计算集合

叉积求多边形面积

#include <bits/stdc++.h>

#define x first

#define y second

using namespace std;

typedef pair<double, double> PDD; // <double, double>

const int N = 110;

int n;

PDD p[N];

double get\_s(PDD a, PDD b, PDD c) { // 计算三角形面积

return ((b.x - a.x) \* (c.y - a.y) - (c.x - a.x) \* (b.y - a.y)) / 2.0;

}

double get\_sum() { // 叉积求多边形面积

double sum = 0;

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

sum += get\_s(p[1], p[i], p[i + 1]);

}

return sum;

}

int main() {

cin >> n;

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

cin >> p[i].x >> p[i].y;

}

cout << get\_sum() << endl;

return 0;

}

字符串

Ac自动机

#include <bits/stdc++.h>

using namespace std;

const int N = 1000010;

int n;

char str[N];

int tr[N][26], idx;

int fail[N];

int q[N], cnt[N];

// AC自动机

void insert() {

int p = 0;

for (int i = 0; str[i]; i++) {

int t = str[i] - 'a';

if (tr[p][t] == 0) tr[p][t] = ++idx;

p = tr[p][t];

}

cnt[p]++;

}

void build() {

int hh = 0, tt = -1;

for (int i = 0; i < 26; i++) {

if (tr[0][i] != 0) {

q[++tt] = tr[0][i];

}

}

while (hh <= tt) {

int t = q[hh++];

for (int k = 0; k < 26; k++) {

if (tr[t][k] == 0) {

tr[t][k] = tr[fail[t]][k];

} else {

fail[tr[t][k]] = tr[fail[t]][k];

q[++tt] = tr[t][k];

}

}

}

}

int main() {

cin >> n;

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

cin >> str;

insert();

}

build();

cin >> str;

int res = 0;

for (int i = 0, j = 0; str[i]; i++) {

int t = str[i] - 'a';

j = tr[j][t];

int p = j;

while (p && cnt[p] != -1) {

res += cnt[p];

cnt[p] = -1;

}

}

cout << res << endl;

return 0; }

字符串

Exkmp

#include <bits/stdc++.h>

using namespace std;

typedef pair<int, int> PII;

const int N = 200010;

int n, m;

int z[N];

char s[N], s1[N], s2[N];

void exkmp() { // 扩展kmp

cin >> (s1 + 1) >> (s2 + 1);

n = strlen(s1 + 1);

m = strlen(s2 + 1);

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

s[i] = s1[i];

}

s[n + 1] = '#';

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

s[n + i + 1] = s2[i];

}

z[1] = 0;

int L = 1, R = 0;

for (int i = 2; i <= n + m + 1; i++) {

if (i > R) {

z[i] = 0;

} else {

int k = i - L + 1;

z[i] = min(z[k], R - i + 1);

}

while (z[i] + i <= n + m + 1 && s[z[i] + 1] == s[z[i] + i]) {

++z[i];

}

if (i + z[i] - 1 > R) {

R = i + z[i] - 1;

L = i;

}

}

}

int main() {

return 0;

}

字符串

kmp

#include <bits/stdc++.h>

using namespace std;

const int N = 1000;

int n, m;

char A[N], B[N];

int ne[N];

void kmp() { // kmp

cin >> (A + 1);

cin >> (B + 1);

n = strlen(A + 1);

m = strlen(B + 1);

int len = 0;

for (int i = 2, j = 0; i <= m; i++) {

while (j && B[i] != B[j + 1]) j = ne[j];

if (B[i] == B[j + 1]) {

j++;

}

ne[i] = j;

}

for (int i = 1, j = 0; i <= n; i++) {

while (j && A[i] != B[j + 1]) j = ne[j];

if (A[i] == B[j + 1]) {

j++;

}

}

}

int main() {

kmp();

return 0;

}

字符串

manacher

#include <bits/stdc++.h>

using namespace std;

typedef long long ll;

typedef pair<ll, int> PII;

const int N = 1000010, M = N \* 2, MOD = 998244353;

int T, n, m, k;

char a[N \* 2], b[N \* 2];

int p[N \* 2];

void manacher() { // 马拉车算法

n = strlen(a + 1);

m = 0;

b[++m] = '#';

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

b[++m] = a[i];

b[++m] = '#';

}

int M = 0, R = 0;

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

if (i > R) {

p[i] = 1;

} else {

p[i] = min(p[2 \* M - i], R - i + 1);

}

while (i - p[i] > 0 && i + p[i] <= m && b[i - p[i]] == b[i + p[i]]) {

++p[i];

}

if (i + p[i] - 1 > R) {

R = i + p[i] - 1;

M = i;

}

}

int ans = 0;

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

ans = max(ans, p[i]);

}

cout << ans - 1 << endl;

}

void solve() {

cin >> (a + 1);

manacher();

}

int main() {

T = 1;

// cin >> T;

while (T--) {

solve();

}

return 0; }

杂

kruskal

#include <bits/stdc++.h>

using namespace std;

const int N = 110, M = 220;

struct Node

{

int a, b, w;

bool operator< (const Node &t) const

{

return w < t.w;

}

}e[M];

int n, m;

int p[N];

int find(int x)

{

return p[x] == x ? p[x] : p[x] = find(p[x]);

}

int main()

{

cin >> n >> m;

for (int i = 1; i <= n; i ++ ) p[i] = i;

for (int i = 0; i < m; i ++ )

{

int a, b, c;

cin >> a >> b >> c;

e[i] = {a, b, c};

}

sort(e, e + m);

int ans = 0;

for (int i = 0; i < m; i ++ )

{

int a = e[i].a, b = e[i].b, c = e[i].w;

a= find(a), b = find(b);

if (a != b) p[a] = b;

else ans += c;

}

cout << ans << endl;

return 0;

}

杂

prim

#include <bits/stdc++.h>

using namespace std;

const int N = 110;

int n;

int g[N][N];

int dist[N];

bool st[N];

int prim()

{

int ans = 0;

memset(dist, 0x3f, sizeof dist);

// 从起点1出发的最小生成树

dist[1] = 0;

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

{

int t = -1;

for (int j = 1; j <= n; j ++ )

if (!st[j] && (t == -1 || dist[t] > dist[j]))

t = j;

st[t] = true;

ans += dist[t];

for (int j = 1; j <= n; j ++ ) dist[j] = min(dist[j], g[t][j]);

}

return ans;

}

int main()

{

cin >> n;

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

for (int j = 1; j <= n; j ++ )

cin >> g[i][j];

cout << prim() << endl;

return 0;

