**Mục lục:**

**1.Luồng trang 1**

**2.cặp ghép trang 2**

**3.Khớp cầu trang 3**

**4.TPLT mạnh trang 3**

**5.LCA trang 4**

**6.Eratos trang 4**

**7.Extendedeuclid trang 5**

**8.Bellforman trang 5**

**9.Phi hàm Euler trang 5**

**10.Lehmer trang 6**

**11.Bao lồi trang 7**

**12.BIGNUM trang 7**

**13.Phương trình bậc nhất 3 ẩn trang 8**

**14.Cặp điểm gần nhất trang 8**

**15.KMP trang 9**

**16.HASH trang 9**

**17.Z function trang 10**

**18.RMQ trang 9**

**19.TRIE trang 10**

**20.SEGMENT TREE trang 11**

**21.BIT trang 11**

**22.Rabin-Miller trang 12**

**23.SEGMENT TREE LAZY trang 12**

**24.Nhân Ma Trận trang 13**

**25.nCk modulo số nguyên tố trang 14**

**26.nCk modulo hợp số trang 15**

**27.BAO LỒI (MONOTONE) trang 16**

**28.2-SAT trang 17**

**29.convexhulltrick trang 18**

**30.FLOW trang 18**

**31.MO trang 20**

**32.Ternary search trang 21**

**33.Chinese remainder theorem trang 22**

**34 Index Tree trang 22**

**35 COMPRESS trang 22**

**36 Random trang 23**

**37 POLLARD RHO trang 23**

**38.Theory and formulaformula**

**39. Cây khung nhỏ nhất Kruskal**

**40. Đường đi ngắn nhất Dijkstra**

**1.Luồng:**

**Dinizt:**

const int N = 1003, oo = 0x3c3c3c3c;

int n, m, S, T;

int d[N], c[N][N], f[N][N];

int Dfs[N], t = 0;

vector<int> a[N];

bool bfs(int S, int T) {

    memset(d, 0, sizeof d);

    queue<int> qu;

    qu.push(S);

    d[S] = 1;

    while (qu.size()) {

        int u = qu.front();

        qu.pop();

        if (u == T)

            return true;

        for (int v : a[u])

            if (!d[v] && f[u][v] < c[u][v]) {

                qu.push(v);

                d[v] = d[u] + 1;

            }

    }

    return false;

}

int visit(int u, int Min) {

    if (u == T)

        return Min;

    if (Dfs[u] != t)

        Dfs[u] = t;

    else

        return 0;

    for (int v : a[u])

        if (f[u][v] < c[u][v])

            if (Dfs[v] != t && d[v] == d[u] + 1)

 if (int x = visit(v, min(Min, c[u][v] - f[u][v]))) {

                    f[u][v] += x;

                    f[v][u] -= x;

                    return x;

                }

    return 0;

}

main() {

    cin >> n >> m >> S >> T;

    for (int i = 1; i <= m; i++) {

        int x, y, z;

        scanf("%d%d%d", &x, &y, &z);

        a[x].push\_back(y);

        a[y].push\_back(x);

        c[x][y] += z;

    }

    int Sum = 0;

    while (bfs(S, T)) {

        while (int x = (t++, visit(S, oo))) {

            Sum += x;

            //printf("Sum=%d\n", Sum);

        }

    }

    cout << Sum << endl;

}

* + 1. **Edmondskarp:**

#define long long long

void minimize(int &a, int b) {

    if (a > b)

        a = b;

}

int n, m, source, target;

vector<int> a[12309];

int c[2309][2309];

int f[2309][2309];

int d[12309];

bool findpath(int start, int target) {

    queue<int> qu;

    for (int i = 1; i <= n; i++)

        d[i] = 0;

    d[start] = 1000111000;

    qu.push(start);

    while (qu.size()) {

        int u = qu.front();

        qu.pop();

        //printf("u=%d\n", u);

        if (u == target)

            return true;

        for (int v : a[u])

            if (d[v] == 0 && c[u][v] > f[u][v]) {

                d[v] = u;

                qu.push(v);

            }

    }

    return false;

}

void enlarge() {

    int u, v, delta = 1000111000;

    for (v = target; (u = d[v]) != 1000111000; v = u)

        minimize(delta, c[u][v] - f[u][v]);

    for (v = target; v != source; v = u) {

        u = d[v];

        f[u][v] += delta;

        f[v][u] -= delta;

