l, int r, int pos, int val){
    int u = idx++;
    tr[u] = tr[old];
    if (l == pos && r == pos){
        tr[u].cnt++;
        tr[u].mex = val;
        return u;
    }
    int mid = l + r >> 1;
    if (pos <= mid) tr[u].ls = update(tr[old].ls, l, mid, pos, val);
    else tr[u].rs = update(tr[old].rs, mid + 1, r, pos, val);
    push_up(u);
    return u;
}

int queryMex(int u, int l, int r, int pos){
    if (l == r) return l;
    int mid = l + r >> 1;
    if (tr[tr[u].ls].mex < pos) return queryMex(tr[u].ls, l, mid, pos);
    else return queryMex(tr[u].rs, mid + 1, r, pos);
}

int queryVal(int s, int t, int L, int R, int l, int r) {
    if (l == L && R == r){return tr[t].cnt - tr[s].cnt;}
    int mid = L + R >> 1;
    if (r <= mid) return queryVal(tr[s].ls, tr[t].ls, L, mid, l, r);
    else if (l <= mid){
        int res = queryVal(tr[s].ls, tr[t].ls, L, mid, l, mid);
        res += queryVal(tr[s].rs, tr[t].rs, mid + 1, R, mid + 1, r);
        return res;
    }
    else return queryVal(tr[s].rs, tr[t].rs, mid + 1, R, l, r);
}

void solve(){
    int n;
    cin >> n;
    vi a(n + 1);
    rt[0] = build(1, n + 1);
    for (int i = 1; i <= n; ++i){
        cin >> a[i];
        if (a[i] > n) a[i] = n + 1;
        rt[i] = update(rt[i - 1], 1, n + 1, a[i], i);
    }
    int m, l, r;
    cin >> m;
    while (m--){
        cin >> l >> r;
```

```cpp
        int mex = queryMex(rt[r], 1, n + 1, l);
        int res = queryVal(rt[l - 1], rt[r], 1, n + 1, 1, mex);
        res = r - l + 1 - res;
        cout << res << endl;
    }
}

signed main(){
    ios;
    //freopen("sample.txt", "r", stdin);
    //freopen("resout.txt", "w", stdout);
    int t = 1;
    //cin >> t;
    while(t--){
        solve();
    }
    return 0;
}
```

可持久化 01Trie

- 区间 xorK 意义下的最大值

```cpp
#include <algorithm>
#include <cstdio>
#include <cstring>
using namespace std;
const int maxn = 600010;
int n, q, a[maxn], s[maxn], l, r, x;
char op;

struct Trie {
    int cnt, rt[maxn], ch[maxn * 33][2], val[maxn * 33];

    void insert(int o, int lst, int v) {
        for (int i = 28; i >= 0; i--) {
            val[o] = val[lst] + 1;  // 在原版本的基础上更新
            if ((v & (1 << i)) == 0) {
                if (!ch[o][0]) ch[o][0] = ++cnt;
                ch[o][1] = ch[lst][1];
                o = ch[o][0];
                lst = ch[lst][0];
            } else {
                if (!ch[o][1]) ch[o][1] = ++cnt;
                ch[o][0] = ch[lst][0];
                o = ch[o][1];
                lst = ch[lst][1];
            }
        }
        val[o] = val[lst] + 1;
        // printf("%d\n",o);
    }

    int query(int o1, int o2, int v) {
        int ret = 0;
        for (int i = 28; i >= 0; i--) {
            // printf("%d %d %d\n",o1,o2,val[o1]-val[o2]);
            int t = ((v & (1 << i)) ? 1 : 0);
            if (val[ch[o1][!t]] - val[ch[o2][!t]])
                ret += (1 << i), o1 = ch[o1][!t],
                                 o2 = ch[o2][!t];  // 尽量向不同的地方跳
            else
                o1 = ch[o1][t], o2 = ch[o2][t];
        }
        return ret;
    }
} st;

int main() {
    scanf("%d%d", &n, &q);
    for (int i = 1; i <= n; i++) scanf("%d", a + i), s[i] = s[i - 1] ^ a[i];
    for (int i = 1; i <= n; i++)
        st.rt[i] = ++st.cnt, st.insert(st.rt[i], st.rt[i - 1], s[i]);
    while (q--) {
        scanf(" %c", &op);
        if (op == 'A') {
            n++;
            scanf("%d", a + n);
            s[n] = s[n - 1] ^ a[n];
            st.rt[n] = ++st.cnt;
            st.insert(st.rt[n], st.rt[n - 1], s[n]);
        }
        if (op == 'Q') {
            scanf("%d%d%d", &l, &r, &x);
            l--;
            r--;
            if (l == 0)
                printf("%d\n", max(s[n] ^ x, st.query(st.rt[r], st.rt[0], s[n] ^ x)));
            else
                printf("%d\n", st.query(st.rt[r], st.rt[l - 1], s[n] ^ x));
        }
    }
    return 0;
}
```

树形 DP

- 树的重心

```
#include <bits/stdc++.h>
#define endl '\n'
#define ios ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
const int maxn = 2e5 + 10;
typedef long long ll;
int n;

void solve()
{
    vector<vector<int>> g(n + 1);
    for (int i = 0; i < n - 1; i++)
    {
        int x, y;
        cin >> x >> y;
        x--, y--;
        g[x].push_back(y);
        g[y].push_back(x);
    }
    vector<int> siz(n + 1);
    int id = 1e9, Min = 1e9;
    function<void(int, int)> dfs = [&](int x, int fa)
    {
        siz[x] = 1;
        for (auto y : g[x])
        {
            if (y == fa)
                continue;
            dfs(y, x);
            siz[x] += siz[y];
            int v = max(siz[x], n - siz[x]);
            if (v <= Min)
            {
                if (v < Min)
                    Min = v, id = x;
                else if (x < id)
                    id = x;
            }
        }
    };
    dfs(0, 0);
    cout << id + 1 << " " << Min - 1 << "\n";
}
```

```cpp

int main()
{
    ios;
    while (cin >> n)
    {
        solve();
    }
    return 0;
}
```

- 树的最大独立集

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

struct edge {
  int v, next;
} e[6005];

int head[6005], n, cnt, f[6005][2], ans, is_h[6005], vis[6005];

void addedge(int u, int v) {  // 建图
  e[++cnt].v = v;
  e[cnt].next = head[u];
  head[u] = cnt;
}

void calc(int k) {
  vis[k] = 1;
  for (int i = head[k]; i; i = e[i].next) {  // 枚举该结点的每个子结点
    if (vis[e[i].v]) continue;
    calc(e[i].v);
    f[k][1] += f[e[i].v][0];
    f[k][0] += max(f[e[i].v][0], f[e[i].v][1]);  // 转移方程
  }
  return;
}

int main() {
  scanf("%d", &n);
  for (int i = 1; i <= n; i++) scanf("%d", &f[i][1]);
  for (int i = 1; i < n; i++) {
    int l, k;
    scanf("%d%d", &l, &k);
    is_h[l] = 1;
    addedge(k, l);
  }
  for (int i = 1; i <= n; i++)
    if (!is_h[i]) {  // 从根结点开始 DFS
      calc(i);
      printf("%d", max(f[i][1], f[i][0]));
      return 0;
    }
}
```

- 树的最小支配集

```cpp
#include <bits/stdc++.h>
using namespace std;
#define N 10010

int n;
int e[N * 2], ne[N * 2], h[N], idx = 0;
int f[N][3];
/***
 * f[i][0] 选 i 且 i 及 i 的子树都被覆盖了
 * f[i][1] 不选 i 且 i 被其儿子覆盖
 * f[i][2] 不选 i 且 i 被其父亲覆盖 (儿子可选可不选)
*/
void add(int a, int b)
{
    e[idx] = b, ne[idx] = h[a], h[a] = idx++;
}

void dfs(int u, int pre)
{
    f[u][0] = 1, f[u][1] = f[u][2] = 0;
    bool flag = true;
    int tmp = 0x3f3f3f3f;
    for (int i = h[u]; ~i; i = ne[i])
    {
        int v = e[i];
        if (v == pre)
            continue;
        dfs(v, u);
        f[u][2] += min(f[v][1], f[v][0]);
        f[u][0] += min(min(f[v][0], f[v][1]), f[v][2]);
        if (f[v][0] <= f[v][1])
        {
            flag = false;
            f[u][1] += f[v][0];
        }
        else
        {
            f[u][1] += f[v][1];
            tmp = min(tmp, f[v][0] - f[v][1]);
        }
    }
    if (flag)
        f[u][1] += tmp;
}

int main()
{
    memset(f, 0x3f, sizeof f);
    memset(h, -1, sizeof h);
    scanf("%d", &n);
    for (int i = 1; i < n; i++)
    {
        int a, b;
        scanf("%d%d", &a, &b);
        add(a, b), add(b, a);
    }
    dfs(1, -1);
    int ans = min(f[1][0], f[1][1]);
    cout << ans << endl;
    return 0;
}
```

- 树的最小覆盖点

```cpp
#include <bits/stdc++.h>
#define endl '\n'
#define mset(s, _) memset(s, _, sizeof(s))
#define rep(i, l, r) for (int i = l; i <= r; ++i)
#define ios ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
const int N = 4e3 + 10, mod = 1e9 + 7;
int n, m;
int h[N], nex[N], v[N], idx;
void add(int a, int b) {
    v[idx] = b; nex[idx] = h[a]; h[a] = idx ++ ;
}

int f[N][5], st[N];
void init() {
    mset(h, -1); mset(f, 0); mset(st, 0); idx = 0;
}

void dp(int u) {
    bool fg = 0;
    for(int i = h[u]; ~i; i = nex[i]) {
        int j = v[i];
        fg = 1;
        dp(j);
        f[u][0] += f[j][1];
        f[u][1] += min(f[j][0], f[j][1]);
    }
    f[u][1] += 1;
    if(!fg) {
        f[u][0] = 0; f[u][1] = 1;
    }
}

int main() {
    while(cin >> n) {
        init();
        rep(i, 1, n) {
            int a, num, b; char t;
            cin >> a >> t >> t >> num >> t;
            rep(j, 1, num) {
                cin >> b; add(a, b); st[b] = 1;
            }
        }
        int root = 0;
        while(st[root]) root ++ ;
        dp(root);
        cout << min(f[root][1], f[root][0]) << endl;
    }

    return 0;
}
```

- 树上背包

最多切 q 条边，剩下多少东西

```
#include <bits/stdc++.h>
using namespace std;
const int N = 105;
int dp[N][N];//dp[i][j] 以 i 为根的子树保留 j 个分支可以得到的最大的苹果数量
int h[N], e[N << 1], nx[N << 1], w[N << 1];
int cnt = 1;
void add(int a, int b, int v)
{
    e[cnt] = b;
    w[cnt] = v;
    nx[cnt] = h[a];
    h[a] = cnt++;
}
int n, q;
void dfs(int u, int f)
{
    for (int i = h[u]; i; i = nx[i])
    {
        int v = e[i];
        if (v != f)
        {
            dfs(v, u);
            for (int j = q; j >= 1; j--)
            {
                for (int k = 0; k <= j - 1; k++)
                {
                    dp[u][j] = max(dp[u][j], dp[u][k] + w[i] + dp[v][j - k - 1]);
                }
            }
        }
    }
}
int main()
{

    cin >> n >> q;
    int a, b, v;
    for (int i = 0; i < n - 1; i++)
    {
        cin >> a >> b >> v;
        add(a, b, v);
        add(b, a, v);
    }
    dfs(1, -1);
    cout << dp[1][q];
}
```

- 树的直径 (带点权)

```
#include <bits/stdc++.h>
#define endl '\n'
#define pll pair<ll, ll>
#define tll tuple<ll, ll, ll>
#define ios ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
typedef long long ll;
const ll maxn = 2e5 + 10;
const ll mod = 998244353;
vector<ll> G[maxn];
ll w[maxn], dis[maxn], ans = -1e18;

void solve(){
    int n;
    cin >> n;
    for (int i = 1; i <= n; ++i){
        cin >> w[i];
    }
    for (int i = 1; i <= n - 1; ++i){
        int u, v;
        cin >> u >> v;
        G[u].push_back(v);
        G[v].push_back(u);
    }
    function<void(int, int)> dfs = [&](int u, int fa){
        ll tmp = 0, mx1 = 0, mx2 = 0;
        for (auto v: G[u]){
            if (v == fa) continue;
            dfs(v, u);
            tmp = dis[v];
            if (tmp >= mx1){
                mx2 = mx1;
                mx1 = tmp;
            }else if (tmp >= mx2){
                mx2 = tmp;
            }
        }
        ans = max(ans, mx1 + mx2 + w[u]);
        dis[u] = mx1 + w[u];
    };
    dfs(1, 0);
    cout << ans << endl;
}

int main(){
    ios;
    int t = 1;
    //cin >> t;
    while(t--){
        solve();
    }
    return 0;
}
```

区间问题

莫队

- 区间取两个数相同概率

```cpp
#include <algorithm>
#include <cmath>
#include <cstdio>
using namespace std;
const int N = 50005;
int n, m, maxn;
int c[N];
long long sum;
int cnt[N];
long long ans1[N], ans2[N];

struct query {
  int l, r, id;

  bool operator<(const query &x) const {  // 重载 < 运算符
    if (l / maxn != x.l / maxn) return l < x.l;
    return (l / maxn) & 1 ? r < x.r : r > x.r;
  }
} a[N];

void add(int i) {
  sum += cnt[i];
  cnt[i]++;
}

void del(int i) {
  cnt[i]--;
  sum -= cnt[i];
}

long long gcd(long long a, long long b) { return b ? gcd(b, a % b) : a; }

int main() {
  scanf("%d%d", &n, &m);
  maxn = sqrt(n);
  for (int i = 1; i <= n; i++) scanf("%d", &c[i]);
  for (int i = 0; i < m; i++) scanf("%d%d", &a[i].l, &a[i].r), a[i].id = i;
  sort(a, a + m);
  for (int i = 0, l = 1, r = 0; i < m; i++) {  // 具体实现
    if (a[i].l == a[i].r) {
      ans1[a[i].id] = 0, ans2[a[i].id] = 1;
      continue;
    }
    while (l > a[i].l) add(c[--l]);
    while (r < a[i].r) add(c[++r]);
    while (l < a[i].l) del(c[l++]);
    while (r > a[i].r) del(c[r--]);
    ans1[a[i].id] = sum;
    ans2[a[i].id] = (long long)(r - l + 1) * (r - l) / 2;
  }
  for (int i = 0; i < m; i++) {
    if (ans1[i] != 0) {
      long long g = gcd(ans1[i], ans2[i]);
      ans1[i] /= g, ans2[i] /= g;
    } else
      ans2[i] = 1;
    printf("%lld/%lld\n", ans1[i], ans2[i]);
  }
  return 0;
}
```

CDQ

- 逆序对

```cpp
#include <bits/stdc++.h>
#define endl '\n'
#define pll pair<ll, ll>
#define tll tuple<ll, ll, ll>
#define vi vector<int>
#define vl vector<ll>
```

```cpp
#define x first
#define y second
#define rep(i, j, k) for(int i = (j); i <= (k); i++)
#define per(i, j ,k) for(int i = (j); i >= (k); i--)
#define ios ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
typedef long long ll;
const ll maxn = 2e5 + 10;
const ll mod = 998244353;
const ll inf = 0x3f3f3f3f;

void solve(){
    int n;
    cin >> n;
    vi a(n + 1), temp(n + 1);
    ll ans = 0;
    rep(i, 1, n) cin >> a[i];
    function<void(int, int)> cdq = [&](int l, int r){
        if (l == r) return;
        int mid = l + r >> 1;
        cdq(l, mid);
        cdq(mid + 1, r);
        int p1 = l, p2 = mid + 1, idx = l;
        while (p1 <= mid && p2 <= r){
            if (a[p1] > a[p2]) temp[idx++] = a[p1++];
            else temp[idx++] = a[p2++], ans += p1 - l;
        }
        while (p1 <= mid) temp[idx++] = a[p1++];
        while (p2 <= r) temp[idx++] = a[p2++], ans += p1 - l;
        for (int i = l; i <= r; ++i) a[i] = temp[i];
    };
    cdq(1, n);
    cout << ans << endl;
}

int main(){
    ios;
    //freopen("sample.txt", "r", stdin);
    //freopen("resout.txt", "w", stdout);
    int t = 1;
    //cin >> t;
    while(t--){
        solve();
    }
    return 0;
}
```

- 求最长不上升子序列和最长上升子序列

```cpp
#include<bits/stdc++.h>
using namespace std;
const int MAXN = 100005;
int n, x, dp[MAXN], a[MAXN], ans;
pair<int, int> temp[MAXN][20]; //val, pos

bool cmp(const pair<int, int> &A, const pair<int, int> &B, const int &type) {
    return type ? A.first != B.first ? A.first > B.first : A.second < B.second : A.first != B.first ? A.first <
↪   B.first: A.second > B.second;
}

void mergeSort(int l, int r, int deep, const int &cmptype) {
    if (l == r) {
        temp[l][deep].first = a[l];
        temp[l][deep].second = l;
        return;
    }
    int mid = (l + r) >> 1;
    mergeSort(l, mid, deep + 1, cmptype);
    mergeSort(mid + 1, r, deep + 1, cmptype);
    int p1 = l, p2 = mid + 1;
    while (p1 <= mid && p2 <= r) {
        if (cmp(temp[p1][deep + 1], temp[p2][deep + 1], cmptype)) {
            temp[l++][deep] = temp[p1++][deep + 1];
        } else {
            temp[l++][deep] = temp[p2++][deep + 1];
        }
    }
    while (p1 <= mid) {
        temp[l++][deep] = temp[p1++][deep + 1];
    }
    while (p2 <= r) {
        temp[l++][deep] = temp[p2++][deep + 1];
    }
}

void cdqDivAlgorithm(int l, int r, int deep, const int &cmptype) {
    if (l == r) {
        dp[l] = max(dp[l], 1);
        ans = max(ans, dp[l]);
        return;
    }
    int mid = (l + r) >> 1;
    cdqDivAlgorithm(l, mid, deep + 1, cmptype);
    int p1 = l, p2 = mid + 1, premax = 0;
    while (p1 <= mid && p2 <= r) {
        if (cmp(temp[p1][deep + 1], temp[p2][deep + 1], cmptype)) {
            premax = max(premax, dp[temp[p1++][deep + 1].second]);
        } else {
            dp[temp[p2][deep + 1].second] = max(premax + 1, dp[temp[p2][deep + 1].second]);
            p2++;
        }
    }
    while (p2 <= r) {
        dp[temp[p2][deep + 1].second] = max(premax + 1, dp[temp[p2][deep + 1].second]);
        p2++;
    }
    cdqDivAlgorithm(mid + 1, r, deep + 1, cmptype);
}

int main()
{
    while (scanf("%d", &x) != EOF)a[++n] = x;
    mergeSort(1, n, 0, 1);
    cdqDivAlgorithm(1, n, 0, 1);
    printf("%d\n", ans);
    memset(dp, 0, sizeof(dp));
    ans = 0;
    mergeSort(1, n, 0, 0);
    cdqDivAlgorithm(1, n, 0, 0);
    printf("%d\n", ans);
    return 0;
}
```