}

杂

tajan求lca

#include <bits/stdc++.h>

#define x first

#define y second

using namespace std;

typedef pair<int, int> PII;

const int N = 10010, M = 40010;

int n, m;

int h[N], e[M], w[M], ne[M], idx;

int dist[N], res[M], st[N];

int p[N];

vector<PII> op[N];

int find(int x)

{

return p[x] == x ? p[x] : p[x] = find(p[x]);

}

void add(int a, int b, int c)

{

e[idx] = b, w[idx] = c, ne[idx] = h[a], h[a] = idx ++ ;

}

void dfs(int u, int father)

{

for (int i = h[u]; ~i; i = ne[i])

{

int j = e[i];

if (j != father)

{

dist[j] = dist[u] + w[i];

dfs(j, u);

}

}

}

void tarjan(int u)

{

st[u] = 1;

for (int i = h[u]; ~i; i = ne[i])

{

int j = e[i];

if (!st[j])

{

tarjan(j);

p[j] = u;

}

}

for (auto t : op[u])

{

int k = t.x, id = t.y;

if (st[k] == 2)

{

int f = find(k);

res[id] = dist[k] + dist[u] - 2 \* dist[f];

}

}

st[u] = 2;

}

int main()

{

cin >> n >> m;

memset(h, -1, sizeof h);

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

{

int a, b, c;

cin >> a >> b >> c;

add(a, b, c), add(b, a, c);

}

for (int i = 0; i < m; i ++ )

{

int a, b;

cin >> a >> b;

op[a].push\_back({b, i});

op[b].push\_back({a, i});

}

for (int i = 1; i <= n; i ++ ) p[i] = i;

dfs(1, -1);

tarjan(1);

for (int i = 0; i < m; i ++ ) cout << res[i] << endl;

return 0;

}

杂

差分约束

#include <bits/stdc++.h>

using namespace std;

int main()

{

//最小值用最长路 dist[j] < dist[t] + w[i]

// a >= b + 1

add(b, a, 1);

// 最大值用最短路 dist[j] > dist[t] + w[i]

// a <= b + 1

add(b, a, 1);

return 0;

}

杂

数字整除

/\*

小学数奥

n 能被 3 整除

各位数字之和能被 3 整除

n 能被 9 整除

各位数字之和能被 9 整除

n 能被 5 整除

最后一位能被 5 整除

n 能被 2 整除

最后一位能被 2 整除

n 能被 4 整除

最后俩位能被整除

n 能被 8 整除

最后三位能被整除

\*/

杂

树的直径

#include <bits/stdc++.h>

using namespace std;

const int N = 100010, M = N \* 2;

int n, m, res;

int h[N], e[M], w[M], ne[M], idx;

int d[N], q[N], pre[N];

int ans;

void add(int a, int b, int c) {

e[idx] = b; w[idx] = c; ne[idx] = h[a]; h[a] = idx++;

}

int bfs(int u) //bfs求距离u最远的节点

{

memset(d, -1, sizeof d);

d[u] = 0; //起点

int hh = 0, tt = 0;

q[0] = u; //起点加入队列

while(hh <= tt) //bfs求所有点到u的最短距离

{

int t = q[hh++];

for(int i = h[t]; i != -1; i = ne[i])

{

int j = e[i];

if(d[j] == -1)

{

d[j] = d[t] + 1;

pre[j] = i;

q[++tt] = j;

}

}

}

//枚举所有点找出距离u最远的点

int p = u;

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

if(d[i] > d[p])

p = i;

return p;

}

int dfs(int u, int father) //用dfs求树的直径

{

int dist1 = 0, dist2 = 0;

for(int i = h[u]; i != -1; i = ne[i])

{

int j = e[i];

if(j == father) continue;

int dist = dfs(j, u) + w[i];

if(dist >= dist1) dist2 = dist1, dist1 = dist;

else if(dist > dist2) dist2 = dist;

}

ans = max(ans, dist1 + dist2);

return dist1;

}

void update(int ed, int st) //从 ed 回推到 st，并将路上经过的边都取反

{

while(ed != st)

{

w[pre[ed]] = -1; //正向边取反

w[pre[ed] ^ 1] = -1; //反向边取反

ed = e[pre[ed] ^ 1]; //退到这条边的入点，即上一步走到的点

}

}

int main() {

cin >> n >> m;

memset(h, -1, sizeof h);

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

int a, b;

cin >> a >> b;

add(a, b, 1); add(b, a, 1);

}

int p = bfs(1);

int q = bfs(p);

res = 2 \* (n - 1) - d[q] + 1;

if (m == 2) {

update(q, p);

dfs(1, -1);

res = res - ans + 1;

}

cout << res << endl;

return 0;

}

杂

自定义排序

#include <bits/stdc++.h>

using namespace std;

typedef long long ll;

typedef pair<int, int> PII;

const int N = 100010, M = 210, MOD = 998244353;

struct Node {

int val;

bool operator< (const Node &t) const {

return val < t.val; // 从小到大排序

}

}a[N];

int T, n, m, k;

multiset<int> set;

void solve() {

cin >> n;

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

cin >> a[i].val;

}

sort(a + 1, a + 1 + n);

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

cout << a[i].val << " ";

}

}

int main() {

T = 1;

// cin >> T;

while (T--) {

solve();

}

return 0;

}

杂

最小表示法

#include <bits/stdc++.h>

using namespace std;

const int N = 100000;

int T, n, m, k;

char str[M];

int get\_min(char \*s) {

int i = 0, j = 1, k = 0;

while (i < n && j < n) {

for (k = 0; k < n && s[i + k] == s[j + k]; k++);

if (k == n) break;

if (s[i + k] > s[j + k]) {

i += k + 1;

if (i == j) i++;

} else {

j += k + 1;

if (i == j) j++;

}

}

return min(i, j);

}

int main() {

cin >> str;

n = strlen(str);

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

str[j] = str[i];

}

int t = get\_min(str);

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

cout << str[i + t];

}

cout << endl;

return 0;

}

图论

2-SAT

#include <bits/stdc++.h>

using namespace std;

const int N = 110 \* 2, M = 2010;

int T, n, m, depth, scc;

int h[N], e[M], ne[M], idx;

int dfn[N], low[N], bel[N];

bool st[N];

stack<int> stk;

void add(int a, int b) {

e[idx] = b; ne[idx] = h[a]; h[a] = idx++;