    }

}

long answer(int u) {

    long r = 0;

    for (int v : a[u])

        r += f[u][v];

    return r;

}

main() {

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

    for (int i = 1; i <= m; i++) {

        int p, q, w;

        scanf("%d%d%d", &p, &q, &w);

        a[p].push\_back(q);

        a[q].push\_back(p);

        c[p][q] = w;

    }

    while (findpath(source, target))

        enlarge();

    printf("%lld\n", answer(source));

}

**2.Cặp ghép cực đại trên đồ thị hai phía không trọng số:**

const int N = 102;

int n, m, Assigned[N];

int Visited[N], t = 0;

vector<int> a[N];

bool visit(int u) {

    if (Visited[u] != t)

        Visited[u] = t;

    else

        return false;

    for (int i = 0; i < a[u].size(); i++) {

        int v = a[u][i];

        if (!Assigned[v] || visit(Assigned[v])) {

            Assigned[v] = u;

            return true;

        }

    }

    return false;

}

int main() {

    scanf("%d%d", &m, &n);

    int x, y;

    while (scanf("%d%d", &x, &y) > 0)

        a[x].push\_back(y);

    int Count = 0;

    for (int i = 1; i <= m; i++) {

        t++;

        Count += visit(i);

    }

    printf("%d\n", Count);

    for (int i = 1; i <= n; i++)

        if (int j = Assigned[i])

            printf("%d %d\n", j, i);

}

**3.Khớp cầu:**

const int N = 100005;

int n, m;

vector<int> a[N];

int CriticalEdge = 0;

bool CriticalNode[N];

int Num[N], Low[N], Time = 0;

void visit(int u, int p) {

    int NumChild = 0;

    Low[u] = Num[u] = ++Time;

    for (int v : a[u])

        if (v != p) {

            if (Num[v] != 0)

                Low[u] = min(Low[u], Num[v]);

            else {

                visit(v, u);

                NumChild++;

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

                if (Low[v] >= Num[v])

                    CriticalEdge++;

                if (u == p) {

                    if (NumChild >= 2)

                        CriticalNode[u] = true;

                } else {

                    if (Low[v] >= Num[u])

                        CriticalNode[u] = true;

                }

            }

        }

}

int main() {

    scanf("%d%d", &n, &m);

    for (int i = 1; i <= m; i++) {

        int x, y;

        scanf("%d%d", &x, &y);

        a[x].push\_back(y);

        a[y].push\_back(x);

    }

    for (int i = 1; i <= n; i++)

        if (!Num[i]) visit(i, i);

    int Count = 0;

    for (int i = 1; i <= n; i++)

        if (CriticalNode[i]) Count++;

    printf("%d %d\n", Count, CriticalEdge);

}

**4.TPLT mạnh (Tarjan):**

const int N = 100005;

const int oo = 0x3c3c3c3c;

int n, m, Num[N], Low[N], cnt = 0;

vector<int> a[N];

stack<int> st;

int Count = 0;

void visit(int u) {

    Low[u] = Num[u] = ++cnt;

    st.push(u);

    for (int v : a[u])

        if (Num[v])

            Low[u] = min(Low[u], Num[v]);

        else {

            visit(v);

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

        }

    if (Num[u] == Low[u]) {  // found one

        Count++;

        int v;

        do {

            v = st.top();

            st.pop();

            Num[v] = Low[v] = oo;  // remove v from graph

        } while (v != u);

    }

}

int main() {

    scanf("%d%d", &n, &m);

    for (int i = 1; i <= m; i++) {

        int x, y;

        scanf("%d%d", &x, &y);

        a[x].push\_back(y);

    }

    for (int i = 1; i <= n; i++)

        if (!Num[i]) visit(i);

    cout << Count << endl;

}

**5.LCA:**

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));

    }

}

int 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;

        }

    }

}

**6.Eratos:**

void eratos(int n) {

    nonpr[0] = nonpr[1] = true;

    for (int i = 2; i <= n; i++)

        if (!nonpr[i])

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

                nonpr[j] = true;

    for (int i = 1; i <= n; i++)

        if (!nonpr[i]) prime[++nPrime] = i;