- 求地毯覆盖（最多取多少个不相互覆盖）

```cpp
#include<bits/stdc++.h>
using namespace std;
const int MAXN = 1000005;
int n, L[MAXN], R[MAXN], id[MAXN], dp[MAXN], ans;
int temp[MAXN];
void cdqDivAlgorithm(int l, int r) {
    if (l == r) {
        dp[id[l]] = max(1, dp[id[l]]);
        ans = max(ans, dp[id[l]]);
        return;
    }
    int mid = (l + r) >> 1;
    cdqDivAlgorithm(l, mid);
    int p1 = l, pl, p2 = mid + 1, premax = 0;
    while (p1 <= mid && p2 <= r) {
        if (R[id[p1]] <= L[id[p2]]) {
            premax = max(premax, dp[id[p1++]]);
        } else {
            dp[id[p2]] = max(premax + 1, dp[id[p2]]);
            ++p2;
        }
    }
    while (p2 <= r) {
        dp[id[p2]] = max(premax + 1, dp[id[p2]]);
        ++p2;
    }
    cdqDivAlgorithm(mid + 1, r);
    p1 = l, pl = l, p2 = mid + 1;
    while (p1 <= mid && p2 <= r) {
        if (R[id[p1]] < R[id[p2]]) {
            temp[pl++] = id[p1++];
        } else {
            temp[pl++] = id[p2++];
        }
    }
    while (p1 <= mid) {
        temp[pl++] = id[p1++];
    }
    while (p2 <= r) {
        temp[pl++] = id[p2++];
    }
    for (int i = l; i <= r; ++i) {
        id[i] = temp[i];
    }
}
int main()
{
    scanf("%d", &n);
    for (int i = 1; i <= n; ++i) {
        scanf("%d %d", &L[i], &R[i]);
        id[i] = i;
    }
    sort(id + 1, id + 1 + n, [](const int &A, const int &B) {
        return L[A] < L[B];
    });
    cdqDivAlgorithm(1, n);
    printf("%d\n", ans);
    return 0;
}
```

- 动态凸包

第一行：一个整数 N ，表示方案和询问的总数。接下来 N 行，每行开头一个单词 “Query” 或 “Project” 。若单词为 Query，则后接一个整数 T，表示 Blue Mary 询问第 T 天的最大收益。若单词为 Project，则后接两个实数 S，P，表示该种设计方案第一天的收益 S，以及以后每天比上一天多出的收益 P。对于每一个 Query，输出一个整数，表示询问的答案，并精确到整百元 1 <= N <= 100000 1 <= T <=50000 0 < P < 100，｜S｜<= 10^6

```
#include<bits/stdc++.h>
using namespace std;
const int MAXN = 100005;
const double eps = 1e-6;
int m, n, id[MAXN], qid[MAXN], type[MAXN], x[MAXN], temp[MAXN], top;
double k[MAXN], b[MAXN], ans[MAXN];
char op[55];
inline bool cmp(const int &A, const int &B) {
    return type[A] != type[B] ? type[A] < type[B] : type[A] ? x[A] < x[B] : k[A] < k[B];
}
inline int dcmp(double x) {
    return x > eps ? 1 : x < -eps ? -1 : 0;
}
inline double getCross(const double &k1, const double &b1, const double &k2, const double &b2) {
    return (b2 - b1) / (k1 - k2);
}
inline double getVal(const double &k, const double &b, const int &x)
{
    return k * x + b;
}
pair<double, double>stk[MAXN];
void stkClear() {
    top = 0;
    stk[++top] = make_pair(0, 0);
}
void stkInsert(double k, double b) {
    if (dcmp(stk[top].first - k) == 0 && dcmp(stk[top].second - b) < 0)top--;
    if (dcmp(stk[top].first - k) == 0 && dcmp(stk[top].second - b) >= 0)return;
    while (top >= 2 && dcmp(getCross(stk[top].first, stk[top].second, stk[top - 1].first, stk[top - 1].second) -
    getCross(stk[top].first, stk[top].second, k, b)) > 0)top--;
    stk[++top] = make_pair(k, b);
}
double stkQuery(int x) {
    while (top >= 2 && dcmp(getVal(stk[top].first, stk[top].second, x) - getVal(stk[top - 1].first, stk[top -
    1].second, x)) < 0)--top;
    return getVal(stk[top].first, stk[top].second, x);
}
void cdqDivAlgorithm(int l, int r) {
    if (l == r)return;
    int mid = (l + r) >> 1;
    cdqDivAlgorithm(l, mid);
    cdqDivAlgorithm(mid + 1, r);
    stkClear();
    for (int i = l; i <= mid && !type[id[i]]; ++i) {
        stkInsert(k[id[i]], b[id[i]]);
    }
    for (int i = r; i > mid && type[id[i]]; --i) {
        ans[qid[id[i]]] = max(ans[qid[id[i]]], stkQuery(x[id[i]]));
    }
    int p1 = l, pl = l, p2 = mid + 1;
    while (p1 <= mid && p2 <= r) {
        if (cmp(id[p1], id[p2])) {
            temp[pl++] = id[p1++];
        } else {
            temp[pl++] = id[p2++];
        }
    }
    while (p1 <= mid) {
        temp[pl++] = id[p1++];
    }
    while (p2 <= r) {
        temp[pl++] = id[p2++];
    }
    for (int i = l; i <= r; ++i) {
        id[i] = temp[i];
    }
}
int main() {
    scanf("%d", &n);
    for (int i = 1; i <= n; ++i) {
        id[i] = i;
        scanf("%s", op);
        if (*op == 'P') {
            type[i] = 0;
            scanf("%lf %lf", &b[i], &k[i]);
            b[i] -= k[i];
        }
        else {
            type[i] = 1;
            qid[i] = ++m;
            scanf("%d", &x[i]);
        }
    }
    cdqDivAlgorithm(1, n);
    for (int i = 1; i <= m; ++i) {
        printf("%d\n", (int)ans[i] / 100);
    }
    return 0;
}
```

树上问题

树剖

- 2018ICPC 青岛网络赛（多测时候用来剖的）

```cpp
#include <bits/stdc++.h>
#define endl '\n'
#define pll pair<ll, ll>
#define tll tuple<ll, ll, ll>
#define vi vector<int>
#define vl vector<ll>
#define x first
#define y second
#define rep(i, j, k) for(int i = (j); i <= (k); i++)
#define per(i, j, k) for(int i = (j); i >= (k); i--)
#define ios ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
typedef long long ll;
const ll maxn = 1e5 + 10;
const ll mod = 998244353;
const ll inf = 0x3f3f3f3f;

void solve()
{
    int n, m, q, k, cnt = 0;
    cin >> n >> m >> q;
    vi red(n + 1);
    vector<vector<pll>> G(n + 1);
    vl dis(n + 1), dep(n + 1), v(n + 1);
    vi dfn(n + 1), idx(n + 1);
    vi son(n + 1, -1), sz(n + 1), fa(n + 1), top(n + 1);
    function<void(int, int)> dfs1 = [&](int u, int f) {
        son[u] = -1;
        sz[u] = 1;
        if(!red[u])
            red[u] = red[f];
        for(auto [v, w] : G[u]) {
            if(v == f)
                continue;
            dep[v] = dep[u] + 1;
            dis[v] = dis[u] + w;
            fa[v] = u;
            dfs1(v, u);
            sz[u] += sz[v];
            if(son[u] == -1 || sz[v] > sz[son[u]])
                son[u] = v;
        }
    };
    function<void(int, int)> dfs2 = [&](int u, int t) {
        top[u] = t;
        dfn[u] = ++cnt;
        idx[cnt] = u;
        if(son[u] == -1)
            return;
        dfs2(son[u], t);
        for(auto [v, w] : G[u])
            if(v != son[u] && v != fa[u])
                dfs2(v, v);
    };
    auto lca = [&](int u, int v) {
        while(top[u] != top[v]) {
            if(dep[top[u]] > dep[top[v]])
                u = fa[top[u]];
            else
                v = fa[top[v]];
        }
        return dep[u] > dep[v] ? v : u;
    };
    for(int i = 1, x; i <= m; ++i)
        cin >> x, red[x] = x;
    for(int i = 1; i < n; ++i) {
        int u, v, w;
        cin >> u >> v >> w;
        G[u].push_back({v, w});
        G[v].push_back({u, w});
    }
    dfs1(1, 0);
    dfs2(1, 1);
    for(int i = 1; i <= n; ++i)
        v[i] = dis[i] - dis[red[i]];
    while(q--) {
        cin >> k;
        vector<int> p(k + 1);
        auto check = [&](ll st) {
            vector<int> q;
            for(int i = 1; i <= k; ++i)
                if(v[p[i]] > st)
                    q.push_back(p[i]);
            if(q.size() == 0)
                return true;
            int mnd = n + 1, mxd = 0;
            for(int i = 0; i < q.size(); ++i) {
                mnd = min(mnd, dfn[q[i]]);
                mxd = max(mxd, dfn[q[i]]);
            }
            int ca = lca(idx[mnd], idx[mxd]);
            for(int i = 0; i < q.size(); ++i)
                if(dis[q[i]] - dis[ca] > st)
                    return false;
            return true;
        };
        ll mx = 0;
        for(int i = 1; i <= k; ++i) {
            cin >> p[i];
            mx = max(mx, v[p[i]]);
        }
        ll l = 0, r = mx;
        while(l < r) {
            ll mid = (l + r) >> 1;
            if(check(mid))
                r = mid;
            else
                l = mid + 1;
        }
        cout << l << endl;
    }
}

int main()
{
    ios;
    // freopen("sample.txt", "r", stdin);
    // freopen("resout.txt", "w", stdout);
    int t = 1;
    cin >> t;
    while(t--) {
        solve();
    }
    return 0;
}

```

- 树上操作

1. 节点 x 加上 a
2. 节点 x 的子树中所有点的点权加 a
3. 询问某个点 x 到根节点

```
#include <bits/stdc++.h>
#define endl '\n'
#define pll pair<ll, ll>
```

```
#define tll tuple<ll, ll, ll>
#define vi vector<int>
#define vl vector<ll>
#define x first
#define y second
#define rep(i, j, k) for(int i = (j); i <= (k); i++)
#define per(i, j ,k) for(int i = (j); i >= (k); i--)
#define ios ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
typedef long long ll;
const ll maxn = 2e5 + 10;
const ll mod = 998244353;
const ll inf = 0x3f3f3f3f;
const int N = 1e5 + 10, M = N * 2;

int n, m;
// w 为节点权值
int h[N], w[N], e[M], ne[M], idx;
// id[x] 为节点 x 的新编号, nw[x] 是新编号为 x 的节点的权值
int id[N], nw[N], cnt;
// dep 为深度, sz 为子树大小, top[x] 是 x 所在重链的头结点,
// fa[x] 为 x 父亲, son[x] 为 x 的重儿子
int dep[N], sz[N], top[N], fa[N], son[N];
struct Tree {
    int l, r;
    ll sum, add;
} tr[N << 2];

void add(int a, int b) {
    e[idx] = b, ne[idx] = h[a], h[a] = idx++;
}

// 第一次 dfs, 求节点深度、父亲、子树大小和重儿子
void dfs1(int u, int from, int depth) {
    dep[u] = depth, fa[u] = from, sz[u] = 1;
    for (int i = h[u]; ~i; i = ne[i]) {
        int v = e[i];
        if (v == from) continue;
        dfs1(v, u, depth + 1);
        sz[u] += sz[v];
        if (sz[son[u]] < sz[v]) son[u] = v;
    }
}

// 第二次 dfs, t 为 u 重链头结点
void dfs2(int u, int t) {
    id[u] = ++cnt, nw[cnt] = w[u], top[u] = t;
    // 到叶子了, 直接返回
    if (!son[u]) return;
    // 先遍历重儿子
    dfs2(son[u], t);
    // 遍历轻儿子
    for (int i = h[u]; ~i; i = ne[i]) {
        int v = e[i];
        if (v == fa[u] || v == son[u]) continue;
        dfs2(v, v);
    }
}

void pushup(int u) {
    tr[u].sum = tr[u << 1].sum + tr[u << 1 | 1].sum;
}

void pushdown(int u) {
    auto &root = tr[u], &left = tr[u << 1], &right = tr[u << 1 | 1];
    if (root.add) {
        left.sum += root.add * (left.r - left.l + 1);
        left.add += root.add;
        right.sum += root.add * (right.r - right.l + 1);
        right.add += root.add;
        root.add = 0;
```

```cpp
    }
}

void build(int u, int l, int r) {
    tr[u] = {l, r, nw[l], 0};
    if (l == r) return;
    int mid = l + r >> 1;
    build(u << 1, l, mid), build(u << 1 | 1, mid + 1, r);
    pushup(u);
}

void update(int u, int l, int r, ll k) {
    if (l <= tr[u].l && tr[u].r <= r) {
        tr[u].add += k;
        tr[u].sum += k * (tr[u].r - tr[u].l + 1);
        return;
    }
    pushdown(u);
    int mid = tr[u].l + tr[u].r >> 1;
    if (l <= mid) update(u << 1, l, r, k);
    if (r > mid) update(u << 1 | 1, l, r, k);
    pushup(u);
}

ll query(int u, int l, int r) {
    if (l <= tr[u].l && tr[u].r <= r) return tr[u].sum;
    pushdown(u);
    int mid = tr[u].l + tr[u].r >> 1;
    ll res = 0;
    if (l <= mid) res += query(u << 1, l, r);
    if (r > mid) res += query(u << 1 | 1, l, r);
    return res;
}

void update_path(int u, int v, ll k) {
    while (top[u] != top[v]) {
        if (dep[top[u]] < dep[top[v]]) swap(u, v);
        // u 的重链头更深，并且 u 重链头在 dfs 序里下标更小，直接更新 u 重链头到 u 这段区间
        update(1, id[top[u]], id[u], k);
        // u 跳到重链头上面
        u = fa[top[u]];
    }
    if (dep[u] < dep[v]) swap(u, v);
    update(1, id[v], id[u], k);
}

ll query_path(int u, int v) {
    ll res = 0;
    while (top[u] != top[v]) {
        if (dep[top[u]] < dep[top[v]]) swap(u, v);
        res += query(1, id[top[u]], id[u]);
        u = fa[top[u]];
    }
    if (dep[u] < dep[v]) swap(u, v);
    res += query(1, id[v], id[u]);
    return res;
}

void update_tree(int u, ll k) {
    update(1, id[u], id[u] + sz[u] - 1, k);
}

ll query_tree(int u) {
    return query(1, id[u], id[u] + sz[u] - 1);
}

void solve() {
    int n, q;
    memset(h, -1, sizeof h);
    cin >> n >> q;
    int cnt = 0;
    for (int i = 1; i <= n; ++i) cin >> w[i];
    for (int i = 1; i <= n - 1; ++i) {
        int u, v;
        cin >> u >> v;
        add(u, v);
        add(v, u);
    }
    dfs1(1, 1, 0);
    dfs2(1, 1);
    build(1, 1, n);
    while (q--) {
        int t, u;
        ll k;
        cin >> t >> u;
        if (t == 1) {
            cin >> k;
            update_path(u, u, k);
        } else if (t == 2) {
            cin >> k;
            update_tree(u, k);
        } else cout << query_path(1, u) << endl;
    }
}

int main() {
    ios;
    //freopen("sample.txt", "r", stdin);
    //freopen("resout.txt", "w", stdout);
    int t = 1;
    //cin >> t;
    while (t--) {
        solve();
    }
    return 0;
}
```

- 树上路径

1. 将以 u 为根的子树内节点 (包括 u) 的权值加 val
2. 将 (u, v) 路径上的节点权值加 val
3. 询问 (u, v) 路径上节点的权值两两相乘的和

```cpp
#include <bits/stdc++.h>
#define endl '\n'
#define pll pair<ll, ll>
#define tll tuple<ll, ll, ll>
#define vi vector<int>
#define vl vector<ll>
#define int ll
#define x first
#define y second
#define rep(i, j, k) for(int i = (j); i <= (k); i++)
#define per(i, j ,k) for(int i = (j); i >= (k); i--)
#define ios ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
typedef long long ll;
const ll mod = 1e9 + 7;
const ll inf = 0x3f3f3f3f;
const int N = 1e5 + 10, M = N * 2;

int n, m;
int h[N], a[N], e[M], ne[M], idx;
int id[N], cnt, rnk[N];
int dep[N], sz[N], top[N], fa[N], son[N];
ll inv2;

void add(int u, int v) {
    e[idx] = v, ne[idx] = h[u], h[u] = idx++;
}
ll qmi(ll x, ll k) {
    ll res = 1;
    while (k) {
        if (k & 1) res = res * x % mod;
        x = x * x % mod;
        k >>= 1;
    }
    return res;
}
struct Segment {
    struct Node {
        int l, r;
        ll sum, psum, add;
    } tr[N * 4];
    void pushup(int u) {
        tr[u].sum = (tr[u << 1].sum + tr[u << 1 | 1].sum) % mod;
        tr[u].psum = (tr[u << 1].psum + tr[u << 1 | 1].psum) % mod;
        return;
    }
    void pushdown(Node& u, Node& l, Node& r) {
        if (u.add) {
            ll x = u.add;
            l.psum = (l.psum + 2 * l.sum * x % mod + (ll)x * x % mod * (l.r - l.l + 1) % mod) % mod;
            r.psum = (r.psum + 2 * r.sum * x % mod + (ll)x * x % mod * (r.r - r.l + 1) % mod) % mod;
            l.sum = (l.sum + (ll)x * (l.r - l.l + 1) % mod) % mod;
            r.sum = (r.sum + (ll)x * (r.r - r.l + 1) % mod) % mod;
            l.add = (l.add + x) % mod;
            r.add = (r.add + x) % mod;
            u.add = 0;
        }
        return;
    }
    void pushdown(int u) {
        pushdown(tr[u], tr[u << 1], tr[u << 1 | 1]);
    }
    void build(int u, int l, int r) {
        tr[u] = {l, r};
        if (l == r) {
            tr[u].sum = a[rnk[l]];
            tr[u].psum = (ll)a[rnk[l]] * a[rnk[l]] % mod;
            return;
        }
        int mid = (l + r) >> 1;
        build(u << 1, l, mid);
        build(u << 1 | 1, mid + 1, r);
        pushup(u);
        return;
    }
    void update(int u, int l, int r, ll x) {
        if (l <= tr[u].l && tr[u].r <= r) {
           tr[u].psum = (tr[u].psum + 2 * tr[u].sum * x % mod + (ll)x * x % mod * (tr[u].r - tr[u].l + 1) % mod) % mod;
            tr[u].sum = (tr[u].sum + (ll)(tr[u].r - tr[u].l + 1) * x % mod) % mod;
            tr[u].add = (tr[u].add + x) % mod;
            return;
        }
        pushdown(u);
        int mid = (tr[u].l + tr[u].r) >> 1;
        if (l <= mid) update(u << 1, l, r, x);
        if (mid < r) update(u << 1 | 1, l, r, x);
        pushup(u);
        return;
    }
    ll query_sum(int u, int l, int r) {
        if (l <= tr[u].l && tr[u].r <= r) return tr[u].sum;
        pushdown(u);
        int mid = (tr[u].l + tr[u].r) >> 1;
        ll res = 0;
        if (l <= mid) res = (res + query_sum(u << 1, l, r)) % mod;
        if (mid < r) res = (res + query_sum(u << 1 | 1, l, r)) % mod;
        return res;
    }
    ll query_psum(int u, int l, int r) {
        if (l <= tr[u].l && tr[u].r <= r) return tr[u].psum;
        pushdown(u);
        int mid = (tr[u].l + tr[u].r) >> 1;
        ll res = 0;
        if (l <= mid) res = (res + query_psum(u << 1, l, r)) % mod;
        if (mid < r) res = (res + query_psum(u << 1 | 1, l, r)) % mod;
        return res;
    }
} Tr;

//Tree
void dfs1(int u, int from, int depth) {
    dep[u] = depth, fa[u] = from, sz[u] = 1;
    for (int i = h[u]; ~i; i = ne[i]) {
        int v = e[i];
        if (v == from) continue;
        dfs1(v, u, depth + 1);
        sz[u] += sz[v];
        if (sz[son[u]] < sz[v]) son[u] = v;
    }
}
void dfs2(int u, int t) {
    id[u] = ++cnt, top[u] = t;
    rnk[cnt] = u;
    if (!son[u]) return;
    dfs2(son[u], t);
    for (int i = h[u]; ~i; i = ne[i]) {
        int v = e[i];
        if (v == fa[u] || v == son[u]) continue;
        dfs2(v, v);
    }
}
void update_path(int u, int v, ll k) {//更新路径
    while (top[u] != top[v]) {
        if (dep[top[u]] < dep[top[v]]) swap(u, v);
        Tr.update(1, id[top[u]], id[u], k);
        u = fa[top[u]];
    }
    if (dep[u] < dep[v]) swap(u, v);
    Tr.update(1, id[v], id[u], k);
}
ll query_path(int u, int v) {
    ll res_sum = 0, res_psum = 0;
    while (top[u] != top[v]) {
        if (dep[top[u]] < dep[top[v]]) swap(u, v);
        res_sum = (res_sum + Tr.query_sum(1, id[top[u]], id[u])) % mod;
        res_psum = (res_psum + Tr.query_psum(1, id[top[u]], id[u])) % mod;
        u = fa[top[u]];
    }
    if (dep[u] < dep[v]) swap(u, v);
    res_sum = (res_sum + Tr.query_sum(1, id[v], id[u])) % mod;
    res_psum = (res_psum + Tr.query_psum(1, id[v], id[u])) % mod;
    return (res_sum * res_sum % mod - res_psum + mod) % mod * inv2 % mod;
}
//Tree

void solve() {
    inv2 = qmi(2, mod - 2);
    cin >> n >> m;
    for (int i = 1; i <= n; ++i) cin >> a[i];
    memset(h, -1, sizeof h);
    for (int i = 1; i <= n - 1; ++i) {
        int u, v;
        cin >> u >> v;
        add(u, v);
        add(v, u);
    }
    dfs1(1, 0, 1);
    dfs2(1, 1);
    Tr.build(1, 1, n);

    while (m--) {
```

```cpp
        int op;
        cin >> op;
        ll u, v, k;
        if (op == 1) {
            cin >> u >> k;
            Tr.update(1, id[u], id[u] + sz[u] - 1, k);
        } else if (op == 2) {
            cin >> u >> v >> k;
            update_path(u, v, k);
        } else {
            cin >> u >> v;
            cout << query_path(u, v) << endl;
        }
    }
}

signed main() {
    ios;
    //freopen("sample.txt", "r", stdin);
    //freopen("resout.txt", "w", stdout);
    int t = 1;
    //cin >> t;
    while (t--) {
        solve();
    }
    return 0;
}
```

