**Team Note**

**ITPTIT\_ROYAL NEVER GIVE UP**

**  **

**Tạo ra 1 class Trie:**

1. **class** Trie {
3. }

Thứ nhất, Trie là cây, mà cây thì phải có nút:

1. **struct** node {
2. **int** num, index, w\_count**;**
3. **int** child[26]**;**
4. }**;**

trong đó, index là địa chỉ của node, num là số từ đi qua node đó lúc insert, w\_count là số từ kết thúc tại node đó, child là mảng lưu địa chỉ tới con của node đó, child[26] nghĩa là 26 chữ cái.

**Khởi tạo và phá bỏ cây:**

1. vector<node> tree**;**
2. **inline** **void** new\_node() {
3. node aw**;**
4. aw.num = 0**;**
5. **for** (**int** i = 0**;** i < 10**;** i++) aw.child[i] = -1**;**
6. aw.index = tree.size()**;**
7. tree.push\_back(aw)**;**
8. }
9. **inline** **void** destruct() {
10. tree.clear()**;**
11. }
13. **inline** **void** init() {
14. new\_node()**;**
15. }

Khởi tạo gốc là node rỗng tương đương với việc cho xâu "" vào trong cây

**Thao tác thêm từ vào cây (add\_word):**

1. **inline** **void** add\_word(**const** string& s) {
2. **int** w**;**
3. **int** indx = 0**;**
4. **for** (**int** i = 0**;** i < s.size()**;** i++) {
5. tree[indx].num++**;**
6. w = s[i] - 'a'**;**
7. **if** (tree[indx].child[w] == -1) {
8. new\_node()**;**
9. tree[indx].child[w] = tree.size() - 1**;**
10. }
11. indx = tree[indx].child[w]**;**
12. }
13. tree[indx].num++**;**
14. tree[indx].w\_count++**;**
15. }

**Thao tác đếm xem có bao nhiêu từ bắt đầu bởi s (count\_pref):**

1. **inline** **int** count\_pref(**const** string& s) {
2. **int** w**;**
3. **int** indx = 0**;**
4. **for** (**int** i = 0**;** i < s.size()**;** i++) {
5. w = s[i] - 'a'**;**
6. **if** (tree[indx].child[w] == -1) **return** 0**;**
7. indx = tree[indx].child[w]**;**
8. }
9. **return** tree[indx].num**;**
10. }

**Đếm xem có bao nhiêu xâu s trong dãy (count\_word):**

1. **inline** **int** count\_word(**const** string& s) {
2. **int** w**;**
3. **int** indx = 0**;**
4. **for** (**int** i = 0**;** i < s.size()**;** i++) {
5. w = s[i] - 'a'**;**
6. **if** (tree[indx].child[w] == -1) **return** 0**;**
7. indx = tree[indx].child[w]**;**
8. }
9. **return** tree[indx].w\_count**;**
10. }

**Thao tác xóa xâu khỏi cây (del\_word):**

1. **inline** **void** del\_word(**const** string& s) {
2. **if** (!count\_pref(s)) **return;** *// không có thì không xóa*
3. **int** w**;**
4. **int** indx = 0**;**
5. **for** (**int** i = 0**;** i < s.size()**;** i++) {
6. tree[indx].num--**;**
7. w = s[i] - 'a'**;**
8. indx = tree[indx].child[w]**;**
9. }
10. tree[indx].num--**;**
11. tree[indx].w\_count--**;**
12. }

**Hash :**

#include <stdio.h>  
#include <string.h>  
#include <iostream>  
#include <algorithm>  
using namespace std;  
  
#define long long long  
const int N=1000006, BASE=1000000007;  
int m, n;  
char a[N], b[N];  
long A[N], B[N], M[N];  
  
**void hash\_build(char a[], int n, long H[]) {**  
    for (int i=1; i<=n; i++)  
    H[i] = (H[i-1] \* M[1] + a[i]) % BASE;  
}  
  