}

### 7.Extendedeuclid:

#define long long long

typedef pair<long, long> ii;

typedef pair<long, ii> triple;

#define X first

#define Y second

ii extended\_gcd(long a, long b) {

    ii qr, st;

    if (b == 0)

        return ii(1, 0);

    else {

        qr = ii(a / b, a % b);

        st = extended\_gcd(b, qr.Y);

        return ii(st.Y, st.X - qr.X \* st.Y);

    }

}

long gcd(long a, long b) {

    if (a == 0)

        return b;

    else

        return gcd(b % a, a);

}

int main() {

    for (;;) {

        int p, q;

        if (scanf("%lld%lld", &p, &q) < 0) return 0;

        ii ww = extended\_gcd(p, q);

        printf("%lld %lld %lld\n", ww.X, ww.Y, gcd(p, q));

    }

}

**8.Fordbellman:**

typedef pair<int, int> ii;

vector<ii> a[230997];

int n, m;

int d[230997];

bool inqueue[230997];

void bellman(int u) {

    queue<int> qu;

    for (int i = 1; i <= n; i++)

        d[i] = 1000111000;

    d[u] = 0;

    qu.push(u);

    inqueue[u] = true;

    while (qu.size()) {

        u = qu.front();

        inqueue[u] = false;

        qu.pop();

        for (int i = 0; i < a[u].size(); i++) {

            int v = a[u][i].Y;

            int uv = a[u][i].X;

            if (d[v] > d[u] + uv) {

                d[v] = d[u] + uv;

                if (!inqueue[v]) {

                    qu.push(v);

                    inqueue[v] = true;

                }

            }

        }

    }

}

int main() {

    int u, v;

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

    u++;

    v++;

    while (m--) {

        int p, q, w;

        scanf("%d%d%d", &p, &q, &w);

        p++;

        q++;

        a[p].push\_back(ii(w, q));

        a[q].push\_back(ii(w, p));

    }

    bellman(u);

    printf("%d\n", d[v]);

}

**9.Phi hàm Euler:**

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 om = m;

    long r = 1;

    if (Phi[om] > 0) return Phi[om];

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

        int 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;

int main() {

    for (;;) {

        scanf("%d", &n);

        if (n == 0) return 0;

        long r = phi(n);

        printf("%lld\n", r);

    }

}

### 10.Lehmer - Đếm số lượng số nguyên tố nhỏ hơn n:

#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;

}

int main() {

    eratos();

    long n;

    while (cin >> n)

        cout << pi(n) << endl;

}

**11.Tìm bao lồi (Graham):**

typedef pair<int, int> ii;

#define X first

#define Y second

ii origin;

void operator-=(ii &A, ii B) {

    A.X -= B.X;

    A.Y -= B.Y;

}

bool ccw(ii O, ii A, ii B) {

    A -= O, B -= O;

    return A.X \* B.Y > A.Y \* B.X;

}

bool cmp(ii A, ii B) { return ccw(origin, A, B); }

int n;

ii a[12309];

int main() {

    scanf("%d", &n);

    for (int i = 1; i <= n; i++)

        scanf("%d%d", &a[i].X, &a[i].Y);

    sort(a + 1, a + n + 1);

    origin = a[1];

    sort(a + 2, a + n + 1, cmp);

    a[0] = a[n];

    a[n + 1] = a[1];

    int j = 1;

    for (int i = 1; i <= n + 1; i++) {  // a[1] and a[n+1] will be both added

        while (j > 2 && !ccw(a[j - 2], a[j - 1], a[i])) j--;

        a[j++] = a[i];

    }

    n = j - 2;

    for (int i = 1; i <= n; i++) printf("%d %d\n", a[i].X, a[i].Y);

}

**12.BIGNUM:**

string add(string a, string b)

{

    string res="";

    while(a.length() < b.length()) a="0"+a;

    while(b.length() < a.length()) b="0"+b;

    int carry=0;

    for(int i=a.length()-1;i>=0;i--)

    {

        int tmp=a[i]-48 + b[i]-48 + carry;

        carry=tmp/10;

        tmp=tmp%10;

        res=(char)(tmp+48)+res;

    }

    if(carry>0) res="1"+res;

    return res;

}

string sub(string a, string b)