dsu

- 树上数颜色

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

const int N = 2e5 + 5;

int n;

// g[u]: 存储与 u 相邻的结点
vector<int> g[N];

// sz: 子树大小
// big: 重儿子
// col: 结点颜色
// L[u]: 结点 u 的 DFS 序
// R[u]: 结点 u 子树中结点的 DFS 序的最大值
// Node[i]: DFS 序为 i 的结点
// ans: 存答案
// cnt[i]: 颜色为 i 的结点个数
// totColor: 目前出现过的颜色个数
int sz[N], big[N], col[N], L[N], R[N], Node[N], totdfn;
int ans[N], cnt[N], totColor;

void add(int u) {
  if (cnt[col[u]] == 0) ++totColor;
  cnt[col[u]]++;
}

void del(int u) {
  cnt[col[u]]--;
  if (cnt[col[u]] == 0) --totColor;
}

int getAns() { return totColor; }

void dfs0(int u, int p) {
  L[u] = ++totdfn;
  Node[totdfn] = u;
  sz[u] = 1;
  for (int v : g[u])
```

```
    if (v != p) {
      dfs0(v, u);
      sz[u] += sz[v];
      if (!big[u] || sz[big[u]] < sz[v]) big[u] = v;
    }
  R[u] = totdfn;
}

void dfs1(int u, int p, bool keep) {
  // 计算轻儿子的答案
  for (int v : g[u])
    if (v != p && v != big[u]) {
      dfs1(v, u, false);
    }
  // 计算重儿子答案并保留计算过程中的数据（用于继承）
  if (big[u]) {
    dfs1(big[u], u, true);
  }
  for (int v : g[u])
    if (v != p && v != big[u]) {
      // 子树结点的 DFS 序构成一段连续区间，可以直接遍历
      for (int i = L[v]; i <= R[v]; i++) {
        add(Node[i]);
      }
    }
  add(u);
  ans[u] = getAns();
  if (keep == false) {
    for (int i = L[u]; i <= R[u]; i++) {
      del(Node[i]);
    }
  }
}

int main() {
  scanf("%d", &n);
  for (int i = 1; i <= n; i++) scanf("%d", &col[i]);
  for (int i = 1; i < n; i++) {
    int u, v;
    scanf("%d%d", &u, &v);
    g[u].push_back(v);
    g[v].push_back(u);
  }
  dfs0(1, 0);
  dfs1(1, 0, false);
  for (int i = 1; i <= n; i++) printf("%d%c", ans[i], " \n"[i == n]);
  return 0;
}
```

- 子树权值不大于 k 的数量

```
#include <bits/stdc++.h>
#define endl '\n'
#define pll pair<ll, ll>
#define tll tuple<ll, ll, ll>
#define vi vector<int>
#define vl vector<ll>
#define x first
#define y second
#define rep(i, j, k) for(int i = (j); i <= (k); i++)
#define per(i, j ,k) for(int i = (j); i >= (k); i--)
#define ios ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
typedef long long ll;
const ll N = 1e6 + 10;
const ll mod = 998244353;
const ll inf = 0x3f3f3f3f;
int tr[N];
int h[N], to[2 * N], ne[2 * N], cnt;
int sz[N], dep[N], fa[N], son[N];
int top[N], dfn, L[N], R[N], idx[N], skp;
int a[N], sum, ans[N];
int n, m;

vector<pll> q[N];
int lowbit(int x) {return x & -x;}
void add(int p, int k){for (int i = p; i < N; i += lowbit(i)) tr[i] += k;}
int query(int p){int res = 0; for (int i = p; i; i -= lowbit(i)) res += tr[i]; return res;}

void addedge(int u, int v){
    to[++cnt] = v;
    ne[cnt] = h[u];
    h[u] = cnt;
}

void dfs1(int u, int f){
    sz[u] = 1;
    dep[u] = dep[f] + 1;
    fa[u] = f;
    for (int i = h[u]; i; i = ne[i]){
        int v = to[i];
        if (v == f) continue;
        dfs1(v, u);
        sz[u] += sz[v];
        if (!son[u] || sz[son[u]] < sz[v]) son[u] = v;
    }
}

void dfs2(int u, int t){
    L[u] = ++dfn;
    idx[dfn] = u;
    top[u] = t;
    if (son[u]) dfs2(son[u], t);
    for (int i = h[u]; i; i = ne[i]){
        int v = to[i];
        if (v != fa[u] && v != son[u])
            dfs2(v, v);
    }
    R[u] = dfn;
}

void get(int u, int op){
    for (int i = L[u]; i <= R[u]; ++i){
        if (idx[i] == skp){i = R[idx[i]]; continue;}
        add(a[idx[i]], op);
    }
    if (op == -1) return;
    for (auto x : q[u]) ans[x.second] = query(x.first);
}

void dsu(int u){
    for (int i = h[u]; i; i = ne[i]){
        int v = to[i];
        if (v == fa[u] || v == son[u]) continue;
        dsu(v);
    }
    if (son[u]) {dsu(son[u]), skp = son[u];}
    get(u, 1);
    if (u == top[u]){
        skp = 0;
        get(u, -1);
    }
}
void solve() {
    cin >> n;
    rep(i, 1, n) cin >> a[i];
    int u, v;
    for (int i = 1; i <= n - 1; ++i){
        cin >> u >> v;
        addedge(u, v);
        addedge(v, u);
    }
    cin >> m;
```

```cpp
    int x, k;
    rep(i, 1, m){
        cin >> x >> k;
        q[x].push_back({k, i});
    }
    dfs1(1, 0);
    dfs2(1, 1);
    dsu(1);
    for (int i = 1; i <= m; ++i) cout << ans[i] << endl;
}

int main() {
    ios;
    //freopen("sample.txt", "r", stdin);
    //freopen("resout.txt", "w", stdout);
    int t = 1;
    //cin >> t;
    while (t--) {
        solve();
    }
    return 0;
}
```

- 子树查询类问题

现在将会问你 m 个问题。对于每个问题，它将会给你三个参数 x,l,r 表示询问以 x 为根的子树中，节点深度在该子树中不小于 l 且不大于 r 的所有节点。你需要告诉智乃酱三个信息，所有符合条件节点的最小值，最大值，以及它们的和。

```cpp
#include <bits/stdc++.h>
#define endl '\n'
#define pll pair<ll, ll>
#define tll tuple<ll, ll, ll>
#define vi vector<int>
#define vl vector<ll>
#define x first
#define y second
#define rep(i, j, k) for(int i = (j); i <= (k); i++)
#define per(i, j ,k) for(int i = (j); i >= (k); i--)
#define ios ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
typedef long long ll;
const ll maxn = 1e5 + 10;
const ll mod = 998244353;
const ll inf32 = 1e9;
const ll inf64 = 2e18;

int tot, h[maxn], len[maxn], L[maxn], R[maxn], fa[maxn], son[maxn], dfn, n, m, x, l, r, u, v;
ll val[maxn];

struct node {
    ll Sum, Max, Min;
} ans[maxn];

struct qnode {
    int id;
    int l, r;
    qnode(int _id = 0, int _l = 0, int _r = 0) {id = _id, l = _l, r = _r;}
};

struct edges {
    int to, next;
} e[2 * maxn];
vector<qnode> lis[maxn];

struct tnode
{
    ll Sum, Max, Min;
    int l, r;
};
tnode operator + (const tnode &a, const tnode &b)
{
    tnode c;
    c.l = a.l;
    c.r = b.r;
    c.Sum = a.Sum + b.Sum;
    c.Max = max(a.Max, b.Max);
    c.Min = min(a.Min, b.Min);
    return c;
}

struct Segment_Tree
{
    tnode t[4 * maxn];
    int mp[maxn];
    void update (int root)
    {
        int ch = root << 1;
        t[root] = t[ch] + t[ch + 1];
    }
    void buildt(int root, int l, int r)
    {
        t[root].l = l;
        t[root].r = r;
        if (l != r)
        {
            int mid = (l + r) >> 1;
            int ch = root << 1;
            buildt(ch, l, mid);
            buildt(ch + 1, mid + 1, r);
            update(root);
        }
        else
        {
            mp[l] = root;
            t[root].Max = -inf64;
            t[root].Min = inf64;
            t[root].Sum = 0;
        }
    }
    void change(int pos, long long delta, long long nmax, long long nmin)
    {
        int root = mp[pos];
        t[root].Sum += delta;
        t[root].Max = max(t[root].Max, nmax);
        t[root].Min = min(t[root].Min, nmin);
        while (root >>= 1)update(root);
    }
    tnode getdata(int pos)
    {
        return t[mp[pos]];
    }
    tnode getseg(int root, int l, int r)
    {
        if (t[root].l == l && t[root].r == r)
        {
            return t[root];
        }
        int mid = (t[root].l + t[root].r) >> 1;
        int ch = root << 1;
        if (r <= mid)return getseg(ch, l, r);
        else if (l > mid)return getseg(ch + 1, l, r);
        else return getseg(ch, l, mid) + getseg(ch + 1, mid + 1, r);
    }
};
Segment_Tree ST;

void add_edge(int u, int to)
{
    e[++tot].to = to;
    e[tot].next = h[u];
    h[u] = tot;
    return;
```

```
}


void dfs1(int x, int father)
{
    fa[x] = father;
    for (int i = h[x]; i; i = e[i].next)
    {
        if (e[i].to != father)
        {
            dfs1(e[i].to, x);
            if (!son[x] || len[son[x]] < len[e[i].to])son[x] = e[i].to;
        }
    }
    len[x] = len[son[x]] + 1;
    return;
}

void dfs2(int x)
{
    L[x] = ++dfn;
    R[x] = L[x] + len[x] - 1;
    if (son[x])dfs2(son[x]);
    for (int i = h[x]; i; i = e[i].next)
    {
        if (e[i].to != fa[x] && e[i].to != son[x])
        {
            dfs2(e[i].to);
        }
    }
    return;
}

void dfs(int x)
{
    if (son[x])
    {
        dfs(son[x]);
    }
    for (int i = h[x]; i; i = e[i].next)
    {
        if (e[i].to != fa[x] && e[i].to != son[x])
        {
            dfs(e[i].to);
            for (int j = L[e[i].to], k = 1; j <= R[e[i].to]; ++j, ++k)
            {
                tnode temp = ST.getdata(j);
                ST.change(L[x] + k, temp.Sum, temp.Max, temp.Min);
            }
        }
    }
    ST.change(L[x], val[x], val[x], val[x]);
    for (auto &i : lis[x])
    {
        tnode temp = ST.getseg(1, L[x] + i.l, L[x] + i.r);
        ans[i.id].Sum = temp.Sum;
        ans[i.id].Max = temp.Max;
        ans[i.id].Min = temp.Min;
    }
    return;
}

void solve() {
    cin >> n;
    for (int i = 1; i <= n; ++i) cin >> val[i];
    for (int i = 1; i <= n - 1; ++i) {
        cin >> u >> v;
        add_edge(u, v);
        add_edge(v, u);
    }
    dfs1(1, 0);
    dfs2(1);
    ST.buildt(1, 1, n);
    cin >> m;
    for (int i = 1; i <= m; ++i) {
        cin >> x >> l >> r;
        lis[x].push_back(qnode(i, l, r));
    }
    dfs(1);
    for (int i = 1; i <= m; ++i)
    {
        cout << ans[i].Min << " " <<  ans[i].Max << " " << ans[i].Sum << endl;
    }
}

int main() {
    ios;
    //freopen("sample.txt", "r", stdin);
    //freopen("resout.txt", "w", stdout);
    int t = 1;
    //cin >> t;
    while (t--) {
        solve();
    }
    return 0;
}

```

- 小 Q 与树

u ∑ v ∑ min(a[u], a[v]) * dis(u, v)

```
#include <bits/stdc++.h>
#define endl '\n'
#define pll pair<ll, ll>
#define tll tuple<ll, ll, ll>
#define vi vector<int>
#define vl vector<ll>
#define int ll
#define rep(i, j, k) for(int i = (j); i <= (k); i++)
#define per(i, j ,k) for(int i = (j); i >= (k); i--)
#define ios ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
typedef long long ll;
const ll maxn = 2e5 + 10;
const ll mod = 998244353;
const ll inf = 0x3f3f3f3f;

int n, h[maxn], to[maxn << 1], nxt[maxn << 1], cnt = 0;
int sz[maxn], son[maxn], dep[maxn], L[maxn], R[maxn], f[maxn], idx[maxn], top[maxn], dfn = 0;
ll sum[maxn], ans = 0;
struct node {
    int x, id;
} a[maxn];

void add(int u, int v) {
    to[++cnt] = v;
    nxt[cnt] = h[u];
    h[u] = cnt;
}

void dfs1(int u, int fa) {
    f[u] = fa;
    dep[u] = dep[fa] + 1;
    sz[u] = 1;
    for (int i = h[u]; i; i = nxt[i]) {
        int v = to[i];
        if (v == fa) continue;
        dfs1(v, u);
        sz[u] += sz[v];
        if (sz[v] > sz[son[u]])
            son[u] = v;
    }
```