}

void tarjan(int u) {

dfn[u] = low[u] = ++depth;

st[u] = true;

stk.push(u);

for (int i = h[u]; i != -1; i = ne[i]) {

int j = e[i];

if (dfn[j] == 0) {

tarjan(j);

low[u] = min(low[u], low[j]);

} else {

if (st[j]) low[u] = min(low[u], dfn[j]);

}

}

if (dfn[u] == low[u]) {

++scc;

while (true) {

int t = stk.top();

stk.pop();

st[t] = false;

bel[t] = scc;

if (t == u) break;

}

}

}

void solve() {

cin >> n >> m;

idx = scc = depth = 0;

memset(h, -1, sizeof h);

for (int i = 0; i < 2 \* n; i++) {

dfn[i] = low[i] = 0;

}

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

char o1, o2;

int x1, x2;

cin >> o1 >> x1 >> o2 >> x2;

x1--; x2--;

int a = x1 \* 2 + (o1 == 'h'), b = x2 \* 2 + (o2 == 'h');

/\*

a | b == 1;

a == 0 -> b == 1

b == 0 -> a == 1

(a ^ 1) == 0 -> b == 1

(b ^ 1) == 0 -> a == 1

\*/

add(a ^ 1, b);

add(b ^ 1, a);

}

for (int i = 0; i < 2 \* n; i++) {

if (dfn[i] == 0) {

tarjan(i);

}

}

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

if (bel[2 \* i] == bel[2 \* i + 1]) {

cout << "BAD" << endl;

return;

}

}

cout << "GOOD" << endl;

}

int main() {

cin >> T;

while (T--) {

solve();

}

return 0;

}

图论

点分治

#include <bits/stdc++.h>

using namespace std;

typedef long long ll;

const int N = 50010, M = N \* 2, T = 510;

int n, k, rt, rts;

int h[N], e[M], ne[M], idx;

int sz[N], f[T], t[T];

bool vis[N];

ll ans;

void add(int a, int b) {

e[idx] = b; ne[idx] = h[a]; h[a] = idx++;

}

void dfs\_root(int u, int fa) {

sz[u] = 1;

int maxs = 0;

for (int i = h[u]; i != -1; i = ne[i]) {

int j = e[i];

if (j == fa || vis[j]) continue;

dfs\_root(j, u);

sz[u] += sz[j];

maxs = max(maxs, sz[j]);

}

maxs = max(maxs, sz[0] - sz[u]);

if (rts > maxs) {

rts = maxs;

rt = u;

}

}

void get\_dist(int u, int fa, int d) {

if (d > k) return;

t[d]++;

for (int i = h[u]; i != -1; i = ne[i]) {

int j = e[i];

if (j == fa || vis[j]) continue;

get\_dist(j, u, d + 1);

}

}

void solove(int u) {

for (int i = h[u]; i != -1; i = ne[i]) {

int j = e[i];

if (vis[j]) continue;

get\_dist(j, u, 1);

for (int p = 1; p < k; p++) {

int q = k - p;

ans = ans + f[p] \* t[q];

}

ans += t[k];

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

f[i] += t[i];

t[i] = 0;

}

}

memset(f, 0, sizeof f);

}

void divide(int u) {

vis[u] = true;

solove(u);

for (int i = h[u]; i != -1; i = ne[i]) {

int j = e[i];

if (vis[j]) continue;

rts = sz[0] = sz[j];

dfs\_root(j, u);

// cout << rt << endl;

divide(rt);

}

}

int main() {

cin >> n >> k;

memset(h, -1, sizeof h);

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

int a, b;

cin >> a >> b;

add(a, b);

add(b, a);

}

rts = sz[0] = n;

dfs\_root(1, -1);

// cout << rt << endl;

divide(rt);

cout << ans << endl;

return 0;

}

图论

割边

#include <bits/stdc++.h>

using namespace std;

const int N = 100010, M = 300010 \* 2;

int n, m, depth;

int dfn[N], low[N];

int h[N], e[M], ne[M], idx;

vector<int> ans;

void add(int a, int b) {

e[idx] = b; ne[idx] = h[a]; h[a] = idx++;

}

void dfs(int u, int id) {

dfn[u] = low[u] = ++depth;

for (int i = h[u]; i != -1; i = ne[i]) {

int j = e[i];

if (!dfn[j]) {

dfs(j, i);

low[u] = min(low[u], low[j]);

if (dfn[j] == low[j]) ans.push\_back(id + 1);

} else if ((id ^ 1) != i) low[u] = min(low[u], dfn[j]);

}

}

int main() {

cin >> n >> m;

memset(h, -1, sizeof h);

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

int a, b;

cin >> a >> b;

add(a, b);

add(b, a);

}

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

if (dfn[i] == 0) {

dfs(i, -1);

}

}

cout << ans.size() << endl;

return 0;

}

图论

割点

#include <bits/stdc++.h>

using namespace std;

const int N = 20010, M = 100010 \* 2;

int n, m, depth, root;

int h[N], e[M], ne[M], idx;

int dfn[N], low[N], cut[N];

void add(int a, int b) {

e[idx] = b; ne[idx] = h[a]; h[a] = idx++;

}

void tarjan(int u, int fa) {

dfn[u] = low[u] = ++depth;

int s = 0;

for (int i = h[u]; i != -1; i = ne[i]) {

int j = e[i];

if (dfn[j] == 0) {

tarjan(j, u);

s++;

low[u] = min(low[u], low[j]);

if (low[j] >= dfn[u]) {

cut[u] = 1;

}

} else {

if (j != fa) low[u] = min(low[u], dfn[j]);

}

}

if (root == u && s <= 1) cut[u] = 0;

}

int main() {

cin >> n >> m;

memset(h, -1, sizeof h);

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

int a, b;

cin >> a >> b;

add(a, b);

add(b, a);

}

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

if (dfn[i] == 0) {

root = i;

tarjan(i, 0);

}

vector<int> ans;

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

if (cut[i]) {

ans.push\_back(i);

}

cout << ans.size() << endl;

for (auto ver : ans) {

cout << ver << " ";

}

cout << endl;

return 0; }

图论

强连通分量

#include <bits/stdc++.h>

using namespace std;

typedef long long ll;

const int N = 100010, M = 1000010;

ll Hash = 10000000l;

int n, r, c, idx, depth, scc, top;

ll ans;

int h[M], e[M], ne[M];

int x[N], y[N], t[N];

vector<int> rs[M], cs[M];

map<ll, int> mp;

map<int, int> row, col;

int dfn[M], low[M], id[M];

int stk[M];

bool st[M];

ll f[N];

void tarjan(int u) {

dfn[u] = low[u] = ++depth;

st[u] = true;

stk[++top] = u;

for (int i = h[u]; i != -1; i = ne[i]) {

int j = e[i];

if (dfn[j] == 0) {

tarjan(j);

low[u] = min(low[u], low[j]);