**long hash\_range(long H[], int L, int R) {**  
    return (H[R] - H[L-1]\*M[R-L+1] + 1LL\*BASE\*BASE) % BASE;  
}  
  
**main() {**  
    M[0]=1; M[1]=2309;  
    for (int i=2; i<N; i++)  
    M[i] = M[i-1] \* M[1] % BASE;  
      
    scanf("%s", a+1); m=strlen(a+1);  
    scanf("%s", b+1); n=strlen(b+1);  
    hash\_build(a, m, A);  
    hash\_build(b, n, B);  
    for (int i=1; i<=m-n+1; i++) {  
        if (hash\_range(A, i, i+n-1) == B[n])  
        printf("%d ", i);  
    }  
    printf("\n");  
}

**Dãy con tăng dài nhất :**

int n;

ll ans = 0, ans1 = 0;

cin >> n;

ll a,b,h;

pair < ll, pair < ll, ll > > p[n + 1];

stack < int > s;

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

cin >> a >> b >> h;

p[i].first = b;

p[i].second.first = a;

p[i].second.second = h;

}

sort(p, p + n);

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

while (!s.empty() && p[i].first <= p[s.top()].second.first) {

ans -= p[s.top()].second.second;

s.pop();

}

s.push(i);

ans += p[i].second.second;

ans1 = max(ans1, ans);

}

cout << ans1 << endl;

**Segment Tree:**

*// Truy vấn: A(i) = v*

*// Hàm cập nhật trên cây ST, cập nhật cây con gốc id quản lý đọan [l, r]*

**void** **update**(**int** id, **int** l, **int** r, **int** i, **int** v) {

**if** (i **<** l **||** r **<** i) {

*// i nằm ngoài đoạn [l, r], ta bỏ qua nút i*

**return** ;

}

*// i nằm trong đoạn [l, r], ta cần cập nhật nút id*

ST[id] **=** max(ST[id], v);

**if** (l **==** r) {

*// Đoạn chỉ gồm 1 phần tử, không có nút con*

**return** ;

}

*// Gọi đệ quy để xử lý các nút con của nút id*

**int** mid **=** (l **+** r) **/** 2;

update(id**\***2, l, mid, i, v);

update(id**\***2 **+** 1, mid**+**1, r, i, v);

}

*// Truy vấn: tìm max đoạn [u, v]*

*// Hàm tìm max các phần tử trên cây ST nằm trong cây con gốc id - quản lý đoạn [l, r]*

**int** **get**(**int** id, **int** l, **int** r, **int** u, **int** v) {

**if** (v **<** l **||** r **<** u) {

*// Đoạn [u, v] không giao với đoạn [l, r], ta bỏ qua đoạn này*

**return** **-**INFINITY;

}

**if** (u **<=** l **&&** r **<=** v) {

*// Đoạn [l, r] nằm hoàn toàn trong đoạn [u, v] mà ta đang truy vấn, ta trả lại*

*// thông tin lưu ở nút id*

**return** ST[id];

}

**int** mid **=** (l **+** r) **/** 2;

*// Gọi đệ quy với các con của nút id*

**return** max(get(id**\***2, l, mid, u, v), get(id**\***2 **+** 1, mid**+**1, r, u, v));

}

**IT Lazy Update**

Ta có kiểu dữ liệu cho 1 nút của ST như sau:

**struct** Node {

**int** lazy; *// giá trị T trong phân tích trên*

**int** val; *// giá trị lớn nhất.*

} nodes[MAXN **\*** 4];

Hàm "đẩy" giá trị TT xuống các con:

**void** **down**(**int** id) {

**int** t **=** nodes[id].lazy;

nodes[id**\***2].lazy **+=** t;

nodes[id**\***2].val **+=** t;

nodes[id**\***2**+**1].lazy **+=** t;

nodes[id**\***2**+**1].val **+=** t;

nodes[id].lazy **=** 0;

}

Hàm cập nhật:

**void** **update**(**int** id, **int** l, **int** r, **int** u, **int** v, **int** val) {

**if** (v **<** l **||** r **<** u) {

**return** ;

}

**if** (u **<=** l **&&** r **<=** v) {

*// Khi cài đặt, ta LUÔN ĐẢM BẢO giá trị của nút được cập nhật ĐỒNG THỜI với*

*// giá trị lazy propagation. Như vậy sẽ tránh sai sót.*

nodes[id].val **+=** val;

nodes[id].lazy **+=** val;