{

    string res="";

    while(a.length() < b.length()) a="0"+a;

    while(b.length() < a.length()) b="0"+b;

    bool neg=false;

    if(a<b)

    {

        swap(a,b);

        neg=true;

    }

    int borrow=0;

    for(int i=a.length()-1; i>=0;i--)

    {

        int tmp=a[i]-b[i]-borrow;

        if(tmp<0)

        {

            tmp+=10;

            borrow=1;

        }

        else borrow=0;

        res=(char)(tmp%10 + 48) + res;

    }

    while(res.length()>1 && res[0]=='0') res.erase(0,1);

    if(neg) res="-"+res;

    return res;

}

string mul(string a, string b)

{

    string res="";

    int n=a.length();

    int m=b.length();

    int len=n+m-1;

    int carry=0;

    for(int i=len;i>=0;i--)

    {

        int tmp=0;

        for(int j=n-1;j>=0;j--)

            if(i-j<=m && i-j>=1)

            {

                int a1=a[j]-48;

                int b1=b[i-j-1]-48;

                tmp+=a1\*b1;

            }

            tmp+=carry;

            carry=tmp/10;

            res=(char)(tmp%10 + 48)+res;

    }

    while(res.length()>1 && res[0]=='0') res.erase(0,1);

    return res;

}

**13.Phương trình bậc nhất 3 ẩn:**

#define long long long

long det(int a1, int b1, int c1, int a2, int b2, int c2, int a3, int b3, int c3) {

    return a1 \* (b2 \* c3 - b3 \* c2) - b1 \* (a2 \* c3 - a3 \* c2) + c1 \* (a2 \* b3 - a3 \* b2);

}

int main() {

    int a1, b1, c1, d1;

    int a2, b2, c2, d2;

    int a3, b3, c3, d3;

    cin >> a1 >> b1 >> c1 >> d1;

    cin >> a2 >> b2 >> c2 >> d2;

    cin >> a3 >> b3 >> c3 >> d3;

    long D = det(a1, b1, c1, a2, b2, c2, a3, b3, c3);

    double DX = det(d1, b1, c1, d2, b2, c2, d3, b3, c3);

    double DY = det(a1, d1, c1, a2, d2, c2, a3, d3, c3);

    double DZ = det(a1, b1, d1, a2, b2, d2, a3, b3, d3);

    if (D == 0)

        cout << "Math error" << endl;

    else

        cout << DX / D << " " << DY / D << " " << DZ / D << endl;

}

**14.Cặp điểm gần nhất:**

struct point {

double x, y;

};

bool cmp\_x(const point &a, const point &b) {

return a.x < b.x;

}

bool cmp\_y(const point &a, const point &b) {

return a.y < b.y;

}

#define MAXN 100000

point a[MAXN];

double mindist; // biến lưu kết quả bài toán

// tính khoảng cách giữa a và b rồi update kết quả

void upd\_ans(const point &a, const point &b) {

double dist = sqrt((a.x-b.x)\*(a.x-b.x) + (a.y-b.y)\*(a.y-b.y));

if (dist < mindist) mindist = dist;

}

void find(int l, int r) {

if (r <= l) return;

// đoạn [l,r] có 2 phần tử

if (r == l + 1) {

upd\_ans(a[l], a[r]);

// sắp các phần tử lại theo y

if (!cmp\_y(a[l], a[r])) swap(a[l], a[r]);

return;

}

int m = (l + r) / 2;

double midx = a[m].x;

find(l, m);

find(m+1, r);

static point t[MAXN];

// trộn a[l,m] và a[m+1,r] lại, lưu vào mảng tạm t

merge(a+l, a+m+1, a+m+1, a+r+1, t, cmp\_y);

// copy từ t về lại a

copy(t, t+r-l+1, a+l);

// mảng t ở đây lưu các phần tử thỏa |x\_i - midx| < mindist,

// với số lượng phần tử là tm

// do đã sort nên các phần tử sẽ có y tăng dần

int tm = 0;

for (int i=l; i<=r; i++) if (abs(a[i].x-midx) < mindist) {

for (int j=tm-1; j>=0 && t[j].y > a[i].y-mindist; j--)

upd\_ans(a[i], t[j]);

t[tm++] = a[i];