} else {

if (st[j]) low[u] = min(low[u], dfn[j]);

}

}

if (dfn[u] == low[u]) {

++scc;

int cnt = 0;

f[scc] = 0;

while (true) {

int t = stk[top--];

st[t] = false;

id[t] = scc;

if (t <= n) cnt++;

for (int i = h[t]; i != -1; i = ne[i]) {

int j = e[i];

if (id[j] != 0 && id[j] != scc) {

f[scc] = max(f[scc], f[id[j]]);

}

}

if (t == u) break;

}

f[scc] += cnt;

ans = max(ans, f[scc]);

}

}

void add(int a, int b) {

e[idx] = b; ne[idx] = h[a]; h[a] = idx++;

}

int main() {

scanf("%d%d%d", &n, &r, &c);

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

scanf("%d%d%d", &x[i], &y[i], &t[i]);

ll s = (x[i]) \* Hash + y[i];

rs[x[i]].push\_back(i);

cs[y[i]].push\_back(i);

mp[s] = i;

}

memset(h, -1, sizeof h);

int tot = n;

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

if (t[i] == 1) {

if (row[x[i]] == 0) {

row[x[i]] = ++tot;

for (auto t : rs[x[i]]) {

add(tot, t);

}

}

add(i, row[x[i]]);

} else if (t[i] == 2) {

if (col[y[i]] == 0) {

col[y[i]] = ++tot;

for (auto t : cs[y[i]]) {

add(tot, t);

}

}

add(i, col[y[i]]);

} else if (t[i] == 3) {

for (int dx = -1; dx <= 1; dx++) {

for (int dy = -1; dy <= 1; dy++) {

if (dx == 0 && dy == 0) continue;

int tx = x[i] + dx, ty = y[i] + dy;

if (tx <= 0 || ty <= 0 || tx > r || ty > c) continue;

ll s = (tx) \* Hash + ty;

if (mp[s] != 0) {

add(i, mp[s]);

}

}

}

}

}

for (int i = 1; i <= tot; i++)

if (dfn[i] == 0) {

tarjan(i);

}

printf("%lld\n", ans);

}

图论

树上启发式合并

#include <bits/stdc++.h>

using namespace std;

typedef long long ll;

const int N = 100010, M = N \* 2, MOD = (int)1e9 + 7;

int n, m, idx, tot;

int h[N], e[M], ne[M];

int col[N];

int l[N], r[N], id[N], hs[N], sz[N];

int cnt[N];

ll ans[N];

ll maxcnt, maxsum;

void add(int a, int b) {

e[idx] = b; ne[idx] = h[a]; h[a] = idx++;

}

void dfs1(int u, int fa) {

l[u] = ++tot;

id[tot] = u;

hs[u] = -1;

sz[u] = 1;

for (int i = h[u]; i != -1; i = ne[i]) {

int j = e[i];

if (j == fa) continue;

dfs1(j, u);

sz[u] += sz[j];

if (hs[u] == -1 || sz[j] > sz[hs[u]]) {

hs[u] = j;

}

}

r[u] = tot;

}

void add(int u) {

u = col[u];

cnt[u]++;

if (cnt[u] > maxcnt) {

maxcnt = cnt[u];

maxsum = 0;

}

if (maxcnt == cnt[u]) maxsum += u;

}

void del(int u) {

u = col[u];

cnt[u]--;

}

void dfs2(int u, int fa, bool flag) {

for (int i = h[u]; i != -1; i = ne[i]) {

int j = e[i];

if (j == fa || j == hs[u]) continue;

dfs2(j, u, false);

}

if (hs[u] != -1) {

dfs2(hs[u], u, true);

}

for (int i = h[u]; i != -1; i = ne[i]) {

int j = e[i];

if (j == fa || j == hs[u]) continue;

for (int t = l[j]; t <= r[j]; t++) {

add(id[t]);

}

}

add(u);

ans[u] = maxsum;

if (!flag) {

maxsum = maxcnt = 0;

for (int i = l[u]; i <= r[u]; i++)

del(id[i]);

}

}

int main() {

cin >> n;

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

cin >> col[i];

}

memset(h, -1, sizeof h);

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

int a, b;

cin >> a >> b;

add(a, b);

add(b, a);

}

dfs1(1, 0);

dfs2(1, 0, false);

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

cout << ans[i] << " ";

}

cout << endl; }

算法

倍增求lca

#include <bits/stdc++.h>

using namespace std;

const int N = 40010, M = N \* 2;

int n, m;

int h[N], e[M], ne[M], idx;

int q[N], depth[N], fa[N][16];

void add(int a, int b)

{

e[idx] = b, ne[idx] = h[a], h[a] = idx ++ ;

}

void bfs(int root)

{

memset(depth, 0x3f, sizeof depth);

int head = 0, tail = -1;

q[ ++ tail] = root, depth[0] = 0, depth[root] = 1;

while (head <= tail)

{

int t = q[head ++ ];

for (int i = h[t]; ~i; i = ne[i])

{

int j = e[i];

if (depth[j] > depth[t] + 1)

{

depth[j] = depth[t] + 1;

q[ ++ tail] = j;

fa[j][0] = t;

for (int k = 1; k <= 15; k ++ )

fa[j][k] = fa[fa[j][k - 1]][k - 1];

}

}

}

}

int lca(int a, int b)

{

if (depth[a] < depth[b]) swap(a, b);

for (int k = 15; k >= 0; k -- )

if (depth[fa[a][k]] >= depth[b])

a = fa[a][k];

if (a == b) return a;

for (int k = 15; k >= 0; k -- )

if (fa[a][k] != fa[b][k])

{

a = fa[a][k];

b = fa[b][k];

}

return fa[a][0];

}

int main()

{

cin >> n;

memset(h, -1, sizeof h);

int root;

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

{

int a, b;

cin >> a >> b;

if (b == -1) root = a;

else add(a, b), add(b, a);

}

bfs(root);

cin >> m;

for (int i = 0; i < m; i ++ )

{

int a, b;

int t = lca(a, b);

if (t == a) cout << 1 << endl;

else if (t == b) cout << 2 << endl;

else cout << 0 << endl;

}

return 0;