**return** ;

}

**int** mid **=** (l **+** r) **/** 2;

down(id); *// đẩy giá trị lazy propagation xuống các con*

update(id**\***2, l, mid, u, v, val);

update(id**\***2**+**1, mid**+**1, r, u, v, val);

nodes[id].val **=** max(nodes[id**\***2].val, nodes[id**\***2**+**1].val);

}

Hàm lấy giá trị lớn nhất:

**int** **get**(**int** id, **int** l, **int** r, **int** u, **int** v) {

**if** (v **<** l **||** r **<** u) {

**return** **-**INFINITY;

}

**if** (u **<=** l **&&** r **<=** v) {

**return** nodes[id].val;

}

**int** mid **=** (l **+** r) **/** 2;

down(id); *// đẩy giá trị lazy propagation xuống các con*

**return** max(get(id**\***2, l, mid, u, v),

get(id**\***2**+**1, mid**+**1, r, u, v));

*// Trong các bài toán tổng quát, giá trị ở nút id có thể bị thay đổi (do ta đẩy lazy propagation*

*// xuống các con). Khi đó, ta cần cập nhật lại thông tin của nút id dựa trên thông tin của các con.*

}

**Kruscal – Cây khung nhỏ nhất**

#include <bits/stdc++.h>

#define FOR(x, a, b) for (int x = a; x <= b; ++x)

#define FOD(x, a, b) for (int x = a; x >= b; --x)

#define REP(x, a, b) for (int x = a; x < b; ++x)

#define fi first

#define se second

#define ll long long

using namespace std;

const int maxn = 1e5+5;

typedef pair <int, int> ii;

typedef pair <int, ii> iii;

int n, m;

int par[maxn];

vector <iii> v;

int root(int p) {

if(par[p] == p) return p;

return par[p] = root(par[p]);

}

int kruskal() {

int r = 0, x, y;

sort(v.begin(), v.end());

vector <iii> :: iterator it;

for(it = v.begin(); it != v.end(); it++) {

if((x = root(it -> se.fi)) != (y = root(it -> se.se))) {

r += it -> fi;

par[x] = y;

}

}

return r;

}

int main(){

std::ios::sync\_with\_stdio(false);

int x, y, w;

cin >> n >> m;

FOR(i, 1, n) {

par[i] = i;

}

FOR(i, 1, m) {

cin >> x >> y >> w;

v.push\_back(iii(w, ii(x, y)));

}

cout << kruskal();

return 0;

}

**Prim – Cây khung nhỏ nhất**

#include <bits/stdc++.h>

#define FOR(x, a, b) for (int x = a; x <= b; ++x)

#define FOD(x, a, b) for (int x = a; x >= b; --x)

#define REP(x, a, b) for (int x = a; x < b; ++x)

#define fi first

#define se second

#define ll long long

using namespace std;

const int maxn = 1e5+5;

const int oo = 1e9;

typedef pair<int, int> ii;

int n, m, d[maxn];

vector <ii> a[maxn];

int prim() {

int r = 0;

priority\_queue <ii, vector<ii>, greater<ii> > pq;

FOR(i, 1, n) {

d[i] = oo;

}

d[1] = 0;

pq.push(ii(0, 1));

while(!pq.empty()) {

int u = pq.top().se, du = pq.top().fi;

pq.pop();

if(du != d[u]) continue;

r += d[u];

d[u] = 0;

for(int i = 0; int v = a[u][i].se; ++i) {

int uv = a[u][i].fi;

if(d[v] > uv) {

d[v] = uv;

pq.push(ii(d[v], v));

}

}

}

return r;

}

int main(){

std::ios::sync\_with\_stdio(false);

int x, y, w;

cin >> n >> m;

while(m--) {

cin >> x >> y >> w;

a[x].push\_back(ii(w, y));

a[y].push\_back(ii(w, x));

}

FOR(i, 1, n) {

a[i].push\_back(ii(0, 0));

}

cout << prim();

return 0;