```cpp
}

void dfs2(int u, int t) {
    top[u] = t;
    L[u] = ++dfn;
    idx[dfn] = u;
    if (son[u]) dfs2(son[u], t);
    for (int i = h[u]; i; i = nxt[i]) {
        int v = to[i];
        if (v == f[u] || v == son[u]) continue;
        dfs2(v, v);
    }
    R[u] = dfn;
}

struct Segment {
    struct Node {
        int l, r;
        int sum, add;
    } tr[maxn * 4];
    void pushup(int u) {
        tr[u].sum = tr[u << 1].sum + tr[u << 1 | 1].sum;
    }
    void pushdown(int u) {
        if (tr[u].add) {
            int x = tr[u].add;
            tr[u << 1].sum += (tr[u << 1].r - tr[u << 1].l + 1) * x;
            tr[u << 1 | 1].sum += (tr[u << 1 | 1].r - tr[u << 1 | 1].l + 1) * x;
            tr[u << 1].add += x;
            tr[u << 1 | 1].add += x;
            tr[u].add = 0;
        }
        return;
    }
    void build(int u, int l, int r) {
        tr[u] = {l, r};
        if (l == r) return;
        int mid = (l + r) >> 1;
        build(u << 1, l, mid);
        build(u << 1 | 1, mid + 1, r);
        return;
    }
    void modify(int u, int l, int r, int x) {
        if (l <= tr[u].l && tr[u].r <= r) {
            tr[u].add += x;
            tr[u].sum += (tr[u].r - tr[u].l + 1) * x;
            return;
        }
        pushdown(u);
        int mid = (tr[u].l + tr[u].r) >> 1;
        if (l <= mid) modify(u << 1, l, r, x);
        if (mid < r) modify(u << 1 | 1, l, r, x);
        pushup(u);
        return;
    }
    int query(int u, int l, int r) {
        if (l <= tr[u].l && tr[u].r <= r) return tr[u].sum;
        pushdown(u);
        int mid = (tr[u].l + tr[u].r) >> 1;
        int res = 0;
        if (l <= mid) res += query(u << 1, l, r);
        if (mid < r) res += query(u << 1 | 1, l, r);
        return res;
    }
} Tr;

void tree_add(int x, int y) {
    while (top[x] != top[y]) {
        if (dep[top[x]] < dep[top[y]]) swap(x, y);
        Tr.modify(1, L[top[x]], L[x], 1);
        x = f[top[x]];
    }
    if (dep[x] < dep[y]) swap(x, y);
    Tr.modify(1, L[y], L[x], 1);
    return;
}

int tree_sum(int x, int y) {
    int res = 0;
    while (top[x] != top[y]) {
        if (dep[top[x]] < dep[top[y]]) swap(x, y);
        res += Tr.query(1, L[top[x]], L[x]);
        x = f[top[x]];
    }
    if (dep[x] < dep[y]) swap(x, y);
    res += Tr.query(1, L[y], L[x]);
    return res;
}

void solve() {
    cin >> n;
    rep(i, 1, n) {
        cin >> a[i].x;
        a[i].id = i;
    }
    sort(a + 1, a + n + 1, [&](node p, node q) {return p.x > q.x;});
    rep(i, 1, n - 1) {
        int u, v;
        cin >> u >> v;
        add(u, v);
        add(v, u);
    }
    dfs1(1, 0);
    dfs2(1, 1);
    Tr.build(1, 1, n);
    rep(i, 1, n) sum[i] = sum[i - 1] + dep[a[i].id];
    rep(i, 1, n){
        int x = a[i].id;
        ans = (ans + (ll)dep[x] * (i - 1) * a[i].x % mod + sum[i - 1] * a[i].x % mod - 2ll * a[i].x * tree_sum(1, x) % mod + mod) % mod;
        tree_add(1, x);
    }
    cout << ans * 2ll % mod << endl;
}
signed main() {
    ios;
    //freopen("sample.txt", "r", stdin);
    //freopen("resout.txt", "w", stdout);
    int t = 1;
    //cin >> t;
    while (t--) {
        solve();
    }
    return 0;
}
```

计算几何

点线

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

typedef double db;
const db EPS = 1e-9;

inline int sign(db a) {
    return a < -EPS ? -1 : a > EPS;
}

inline int cmp(db a, db b) {
    return sign(a - b);
}

struct P {
    db x, y;
    P() {}
    P(db _x, db _y) : x(_x), y(_y) {}
    //重构加减乘除
    P operator+(P p) { return {x + p.x, y + p.y}; }
    P operator-(P p) { return {x - p.x, y - p.y}; }
    P operator*(db d) { return {x * d, y * d}; }
    P operator/(db d) { return {x / d, y / d}; }

    bool operator<(P p) const {
        int c = cmp(x, p.x);
        if (c)
            return c == -1;
        return cmp(y, p.y) == -1;
    }

    bool operator==(P o) const { return cmp(x, o.x) == 0 && cmp(y, o.y) == 0; }

    db dot(P p) { return x * p.x + y * p.y; }  //点积
    db det(P p) { return x * p.y - y * p.x; }  //叉积

    db distTo(P p) { return (*this - p).abs(); }
    db alpha() { return atan2(y, x); }
    void read() { cin >> x >> y; }
    void write() { cout << "(" << x << "," << y << ")" << endl; }
    db abs() { return sqrt(abs2()); }
    db abs2() { return x * x + y * y; }
    P rot90() { return P(-y, x); }
    P unit() { return *this / abs(); }
    int quad() const { return sign(y) == 1 || (sign(y) == 0 && sign(x) >= 0); }
    P rot(db an) {
        return {x * cos(an) - y * sin(an), x * sin(an) + y * cos(an)};
    }
};

#define cross(p1, p2, p3) \
    ((p2.x - p1.x) * (p3.y - p1.y) - (p3.x - p1.x) * (p2.y - p1.y))
#define crossOp(p1, p2, p3) sign(cross(p1, p2, p3))

// 直线 p1p2, q1q2 是否恰有一个交点
bool chkLL(P p1, P p2, P q1, P q2) {
    db a1 = cross(q1, q2, p1), a2 = -cross(q1, q2, p2);
    return sign(a1 + a2) != 0;
}

// 求直线 p1p2, q1q2 的交点
P isLL(P p1, P p2, P q1, P q2) {
    db a1 = cross(q1, q2, p1), a2 = -cross(q1, q2, p2);
    return (p1 * a2 + p2 * a1) / (a1 + a2);
}

// 判断区间 [l1, r1], [l2, r2] 是否相交
bool intersect(db l1, db r1, db l2, db r2) {
    if (l1 > r1)
        swap(l1, r1);
    if (l2 > r2)
        swap(l2, r2);
    return !(cmp(r1, l2) == -1 || cmp(r2, l1) == -1);
}

// 线段 p1p2, q1q2 相交
bool isSS(P p1, P p2, P q1, P q2) {
    return intersect(p1.x, p2.x, q1.x, q2.x) &&
           intersect(p1.y, p2.y, q1.y, q2.y) &&
           crossOp(p1, p2, q1) * crossOp(p1, p2, q2) <= 0 &&
           crossOp(q1, q2, p1) * crossOp(q1, q2, p2) <= 0;
}
```

```cpp

// 线段 p1p2, q1q2 严格相交
bool isSS_strict(P p1, P p2, P q1, P q2) {
    return crossOp(p1, p2, q1) * crossOp(p1, p2, q2) < 0 &&
           crossOp(q1, q2, p1) * crossOp(q1, q2, p2) < 0;
}

// m 在 a 和 b 之间
bool isMiddle(db a, db m, db b) {
    /*if (a > b) swap(a, b);
    return cmp(a, m) <= 0 && cmp(m, b) <= 0;*/
    return sign(a - m) == 0 || sign(b - m) == 0 || (a < m != b < m);
}

bool isMiddle(P a, P m, P b) {
    return isMiddle(a.x, m.x, b.x) && isMiddle(a.y, m.y, b.y);
}

// 点 p 在线段 p1p2 上
bool onSeg(P p1, P p2, P q) {
    return crossOp(p1, p2, q) == 0 && isMiddle(p1, q, p2);
}
// q1q2 和 p1p2 的交点 在 p1p2 上?

// 点 p 严格在 p1p2 上
bool onSeg_strict(P p1, P p2, P q) {
    return crossOp(p1, p2, q) == 0 &&
           sign((q - p1).dot(p1 - p2)) * sign((q - p2).dot(p1 - p2)) < 0;
}

// 求 q 到 直线 p1p2 的投影（垂足） ⊠ : p1 != p2
P proj(P p1, P p2, P q) {
    P dir = p2 - p1;
    return p1 + dir * (dir.dot(q - p1) / dir.abs2());
}

// 求 q 以 直线 p1p2 为轴的反射
P reflect(P p1, P p2, P q) {
    return proj(p1, p2, q) * 2 - q;
}

// 求 q 到 线段 p1p2 的最小距离
db nearest(P p1, P p2, P q) {
    if (p1 == p2)
        return p1.distTo(q);
    P h = proj(p1, p2, q);
    if (isMiddle(p1, h, p2))
        return q.distTo(h);
    return min(p1.distTo(q), p2.distTo(q));
}

// 求 线段 p1p2 与 线段 q1q2 的距离
db disSS(P p1, P p2, P q1, P q2) {
    if (isSS(p1, p2, q1, q2))
        return 0;
    return min(min(nearest(p1, p2, q1), nearest(p1, p2, q2)),
               min(nearest(q1, q2, p1), nearest(q1, q2, p2)));
}

// 极角排序
sort(p, p + n, [&](P a, P b) {
    int qa = a.quad(), qb = b.quad();
    if (qa != qb)
        return qa < qb;
    else
        return sign(a.det(b)) > 0;
});
```

多边形

```cpp
#include <bits/stdc++.h>
using namespace std;
#define rep(i, a, n) for (int i = a; i < n; i++)
typedef double db;
const db EPS = 1e-9;

//求多边形面积
db area(vector<P> ps) {
    db ret = 0;
    rep(i, 0, ps.size()) ret += ps[i].det(ps[(i + 1) % ps.size()]);
    return ret / 2;
}
//点包含
int contain(vector<P> ps, P p) {  // 2:inside,1:on_seg,0:outside
    int n = ps.size(), ret = 0;
    rep(i, 0, n) {
        P u = ps[i], v = ps[(i + 1) % n];
        if (onSeg(u, v, p))
            return 1;
        if (cmp(u.y, v.y) <= 0)
            swap(u, v);
        if (cmp(p.y, u.y) > 0 || cmp(p.y, v.y) <= 0)
            continue;
        ret ^= crossOp(p, u, v) > 0;
    }
    return ret * 2;
}
//严格凸包
vector<P> convexHull(vector<P> ps) {
    int n = ps.size();
    if (n <= 1)
        return ps;
    sort(ps.begin(), ps.end());
    vector<P> qs(n * 2);
    int k = 0;
    for (int i = 0; i < n; qs[k++] = ps[i++])
        while (k > 1 && crossOp(qs[k - 2], qs[k - 1], ps[i]) <= 0)
            --k;
    for (int i = n - 2, t = k; i >= 0; qs[k++] = ps[i--])
        while (k > t && crossOp(qs[k - 2], qs[k - 1], ps[i]) <= 0)
            --k;
    qs.resize(k - 1);
    return qs;
}

//不严格凸包
vector<P> convexHullNonStrict(vector<P> ps) {
    // caution: need to unique the Ps first
    int n = ps.size();
    if (n <= 1)
        return ps;
    sort(ps.begin(), ps.end());
    vector<P> qs(n * 2);
    int k = 0;
    for (int i = 0; i < n; qs[k++] = ps[i++])
        while (k > 1 && crossOp(qs[k - 2], qs[k - 1], ps[i]) < 0)
            --k;
    for (int i = n - 2, t = k; i >= 0; qs[k++] = ps[i--])
        while (k > t && crossOp(qs[k - 2], qs[k - 1], ps[i]) < 0)
            --k;
    qs.resize(k - 1);
    return qs;
}
//旋转卡壳
db convexDiameter(vector<P> ps) {
    int n = ps.size();
    if (n <= 1)
        return 0;
    int is = 0, js = 0;
    rep(k, 1, n) is = ps[k] < ps[is] ? k : is, js = ps[js] < ps[k] ? k : js;
    int i = is, j = js;
    db ret = ps[i].distTo(ps[j]);
    do {
        if ((ps[(i + 1) % n] - ps[i]).det(ps[(j + 1) % n] - ps[j]) >= 0)
            (++j) %= n;
        else
            (++i) %= n;
        ret = max(ret, ps[i].distTo(ps[j]));
    } while (i != is || j != js);
    return ret;
}

//切多边形
vector<P> convexCut(const vector<P>& ps, P q1, P q2) {
    vector<P> qs;
    int n = ps.size();
    rep(i, 0, n) {
        P p1 = ps[i], p2 = ps[(i + 1) % n];
        int d1 = crossOp(q1, q2, p1), d2 = crossOp(q1, q2, p2);
        if (d1 >= 0)
            qs.push_back(p1);
        if (d1 * d2 < 0)
            qs.push_back(isLL(p1, p2, q1, q2));
    }
    return qs;
}
```

圆

```cpp
#define rep(i, a, n) for (int i = a; i < n; i++)
const double PI = acos(-1.0);

//判断两个圆的关系
int type(P o1, db r1, P o2, db r2) {
    db d = o1.distTo(o2);
    if (cmp(d, r1 + r2) == 1)
        return 4;
    if (cmp(d, r1 + r2) == 0)
        return 3;
    if (cmp(d, abs(r1 - r2)) == 1)
        return 2;
    if (cmp(d, abs(r1 - r2)) == 0)
        return 1;
    return 0;
}
//圆和线相交
vector<P> isCL(P o, db r, P p1, P p2) {
    if (cmp(abs((o - p1).det(p2 - p1) / p1.distTo(p2)), r) > 0)
        return {};
    db x = (p1 - o).dot(p2 - p1), y = (p2 - p1).abs2(),
       d = x * x - y * ((p1 - o).abs2() - r * r);
    d = max(d, (db)0.0);
    P m = p1 - (p2 - p1) * (x / y), dr = (p2 - p1) * (sqrt(d) / y);
    return {m - dr, m + dr};  // along dir: p1->p2
}

//两个圆相交的交点
vector<P> isCC(P o1,
               db r1,
               P o2,
               db r2) {  // need to check whether two circles are the same
    db d = o1.distTo(o2);
    if (cmp(d, r1 + r2) == 1)
        return {};
    if (cmp(d, abs(r1 - r2)) == -1)
        return {};
    d = min(d, r1 + r2);
    db y = (r1 * r1 + d * d - r2 * r2) / (2 * d), x = sqrt(r1 * r1 - y * y);
    P dr = (o2 - o1).unit();
    P q1 = o1 + dr * y, q2 = dr.rot90() * x;
    return {q1 - q2, q1 + q2};  // along circle 1
}

//求切线，默认求外公切线，求内公切线的话，r2 改成负数，求点到圆的切线,r2 改成 0
// extanCC, intanCC : -r2, tanCP : r2 = 0
vector<pair<P, P>> tanCC(P o1, db r1, P o2, db r2) {
    P d = o2 - o1;
    db dr = r1 - r2, d2 = d.abs2(), h2 = d2 - dr * dr;
    if (sign(d2) == 0 || sign(h2) < 0)
        return {};
    h2 = max((db)0.0, h2);
    vector<pair<P, P>> ret;
    for (db sign : {-1, 1}) {
        P v = (d * dr + d.rot90() * sqrt(h2) * sign) / d2;
        ret.push_back({o1 + v * r1, o2 + v * r2});
    }
    if (sign(h2) == 0)
        ret.pop_back();
    return ret;
}

db rad(P p1, P p2) {
    return atan2l(p1.det(p2), p1.dot(p2));
}
//圆和三角形的面积交
db areaCT(db r, P p1, P p2) {
    vector<P> is = isCL(P(0, 0), r, p1, p2);
    if (is.empty())
        return r * r * rad(p1, p2) / 2;
    bool b1 = cmp(p1.abs2(), r * r) == 1, b2 = cmp(p2.abs2(), r * r) == 1;
    if (b1 && b2) {
        P md = (is[0] + is[1]) / 2;
        if (sign((p1 - md).dot(p2 - md)) <= 0)
            return r * r * (rad(p1, is[0]) + rad(is[1], p2)) / 2 +
                   is[0].det(is[1]) / 2;
        else
            return r * r * rad(p1, p2) / 2;
    }
    if (b1)
        return (r * r * rad(p1, is[0]) + is[0].det(p2)) / 2;
    if (b2)
        return (p1.det(is[1]) + r * r * rad(is[1], p2)) / 2;
    return p1.det(p2) / 2;
}

//内心
P inCenter(P A, P B, P C) {
    double a = (B - C).abs(), b = (C - A).abs(), c = (A - B).abs();
    return (A * a + B * b + C * c) / (a + b + c);
}
//外心
P circumCenter(P a, P b, P c) {
    P bb = b - a, cc = c - a;
    double db = bb.abs2(), dc = cc.abs2(), d = 2 * bb.det(cc);
    return a - P(bb.y * dc - cc.y * db, cc.x * db - bb.x * dc) / d;
}
//垂心
P othroCenter(P a, P b, P c) {
    P ba = b - a, ca = c - a, bc = b - c;
    double Y = ba.y * ca.y * bc.y, A = ca.x * ba.y - ba.x * ca.y,
           x0 = (Y + ca.x * ba.y * b.x - ba.x * ca.y * c.x) / A,
           y0 = -ba.x * (x0 - c.x) / ba.y + ca.y;
    return {x0, y0};
}

//最小圆覆盖, 随机增量法
pair<P, db> min_circle(vector<P> ps) {
    random_device rd;
    mt19937 rng(rd());
    shuffle(ps.begin(), ps.end(), rng);
    // random_shuffle(ps.begin(), ps.end());
    int n = ps.size();
    P o = ps[0];
    db r = 0;
    rep(i, 1, n) if (o.distTo(ps[i]) > r + EPS) {
        o = ps[i], r = 0;
        rep(j, 0, i) if (o.distTo(ps[j]) > r + EPS) {
            o = (ps[i] + ps[j]) / 2;
            r = o.distTo(ps[i]);
            rep(k, 0, j) if (o.distTo(ps[k]) > r + EPS) {
                o = circumCenter(ps[i], ps[j], ps[k]);
                r = o.distTo(ps[i]);
            }
        }
    }
    return {o, r};
}

db intergal(db x, db y, db r, db L, db R) {
    return r * r * (R - L) + x * r * (sinl(R) - sinl(L)) +
           y * r * (-cosl(R) + cosl(L));
}

db calc_area_circle(P c, db r, db L, db R) {
    return intergal(c.x, c.y, r, L, R) / 2;
}

db norm(db x) {
    while (x < 0)
        x += 2 * PI;
    while (x > 2 * PI)
        x -= 2 * PI;
    return x;
}



//圆面积并
P cs[N];
db rs[N];

void work() {
    vector<int> cand = {};
    rep(i, 0, n) {
        bool ok = 1;
        rep(j, 0, n) if (i != j) {
            if (rs[j] > rs[i] + EPS &&
                rs[i] + cs[i].distTo(cs[j]) <= rs[j] + EPS) {
                ok = 0;
                break;
            }
            if (cs[i] == cs[j] && cmp(rs[i], rs[j]) == 0 && j < i) {
                ok = 0;
                break;
            }
        }
        if (ok)
            cand.push_back(i);
    }

    rep(i, 0, cand.size()) cs[i] = cs[cand[i]], rs[i] = rs[cand[i]];
    n = cand.size();

    db area = 0;

    // work
    rep(i, 0, n) {
        vector<pair<db, int>> ev = {{0, 0}, {2 * PI, 0}};

        int cur = 0;

        rep(j, 0, n) if (j != i) {
            auto ret = isCC(cs[i], rs[i], cs[j], rs[j]);
            if (!ret.empty()) {
                db l = (ret[0] - cs[i]).alpha();
                db r = (ret[1] - cs[i]).alpha();
                l = norm(l);
                r = norm(r);
                ev.push_back({l, 1});
                ev.push_back({r, -1});
                if (l > r)
                    ++cur;
            }
        }

        sort(ev.begin(), ev.end());
        rep(j, 0, ev.size() - 1) {
            cur += ev[j].se;
            if (cur == 0) {
                area += calc_area_circle(cs[i], rs[i], ev[j].fi, ev[j + 1].fi);
            }
        }
    }
}
```

字符串

字符串哈希

- 取双模

```
#include<bits/stdc++.h>
#include<unordered_map>
#define debug cout << "debug---  "
#define debug_ cout << "\n---debug---\n"
#define oper(a) operator<(const a& ee)const
#define forr(a,b,c) for(int a=b;a<=c;a++)
#define mem(a,b) memset(a,b,sizeof a)
#define cinios (ios::sync_with_stdio(false),cin.tie(0),cout.tie(0))
#define all(a) a.begin(),a.end()
#define sz(a) (int)a.size()
#define endl "\n"
#define ul (u << 1)
#define ur (u << 1 | 1)
using namespace std;

typedef unsigned long long ull;
typedef long long ll;
typedef pair<ll, int> PII;

const int N = 1e5 + 10, M = 2e6 + 10, mod = 1e9 + 7;
int INF = 0x3f3f3f3f; ll LNF = 0x3f3f3f3f3f3f3f3f;
int n, m, B = 10, ki;

const int mod1 = 1e9 + 9;

ll p1[N], P1 = 131, p2[N], P2 = 13331;
ll h[N], h2[N];
//乘法开 ll, mod 取 int

int get1(int l, int r) {
    return (h[r] - (h[l - 1] * p1[r - l + 1]) % mod + mod) % mod;
}
int get2(int l, int r) {
    return (h2[r] - (h2[l - 1] * p2[r - l + 1]) % mod1 + mod1) % mod1;
}

char str[N];

void solve() {
    cin >> n >> m;
    cin >> str + 1;
```

```cpp

    p1[0] = p2[0] = 1;

    for (int i = 1; i <= n; i++) {
        p1[i] = (p1[i - 1] * P1) % mod;
        p2[i] = (p2[i - 1] * P2) % mod1;

        h[i] = ((h[i - 1] * P1) % mod + str[i] - '0' + 1) % mod;
        h2[i] = ((h2[i - 1] * P2) % mod1 + str[i] - '0' + 1) % mod1;
    }

    while (m--)
    {
        int l1, r1, l2, r2;
        cin >> l1 >> r1 >> l2 >> r2;
        if (get1(l1, r1) == get1(l2, r2) && get2(l1, r1) == get2(l2, r2)) cout << "Yes\n";
        else cout << "No\n";
    }
}

signed main() {
    cinios;
    int T = 1;
    for (int t = 1; t <= T; t++) {
        solve();
    }
    return 0;
}
```