}

算法

数位dp

#include <bits/stdc++.h>

using namespace std;

typedef long long ll;

typedef pair<ll, ll> PII;

typedef pair<PII, int> PIII;

const int N = 105, M = 1 << 11, MOD = 1e9 + 7, INF = 1e9, Eps = 1e-7;

int T, n, m, k;

ll l, r;

ll f[11][N][M];

vector<int> ve;

ll dp(int u, int state, int pre, int limit) {

if (u < 0) {

return !state;

}

if (!limit && !pre && f[k][u][state] != -1) return f[k][u][state];

int up = limit ? ve[u] : k - 1;

ll ans = 0;

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

int t = state;

if (!pre || i != 0) t = t ^ (1 << i);

ans += dp(u - 1, t, pre && i == 0, limit && i == up);

}

if (!limit && !pre) f[k][u][state] = ans;

return ans;

}

ll calc(ll x) {

ve.clear();

while (x) {

ve.push\_back(x % k);

x /= k;

}

return dp(ve.size() - 1, 0, 1, 1);

}

void solve() {

cin >> k >> l >> r;

cout << calc(r) - calc(l - 1) << endl;

}

int main() {

memset(f, -1, sizeof f);

T = 1;

cin >> T;

while (T--) {

solve();

}

return 0;

}

数论

乘法逆元

#include <bits/stdc++.h>

using namespace std;

typedef long long ll;

int main()

{

// MOD 是质数

// inv[0] = inv[1] = 1

// inv[i] = (MOD - MOD / i) \* inv[MOD % i] % MOD

// inv[i] = pow(i, MOD - 2)

}

数论

整除分块

typedef long long ll;

typedef pair<int, int> PII;

const int N = 1000010, M = 55, MOD = 1e9 + 7;

int T, n, m, k;

void solve() {

ll n, m;

cin >> n >> m;

ll res = n \* 1ll \* m;

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

ll d = m / i, l = i, r;

if (d == 0) {

r = n;

} else {

r = min(m / d, n);

}

res -= d \* (l + r) \* (r - l + 1) / 2;

i = r;

}

cout << res << endl;

}

int main() {

T = 1;

// cin >> T;

while (T--) {

solve();

}

return 0;

}

数论

高斯消元

typedef long long ll;

typedef pair<int, int> PII;

const int N = 110, M = N \* 2, MOD = 998244353;

const double eps = 1e-10;

int T, n, m, k;

double a[N][N], b[N];

void gauss() {

int l = 1;

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

for (int j = l; j <= n; j++)

if (abs(a[l][i]) < abs(a[j][i])) {

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

swap(a[l][k], a[j][k]);

}

swap(b[l], b[j]);

}

if (abs(a[l][i]) < eps) continue;

for (int j = 1; j <= n; j++)

if (j != l && abs(a[j][i]) > eps) {

double d = a[j][i] / a[l][i];

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

a[j][k] -= d \* a[l][k];

}

b[j] -= b[l] \* d;

}

l++;

}

if (l <= n) {

cout << "No Solution" << endl;

} else {

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

printf("%.2f\n", b[i] / a[i][i]);

}

}

}

void solve() {

cin >> n;

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

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

cin >> a[i][j];

}

cin >> b[i];

}

gauss();

}

int main() {

T = 1;

// cin >> T;

while (T--) {

solve();

}

return 0; }

数论

矩阵乘法快速幂

typedef long long ll;

typedef pair<int, int> PII;

const int N = 50010, M = N \* 2, MOD = 1e9 + 7;

int T;

ll n, b, k, x;

ll f[110], g[110][110];

void mul(ll a[][110], ll b[][110]) {

ll t[110][110] = {0};

for (int i = 0; i < x; i++)

for (int k = 0; k < x; k++)

if (a[i][k])

for (int j = 0; j < x; j++)

if (b[k][j])

t[i][j] = (t[i][j] + a[i][k] \* b[k][j]) % MOD;

memcpy(a, t, sizeof t);

}

void mul(ll a[], ll b[][110]) {

ll t[10] = {0};

for (int i = 0; i < x; i++)

for (int j = 0; j < x; j++)

t[i] = (t[i] + a[j] \* b[j][i]) % MOD;

memcpy(a, t, sizeof t);

}

void solve() {

cin >> n >> b >> k >> x;

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

ll t;

cin >> t;

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

g[j][(j \* 10 + t) % x]++;

}

}

f[0] = 1;

while (b) {

if (b & 1) {

mul(f, g);

}

mul(g, g);

b >>= 1;

}

cout << f[k] << endl;

}

int main() {

T = 1;

// cin >> T;

while (T--) {

solve();

}

return 0; }

数论

扩展欧几里得

#include <bits/stdc++.h>

using namespace std;

/\*

ax + by = c

d = exgcd(a, b, x, y)

x1 = x \* (c / d)

y1 = y \* (c / d)

-d / 2 <= t <= d / 2

x = x1 - (b / d) \* t

y = y1 + (a / d) \* t

最小正整数解 x = (x % (b / d) + (b / d)) % (b / d)

\*/

int exgcd(int a, int b, int &x, int &y) {

if (b == 0) {

x = 1;

y = 0;

return a;

}

int d = exgcd(b, a % b, y, x);

y -= a / b \* x;

return d;

}

int main() {

int a, b, c;

cin >> a >> b >> c;

int x, y;

int d = exgcd(a, b, x, y);

if (c % d != 0) {

cout << "-1" << endl;

} else {

int t = c / d;

cout << (-x \* t) << " " << (-y \* t) << endl;

}

return 0; }

数论

算术基本定理

#include <bits/stdc++.h>

using namespace std;

const int N = 1000010, M = 2010;

int n;

int a[N];

int primes[N], cnt;

bool st[N];

void get\_prime(int x)

{

for (int i = 2; i <= x; i ++ )

{

if (!st[i]) primes[cnt ++ ] = i;

for (int j = 0; i \* primes[j] <= x; j ++ )

{

st[i \* primes[j]] = true;

if (i % primes[j] == 0) break;

}

}

}

int main()

{

cin >> n;

get\_prime(n); // 求 n 的阶乘

for (int i = 0; i < cnt; i ++ )

{

int p = primes[i];

int s = 0;

for (int j = n; j; j /= p) s += j / p;

cout << p << " " << s << endl;

}

return 0;

}

数论

线性筛

#include <bits/stdc++.h>

using namespace std;

typedef long long LL;

const int N = (1 << 20) + 10;

int primes[N], cnt;

int minp[N];

bool st[N];

int sum[N];

void get\_primes(int n)

{

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

{

if (!st[i])

{

primes[cnt ++ ] = i;

minp[i] = i;

}

for (int j = 0; primes[j] \* i <= n; j ++ )