}

**Dijkstra – Đường đi ngắn nhất**

#include <bits/stdc++.h>

#define FOR(x, a, b) for (int x = a; x <= b; ++x)

#define FOD(x, a, b) for (int x = a; x >= b; --x)

#define REP(x, a, b) for (int x = a; x < b; ++x)

#define fi first

#define se second

#define ll long long

using namespace std;

typedef pair <int, int> ii;

const int maxn = 1e4+5;

const int oo = 1e9;

int n, m, d[maxn];

vector <ii> a[maxn];

priority\_queue <ii, vector <ii>, greater <ii> > pq;

void Dijkstra(int s) {

int u, v, du, uv;

FOR(i, 1, n) {

d[i] = oo;

}

d[s] = 0;

pq.push(ii(s, 0));

while(!pq.empty()) {

u = pq.top().fi;

du = pq.top().se;

pq.pop();

if(du != d[u]) continue;

for(int i = 0; v = a[u][i].fi; ++i) {

uv = a[u][i].se;

if(d[v] > du+uv) {

d[v] = du+uv;

pq.push(ii(v, d[v]));

}

}

}

}

int main(){

std::ios::sync\_with\_stdio(false);

int x, y, w, s;

cin >> n >> m >> s;

while(m--) {

cin >> x >> y >> w;

a[x].push\_back(ii(y, w));

a[y].push\_back(ii(x, w));

}

FOR(i, 1, n) {

a[i].push\_back(ii(0, 0));

}

Dijkstra(s);

FOR(i, 1, n) {

cout << d[i] << "\n";

}

return 0;

}

**Floy – Đường đi ngắn nhất**

#include <bits/stdc++.h>

#define FOR(x, a, b) for (int x = a; x <= b; ++x)

#define FOD(x, a, b) for (int x = a; x >= b; --x)

#define REP(x, a, b) for (int x = a; x < b; ++x)

#define fi first

#define se second

#define ll long long

using namespace std;

const int maxn = 1e3+5;

const int oo = 1e9;

int n, m;

int d[maxn][maxn];

int main(){

std::ios::sync\_with\_stdio(false);

memset(d, oo, sizeof d);

int x, y, w;

FOR(i, 1, n) {

d[i][i] = 0;

}

cin >> n >> m;

while(m--) {

cin >> x >> y >> w;

d[x][y] = d[y][x] = w;

}

FOR(k, 1, n) {

FOR(i, 1, n) {

FOR(j, 1, n) {

d[i][j] = min(d[i][j], d[i][k] + d[k][j]);

}

}

}

return 0;

}

**Đếm số khớp, cầu của đồ thị**

#include <bits/stdc++.h>

#define FOR(x, a, b) for (int x = a; x <= b; ++x)

#define FOD(x, a, b) for (int x = a; x >= b; --x)

#define REP(x, a, b) for (int x = a; x < b; ++x)

#define fi first

#define se second

#define ll long long

using namespace std;

const int maxn = 1e4+5;

int n, m, x, y;

vector<int> a[maxn];

int c = 0, dem = 0;

bool k[maxn];

int num[maxn], low[maxn], d = 0;

void visit(int u, int p) {

int v, child = 0;

num[u] = low[u] = ++d;

for(int i = 0; v = a[u][i]; ++i) {

if(v != p) {

if(num[v]) {

low[u] = min(low[u], num[v]);

}

else {

visit(v, u);

++child;

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

if(low[v] >= num[v]) {

++c;

}

if(u == p) {

if(child >= 2) {

k[u] = true;

}

}

else {

if(low[v] >= num[u]) {

k[u] = true;

}

}

}

}

}

}

int main(){

std::ios::sync\_with\_stdio(false);

cin >> n >> m;

while(m--) {

cin >> x >> y;

a[x].push\_back(y);

a[y].push\_back(x);

}

FOR(i, 1, n) {

a[i].push\_back(0);

}

FOR(i, 1, n) {

if(!num[i]) {

visit(i, i);

}

}

FOR(i, 1, n) {

if(k[i]) {

++dem;

}

}

cout << dem << " " << c;

return 0;