KMP

- KMP 模板

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

using namespace std;

const int N = 1e6 + 10;

vector<int> prefix_function(string s)
{
    int n = (int)s.length();
    vector<int> pi(n);
    for (int i = 2; i < n; i++)
    {
        pi[i] = pi[i - 1];
        while (pi[i] && s[i] != s[pi[i] + 1])
            pi[i] = pi[pi[i]];
        pi[i] += (s[i] == s[pi[i] + 1]);
    }
    return pi;
}

int main(void)
{
    ios::sync_with_stdio(false), cin.tie(0), cout.tie(0);
    string s1, s2;
    cin >> s1 >> s2;
    s1 = " " + s1;
    s2 = " " + s2;
    auto nxt = prefix_function(s2);
    for (int i = 1, j = 0; i < s1.size(); i++)
    {
        while (j && s1[i] != s2[j + 1])
            j = nxt[j];
        if (s1[i] == s2[j + 1])
            j++;
        if (j == s2.size() - 1)
        {
            cout << i - j + 1 << "\n";
            j = nxt[j];
        }
    }
    for (int i = 1; i < s2.size(); i++)
        cout << nxt[i] << " ";

    return 0;
}
```

- carpet(二维 KMP) 有一个 n*m 的地毯，aij 表示地毯每格的元素，bij 表示地毯每格的价格，要求选取一块价格最大值最小的地毯，并且这块地毯无限铺开之后，原地毯是其子矩阵

```
#include <bits/stdc++.h>
#define endl '\n'
#define pll pair<ll, ll>
#define tll tuple<ll, ll, ll>
#define x first
#define y second
#define int ll
#define rep(i, j, k) for (int i = (j); i <= (k); i++)
#define per(i, j, k) for (int i = (j); i >= (k); i--)
#define ios ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
typedef long long ll;
const ll maxn = 1e6 + 10;
const ll mod = 998244353;
const ll inf = 0x3f3f3f3f;

vector<int> prefix_function(string s)
{
    int n = (int)s.length();
    vector<int> pi(n);
    for (int i = 2; i < n; i++)
    {
        pi[i] = pi[i - 1];
        while (pi[i] && s[i] != s[pi[i] + 1])
            pi[i] = pi[pi[i]];
        pi[i] += (s[i] == s[pi[i] + 1]);
    }
    return pi;
}

int get_length(vector<string> s)
{
    int len = s[1].size() - 1;
    int ret = len;
    vector<int> cnt(len + 1);
    for (int i = 1; i < s.size(); ++i)
    {
        string tmp = s[i];
        auto nxt = prefix_function(tmp);
        int j = len;
        while (j)
        {
            cnt[len - nxt[j]]++;
            j = nxt[j];
        }
    }
    for (int i = 1; i <= len; ++i)
        if (cnt[i] == s.size() - 1)
        {
            ret = i;
            break;
        }
    return ret;
}

void solve()
{
    int n, m;
    cin >> n >> m;
    vector<string> s1(n + 1);
```

```cpp
    for (int i = 1; i <= n; ++i)
        cin >> s1[i], s1[i] = " " + s1[i];
    vector<string> s2(m + 1);
    for (int i = 1; i <= m; ++i)
    {
        string tmp = " ";
        for (int j = 1; j <= n; ++j)
            tmp += s1[j][i];
        s2[i] = tmp;
    }
    vector<vector<int>> a(n + 1, vector<int>(m + 1, 0));
    for (int i = 1; i <= n; ++i)
        for (int j = 1; j <= m; ++j)
            cin >> a[i][j];
    int p = get_length(s1), q = get_length(s2);
    ll ans = 1e9;
    deque<int> dq;
    auto b = a;
    for (int i = 1; i <= n; ++i){
        while (dq.size()) dq.pop_back();
        for (int j = 1; j <= m; ++j){
            while (dq.size() && j - dq.front() + 1 > p) dq.pop_front();
            while (dq.size() && a[i][dq.back()] <= a[i][j]) dq.pop_back();
            dq.push_back(j);
            b[i][j] = a[i][dq.front()];
        }
    }
    for (int j = 1; j <= m; ++j){
        while (dq.size()) dq.pop_back();
        for (int i = 1; i <= n; ++i){
            while (dq.size() && i - dq.front() + 1 > q) dq.pop_front();
            while (dq.size() && b[dq.back()][j] <= b[i][j]) dq.pop_back();
            dq.push_back(i);
            if (i >= q && j >= p)
                ans = min(ans, 1ll * b[dq.front()][j]);
        }
    }
    ans = ans * (p + 1) * (q + 1);
    cout << ans << endl;
}

signed main()
{
    ios;
    // freopen("sample.txt", "r", stdin);
    // freopen("resout.txt", "w", stdout);
    int t = 1;
    // cin >> t;
    while (t--)
    {
        solve();
    }
    return 0;
}

```

Trie

- trie & topo

可自定字符间大小关系，求多少个串可以成为字典序最小的串

```cpp
#include <bits/stdc++.h>
#define endl '\n'
#define pll pair<ll, ll>
#define tll tuple<ll, ll, ll>
#define x first
#define y second
#define rep(i, j, k) for (int i = (j); i <= (k); i++)
#define per(i, j, k) for (int i = (j); i >= (k); i--)
#define ios ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
```

```cpp
using namespace std;
typedef long long ll;
const ll maxn = 3e4 + 10, maxm = (3e4 + 10) * 26;
const ll mod = 998244353;
const ll inf = 0x3f3f3f3f;

int tr[maxm][26], idx = 0;
bool vis[maxm];
vector<int> edge[26];
int in[26];

void insert(string s)
{
    int x = 0;
    for (auto op : s)
    {
        auto c = op - 'a';
        if (!tr[x][c])
            tr[x][c] = ++idx;
        x = tr[x][c];
    }
    vis[x] = 1;
}

bool query(string s)
{
    auto topo = [&](){
        queue<int> q;
        int cnt = 0;
        for (int i = 0; i < 26; ++i) if (!in[i]) q.push(i);
        while(!q.empty()){
            auto op = q.front();
            q.pop(), cnt++;
            for (auto v : edge[op]){
                if (!--in[v]) q.push(v);
            }
        }
        return cnt == 26;
    };

    int x = 0;
    for (int i = 0; i < s.size(); ++i){
        auto c = s[i] - 'a';
        for (int j = 0; j < 26; ++j){
            if (j == c || !tr[x][j]) continue;
            edge[c].push_back(j);
            in[j]++;
        }
        x = tr[x][c];
        if (vis[x] && i != s.size() - 1) return false;
    }
    return topo();
}

void solve()
{
    int n;
    cin >> n;
    vector<string> v(n + 1);
    for (int i = 1; i <= n; ++i)
    {
        cin >> v[i];
        insert(v[i]);
    }
    vector<string> res;
    for (int op = 1; op <= n; ++op)
    {
        for (int i = 0; i < 26; ++i)
            in[i] = 0, edge[i].clear();
        if (query(v[op]))
            res.push_back(v[op]);
    }
    cout << res.size() << endl;
    for (auto s : res)
        cout << s << endl;
}

int main()
{
    ios;
    // freopen("sample.txt", "r", stdin);
    // freopen("resout.txt", "w", stdout);
    int t = 1;
    //cin >> t;
    while (t--)
    {
        solve();
    }
    return 0;
}
```

01Trie

- 两数最大异或和

```cpp
#include <bits/stdc++.h>
#define endl '\n'
#define pll pair<ll, ll>
#define tll tuple<ll, ll, ll>
#define vi vector<int>
#define vl vector<ll>
#define x first
#define y second
#define rep(i, j, k) for(int i = (j); i <= (k); i++)
#define per(i, j ,k) for(int i = (j); i >= (k); i--)
#define ios ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
typedef long long ll;
const ll maxn = 2e5 + 10;
const ll maxm = maxn * 32;
const ll mod = 998244353;
const ll inf = 0x3f3f3f3f;

int tr[maxm][2], idx, n;

void insert(int x){
    int p = 0;
    for (int i = 31; i >= 0; --i){
        int c = x >> i & 1;
        if (!tr[p][c]) tr[p][c] = ++idx;
        p = tr[p][c];
    }
}

int query(int x){
    int res = 0, p = 0;
    for (int i = 31; i >= 0; --i){
        int c = x >> i & 1;
        if (tr[p][c ^ 1]){
            p = tr[p][c ^ 1];
            res += 1 << i;
        }else
            p = tr[p][c];
    }
    return res;
}

void solve(){
    cin >> n;
    int ans = 0;
    for (int i = 0; i < n; ++i){
        int x; cin >> x;
        ans = max(ans, query(x));
        insert(x);
    }
    cout << ans << endl;
}

int main(){
    ios;
    //freopen("sample.txt", "r", stdin);
    //freopen("resout.txt", "w", stdout);
    int t = 1;
    //cin >> t;
    while(t--){
        solve();
    }
    return 0;
}
```

- 区间异或最大值

```cpp
#include <bits/stdc++.h>
#define endl '\n'
#define pll pair<ll, ll>
#define tll tuple<ll, ll, ll>
#define vi vector<int>
#define vl vector<ll>
#define x first
#define y second
#define rep(i, j, k) for (int i = (j); i <= (k); i++)
#define per(i, j, k) for (int i = (j); i >= (k); i--)
#define ios ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
typedef long long ll;
const ll maxn = 2e5 + 10;
const ll maxm = maxn * 21;
const ll mod = 998244353;
const ll inf = 0x3f3f3f3f;

int a[maxn], s[maxn];
int tr[maxm][2], tot;

void insert(int x)
{
    int p = 0;
    for (int i = 20; i >= 0; --i)
    {
        int c = x >> i & 1;
        if (!tr[p][c])
            tr[p][c] = ++tot;
        p = tr[p][c];
    }
}

int query(int x)
{
    int p = 0, res = 0;
    for (int i = 20; i >= 0; --i)
    {
        int c = x >> i & 1;
        if (tr[p][!c])
        {
            p = tr[p][!c];
            res += 1 << i;
        }
        else
            p = tr[p][c];
    }
    return res;
}

map<int, int> mp;

```

```cpp
void solve()
{
    int n, l, r;
    cin >> n;
    insert(0);
    mp[0] = 0;
    int ans = -1;
    for (int i = 1;i <= n; ++i){
        cin >> a[i];
        s[i] = s[i - 1] ^ a[i];
        insert(s[i]);
        int tmp = query(s[i]);
        if (tmp > ans){
            ans = tmp;
            r = i;
            l = mp[tmp ^ s[i]] + 1;
        }
        mp[s[i]] = i;
    }
    cout << ans << " " << l << " " << r << endl;
}

int main()
{
    ios;
    // freopen("sample.txt", "r", stdin);
    // freopen("resout.txt", "w", stdout);
    int t = 1;
    // cin >> t;
    while (t--)
    {
        solve();
    }
    return 0;
}
```

- Border1

给一个长度为 n 的仅包含小写字母的字符串 S，一个正整数 k，求一个最长的字符串 T，满足: 1. T 为 S 的前缀 2. T 为 S 的后缀 3. T 在 S 中至少出现 k 次

```cpp
#include <bits/stdc++.h>
#define endl '\n'
#define pll pair<ll, ll>
#define tll tuple<ll, ll, ll>
#define vi vector<int>
#define vl vector<ll>
#define x first
#define y second
#define rep(i, j, k) for(int i = (j); i <= (k); i++)
#define per(i, j, k) for(int i = (j); i >= (k); i--)
#define ios ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
typedef long long ll;
const ll maxn = 1e6 + 10;
const ll mod  = 998244353;
const ll inf  = 0x3f3f3f3f;

vi G[maxn];
int sz[maxn];

vector<int> prefix_function(string s)
{
    G[0].push_back(1);
    int n = (int) s.length();
    vector<int> pi(n);
    for(int i = 2; i < n; i++) {
        pi[i] = pi[i - 1];
        while(pi[i] && s[i] != s[pi[i] + 1])
            pi[i] = pi[pi[i]];
        pi[i] += (s[i] == s[pi[i] + 1]);
        G[pi[i]].push_back(i);
    }
    return pi;
}

void dfs(int u)
{
    int sum = 1;
    for (auto v : G[u]){
        dfs(v);
        sum += sz[v];
    }
    sz[u] = sum;
}

void solve()
{
    int n, k;
    cin >> n >> k;
    string s;
    cin >> s;
    s = " " + s;
    auto nxt = prefix_function(s);
    dfs(0);
    int u = n;
    while (u && sz[u] < k) u = nxt[u];
    if (!u) cout << -1 << endl;
    else cout << s.substr(1, u) << endl;
}

int main()
{
    ios;
    // freopen("sample.txt", "r", stdin);
    // freopen("resout.txt", "w", stdout);
    int t = 1;
    //cin >> t;
    while(t--) {
        solve();
    }
    return 0;
}
```

- Border2

给一个长度为 n 的仅包含小写字母的字符串 S，有 Q 次操作：1. 修改操作：1 ch 表示向字符串末尾添加一个字符 ch 2. 查询操作：2 k，求一个最长的字符串 T 满足：T 为 S 的前缀，T 为 S 的后缀，且 T 在 S 中至少出现次

```
#include <bits/stdc++.h>
#define endl '\n'
#define pll pair<ll, ll>
#define tll tuple<ll, ll, ll>
#define vi vector<int>
#define vl vector<ll>
#define x first
#define y second
#define rep(i, j, k) for(int i = (j); i <= (k); i++)
#define per(i, j, k) for(int i = (j); i >= (k); i--)
#define ios ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
typedef long long ll;
const ll maxn = 5e5 + 10;
const ll mod  = 998244353;
const ll inf  = 0x3f3f3f3f;

vi G[maxn];
int tr[maxn], dfn[maxn], low[maxn], f[maxn][21], tot = 0;

int lowbit(int x)
{
    return x & (-x);
```

```cpp
}

vector<pair<int, int>> qry(maxn);

void add(int x, int val)
{
    for(int i = x; i <= tot; i += lowbit(i))
        tr[i] += val;
}
int query(int x)
{
    int res = 0;
    for(int i = x; i; i -= lowbit(i))
        res += tr[i];
    return res;
}

vector<int> prefix_function(string s)
{
    G[0].push_back(1);
    int n = (int) s.length();
    vector<int> pi(n);
    for(int i = 2; i < n; i++) {
        pi[i] = pi[i - 1];
        while(pi[i] && s[i] != s[pi[i] + 1])
            pi[i] = pi[pi[i]];
        pi[i] += (s[i] == s[pi[i] + 1]);
        G[pi[i]].push_back(i);
    }
    return pi;
}

void dfs(int u)
{
    dfn[u] = ++tot;
    for(auto v : G[u]) {
        f[v][0] = u;
        for(int i = 1; i <= 20; ++i)
            f[v][i] = f[f[v][i - 1]][i - 1];
        dfs(v);
    }
    low[u] = tot;
}

void solve()
{
    int n, q;
    string s;
    cin >> n >> q >> s;
    s = " " + s;
    for(int i = 1; i <= q; ++i) {
        cin >> qry[i].x;
        if(qry[i].x == 2)
            cin >> qry[i].y;
        else {
            char ch;
            cin >> ch;
            qry[i].y = ch;
            s += ch;
        }
    }
    auto nxt = prefix_function(s);
    dfs(0);
    for(int i = 1; i <= n; ++i)
        add(dfn[i], 1);
    for(int i = 1; i <= q; ++i) {
        if(qry[i].x == 1)
            add(dfn[++n], 1);
        else {
            int cur = n;
            for(int j = 20; j >= 0; --j) {
                int k = qry[i].y;
                int p = f[cur][j];
                if(query(low[p]) - query(dfn[p] - 1) < k)
                    cur = p;
            }
            int ans = -1;
            if(f[cur][0])
                ans = f[cur][0];
            cout << ans << endl;
        }
    }
}

int main()
{
    ios;
    // freopen("sample.txt", "r", stdin);
    // freopen("resout.txt", "w", stdout);
    int t = 1;
    //cin >> t;
    while(t--) {
        solve();
    }
    return 0;
}
```

ACAM

- AC 自动机模板

```
#include <bits/stdc++.h>
#define endl '\n'
#define pll pair<ll, ll>
#define tll tuple<ll, ll, ll>
#define vi vector<int>
#define vl vector<ll>
#define x first
#define y second
#define rep(i, j, k) for(int i = (j); i <= (k); i++)
#define per(i, j, k) for(int i = (j); i >= (k); i--)
#define ios ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
typedef long long ll;
const ll maxn = 2e5 + 10;
const ll mod = 998244353;
const ll inf = 0x3f3f3f3f;

int tr[maxn][26], cnt = 0;
int sz[maxn], id[maxn], fail[maxn];
vi G[maxn];
string s;

void insert(int x)
{
    int p = 0;
    for(int i = 0; i < s.size(); ++i) {
        int c = s[i] - 'a';
        if(!tr[p][c])
            tr[p][c] = ++cnt;
        p = tr[p][c];
    }
    id[x] = p;
}

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

```cpp
        int u = q.front();
        q.pop();
        for(int i = 0; i < 26; ++i) {
            int &v = tr[u][i];
            if(v) {
                fail[v] = tr[fail[u]][i];
                q.push(tr[u][i]);
            }
            else
                v = tr[fail[u]][i];
        }
    }
}

void dfs(int u)
{
    for(auto v : G[u]) {
        dfs(v);
        sz[u] += sz[v];
    }
}

void solve()
{
    int n;
    cin >> n;
    for(int i = 1; i <= n; ++i) {
        cin >> s;
        insert(i);
    }
    build();
    cin >> s;
    int p = 0;
    for(auto c : s) {
        p = tr[p][c - 'a'];
        sz[p]++;
    }
    for(int i = 1; i <= cnt; ++i)
        G[fail[i]].push_back(i);
    dfs(0);
    for(int i = 1; i <= n; ++i)
        cout << sz[id[i]] << endl;
}

int main()
{
    ios;
    // freopen("sample.txt", "r", stdin);
    // freopen("resout.txt", "w", stdout);
    int t = 1;
    // cin >> t;
    while(t--) {
        solve();
    }
    return 0;
}
```