{

int t = primes[j] \* i;

st[t] = true;

minp[t] = primes[j];

if (i % primes[j] == 0) break;

}

}

}

int main()

{

get\_primes(N - 1);

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

cout << primes[i] << endl;

}

int x;

while (scanf("%d", &x) != -1)

{

int tot = 0, k = 0;

while (x > 1)

{

int t = minp[x];

sum[k] = 0;

while (x % t == 0)

{

x /= t;

sum[k] ++ ;

tot ++ ;

}

k ++ ;

}

LL res = 1;

for (int i = 1; i <= tot; i ++ ) res \*= i;

for (int i = 0; i < k; i ++ )

for (int j = 1; j <= sum[i]; j ++ )

res /= j;

cout << tot << " " << res << endl;

}

return 0;

}

数据结构

dfs序

typedef long long ll;

const int N = 1000010, M = 2 \* N, MOD = 1e9 + 7;

int n, m, rt, idx, tot;

int v[N], l[N], r[N];

ll tr[N];

int h[N], e[M], w[M], ne[M];

void add(int a, int b, int c) {

e[idx] = b; w[idx] = c; ne[idx] = h[a]; h[a] = idx++;

}

void add(int x, ll v) {

for (int i = x; i <= n; i += i & -i) {

tr[i] += v;

}

}

ll query(int x) {

ll ans = 0;

for (int i = x; i > 0; i -= i & -i) {

ans += tr[i];

}

return ans;

}

void dfs(int u, int fa) {

l[u] = ++tot;

for (int i = h[u]; i != -1; i = ne[i]) {

int j = e[i];

if (j == fa) continue;

dfs(j, u);

}

r[u] = tot;

}

int main() {

cin >> n >> m >> rt;

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

cin >> v[i];

}

memset(h, -1, sizeof h);

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

int a, b;

cin >> a >> b;

add(a, b, 0);

add(b, a, 0);

}

dfs(rt, -1);

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

int x = l[i];

add(x, v[i]);

}

while (m-- > 0) {

int type;

cin >> type;

if (type == 1) {

int x, y;

cin >> x >> y;

add(l[x], y);

} else {

int x;

cin >> x;

printf("%lld\n", query(r[x]) - query(l[x] - 1));

}

}

return 0;

}

数据结构

fhq平衡树

#include <bits/stdc++.h>

using namespace std;

const int N = 100010, INF = 1e9;

struct Tree {

int l, r;

int key, val;

int size;

}tr[N];

int T, n, idx;

int root;

int x, y, z;

void pushup(int u) {

tr[u].size = tr[tr[u].l].size + tr[tr[u].r].size + 1;

}

int get\_Node(int val) {

tr[++idx] = {0, 0, rand() \* N, val, 1};

return idx;

}

void split(int u, int val, int &x, int &y) {

if (u == 0) x = y = 0;

else {

if (tr[u].val <= val) {

x = u;

split(tr[u].r, val, tr[u].r, y);

} else {

y = u;

split(tr[u].l, val, x, tr[u].l);

}

pushup(u);

}

}

int merge(int x, int y) {

if (x == 0 || y == 0) return x + y;

if (tr[x].key <= tr[y].key) {

tr[x].r = merge(tr[x].r, y);

pushup(x);

return x;

} else {

tr[y].l = merge(x, tr[y].l);

pushup(y);

return y;

}

}

void insert(int val) {

split(root, val, x, y);

root = merge(x, merge(get\_Node(val), y));

}

void del(int val) {

split(root, val, x, z);

split(x, val - 1, x, y);

y = merge(tr[y].l, tr[y].r);

root = merge(x, merge(y, z));

}

void get\_rank(int val) {

split(root, val - 1, x, y);

cout << tr[x].size << endl;

root = merge(x, y);

}

void get\_val(int rank) {

int u = root;

while (u != 0) {

if (tr[tr[u].l].size + 1 == rank) {

break;

} else if (tr[tr[u].l].size >= rank) {

u = tr[u].l;

} else {

rank -= tr[tr[u].l].size + 1;

u = tr[u].r;

}

}

cout << tr[u].val << endl;

}

void get\_prev(int val) {

split(root, val - 1, x, y);

int u = x;

while (tr[u].r) {

u = tr[u].r;

}

cout << tr[u].val << endl;

root = merge(x, y);

}

void get\_next(int val) {

split(root, val, x, y);

int u = y;

while (tr[u].l) {

u = tr[u].l;

}

cout << tr[u].val << endl;

root = merge(x, y);

}

void solve() {

cin >> n;

insert(-INF);

insert(INF);

while (n--) {

int t, x;

cin >> t >> x;

// if (n == 7) break;

// cout << t << " " << x << endl;

if (t == 1) {

insert(x);

} else if (t == 2) {

del(x);

} else if (t == 3) {

get\_rank(x);

} else if (t == 4) {

get\_val(x + 1);

} else if (t == 5) {

get\_prev(x);

} else {

get\_next(x);

}

}

}

int main() {

T = 1;

// cin >> T;

while (T--) {

solve();

}

return 0;

}

数据结构

st表

#include <bits/stdc++.h>

using namespace std;

const int N = 100010, M = 20;

int n, m;

int a[N];

int f[N][M];

void init() {

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

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

if (j == 0) f[i][j] = a[i];

else f[i][j] = max(f[i][j - 1], f[i + (1 << j - 1)][j - 1]);

}

}

}

int query(int l, int r) {

int k = log(r - l + 1) / log(2);

return max(f[l][k], f[r - (1 << k) + 1][k]);

}

int main() {

scanf("%d%d", &n, &m);

for (int i = 1; i <= n; i++) scanf("%d", &a[i]);

init();

while (m--) {

int l, r;

scanf("%d%d", &l, &r);

printf("%d\n", query(l, r));

}

return 0;

}

数据结构

动态开点

typedef long long ll;

typedef pair<ll, ll> PII;

typedef pair<PII, int> PIII;

const int N = 15001000, M = N \* 2, MOD = 1e9 + 7, INF = 1e9, Eps = 1e-7;

struct Tree {

int l, r;

int sum, lazy;

}tr[N];

int T, n, m, k, tot;

int root;

void pushup(int u) {

tr[u].sum = tr[tr[u].l].sum + tr[tr[u].r].sum;