}

}

int main() {

ios::sync\_with\_stdio(false); cin.tie(0);

int n; cin >> n;

for (int i=0; i<n; i++) cin >> a[i].x >> a[i].y;

mindist = 1E20;

sort(a, a+n, cmp\_x);

find(0, n-1);

printf("%.3lf", mindist);

return 0;

}

**15.KMP:**

const int N = 1000006;

int m, n, Prev[N];

char a[N], b[N];

int main() {

    scanf("%s%s", a + 1, b + 1);

    m = strlen(a + 1), n = strlen(b + 1);

    Prev[0] = -1;

    for (int i = 1; i <= n; i++) {

        Prev[i] = 0;

        for (int j = Prev[i - 1]; j != -1; j = Prev[j])

            if (b[j + 1] == b[i]) {

                Prev[i] = j + 1;

                break;

            }

    }

    int u = 0;

    for (int i = 1; i <= m; i++) {

        while (u != 0 && b[u + 1] != a[i]) u = Prev[u];

        if (b[u + 1] == a[i]) u++;

        if (u == n) printf("%d ", i - n + 1);

    }

    puts("");

}

**16.HASH:**

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;

}

int 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");

}

**18.RMQ:**

int a[230997];

int M[230997][23];

int n, m;

int max(int a, int b) { return a > b ? a : b; }

int main() {

    int i, k;

    scanf("%d%d", &n, &m);

    for (i = 1; i <= n; i++)

        scanf("%d", &a[i]);

    for (i = 1; i <= n; i++) M[i][0] = a[i];

    for (k = 1; (1 << k) <= n; k++)

        for (i = 1; i + (1 << k) - 1 <= n; i++)

            M[i][k] = max(M[i][k - 1], M[i + (1 << (k - 1))][k - 1]);

    int u, v;

    while (m--) {

        scanf("%d%d", &u, &v);

        k = log2(v - u + 1);

        printf("%d\n", max(M[u][k], M[v - (1 << k) + 1][k]));

    }

}

**17.Z function:**

#define N 500005

int n, z[N];

char a[N];

void make\_z(char a[], int n, int F[]) {

    int L = -1, R = -1;

    F[0] = n;

    for (int i = 1; i < n; i++) {

        if (i > R) {

            L = i;

            R = i - 1;

            while (R < n - 1 && a[R + 1] == a[R - L + 1]) R++;

            F[i] = R - L + 1;

        } else {

            if (F[i - L] < R - i + 1)

                F[i] = F[i - L];

            else {

                L = i;

                while (R < n - 1 && a[R + 1] == a[R - L + 1]) R++;

                F[i] = R - L + 1;

            }

        }

    }

}

int main() {

    gets(a);

    n = strlen(a);

    make\_z(a, n, z);

    for (int i = 0; i < n; i++)

        printf("%d ", z[i]);

    printf("\n");

}

**19.Trie:**

1. **struct** **Trie** {
2. **int** ID = 1;
3. **struct** **Node** {
4. **int** size;
5. **int** child[2];
7. **Node**() {
8. size = 0;
9. child[0] = child[1] = 0;
10. }
11. };
13. **vector**<**Node**> node;
14. **Trie**(**int** n) {
15. node.resize(30 \* n + 5);
16. }
18. **void** insert(**int** val) {
19. **int** root = 1;
20. **for** (**int** i = 29; i >= 0; i--) {
21. **int** bit = (**bool**) (val & (1 << i));
23. node[root].size++;
24. **if** (node[root].child[bit] == 0) {
25. node[root].child[bit] = ++ID;
26. }
27. root = node[root].child[bit];
28. }
29. }
31. **long** **long** queryMin(**int** val) {
32. **int** root = 1;
33. **long** **long** res = 0;
34. **for** (**int** i = 29; i >= 0; i--) {
35. **int** bit = (**bool**) (val & (1 << i));
37. **if** (node[root].child[bit] == 0) {
38. root = node[root].child[bit];
39. }
40. **else** {
41. res += (1 << i);
42. root = node[root].child[bit ^ 1];
43. }
44. }
45. **return** res;
46. }
48. **long** **long** queryMax(**int** val) {
49. **int** root = 1;
50. **long** **long** res = 0;
51. **for** (**int** i = 29; i >= 0; i--) {
52. **int** bit = (**bool**) (val & (1 << i));
53. bit ^= 1;
55. **if** (node[root].child[bit] == 0) {
56. root = node[root].child[bit ^ 1];
57. }
58. **else** {
59. res += (1 << i);
60. root = node[root].child[bit];
61. }
62. }
63. **return** res;
64. }
65. };