- 单词 (出现了多少次)

```cpp
#include <bits/stdc++.h>
#define endl '\n'
#define pll pair<ll, ll>
#define tll tuple<ll, ll, ll>
#define vi vector<int>
#define vl vector<ll>
#define x first
#define y second
#define rep(i, j, k) for(int i = (j); i <= (k); i++)
#define per(i, j ,k) for(int i = (j); i >= (k); i--)
#define ios ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
typedef long long ll;
```

```cpp
const ll maxn = 1e6 + 10;
const ll mod = 998244353;
const ll inf = 0x3f3f3f3f;

int tr[maxn][26], cnt = 0;
int sz[maxn], id[maxn], fail[maxn], ed[maxn], ans[maxn];
vi G[maxn];
string s;

void insert(int x)
{
    int p = 0;
    for (int i = 0; i < s.size(); ++i) {
        int c = s[i] - 'a';
        if (!tr[p][c])
            tr[p][c] = ++cnt;
        p = tr[p][c];
        sz[p]++;
    }
    id[x] = p;
    ed[p] = s.size();
}

void build()
{
    queue<int> q;
    for (int i = 0; i < 26; ++i)
        if (tr[0][i])
            q.push(tr[0][i]);
    while (q.size()) {
        int u = q.front();
        q.pop();
        for (int i = 0; i < 26; ++i) {
            int &v = tr[u][i];
            if (v) {
                fail[v] = tr[fail[u]][i];
                q.push(tr[u][i]);
            }
            else
                v = tr[fail[u]][i];
        }
    }
}

void dfs(int u)
{
    for(auto v : G[u]) {
        dfs(v);
        sz[u] += sz[v];
    }
}

void solve() {
    int n;
    cin >> n;
    for (int i = 1; i <= n; ++i) {
        cin >> s;
        insert(i);
    }
    build();
    for (int i = 1; i <= cnt; ++i)
        G[fail[i]].push_back(i);
    dfs(0);
    for (int i = 1; i <= n; ++i)
        cout << sz[id[i]] << endl;
}

int main() {
    ios;
    //freopen("sample.txt", "r", stdin);
    //freopen("resout.txt", "w", stdout);
    int t = 1;
    //cin >> t;
    while (t--) {
        solve();
    }
    return 0;
}
```

- 文本生成器

长度为 n 的串中，出现任一所给字符串的个数的方案书

```
#include <bits/stdc++.h>
#define endl '\n'
#define pll pair<ll, ll>
#define tll tuple<ll, ll, ll>
#define vi vector<int>
#define vl vector<ll>
#define x first
#define y second
#define rep(i, j, k) for (int i = (j); i <= (k); i++)
#define per(i, j, k) for (int i = (j); i >= (k); i--)
#define ios ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
typedef long long ll;
const ll maxn = 4e6 + 10;
const ll mod = 10007;
const ll inf = 0x3f3f3f3f;

int tr[maxn][26], tot;
int fail[maxn];
vi G[maxn];
int ok[maxn];
int f[10010][6010];

void insert(string s)
{
    int p = 0;
    for (int i = 0; i < s.size(); ++i)
    {
        int c = s[i] - 'A';
        if (!tr[p][c])
            tr[p][c] = ++tot;
        p = tr[p][c];
    }
    ok[p] = 1;
}

void build()
{
    queue<int> q;
    for (int i = 0; i < 26; ++i)
        if (tr[0][i])
            q.push(tr[0][i]);
    while (!q.empty())
    {
        auto u = q.front();
        q.pop();
        for (int i = 0; i < 26; ++i){
            auto &v = tr[u][i];
            if (v){
                fail[v] = tr[fail[u]][i];
                q.push(v);
            }else
                v = tr[fail[u]][i];
        }
    }
}

void dfs(int u){
    for (auto v : G[u]){
        if (ok[u]) ok[v] = 1;
```

```cpp
        dfs(v);
    }
}

void solve()
{
    int n, m;
    cin >> n >> m;
    for (int i = 1; i <= n; ++i){
        string s;
        cin >> s;
        insert(s);
    }
    build();
    for (int i = 1; i <= tot; ++i) G[fail[i]].push_back(i);
    dfs(0);
    f[0][0] = 1;
    for (int i = 0; i <= m; ++i){
        for (int j = 0; j <= tot; ++j){
            for (int k = 0; k < 26; ++k){
                if (!ok[tr[j][k]]) f[i + 1][tr[j][k]] = (f[i + 1][tr[j][k]] + f[i][j]) % mod;
            }
        }
    }
    ll ans = 0;
    for (int i = 0; i <= tot; ++i)
        if (!ok[i]) ans += f[m][i];
    ll sum = 1;
    for (int i = 1; i <= m; ++i)
        sum = sum * 26 % mod;
    cout << ((sum - ans) % mod + mod) % mod;
}

int main()
{
    ios;
    // freopen("sample.txt", "r", stdin);
    // freopen("resout.txt", "w", stdout);
    int t = 1;
    // cin >> t;
    while (t--)
    {
        solve();
    }
    return 0;
}
```

manacher

- 查回文

给出 l, r 求 l, r 区间内满足点对下字符串为回文串的方案点对数

位于左半个区间的回文中心，在延伸的过程中只可能被区间的左边界截断，位于右半个区间的，只可能被区间的右边界截断。对于每个 i，我给区间 $[i - r[i] + 1, i]$ 这些位置加一，然后我要求的东西就转化成了区间 $[l, +\infty)$ 的

```cpp
#include <bits/stdc++.h>
#define endl '\n'
#define pll pair<ll, ll>
#define tll tuple<ll, ll, ll>
#define vi vector<int>
#define vl vector<ll>
#define x first
#define y second
#define iinf 0x3f3f3f3f
#define linf (1ll << 60)
#define rep(i, j, k) for (int i = (j); i <= (k); i++)
#define per(i, j, k) for (int i = (j); i >= (k); i--)
#define ios ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
typedef long long ll;
```

```cpp
const ll maxn = 4e5 + 10;
const ll mod = 998244353;
const ll inf = 0x3f3f3f3f;

struct Manacher
{
    int r[maxn], p[maxn], n;
    void clear() { memset(r, 0, sizeof r), memset(p, 0, sizeof p); }
    void calc(string s, int N)
    {
        n = N;
        int i, j, mx(0), center;
        r[0] = -2;
        for (i = 1; i <= N; i++)
            r[2 * i] = s[i];
        for (i = 1; i <= N; i++)
            r[2 * i - 1] = -1;
        r[2 * N + 1] = -1;
        for (i = 1; i <= 2 * N + 1; i++)
        {
            if (mx >= i)
                p[i] = min(p[2 * center - i], mx - i + 1);
            else
                p[i] = 1;
            while (r[i - p[i]] == r[i + p[i]])
                p[i]++;
            if (i + p[i] - 1 > mx)
            {
                mx = i + p[i] - 1;
                center = i;
            }
        }
    }
} mnc;
ll n, id[maxn], l[maxn], r[maxn], ans[maxn], q;
struct SegmentTree
{
    ll mn[maxn << 2], mx[maxn << 2], sum[maxn << 2], add[maxn << 2], set[maxn << 2], L[maxn << 2], R[maxn << 2];
    void maketag_set(ll o, ll v)
    {
        add[o] = 0;
        set[o] = v;
        mx[o] = mn[o] = v;
        sum[o] = (R[o] - L[o] + 1) * v;
    }
    void maketag_add(ll o, ll v)
    {
        add[o] += v;
        mx[o] += v, mn[o] += v;
        sum[o] += (R[o] - L[o] + 1) * v;
    }
    void pushdown(ll o)
    {
        if (L[o] == R[o])
            return;
        if (~set[o])
        {
            maketag_set(o << 1, set[o]);
            maketag_set(o << 1 | 1, set[o]);
            set[o] = -1;
        }
        if (add[o])
        {
            maketag_add(o << 1, add[o]);
            maketag_add(o << 1 | 1, add[o]);
            add[o] = 0;
        }
    }
    void pushup(ll o)
    {
        mx[o] = max(mx[o << 1], mx[o << 1 | 1]);
        mn[o] = min(mn[o << 1], mn[o << 1 | 1]);
        sum[o] = sum[o << 1] + sum[o << 1 | 1];
    }
    void build(ll o, ll l, ll r, ll *array = NULL)
    {
        ll mid(l + r >> 1);
        L[o] = l, R[o] = r;
        add[o] = 0;
        set[o] = -1;
        if (l == r)
        {
            if (array)
                mn[o] = mx[o] = sum[o] = array[l];
            else
                mn[o] = mx[o] = sum[o] = 0;
            return;
        }
        build(o << 1, l, mid, array);
        build(o << 1 | 1, mid + 1, r, array);
        pushup(o);
    }
    void Set(ll o, ll l, ll r, ll v)
    {
        ll mid(L[o] + R[o] >> 1);
        if (l <= L[o] and r >= R[o])
        {
            maketag_set(o, v);
            return;
        }
        pushdown(o);
        if (l <= mid)
            Set(o << 1, l, r, v);
        if (r > mid)
            Set(o << 1 | 1, l, r, v);
        pushup(o);
    }
    void Add(ll o, ll l, ll r, ll v)
    {
        ll mid(L[o] + R[o] >> 1);
        if (l <= L[o] and r >= R[o])
        {
            maketag_add(o, v);
            return;
        }
        pushdown(o);
        if (l <= mid)
            Add(o << 1, l, r, v);
        if (r > mid)
            Add(o << 1 | 1, l, r, v);
        pushup(o);
    }
    ll Sum(ll o, ll l, ll r)
    {
        pushdown(o);
        ll mid(L[o] + R[o] >> 1), ans(0);
        if (l <= L[o] and r >= R[o])
            return sum[o];
        if (l <= mid)
            ans += Sum(o << 1, l, r);
        if (r > mid)
            ans += Sum(o << 1 | 1, l, r);
        return ans;
    }
    ll Min(ll o, ll l, ll r)
    {
        ll mid(L[o] + R[o] >> 1), ans(linf);
        if (l <= L[o] and r >= R[o])
            return mn[o];
        pushdown(o);
        if (l <= mid)
            ans = min(ans, Min(o << 1, l, r));
```

```cpp
        if (r > mid)
            ans = min(ans, Min(o << 1 | 1, l, r));
        return ans;
    }
    ll Max(ll o, ll l, ll r)
    {
        ll mid(L[o] + R[o] >> 1), ans(-linf);
        if (l <= L[o] and r >= R[o])
            return mx[o];
        pushdown(o);
        if (l <= mid)
            ans = max(ans, Max(o << 1, l, r));
        if (r > mid)
            ans = max(ans, Max(o << 1 | 1, l, r));
        return ans;
    }
} segtree;

void solve()
{
    cin >> n >> q;
    string s;
    cin >> s;
    s = " " + s;
    mnc.calc(s, n);
    for (int i = 1; i <= q; ++i){
        cin >> l[i] >> r[i];
        l[i] = 2 * l[i] - 1;
        r[i] = 2 * r[i] + 1;
        id[i] = i;
    }
    sort(id + 1, id + q + 1, [&](ll a, ll b){return l[a] + r[a] < l[b] + r[b];});
    segtree.build(1, 1, 2 * n + 1);
    int j = 1;
    for (int i = 1; i <= q; ++i){
        while(j <= (l[id[i]] + r[id[i]] >> 1)){
            segtree.Add(1, j - mnc.p[j] + 1, j, 1);
            ++j;
        }
        ans[id[i]] += segtree.Sum(1, l[id[i]], 2 * n);
    }
    segtree.build(1, 1, 2 * n + 1);
    j = 2 * n + 1;
    for (int i = q; i >= 1; --i){
        while (j > (l[id[i]] + r[id[i]] >> 1)){
            segtree.Add(1, j, j + mnc.p[j] - 1, 1);
            --j;
        }
        ans[id[i]] += segtree.Sum(1, 1, r[id[i]]);
    }
    for (int i = 1; i <= q; ++i){
        ans[i] -= (r[i] + 1 >> 1) - (l[i] >> 1);
        cout << ans[i] / 2 << endl;
    }
}

int main()
{
    ios;
    // freopen("sample.txt", "r", stdin);
    // freopen("resout.txt", "w", stdout);
    int t = 1;
    //cin >> t;
    while (t--)
    {
        solve();
    }
    return 0;
}
```

- 拉拉队排练

按照女生的个数降序排序之后，前 K 个和谐小群体的女生个数的乘积是多少。由于答案可能很大，只要你告诉她，答案除以 19930726 的余数是多少就行了

```cpp
#include <bits/stdc++.h>
#define endl '\n'
#define pll pair<ll, ll>
#define tll tuple<ll, ll, ll>
#define vi vector<int>
#define vl vector<ll>
#define int ll
#define rep(i, j, k) for(int i = (j); i <= (k); i++)
#define per(i, j ,k) for(int i = (j); i >= (k); i--)
#define ios ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
typedef long long ll;
const ll inf = 0x3f3f3f3f;
const int N = 2e6 + 10;
const int mod = 19930726;
string s;
int d[N];
int mp[N];

int qmi(int a, int n)
{
    int res = 1;
    a %= mod;
    while (n)
    {
        if (n & 1) res = res * a % mod;
        a = a * a % mod;
        n >>= 1;
    }
    return res;
}

void manacher(int n)
{
    d[1] = 1;

    for (int i = 2, l, r = 1; i <= n; i++)
    {
        if (r >= i) d[i] = min(r - i + 1, d[r - i + l]); // 在加速盒子内
        while (s[i - d[i]] == s[i + d[i]]) d[i]++; // 盒外暴力
        if (i + d[i] - 1 > r) r = i + d[i] - 1, l = i - d[i] + 1; // 更新加速盒子（根据有边界）
        mp[d[i] * 2 - 1]++;
    }
}

signed main()
{
    ios;
    int n, k;
    cin >> n >> k >> s;
    s = '@' + s + '.';
    manacher(n);

    if (n % 2 == 0) n--;

    int ans = 1, sum = 0;
    for (int i = n; i > 0; i -= 2)
    {
        sum += mp[i];
        if (k < sum)
        {
            ans = ans * qmi(i, k) % mod;
            k -= sum;
            break;
        }
        else
        {
            ans = ans * qmi(i, sum) % mod;
            k -= sum;
        }

    }
    if (k > 0) cout << -1 << endl;
    else cout << ans << endl;

    return 0;
}
```

- 最长双回文串

```
#include <bits/stdc++.h>
#define endl '\n'
#define pll pair<ll, ll>
#define tll tuple<ll, ll, ll>
#define vi vector<int>
#define vl vector<ll>
#define x first
#define y second
#define rep(i, j, k) for(int i = (j); i <= (k); i++)
#define per(i, j ,k) for(int i = (j); i >= (k); i--)
#define ios ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
typedef long long ll;
#define int ll
const ll maxn = 1e6 + 10;
const ll mod = 998244353;
const ll inf = 0x3f3f3f3f;

char s[maxn];
int p[maxn], L[maxn], R[maxn];
int n;
void Manacher(string t)
{
    s[0] = '@', s[1] = '#';
    int cnt = 1;
    for (auto x : t)
    {
        s[++cnt] = x;
        s[++cnt] = '#';
    }
    n = cnt;
    for (int i = 1, mid = 0, r = 0; i <= n; i++)
    {
        if (i <= r)p[i] = min(p[2 * mid - i], r - i + 1);
        while (s[i - p[i]] == s[i + p[i]])p[i]++;
        if (i + p[i] > r)r = i + p[i] - 1, mid = i;
        int l = i + p[i] - 1;
        int rr = i - p[i] + 1;
        R[rr] = max(R[rr], p[i] - 1);//以 rr 为回文串右端点的最长回文串
        L[l] = max(L[l], p[i] - 1);//以 ll 为回文串左端点的最长回文串
    }
}

void solve(){
    string t;
    cin >> t;
    n = t.length();
    Manacher(t);
    ll ans = 0;
    for (int i = 3; i <= n; i += 2) R[i] = max(R[i], R[i - 2] - 2);
    for (int i = n - 1; i >= 1; i -= 2) L[i] = max(L[i], L[i + 2] - 2);
    for (int i = 1; i <= n; i += 2) if (R[i] && L[i])ans = max(ans, 1ll * (L[i] + R[i]));
    cout << ans << endl;
}

signed main(){
    ios;
    //freopen("sample.txt", "r", stdin);
    //freopen("resout.txt", "w", stdout);
    int t = 1;
    //cin >> t;
    while(t--){
        solve();
    }
    return 0;
}
```

pam

- 本质不同回文字串个数

一个串的本质不同回文子串个数等于回文树的状态数 (排除奇根和偶根两个状态)

- 回文子串出现次数

(最大 (回文字串出现的次数 * 回文子串的长度))

```
#include <bits/stdc++.h>
using namespace std;
const int maxn = 300000 + 5;
typedef long long ll;
namespace pam {
int sz, tot, last;
int cnt[maxn], ch[maxn][26], len[maxn], fail[maxn];
char s[maxn];

int node(int l) {  // 建立一个新节点，长度为 l
    sz++;
    memset(ch[sz], 0, sizeof(ch[sz]));
    len[sz] = l;
    fail[sz] = cnt[sz] = 0;
    return sz;
}

void clear() {  // 初始化
    sz = -1;
    last = 0;
    s[tot = 0] = '$';
    node(0);
    node(-1);
    fail[0] = 1;
}

int getfail(int x) {  // 找后缀回文
    while (s[tot - len[x] - 1] != s[tot]) x = fail[x];
    return x;
}

void insert(char c) {  // 建树
    s[++tot] = c;
    int now = getfail(last);
    if (!ch[now][c - 'a']) {
        int x = node(len[now] + 2);
        fail[x] = ch[getfail(fail[now])][c - 'a'];
        ch[now][c - 'a'] = x;
    }
    last = ch[now][c - 'a'];
    cnt[last]++;
}

ll solve() {
    ll ans = 0;
    for (int i = sz; i >= 0; i--) {
        cnt[fail[i]] += cnt[i];
    }
    for (int i = 1; i <= sz; i++) {  // 更新答案
        ans = max(ans, 1ll * len[i] * cnt[i]);
    }
    return ans;
}
}  // namespace pam
```

```cpp

char s[maxn];

int main() {
    int n;
    cin >> n;
    pam::clear();
    scanf("%s", s + 1);
    for (int i = 1; s[i]; i++) {
        pam::insert(s[i]);
    }
    printf("%lld\n", pam::solve());
    return 0;
}
```

- 例题 (Colourful String)