}

void pushdown(int u, int l, int r) {

if (tr[u].lazy) {

int mid = l + r >> 1;

int d = 1;

if (tr[u].lazy == 1) d = 0;

if (!tr[u].l) tr[u].l = ++tot;

if (!tr[u].r) tr[u].r = ++tot;

tr[tr[u].l].sum = (mid - l + 1) \* d;

tr[tr[u].r].sum = (r - (mid + 1) + 1) \* d;

tr[tr[u].l].lazy = tr[tr[u].r].lazy = tr[u].lazy;

tr[u].lazy = 0;

}

}

void modify(int &u, int L, int R, int l, int r, int lazy) {

if (!u) u = ++tot;

if (L >= l && R <= r) {

int d = 1;

if (lazy == 1) d = 0;

tr[u].sum = (R - L + 1) \* d;

tr[u].lazy = lazy;

return;

}

pushdown(u, L, R);

int mid = L + R >> 1;

if (l <= mid) modify(tr[u].l, L, mid, l, r, lazy);

if (r > mid) modify(tr[u].r, mid + 1, R, l, r, lazy);

pushup(u);

}

int query(int u, int L, int R, int l, int r) {

// cout << L << " " << R << " " << l << " " << r << endl;

if (L >= l && R <= r) {

return tr[u].sum;

}

pushdown(u, L, R);

int mid = L + R >> 1;

int ans = 0;

if (l <= mid) ans += query(tr[u].l, L, mid, l, r);

if (r > mid) ans += query(tr[u].r, mid + 1, R, l, r);

return ans;

}

void solve() {

scanf("%d%d", &n, &m);

modify(root, 1, n, 1, n, 2);

while (m--) {

int l, r, k;

scanf("%d%d%d", &l, &r, &k);

modify(root, 1, n, l, r, k);

printf("%d\n", query(root, 1, n, 1, n));

}

}

int main() {

T = 1;

// cin >> T;

while (T--) {

solve();

}

return 0; }

数据结构

可持久化trie

#include <bits/stdc++.h>

using namespace std;

typedef long long ll;

typedef pair<int, int> PII;

typedef pair<PII, int> PIII;

const int N = 600005, M = 600005 \* 25, MOD = 1e9 + 7, INF = 1e9, Eps = 1e-7;

int T, n, m, k;

int tr[M][2], root[N], max\_id[M], idx;

int s[N];

void insert(int i, int k, int p, int q) {

if (k < 0) {

max\_id[q] = i;

return;

}

int t = s[i] >> k & 1;

if (tr[p][t ^ 1]) tr[q][t ^ 1] = tr[p][t ^ 1];

tr[q][t] = ++idx;

insert(i, k - 1, tr[p][t], tr[q][t]);

max\_id[q] = max(max\_id[tr[q][0]], max\_id[tr[q][1]]);

}

int query(int x, int root, int sum) {

int p = root;

for (int i = 23; i >= 0; i--) {

int t = sum >> i & 1;

if (max\_id[tr[p][t ^ 1]] >= x) {

p = tr[p][t ^ 1];

} else {

p = tr[p][t];

}

}

return sum ^ s[max\_id[p]];

}

void solve() {

cin >> n >> m;

max\_id[0] = -1;

root[0] = ++idx;

insert(0, 23, 0, root[0]);

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

cin >> s[i];

s[i] = s[i] ^ s[i - 1];

root[i] = ++idx;

insert(i, 23, root[i - 1], root[i]);

}

while (m--) {

char ops[3];

cin >> ops;

if (\*ops == 'A') {

int x;

cin >> x;

++n;

s[n] = x ^ s[n - 1];

root[n] = ++idx;

insert(n, 23, root[n - 1], root[n]);

} else {

int l, r, x;

cin >> l >> r >> x;

cout << query(l - 1, root[r - 1], s[n] ^ x) << endl;

}

}

}

int main() {

T = 1;

// cin >> T;

while (T--) {

solve();

}

return 0;

}

数据结构

树链部分

typedef long long ll;

const int N = 30010, M = 2 \* N, MOD = 1e9 + 7;

struct Tree {

int l, r;

int sum, maxs;

}tr[N \* 4];

int n, m, tot;

int h[N], e[M], w[N], ne[M], idx;

int sz[N], fa[N], dep[N], id[N];

int l[N], r[N], hs[N], top[N];

void pushup(int u) {

tr[u].sum = tr[u << 1].sum + tr[u << 1 | 1].sum;

tr[u].maxs = max(tr[u << 1].maxs, tr[u << 1 | 1].maxs);

}

void build(int u, int l, int r) {

if (l == r) {

tr[u] = {l, r, w[id[l]], w[id[l]]};

} else {

tr[u] = {l, r};

int mid = l + r >> 1;

build(u << 1, l, mid);

build(u << 1 | 1, mid + 1, r);

pushup(u);

}

}

void modify(int u, int x, int v) {

if (tr[u].l >= x && tr[u].r <= x) {

tr[u].sum = tr[u].maxs = v;

return;

}

int mid = tr[u].l + tr[u].r >> 1;

if (x <= mid) modify(u << 1, x, v);

if (x > mid) modify(u << 1 | 1, x, v);

pushup(u);

}

int query\_maxs(int u, int l, int r) {

if (tr[u].l >= l && tr[u].r <= r) {

return tr[u].maxs;

}

int mid = tr[u].l + tr[u].r >> 1;

int maxs = -1e9;

if (l <= mid) maxs = max(maxs, query\_maxs(u << 1, l, r));

if (r > mid) maxs = max(maxs, query\_maxs(u << 1 | 1, l, r));

return maxs;

}

int query\_sum(int u, int l, int r) {

if (tr[u].l >= l && tr[u].r <= r) {

return tr[u].sum;

}

int mid = tr[u].l + tr[u].r >> 1;

int sum = 0;

if (l <= mid) sum += query\_sum(u << 1, l, r);

if (r > mid) sum += query\_sum(u << 1 | 1, l, r);

return sum;

}

void add(int a, int b) {

e[idx] = b; ne[idx] = h[a]; h[a] = idx++;

}

void dfs1(int u, int f) {

sz[u] = 1;

hs[u] = -1;

fa[u] = f;

dep[u] = dep[f] + 1;

for (int i = h[u]; i != -1; i = ne[i]) {

int j = e[i];

if (j == f) continue;

dfs1(j, u);

sz[u] += sz[j];

if (hs[u] == -1 || sz[hs[u]] < sz[j]) {

hs[u] = j;

}

}

}

void dfs2(int u, int t) {

l[u] = ++tot;

id[tot] = u;

top[u] = t;

if (hs[u] != -1) {

dfs2(hs[u], t);

}

for (int i = h[u]; i != -1; i = ne[i]) {

int j = e[i];

if (j == fa[u] || j == hs[u]) continue;

dfs2(j, j);