}

**LCA:**

#include <stdio.h>

#include <iostream>

#include <algorithm>

using namespace std;

const int N = 100005;

int n, Root, l[N], P[20][N];

int level(int u) {

if (u==Root) return l[u]=1;

if (l[u]==0) l[u]=level(P[0][u])+1;

return l[u];

}

int lca(int x, int y) {

for (int k=19; k>=0; k--)

if (l[P[k][x]]>=l[y])

x=P[k][x];

for (int k=19; k>=0; k--)

if (l[P[k][y]]>=l[x])

y=P[k][y];

for (int k=19; k>=0; k--)

if (P[k][x]!=P[k][y])

{ x=P[k][x]; y=P[k][y]; }

while (x!=y)

{ x=P[0][x]; y=P[0][y]; }

return x;

}

void solve() {

scanf("%d", &n);

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

int p; scanf("%d", &p);

while (p-->0) {

int q; scanf("%d", &q);

P[0][q] = i;

}

}

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

if (P[0][i]==0) Root=i;

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

level(i); // done l

for (int k=1; k<=19; k++)

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

P[k][i] = P[k-1][P[k-1][i]];

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

while (m-->0) {

int x, y;

scanf("%d%d", &x, &y);

printf("%d\n", lca(x, y));

}

}

main() {

int t; scanf("%d", &t);

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

printf("Case %d:\n", i);

solve();

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

{ l[j]=0; P[0][j]=0; }

}

}

**Đếm số lượng số nguyên tố nhỏ hơn n:**

#include <stdio.h>  
#include <math.h>  
#include <vector>  
#include <map>  
#include <iostream>  
#include <algorithm>  
using namespace std;  
  
#define long long long  
const int N = 100005;  
const int M = 1000000007;  
bool np[N];  
int p[N], pp=0;  
  
**void eratos() {**  
    np[0]=np[1]=true;  
    for (int i=2; i\*i<N; i++) if (!np[i])  
        for (int j=i\*i; j<N; j+=i) np[j]=true;  
    for (int i=2; i<N; i++)  
    if (!np[i]) p[++pp]=i;  
}  
  
**long power(long a, long k) {**  
    long P = 1;  
    while (k) {  
        if (k&1) P=P\*a;  
        k/=2; a=a\*a;  
    }  
    return P;  
}  
  
**long power(long a, long k, long M) {**  
    long P=1;  
    for (a=a%M; k; k/=2)  
    { if (k&1) P=P\*a%M; a=a\*a%M; }  
    return P;  
}  
  
**long root(long n, long k) {**  
    long x = pow(n, 1.0/k);  
    while (power(x, k)%M==power(x, k, M) && power(x, k)<n) x++;  
    while (power(x, k)%M!=power(x, k, M) || power(x, k)>n) x--;  
    return x;  
}  
  
map<long, long> Phi[N];  
  
**long phi(long x, int a) {**  
    if (Phi[a].count(x)) return Phi[a][x];  
    if (a==1) return (x+1)/2;  
    long Result = phi(x, a-1) - phi(x/p[a], a-1);  
    return Phi[a][x] = Result;  
}  
  
**long pi(long x) {**  
    if (x<N)  
        return upper\_bound(p+1, p+pp+1, x) - (p+1);  
    long a = pi(root(x, 4));  
    long b = pi(root(x, 2));  
    long c = pi(root(x, 3));  
    long Sum = phi(x, a) + (b+a-2)\*(b-a+1)/2;  
    for (int i=a+1; i<=b; i++)  
        Sum -= pi(x/p[i]);  
    for (int i=a+1; i<=c; i++) {  
        long bi = pi(root(x/p[i], 2));  
        for (int j=i; j<=bi; j++)  
        Sum -= pi(x/p[i]/p[j]) - (j-1);  
    }  
    return Sum;  
}  
  
**main(){**  
    eratos();  
    long n;  
    while (cin >> n)  
    cout << pi(n) << endl;  
}

**Disjoint\_Set:**

#include <bits/stdc++.h>

#define FOR(x, a, b) for (int x = a; x <= b; ++x)

#define FOD(x, a, b) for (int x = a; x >= b; --x)

#define REP(x, a, b) for (int x = a; x < b; ++x)