子字符串不同颜色的数量的和

```cpp
#include <bits/stdc++.h>
#define endl '\n'
#define pll pair<ll, ll>
#define tll tuple<ll, ll, ll>
#define vi vector<int>
#define vl vector<ll>
#define x first
#define y second
#define rep(i, j, k) for(int i = (j); i <= (k); i++)
#define per(i, j ,k) for(int i = (j); i >= (k); i--)
#define ios ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
typedef long long ll;
const ll maxn = 1e6 + 10;
const ll mod = 998244353;
const ll inf = 0x3f3f3f3f;


int n;
string s;

struct PAM
{
    int last, idx;
    vector<array<int, 26>> tr;
    vector<int> fail, len, cnt, mask;

    PAM(): last(0), idx(0) {
        fail.resize(n + 2), len.resize(n + 2), tr.resize(n + 2), mask.resize(n + 2), cnt.resize(n + 2);
    }
    int newnode(int l) { //新增一个结点，长度为 l
        len[idx] = l;
        tr[idx].fill(0);
        return idx++;
    }
    void init() {
        idx = last = 0;
        newnode(0), newnode(-1); //偶根长度为 0，奇根长度为-1
        s[0] = -1, fail[0] = 1; //偶根的失配边指向奇根，奇根的失配边指向偶根
    }
    int get_fail(int p, int i) {
        while (s[i - len[p] - 1] != s[i])    p = fail[p];
        return p;
    }
    void insert(int i) {
        int u = s[i] - 'a', p = get_fail(last, i);
        if (!tr[p][u]) {
            int now = newnode(len[p] + 2);
            mask[now] = mask[p] | (1 << u);
            fail[now] = tr[get_fail(fail[p], i)][u];
            tr[p][u] = now;
        }
        last = tr[p][u];
        cnt[last]++;
    }
};


void solve() {
    cin >> s;
    n = s.length();
    s = " " + s;
    PAM pam;
    pam.init();
    for (int i = 1; i <= n; ++i) pam.insert(i);
    for (int i = pam.idx - 1; ~i; --i) pam.cnt[pam.fail[i]] += pam.cnt[i];
    ll res = 0;
    for (int i = 2; i < pam.idx; ++i) res += pam.cnt[i] * __builtin_popcount(pam.mask[i]);
    cout << res << endl;
}

int main() {
    ios;
    //freopen("sample.txt", "r", stdin);
    //freopen("resout.txt", "w", stdout);
    int t = 1;
    //cin >> t;
    while (t--) {
        solve();
    }
    return 0;
}
```

SA

- nlog^2n(倍增法)

```
#include <algorithm>
#include <cstdio>
#include <cstring>
#include <iostream>

using namespace std;

const int N = 1000010;

char s[N];
int n, w, sa[N], rk[N << 1], oldrk[N << 1];

// 为了防止访问 rk[i+w] 导致数组越界，开两倍数组。
// 当然也可以在访问前判断是否越界，但直接开两倍数组方便一些。

int main() {
  int i, p;

  scanf("%s", s + 1);
  n = strlen(s + 1);
  for (i = 1; i <= n; ++i) sa[i] = i, rk[i] = s[i];

  for (w = 1; w < n; w <<= 1) {
    sort(sa + 1, sa + n + 1, [](int x, int y) {
      return rk[x] == rk[y] ? rk[x + w] < rk[y + w] : rk[x] < rk[y];
    });  // 这里用到了 lambda
    memcpy(oldrk, rk, sizeof(rk));
    // 由于计算 rk 的时候原来的 rk 会被覆盖，要先复制一份
    for (p = 0, i = 1; i <= n; ++i) {
      if (oldrk[sa[i]] == oldrk[sa[i - 1]] &&
          oldrk[sa[i] + w] == oldrk[sa[i - 1] + w]) {
        rk[sa[i]] = p;
      } else {
        rk[sa[i]] = ++p;
      }  // 若两个子串相同，它们对应的 rk 也需要相同，所以要去重
    }
  }

  for (i = 1; i <= n; ++i) printf("%d ", sa[i]);

  return 0;
}
```

- SA-IS

```
// 后缀类型
#define L_TYPE 0
#define S_TYPE 1

// 判断一个字符是否为 LMS 字符
inline bool is_lms_char(int *type, int x) {
    return x > 0 && type[x] == S_TYPE && type[x - 1] == L_TYPE;
}

// 判断两个 LMS 子串是否相同
inline bool equal_substring(int *S, int x, int y, int *type) {
    do {
        if (S[x] != S[y])
            return false;
        x++, y++;
    } while (!is_lms_char(type, x) && !is_lms_char(type, y));

    return S[x] == S[y];
}

// 诱导排序 (从 * 型诱导到 L 型、从 L 型诱导到 S 型)
// 调用之前应将 * 型按要求放入 SA 中
inline void induced_sort(int *S, int *SA, int *type, int *bucket, int *lbucket,
                         int *sbucket, int n, int SIGMA) {
    for (int i = 0; i <= n; i++)
        if (SA[i] > 0 && type[SA[i] - 1] == L_TYPE)
            SA[lbucket[S[SA[i] - 1]]++] = SA[i] - 1;
    for (int i = 1; i <= SIGMA; i++)  // Reset S-type bucket
        sbucket[i] = bucket[i] - 1;
    for (int i = n; i >= 0; i--)
        if (SA[i] > 0 && type[SA[i] - 1] == S_TYPE)
            SA[sbucket[S[SA[i] - 1]]--] = SA[i] - 1;
}

// SA-IS 主体
// S 是输入字符串, length 是字符串的长度, SIGMA 是字符集的大小
static int *SAIS(int *S, int length, int SIGMA) {
    int n = length - 1;
    int *type = new int[n + 1];  // 后缀类型
    int *position = new int[n + 1];  // 记录 LMS 子串的起始位置
    int *name = new int[n + 1];  // 记录每个 LMS 子串的新名称
    int *SA = new int[n + 1];  // SA 数组
    int *bucket = new int[SIGMA + 1];  // 每个字符的桶
    int *lbucket = new int[SIGMA + 1];  // 每个字符的 L 型桶的起始位置
    int *sbucket = new int[SIGMA + 1];  // 每个字符的 S 型桶的起始位置

    // 初始化每个桶
    memset(bucket, 0, sizeof(int) * (SIGMA + 1));
    for (int i = 0; i <= n; i++)
        bucket[S[i]]++;
    lbucket[0] = sbucket[0] = 0;
    for (int i = 1; i <= SIGMA; i++) {
        bucket[i] += bucket[i - 1];
        lbucket[i] = bucket[i - 1];
        sbucket[i] = bucket[i] - 1;
    }

    // 确定后缀类型 (利用引理 2.1)
    type[n] = S_TYPE;
    for (int i = n - 1; i >= 0; i--) {
        if (S[i] < S[i + 1])
            type[i] = S_TYPE;
        else if (S[i] > S[i + 1])
            type[i] = L_TYPE;
        else
            type[i] = type[i + 1];
    }

    // 寻找每个 LMS 子串
    int cnt = 0;
    for (int i = 1; i <= n; i++)
        if (type[i] == S_TYPE && type[i - 1] == L_TYPE)
            position[cnt++] = i;

    // 对 LMS 子串进行排序
    fill(SA, SA + n + 1, -1);
    for (int i = 0; i < cnt; i++)
        SA[sbucket[S[position[i]]]--] = position[i];
    induced_sort(S, SA, type, bucket, lbucket, sbucket, n, SIGMA);

    // 为每个 LMS 子串命名
    fill(name, name + n + 1, -1);
    int lastx = -1, namecnt = 1;  // 上一次处理的 LMS 子串与名称的计数
    bool flag = false;  // 这里顺便记录是否有重复的字符
    for (int i = 1; i <= n; i++) {
        int x = SA[i];

        if (is_lms_char(type, x)) {
            if (lastx >= 0 && !equal_substring(S, x, lastx, type))
                namecnt++;
            // 因为只有相同的 LMS 子串才会有同样的名称
            if (lastx >= 0 && namecnt == name[lastx])
                flag = true;

            name[x] = namecnt;
            lastx = x;
        }
    }  // for
    name[n] = 0;

    // 生成 S1
    int *S1 = new int[cnt];
    int pos = 0;
    for (int i = 0; i <= n; i++)
        if (name[i] >= 0)
            S1[pos++] = name[i];

    int *SA1;
    if (!flag) {
        // 直接计算 SA1
        SA1 = new int[cnt + 1];

        for (int i = 0; i < cnt; i++)
            SA1[S1[i]] = i;
    } else
        SA1 = SAIS(S1, cnt, namecnt);  // 递归计算 SA1

    // 从 SA1 诱导到 SA
    lbucket[0] = sbucket[0] = 0;
    for (int i = 1; i <= SIGMA; i++) {
        lbucket[i] = bucket[i - 1];
        sbucket[i] = bucket[i] - 1;
    }
    fill(SA, SA + n + 1, -1);
    for (int i = cnt - 1; i >= 0; i--)  // 这里是逆序扫描 SA1，因为 SA 中 S 型桶是倒序的
        SA[sbucket[S[position[SA1[i]]]]--] = position[SA1[i]];
    induced_sort(S, SA, type, bucket, lbucket, sbucket, n, SIGMA);

    // 后缀数组计算完毕
    return SA;
}
```

- 从字符串首尾取字符最小化字典序

暴力做法就是每次最坏 O(n) 地判断当前应该取首还是尾（即比较取首得到的字符串与取尾得到的反串的大小），只需优化这一判断过程

即可。由于需要在原串后缀与反串后缀构成的集合内比较大小，可以将反串拼接在原串后，并在中间加上一个没出现过的字符（如 #，代码中可以直接使用空字符），求后缀数组，即可 O(1) 完成这一判断。

```cpp
#include <cctype>
#include <cstdio>
#include <cstring>
#include <iostream>

using namespace std;

const int N = 1000010;

char s[N];
int n, sa[N], id[N], oldrk[N * 2], rk[N * 2], px[N], cnt[N];

bool cmp(int x, int y, int w) {
  return oldrk[x] == oldrk[y] && oldrk[x + w] == oldrk[y + w];
}

int main() {
  int i, w, m = 200, p, l = 1, r, tot = 0;

  cin >> n;
  r = n;

  for (i = 1; i <= n; ++i)
    while (!isalpha(s[i] = getchar()))
      ;
  for (i = 1; i <= n; ++i)
    rk[i] = rk[2 * n + 2 - i] = s[i];  // 拼接正反两个字符串，中间空出一个字符

  n = 2 * n + 1;
  // 求后缀数组
  for (i = 1; i <= n; ++i) ++cnt[rk[i]];
  for (i = 1; i <= m; ++i) cnt[i] += cnt[i - 1];
  for (i = n; i >= 1; --i) sa[cnt[rk[i]]--] = i;

  for (w = 1; w < n; w *= 2, m = p) {  // m=p 就是优化计数排序值域
    for (p = 0, i = n; i > n - w; --i) id[++p] = i;
    for (i = 1; i <= n; ++i)
      if (sa[i] > w) id[++p] = sa[i] - w;
    memset(cnt, 0, sizeof(cnt));
    for (i = 1; i <= n; ++i) ++cnt[px[i] = rk[id[i]]];
    for (i = 1; i <= m; ++i) cnt[i] += cnt[i - 1];
    for (i = n; i >= 1; --i) sa[cnt[px[i]]--] = id[i];
    memcpy(oldrk, rk, sizeof(rk));
    for (p = 0, i = 1; i <= n; ++i)
      rk[sa[i]] = cmp(sa[i], sa[i - 1], w) ? p : ++p;
  }
  // 利用后缀数组 O(1) 进行判断
  while (l <= r) {
    printf("%c", rk[l] < rk[n + 1 - r] ? s[l++] : s[r--]);
    if ((++tot) % 80 == 0) puts("");  // 回车
  }

  return 0;
}
```

杂项

线性基

- 线性基模板 (总异或最大值)

```cpp
ll p[64];
void insert(ll x){
    for (int i = 63; ~i; --i){
        if (!(x >> i)) continue;
        if (!p[i]){
            p[i] = x;
```

```
                break;
            }
            x ^= p[i];
        }
}
```

- 区间线性基 (区间异或最大值, 强制在线)

```
#include<bits/stdc++.h>
#define M 500009
using namespace std;
int read() {
    int f = 1, re = 0; char ch;
    for (ch = getchar(); !isdigit(ch) && ch != '-'; ch = getchar());
    if (ch == '-') {f = -1, ch = getchar();}
    for (; isdigit(ch); ch = getchar()) re = (re << 3) + (re << 1) + ch - '0';
    return re * f;
}
int pos[M][32], p[M][32], t, n, m, lastans;
void insert(int val, int num, int po) {
    for (int i = 30; i >= 0; i--) {
        if (val & (1ll << i)) {
            if (!p[num][i]) {p[num][i] = val, pos[num][i] = po; return;}
            else if (pos[num][i] < po) {
                swap(val, p[num][i]);
                swap(po, pos[num][i]);
            } val ^= p[num][i];
        }
    } return;
}
int query(int l, int r) {
    int ans = 0;
    for (int i = 30; i >= 0; i--)
        if (pos[r][i] >= l && (p[r][i]^ans) > ans) ans ^= p[r][i];
    return ans;
}
signed main() {
    t = read();
    while (t--) {
        n = read(), m = read(); lastans = 0;
        memset(p, 0, sizeof(p));
        memset(pos, 0, sizeof(pos));
        for (int i = 1; i <= n; i++) {
            int x = read();
            for (int j = 0; j <= 30; j++)
                p[i][j] = p[i - 1][j], pos[i][j] = pos[i - 1][j];
            insert(x, i, i);
        }
        for (int i = 1; i <= m; i++) {
            int opt = read();
            if (opt) {
                int x = read()^lastans; n++;
                for (int j = 0; j <= 30; j++)
                    p[n][j] = p[n - 1][j], pos[n][j] = pos[n - 1][j];
                insert(x, n, n);
            }
            else {
                int l = (read()^lastans) % n + 1;
                int r = (read()^lastans) % n + 1;
                if (l > r) swap(l, r);
                printf("%d\n", lastans = query(l, r));
            }
        }
    } return 0;
}
```

- 区间问题 (异或和, 区间内是否存在异或和为 x)

```
#include <bits/stdc++.h>
#define ll long long
using namespace std;
constexpr ll maxn = 4e5 + 5;
```

```
int pos[65];
ll p[65], t, n, m;
bool ans[maxn];
void insert(ll val, int P)
{
    for (int i = 59; i >= 0; i--)
    {
        if (val & (1ll << i))
        {
            if (!p[i])
            {
                p[i] = val, pos[i] = P;
                return;
            }
            else if (pos[i] < P)
            {
                swap(val, p[i]);
                swap(P, pos[i]);
            }
            val ^= p[i];
        }
    }
    return;
}
bool query(int l, ll val)
{
    for (int i = 59; i >= 0; i--)
    {
        if (val & (1ll << i))
        {
            if (!p[i])
                return false;
            if (pos[i] < l)
                return false;
            val ^= p[i];
        }
    }
    return true;
}
signed main()
{
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    cin >> n >> m;
    vector<ll> a(n + 1);
    vector<tuple<int, int, ll, int>> q(m);
    for (int i = 1; i <= n; i++)
        cin >> a[i];
    for (int i = 0; i < m; i++)
    {
        auto &[l, r, val, id] = q[i];
        cin >> l >> r >> val, id = i;
    }
    sort(q.begin(), q.end(), [&](auto x, auto y)
         {
        auto &[l1,r1,val1,id1] = x;
        auto &[l2,r2,val2,id2] = y;
        return (r1==r2)?(l1<l2):(r1<r2); });
    int R = 0;
    for (int i = 0; i < m; i++)
    {
        auto &[l, r, val, id] = q[i];
        while (R < r)
            insert(a[R + 1], R + 1),R++;
        ans[id] = query(l, val);
    }
    for (int i = 0; i < m; i++)
    {
        cout << (ans[i] ? "Yes\n" : "No\n");
    }

```

```cpp
    return 0;
}

```

Tarjan

- 缩点

```cpp
//Tarjan 缩点（删去一个点，有多少点对不能互通）
#include <bits/stdc++.h>
#define endl '\n'
#define pll pair<ll, ll>
#define tll tuple<ll, ll, ll>
#define x first
#define y second
#define int ll
#define rep(i, j, k) for (int i = (j); i <= (k); i++)
#define per(i, j, k) for (int i = (j); i >= (k); i--)
#define ios ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
typedef long long ll;
typedef __int128 i128;
const ll maxn = 1e6 + 10;
const ll mod = 998244353;
const ll inf = 0x3f3f3f3f;

ll n, m;
ll head[maxn], nxt[maxn], to[maxn], tot = 1;
ll dfn[maxn], low[maxn];
bool vis[maxn];
ll cnt;
ll deg[maxn];
ll ans[maxn];
ll sz[maxn];

void addedge(int u, int v)
{
    nxt[++tot] = head[u];
    to[head[u] = tot] = v;
    nxt[++tot] = head[v];
    to[head[v] = tot] = u;
}

void tarjan(int u, int lst)
{
    dfn[u] = low[u] = ++cnt;
    ll sum = 0;
    sz[u] = 1;
    for (int i = head[u]; i; i = nxt[i])
    {
        if (i != (lst ^ 1))
        {
            int v = to[i];
            if (!dfn[v])
            {
                tarjan(v, i);
                sz[u] += sz[v];
                low[u] = min(low[u], low[v]);
                if (low[v] >= dfn[u])
                {
                    // 找到新的双连通分量
                    ans[u] += 1ll * sz[v] * (n - sz[v]);
                    sum += sz[v];
                    ++deg[u];
                    if (deg[u] > 1 || u != 1)
                        vis[u] = 1;
                }
            }
            else
                low[u] = min(low[u], dfn[v]);
        }
    }
    if (vis[u])
    {
        ans[u] += 1ll * (n - (sum + 1)) * (sum + 1) + n - 1;
    }else
        ans[u] = 2 * (n - 1);
}
void solve()
{
    cin >> n >> m;
    for (int i = 1; i <= m; ++i)
    {
        int u, v;
        cin >> u >> v;
        addedge(u, v);
    }
    tarjan(1, -1);
    for (int i = 1; i <= n; ++i)
    {
        if (vis[i])
        {
            cout << ans[i] << endl;
        }
        else
        {
            cout << 2ll * (n - 1) << endl;
        }
    }
}

signed main()
{
    ios;
    //freopen("sample.txt", "r", stdin);
    //freopen("res.txt", "w", stdout);
    int t = 1;
    // cin >> t;
    while (t--)
    {
        solve();
    }

    return 0;
}
```

位运算基础

```
去掉最后一位
x >> 1
在最后一位加个 0
x << 1
在最后一位加个 1
(x << 1) | 1
把最后一位变成 1
x | 1
把最后一位变成 0
(x | 1) - 1
最后一位取反
x ^ 1
把右数第 k 位变成 1
x | (1 << (k - 1))
把右数第 k 位变成 0
x & (~(1 << (k - 1)))
右数第 k 位取反
x ^ ( 1 << (k - 1))
取末 k 位
x & ((1 << k) - 1)
取右数第 k 位
(x >> (k - 1)) & 1
把末 k 位变成 1
x | ((1 << k) - 1)
```

```
把右边连续的 1 变成 0
x & (x + 1)
把右边第一个 0 变成 1
x | (x + 1)
取右边连续的 1
(x ^ (x + 1)) >> 1
去掉右起第一个 1 的左边
x & (-x)
```