}

r[u] = tot;

}

int q\_maxs(int a, int b) {

int maxs = -1e9;

while (top[a] != top[b]) {

if (dep[top[a]] > dep[top[b]]) {

maxs = max(maxs, query\_maxs(1, l[top[a]], l[a]));

a = fa[top[a]];

} else {

maxs = max(maxs, query\_maxs(1, l[top[b]], l[b]));

b = fa[top[b]];

}

}

if (dep[a] >= dep[b]) {

maxs = max(maxs, query\_maxs(1, l[b], l[a]));

} else {

maxs = max(maxs, query\_maxs(1, l[a], l[b]));

}

return maxs;

}

int q\_sum(int a, int b) {

int sum = 0;

while (top[a] != top[b]) {

if (dep[top[a]] > dep[top[b]]) {

sum += query\_sum(1, l[top[a]], l[a]);

a = fa[top[a]];

} else {

sum += query\_sum(1, l[top[b]], l[b]);

b = fa[top[b]];

}

}

if (dep[a] >= dep[b]) {

sum += query\_sum(1, l[b], l[a]);

} else {

sum += query\_sum(1, l[a], l[b]);

}

return sum;

}

int main() {

cin >> n;

memset(h, -1, sizeof h);

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

int a, b;

cin >> a >> b;

add(a, b);

add(b, a);

}

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

cin >> w[i];

}

dfs1(1, 0);

dfs2(1, 1);

build(1, 1, n);

cin >> m;

while (m--) {

char ops[10];

cin >> ops;

if (ops[1] == 'M') {

int a, b;

cin >> a >> b;

cout << q\_maxs(a, b) << endl;

} else if (ops[1] == 'S') {

int a, b;

cin >> a >> b;

cout << q\_sum(a, b) << endl;

} else if (ops[1] == 'H') {

int x, v;

cin >> x >> v;

modify(1, l[x], v);

}

}

return 0; }

数据结构

文艺平衡树

#include <bits/stdc++.h>

using namespace std;

const int N = 100010, INF = 1e9;

struct Tree {

int l, r;

int key, val;

int lazy, size;

}tr[N];

int T, n, m, idx;

int root;

int x, y, z;

void pushup(int u) {

tr[u].size = tr[tr[u].l].size + tr[tr[u].r].size + 1;

}

void pushdown(int u) {

swap(tr[u].l, tr[u].r);

tr[tr[u].l].lazy ^= 1;

tr[tr[u].r].lazy ^= 1;

tr[u].lazy = 0;

}

int get\_Node(int val) {

tr[++idx] = {0, 0, rand() \* N, val, 0, 1};

return idx;

}

void split(int u, int size, int &x, int &y) {

if (u == 0) x = y = 0;

else {

if (tr[u].lazy) pushdown(u);

if (tr[tr[u].l].size < size) {

x = u;

split(tr[u].r, size - tr[tr[u].l].size - 1, tr[u].r, y);

} else {

y = u;

split(tr[u].l, size, x, tr[u].l);

}

pushup(u);

}

}

int merge(int x, int y) {

if (x == 0 || y == 0) return x + y;

if (tr[x].key <= tr[y].key) {

if (tr[x].lazy) pushdown(x);

tr[x].r = merge(tr[x].r, y);

pushup(x);

return x;

} else {

if (tr[y].lazy) pushdown(y);

tr[y].l = merge(x, tr[y].l);

pushup(y);

return y;

}

}

void insert(int val) {

root = merge(root, get\_Node(val));

}

void prin(int u) {

if (u == 0) return;

if (tr[u].lazy) {

pushdown(u);

}

prin(tr[u].l);

cout << tr[u].val << " ";

prin(tr[u].r);

}

void rev(int l, int r) {

split(root, l - 1, x, y);

split(y, r - l + 1, y, z);

tr[y].lazy ^= 1;

root = merge(x, merge(y, z));

}

void solve() {

cin >> n >> m;

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

insert(i);

}

while (m--) {

int l, r;

cin >> l >> r;

rev(l, r);

}

prin(root);

}

int main() {

T = 1;

// cin >> T;

while (T--) {

solve();

}

return 0;

}

数据结构

主席树

#include <bits/stdc++.h>

using namespace std;

typedef long long ll;

typedef pair<int, int> PII;

typedef pair<PII, int> PIII;

const int N = 100005, M = 600005 \* 25, MOD = 1e9 + 7, INF = 1e9, Eps = 1e-7;

struct Tree {

int l, r;

int cnt;

}tr[N \* 20];

int T, n, m, k;

int a[N], root[N], idx;

vector<int> ve[N];

void pushup(int u) {

tr[u].cnt = tr[tr[u].l].cnt + tr[tr[u].r].cnt;

}

int build(int l, int r) {

int p = ++idx;

if (l == r) return p;

int mid = l + r >> 1;

tr[p].l = build(l, mid);

tr[p].r = build(mid + 1, r);

return p;

}

int insert(int p, int l, int r, int x) {

int q = ++idx;

tr[q] = tr[p];

if (l == r) {

tr[q].cnt++;

return q;

}

int mid = l + r >> 1;

if (x <= mid) tr[q].l = insert(tr[p].l, l, mid, x);

else tr[q].r = insert(tr[p].r, mid + 1, r, x);

pushup(q);

return q;

}

int query(int p, int q, int l, int r, int L, int R) {

// cout << l << " " << r << " " << L << " " << R << endl;

if (l >= L && r <= R) {

return tr[q].cnt - tr[p].cnt;

}

int mid = l + r >> 1;

int ans = 0;

if (L <= mid) ans += query(tr[p].l, tr[q].l, l, mid, L, R);

if (R > mid) ans += query(tr[p].r, tr[q].r, mid + 1, r, L, R);

return ans;

}

void solve() {

cin >> n >> k;

root[0] = build(0, n);

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

cin >> a[i];

ve[a[i]].push\_back(i);

int t = 0;

if (ve[a[i]].size() > k) {

t = ve[a[i]][ve[a[i]].size() - k - 1];

}

root[i] = insert(root[i - 1], 0, n, t);

}

cin >> m;

int last = 0;

while (m--) {

int l, r;

cin >> l >> r;

l = (l + last) % n + 1;

r = (r + last) % n + 1;

if (l > r) swap(l, r);

last = query(root[l - 1], root[r], 0, n, 0, l - 1);

cout << last << endl;

}

}

int main() {

T = 1;

// cin >> T;

while (T--) {

solve();

}

return 0;

}