#define fi first

#define se second

#define ll long long

using namespace std;

const int maxn = 1e4+5;

int par[maxn], p, x, y, z;

int root(int v) {

return par[v] < 0 ? v : (par[v] = root(par[v]));

}

void merge(int x, int y) {

if ((x = root(x)) == (y = root(y))) {

return ;

}

if (par[y] < par[x]) {

swap(x, y);

}

par[x] += par[y];

par[y] = x;

}

int main(){

std::ios::sync\_with\_stdio(false);

memset(par, -1, sizeof par);

cin >> p;

while(p--) {

cin >> x >> y;

if(z == 1) {

merge(x, y);

}

else {

if(root(x) == root(y)) cout << "1\n";

else cout << "0\n";

}

}

return 0;

}

**Thao tác với bit:**

char Getbit(char x, i) {

return (x Shr i) && 1;

}

char Batbit(char x, i) {

return x OR(1 Shl i);

}

Char Tatbit(char x, i) {

return NOT(1 Shl i) And x;

}

**Deque tịnh tiến:**

D[0] **=** 0;

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

**while** (top **>** 0 **&&** a[D[top]] **>=** a[i]) **--**top;

L[i] **=** D[top] **+** 1;

D[**++**top] **=** i;

}

D[0] **=** n**+**1;

**for** (**int** i **=** n; i **>=** 1; **--**i) {

**while** (top **>** 0 **&&** a[D[top]] **>=** a[i]) **--**top;

R[i] **=** D[top] **-** 1;

D[**++**top] **=** i;

}

**Tính phi hàm Euler:**

#include <stdio.h>

#define long long long

long Power[230997][15]; // positive

long power(int a, int k){

    if (k==0) return 1;

    if (Power[a][k] > 0) return Power[a][k];

    long p=power(a, k/2);

    if (k&1) return Power[a][k] = p\*p\*a;

    else return Power[a][k] = p\*p;

}

long phi(int p, int k){

    // phi of p^k with p is a prime

    if (k==0) return 1;

    return (p-1)\*power(p, k-1);

}

long Phi[230997]; // positive

long phi(int m){

    int i, k, om=m;

    long r=1;

    if (Phi[om] > 0) return Phi[om];

    for (i=2; i\*i<=m; i++){

        k=0;

        while (m%i==0) { k++; m/=i; }

        r \*= phi(i, k);

    }

    if (m>1) r \*= phi(m, 1);

    return Phi[om] = r;

}

int n;

main(){

    int i;

    long r=0;

for (;;){

    scanf("%d", &n);

    if (n==0) return 0;

    r = phi(n);

    printf("%lld\n", r);

}

}

**Modul ngược:**

**function** ModuloInverse(a, m);

**begin**

xa:= 1; xm:= 0;

**while** m ≠ 0 **do**

**begin**

q:= a **div** m;

xr:= xa - q \* xm;

xa:= xm;

xm:= xr;

r:= a **mod** m;

a:= m;

m:= r;

**end**;

Result:= xa;

**end**;

**Manacher:**

#include <stdio.h>

#include <string.h>

#include <string>

#include <iostream>

#include <algorithm>

using namespace std;

bool maximize(int &a, int b){ if (a<b) a=b; else return false; return true; }

void prepare(char a[], char b[]){

int cnt=0;

b[++cnt]='#';

for (int i=1; a[i]; i++)

{ b[++cnt]=a[i]; b[++cnt]='#'; }

b[++cnt]=0;

b[0]='^';

}

int manacher(char b[]){

int C=1, R=1, n=strlen(b+1);

int \*P=new int[n+2], r=0;

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

int i\_mirror = 2\*C-i;

P[i] = (R>i) ? min(R-i, P[i\_mirror]) : 0;

while (b[i-1-P[i]] == b[i+1+P[i]]) P[i]++;

maximize(r, P[i]);

if (i+P[i]>R) { C=i; R=i+P[i]; }

}

delete[] P;

return r;

}

#define N 50004

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

main(){

ios :: sync\_with\_stdio(false);

int n;

cin >> n >> a+1;

prepare(a, b);

cout << manacher(b) << endl;

}