虚拟源点

- 843div2D

给定 n 个点，每个点的权值为 ai。两个位置 i, j 存在一个长度为 1 的边当且仅当 gcd(ai, aj) > 1。求 S 到 T 的最短路

```cpp
#include <bits/stdc++.h>
#define endl '\n'
#define pll pair<ll, ll>
#define tll tuple<ll, ll, ll>
#define vi vector<int>
#define vl vector<ll>
#define x first
#define y second
#define rep(i, j, k) for(int i = (j); i <= (k); i++)
#define per(i, j ,k) for(int i = (j); i >= (k); i--)
#define ios ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
typedef long long ll;
const ll maxn = 6e5 + 10;
const ll mod = 998244353;
const ll inf = 0x3f3f3f3f;

int prime[maxn], cnt = 0;
bool vis[maxn];
int minp[maxn];
int idx[maxn];

void init(int n) {
    for (int i = 2; i <= n; ++i) {
        if (vis[i] == false) {
            prime[++cnt] = i;
            minp[i] = i;
            idx[i] = cnt;
        }
        for (int j = 1; j <= cnt && i * prime[j] <= n; ++j) {
            minp[i * prime[j]] = prime[j];
            vis[i * prime[j]] = 1;
            if (i % prime[j] == 0) break;
        }
    }
}

int a[maxn];
set<int> e[maxn];
ll dis[maxn];
priority_queue<pair<int, int>> q;
int vs[maxn], pre[maxn];

void dij(int s, int exn, int n) {
    rep(i, 1, exn) dis[i] = 1e18;
    dis[s] = 0;
    q.push({0, s});
    while (!q.empty()) {
        pair<int, int> cur = q.top();
        q.pop();
        if (vs[cur.y]) continue;
        int u = cur.y;
        vs[u] = 1;
        for (auto v : e[u]) {
            int w = 1;
            if (v > n) w = 0;
            if (dis[v] > dis[u] + w) {
                dis[v] = dis[u] + w;
                pre[v] = u;
                if (!vs[v]) q.push({ -dis[v],   v});
            }
        }
    }
}

void solve() {
    int n;
    cin >> n;
    for (int i = 1; i <= n; ++i) cin >> a[i];
    int s, t;
    cin >> s >> t;
    int exn = n;
    for (int i = 1; i <= n; ++i) {
        int tmp = a[i];
        while (tmp > 1) {
            int tar = idx[minp[tmp]];
            exn = max(exn, n + tar);
            e[n + tar].insert(i);
            e[i].insert(n + tar);
            tmp /= minp[tmp];
        }
    }
    dij(s, exn, n);
    if (dis[t] == 1e18) cout << -1 << endl;
    else {
        vector<int> ans;
        int tmp = t;
        while (tmp != s && tmp != 0) {
            if (tmp <= n) ans.push_back(tmp);
            tmp = pre[tmp];
        }
        ans.push_back(s);
        reverse(ans.begin(), ans.end());
        cout << ans.size() << endl;
        for (auto it : ans) cout << it << " " ;
        cout << endl;
    }
}
int main() {
    ios;
    init(3e5 + 10);
    //freopen("sample.txt", "r", stdin);
    //freopen("resout.txt", "w", stdout);
    int t = 1;
    //cin >> t;
    while (t--) {
        solve();
    }
    return 0;
}
```

简单环

```
#include <bits/stdc++.h>
#define ll long long
using namespace std;
const int MOD = 998244353;
const int maxn = 25;
ll ans[maxn], dp[1 << 20][maxn], num[1 << 20];
int n, m, k;
bool vis[maxn][maxn];
ll ksm(ll a, ll b)
{
    if (b == 0)
        return 1;
    if (b == 1)
        return a % MOD;
```

```cpp
    ll mid = ksm(a, b >> 1);
    if (b & 1)
        return mid * mid % MOD * a % MOD;
    else
        return mid * mid % MOD;
}
void Init()
{
    for (int st = 0; st < (1 << n); st++)
    {
        int cur = st, cnt = 0;
        while (cur)
        {
            if (cur & 1)
                cnt++;
            cur >>= 1;
        }
        num[st] = cnt;
    }
}
int main()
{
    scanf("%d%d%d", &n, &m, &k);
    Init();
    for (int i = 1; i <= m; i++)
    {
        int u, v;
        scanf("%d%d", &u, &v);
        vis[u][v] = true, vis[v][u] = true;
    }
    for (int i = 1; i <= n; i++)
        dp[1 << (i - 1)][i] = 1;
    for (int st = 1; st < (1 << n); st++)
    {
        int lowbit = st & (-st), s = 0;
        while (lowbit)
            s++, lowbit >>= 1;
        for (int j = 1; j <= n; j++)
            if (dp[st][j])
            {
                if (vis[j][s] && num[st] > 2)
                    ans[num[st] % k] = (ans[num[st] % k] + dp[st][j]) % MOD;
                for (int k = s + 1; k <= n; k++)
                    if ((st & (1 << (k - 1))) == 0 && vis[j][k])
                    {
                        int p = st | (1 << (k - 1));
                        dp[p][k] = (dp[p][k] + dp[st][j]) % MOD;
                    }
            }
    }
    ll invtwo = ksm(2, MOD - 2);
    for (int i = 1; i <= k; i++)
        cout << ans[i - 1] * invtwo % MOD << endl;
    return 0;
}
```

数位 dp

```cpp
/*
 * 第一行, 一个整数 T, 代表数据组数对于每组数据,
 * 有三个数字 l,r,n
 * 接下来 n 行, 每行一个数字 x, 接下来一个数 len 表示数字 x 在数字串中连续出现的次数不能大于 len
 * 对于每组数据
 * 输出一个整数, 表示 l,r 中满足约束的数字个数。(对 20020219 取模)
 */
#include <bits/stdc++.h>
#define endl '\n'
#define pll pair<ll, ll>
#define tll tuple<ll, ll, ll>
#define ios ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
```

```cpp
typedef long long ll;
const ll maxn = 20;
const ll mod = 20020219;
ll len[maxn];
ll a[maxn];
ll f[maxn][maxn][maxn];
ll l, r, n;

void solve()
{
    // flag 表示是否能直接返回值，也就是说前 pos-1 位与原数是否不同
    // 相同则这一位收到限制需要继续递归求解
    // 不同则不受限制，如果之前算过了可以直接返回
    function<ll(ll, ll, ll, bool)> dp = [&](ll pos, ll x, ll cnt, bool flag)
    {
        if (pos == 0)
            return 1ll;
        if (flag && f[pos][x][cnt])
            return f[pos][x][cnt];
        int up = flag ? 9 : a[pos];
        ll ans = 0;
        for (int i = 0; i <= up; ++i)
        {
            if (i == x)
            {
                if (cnt + 1 > len[i])
                    continue;
                ans = (ans + dp(pos - 1, i, cnt + 1, flag || (i < up))) % mod;
            }
            else
            {
                ans = (ans + dp(pos - 1, i, 1, flag || (i < up))) % mod;
            }
        }
        if (flag)
            f[pos][x][cnt] = ans % mod;
        return ans % mod;
    };
    function<ll(ll)> clac = [&](ll x)
    {
        int id = 0;
        while (x)
        {
            a[++id] = x % 10;
            x /= 10;
        }
        return dp(id, 0, 0, 0);
    };

    cin >> l >> r >> n;
    memset(len, 0x3f, sizeof(len));
    memset(f, 0, sizeof(f));
    for (int i = 1; i <= n; i++)
    {
        ll x, cnt;
        cin >> x >> cnt;
        len[x] = min(len[x], cnt);
    }
    cout << (clac(r) - clac(l - 1) + mod) % mod << "\n";
}

int main()
{
    ios;
    int t = 1;
    cin >> t;
    while (t--)
    {
        solve();
    }
    return 0;
}
```

很多线段树，树状数组

```
//线段树单点修改，区间查询
void build(int u, int l, int r)
{
    tr[u] = {l, r};
    if (l == r) return;
    int mid = (l + r) >> 1;
    build(u << 1, l, mid);
    build(u << 1 | 1, mid + 1, r);
}
//modify(1, position, valuse)
void modify(int u, int x, int v){
    if (tr[u].l == tr[u].r)
    {
        tr[u].v = v;
        return;
    }
    int mid = (tr[u].l + tr[u].r) >> 1;
    if (x <= mid)
    {
        modify(u << 1, x, v);
    }
    else
    {
        modify(u << 1 | 1, x, v);
    }
    tr[u].v = max(tr[u << 1].v, tr[u << 1 | 1].v);
}
//query(1, l, r)
int query(int u, int l, int r)
{
    if (l <= tr[u].l && r >= tr[u].r)
    {
        return tr[u].v;
    }

    int mid = (tr[u].l + tr[u].r) >> 1;
    int a = 0;
    int b = 0;
    if (l <= mid)
    {
        a = query(u << 1, l, r);
    }
    if (r > mid)
    {
        b = query(u << 1 | 1, l, r);
    }
    return max(a, b);
}

//线段树区间加，区间查询
ll a[maxn], w[maxn * 4], lazyTag[maxn * 4];
void pushup(int u) { w[u] = w[u << 1] + w[u << 1 | 1]; } // 上推
void makeTag(int u, int len, ll x)
{ // 下放 lazytag
    lazyTag[u] += x;
    w[u] += len * x;
}
void pushdown(int u, int L, int R)
{ // 下放 lazytag 的索引
    int M = (L + R) >> 1;
    makeTag(u << 1, M - L + 1, lazyTag[u]);
    makeTag(u << 1 | 1, R - M, lazyTag[u]);
    lazyTag[u] = 0;
}
void build(int u, int L, int R)
{ // 递归建树
    if (L == R)
    {
        w[u] = a[L];
        return;
    }
    int M = (L + R) >> 1;
    build(u << 1, L, M);
    build(u << 1 | 1, M + 1, R);
    pushup(u);
}
bool inRange(int L, int R, int l, int r) { return (l <= L) && (R <= r); }  // 判断 [L,R] 是否被 [l,r] 包含
bool outofRange(int L, int R, int l, int r) { return (L > r) || (R < l); } // 判断 [L,R] 是否与 [l,r] 完全无交集
ll query(int u, int L, int R, int l, int r)
{ // 区间查询
    if (inRange(L, R, l, r))
        return w[u];
    else if (!outofRange(L, R, l, r))
    {
        int M = (L + R) >> 1;
        pushdown(u, L, R);
        return query(u << 1, L, M, l, r) + query(u << 1 | 1, M + 1, R, l, r);
    }
    else
        return 0;
}
void update(int u, int L, int R, int l, int r, ll x)
{ // 区间修改
    if (inRange(L, R, l, r))
        makeTag(u, R - L + 1, x);
    else if (!outofRange(L, R, l, r))
    {
        int M = (L + R) >> 1;
        pushdown(u, L, R);
        update(u << 1, L, M, l, r, x);
        update(u << 1 | 1, M + 1, R, l, r, x);
        pushup(u);
    }
}
int main()
{
    int n, m;
    cin >> n >> m;
    for (int i = 1; i <= n; ++i)
    {
        cin >> a[i];
    }
    build(1, 1, n);
    for (int i = 1; i <= m; ++i)
    {
        int op, x, y;
        ll k;
        cin >> op;
        if (op == 1)
        {
            cin >> x >> y >> k;
            update(1, 1, n, x, y, k);
        }
        else
        {
            cin >> x >> y;
            cout << query(1, 1, n, x, y) << endl;
        }
    }
    return 0;
}
//线段树区间乘法，区间加法，区间查询
ll n, m, p;
int ls(int u) { return u << 1; }
int rs(int u) { return u << 1 | 1; }
int mid(int l, int r) { return (l + r) >> 1; }
struct Node
{
```

```cpp
    ll ad;
    ll mu = 1;
} tag[maxn * 4];
ll a[maxn], w[maxn * 4];
void pushup(int u) { w[u] = (w[ls(u)] + w[rs(u)]) % p; }
void build(int u, int L, int R)
{
    if (L == R)
    {
        w[u] = a[L] % p;
        return;
    }
    int M = mid(L, R);
    build(ls(u), L, M);
    build(rs(u), M + 1, R);
    pushup(u);
}
bool inRange(int L, int R, int l, int r) { return (l <= L) && (R <= r); }  // 判断 [L,R] 是否被 [l,r] 包含
bool outofRange(int L, int R, int l, int r) { return (L > r) || (R < l); } // 判断 [L,R] 是否与 [l,r] 完全无交集
void pushdown(int u, int L, int R)
{
    int M = mid(L, R);
    if (tag[u].mu != 1)
    {
        tag[ls(u)].ad = tag[ls(u)].ad * tag[u].mu % p;
        tag[rs(u)].ad = tag[rs(u)].ad * tag[u].mu % p;
        tag[ls(u)].mu = tag[ls(u)].mu * tag[u].mu % p;
        tag[rs(u)].mu = tag[rs(u)].mu * tag[u].mu % p;
        w[ls(u)] = w[ls(u)] * tag[u].mu % p;
        w[rs(u)] = w[rs(u)] * tag[u].mu % p;
        tag[u].mu = 1;
    }
    if (tag[u].ad)
    {
        w[ls(u)] = (w[ls(u)] + (M - L + 1) * tag[u].ad) % p;
        tag[ls(u)].ad = (tag[ls(u)].ad + tag[u].ad) % p;
        w[rs(u)] = (w[rs(u)] + (R - M) * tag[u].ad) % p;
        tag[rs(u)].ad = (tag[rs(u)].ad + tag[u].ad) % p;
        tag[u].ad = 0;
    }
}
ll query(int u, int L, int R, int l, int r)
{
    if (inRange(L, R, l, r))
        return w[u];
    else if (!outofRange(L, R, l, r))
    {
        int M = mid(L, R);
        pushdown(u, L, R);
        return (query(ls(u), L, M, l, r) + query(rs(u), M + 1, R, l, r)) % p;
    }
    else
        return 0;
}
void update1(int u, int L, int R, int l, int r, ll k)
{
    if (inRange(L, R, l, r))
    {
        tag[u].mu = tag[u].mu * k % p;
        tag[u].ad = tag[u].ad * k % p;
        w[u] = w[u] * k % p;
        return;
    }
    else if (!outofRange(L, R, l, r))
    {
        int M = mid(L, R);
        pushdown(u, L, R);
        update1(ls(u), L, M, l, r, k);
        update1(rs(u), M + 1, R, l, r, k);
        pushup(u);
    }
```

```cpp
}
void update2(int u, int L, int R, int l, int r, ll k)
{
    if (inRange(L, R, l, r))
    {
        tag[u].ad = (tag[u].ad + k) % p;
        w[u] = (w[u] + (R - L + 1) * k) % p;
        return;
    }
    else if (!outofRange(L, R, l, r))
    {
        int M = mid(L, R);
        pushdown(u, L, R);
        update2(ls(u), L, M, l, r, k);
        update2(rs(u), M + 1, R, l, r, k);
        pushup(u);
    }
}
int main()
{
    cin >> n >> m >> p;
    for (int i = 1; i <= n; ++i)
        cin >> a[i];
    build(1, 1, n);
    for (int i = 1; i <= m; ++i)
    {
        int op, x, y;
        ll k;
        cin >> op >> x >> y;
        if (op == 1)
        {
            cin >> k;
            update1(1, 1, n, x, y, k);
        }
        else if (op == 2)
        {
            cin >> k;
            update2(1, 1, n, x, y, k);
        }
        else
        {
            cout << query(1, 1, n, x, y) % p << endl;
        }
    }
    return 0;
}
//树状数组单点修改区间查询
#include <bits/stdc++.h>
using namespace std;
int tree[500010];
int n, m;
int lowbit(int x) { return x & -x; }
void add(int x, int k)
{
    while (x <= n)
    {
        tree[x] += k;
        x += lowbit(x);
    }
}
int sum(int x)
{
    int ans = 0;
    while (x != 0)
    {
        ans += tree[x];
        x -= lowbit(x);
    }
    return ans;
}
int main()
```

```cpp
{
    cin >> n;
    for (int i = 1; i <= n; i++)
    {
        int a;
        cin >> a;
        add(i, a);
    }
    cin >> m;
    for (int i = 1; i <= m; i++)
    {
        int u, v, w;
        cin >> u >> v >> w;
        if (u == 1)
            add(v, w);
        else if (u == 2)
            cout << sum(w) - sum(v - 1) << endl;
    }
    return 0;
}
//树状数组区间修改单点查询
#include <bits/stdc++.h>
using namespace std;
int tree[500010], y[500010];
int n, m;
int lowbit(int x) { return x & -x; }
void add(int x, int k)
{
    while (x <= n)
    {
        tree[x] += k;
        x += lowbit(x);
    }
}
int sum(int x)
{
    int ans = 0;
    while (x != 0)
    {
        ans += tree[x];
        x -= lowbit(x);
    }
    return ans;
}
int main()
{
    cin >> n >> m;
    for (int i = 1; i <= n; i++)
        cin >> y[i];
    for (int i = 1; i <= m; i++)
    {
        int u, a, b, c, v;
        cin >> u;
        if (u == 1)
        {
            cin >> a >> b >> c;
            add(a, c);
            add(b + 1, -c);
        }
        else if (u == 2)
        {
            cin >> v;
            cout << y[v] + sum(v) << endl;
        }
    }
    return 0;
}
//树状数组区间修改区间查询
#include <bits/stdc++.h>
using namespace std;
#define int long long
```

```cpp
#define MAXN (int)(1e6 + 5)

int n,m;
int a[MAXN];
struct BIT{
    int bit1[MAXN],bit2[MAXN];
    int lowbit(int x){return x & (-x);}
    void add(int i, int v){
        int k = v * i;
        while(i <= n){
            bit1[i] += v, bit2[i] += k; // 维护的重点部分
            i += lowbit(i);
        }
    }
    int sum(int *b, int i){
        int res = 0;
        while(i >= 1){
            res += b[i];
            i -= lowbit(i);
        }
        return res;
    }
    int qry(int l, int r){
        return sum(bit1, r) * (r + 1) - sum(bit1, l - 1) * l - (sum(bit2, r) - sum(bit2, l - 1));
    }
}bt;

signed main(){
    cin >> n >> m;
    for(int i = 1; i <= n; i++)
        cin >> a[i], bt.add(i, a[i] - a[i - 1]);
    while(m--){
        int op, l, r, x;
        cin >> op >> l >> r;
        if(op == 1){cin >> x; bt.add(l, x), bt.add(r + 1, -x);}
        if(op == 2){printf("%lld\n", bt.qry(l, r));}
    }
    return 0;
}
```

快速幂

```cpp
int qpow(long long a, int b) {
  int ans = 1;
  a = (a % p + p) % p;
  for (; b; b >>= 1) {
    if (b & 1) ans = (a * ans) % p;
    a = (a * a) % p;
  }
  return ans;
}
```

lucas

```cpp
long long Lucas(long long n, long long m, long long p) {
  if (m == 0) return 1;
  return (C(n % p, m % p, p) * Lucas(n / p, m / p, p)) % p;
}
```

各种背包

```cpp
//01
for (int i = 1; i <= n; i++)
  for (int l = W; l >= w[i]; l--) f[l] = max(f[l], f[l - w[i]] + v[i]);
//完全
for (int i = 1; i <= n; i++)
    for (int l = w[i]; l <= W; l++)
      if (f[l - w[i]] + v[i] > f[l]) f[l] = f[l - w[i]] + v[i];
//分组
```

```
for (int k = 1; k <= ts; k++)           // 循环每一组
  for (int i = m; i >= 0; i--)          // 循环背包容量
    for (int j = 1; j <= cnt[k]; j++)  // 循环该组的每一个物品
      if (i >= w[t[k][j]])             // 背包容量充足
        dp[i] = max(dp[i], dp[i - w[t[k][j]]] + c[t[k][j]]);  // 像 0-1 背包一样状态转移
```

Z 函数

```
vector<int> z_function_trivial(string s) {
  int n = (int)s.length();
  vector<int> z(n);
  for (int i = 1; i < n; ++i)
    while (i + z[i] < n && s[z[i]] == s[i + z[i]]) ++z[i];
  return z;
}
```