### 20.Segment tree

#include <bits/stdc++.h>

using namespace std;

const int N = 1e5 + 10;

int node[4\*N];

void modify(int seg, int l, int r, int p, int val){

if(l == r){

node[seg] += val;

return;

}

int mid = (l + r)/2;

if(p <= mid){

modify(2\*seg + 1, l, mid, p, val);

}else{

modify(2\*seg + 2, mid + 1, r, p, val);

}

node[seg] = node[2\*seg + 1] + node[2\*seg + 2];

}

int sum(int seg, int l, int r, int a, int b){

if(l > b || r < a) return 0;

if(l >= a && r <= b) return node[seg];

int mid = (l + r)/2;

return sum(2\*seg + 1, l, mid, a, b) + sum(2\*seg + 2, mid + 1, r, a, b);

}

int main(){

int n, m;

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

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

char t;

scanf(" %c", &t);

if(t == 'A'){

int p, x;

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

modify(0, 1, n, p, x);

}else{

int a, b;

scanf("%d %d", &a, &b);

printf("%d\n", sum(0, 1, n, a, b));

}

}

return 0;

}

### 21.Binary Indexed tree (Fenwick Tree)

#include <iostream>

using namespace std;

int n, m, k;

long long arr[1000005];

long long tree[1000005];

void update(int idx, int val) {

while (idx <= n) {

tree[idx] += val;

idx += (idx & -idx);

}

}

long long read(int idx) {

long long ret = 0;

while (idx > 0) {

ret += tree[idx];

idx -= (idx & -idx);

}

return ret;

}

int main() {

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

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

scanf("%lld", &arr[i]);

update(i, arr[i]);

}

int a, b, c;

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

scanf("%d %d %d", &a, &b, &c);

if (a == 1) {

update(b, c - arr[b]);

arr[b] = c;

}

else {

printf("%lld\n", read(c) - read(b - 1));

}

}

return 0;

}

**22.Rabin -Miller:**

const int RAB[] = {3,5,7,11,13,17}, R = sizeof(RAB)/sizeof(RAB[0]);

LL pm(LL a, LL e, LL m) {

if (m==1) return 0;

if (!e) return 1;

LL t = 1;

while (e > 1) {

if (e&1) t = t\*a%m;

a = a\*a%m;

e >>= 1;

}

return t\*a%m;

}

bool primeTest(LL n) {

if (n==2) return true;

if (n<2 || (n&1)==0) return false;

LL m = n-1, s = 0;

while ((m&1)==0) {

m >>= 1;

s++;

}

\_for(i,0,R) {

LL k = RAB[i], b = pm(k,m,n);

if (n == k) return true;

if (n%k == 0) return false;

if (b == 1) continue;

bool pass = false;

\_for (j,0,s) {

if (b == n-1) {

pass = true;

break;

}

b = b\*b%n;

}

if (!pass) return false;

}

return true;

}

**23.SG LAZY:**

1. struct Node {
2. i64 val = 0;
3. i64 lazy = 0;
4. };
6. i64 n, m;
7. vector<Node> tree(666666);
9. void update(i64 node, i64 st, i64 en, i64 L, i64 R, i64 x) {
10. if (tree[node].lazy > 0) {
11. tree[node].val += tree[node].lazy;
12. if (st != en) {
13. tree[node\*2].lazy += tree[node].lazy;
14. tree[node\*2+1].lazy += tree[node].lazy;
15. }
16. tree[node].lazy = 0;
17. }
18. if (st > en || st > R || en < L) return;
19. if (L <= st && en <= R) {
20. tree[node].val += x;
21. if (st != en) {
22. tree[node\*2].lazy += x;
23. tree[node\*2+1].lazy += x;
24. }
25. return;
26. }
27. update(node\*2, st, (st+en)/2, L, R, x);
28. update(node\*2+1, (st+en)/2+1, en, L, R, x);
29. tree[node].val = max(tree[node\*2].val, tree[node\*2+1].val);
30. }
32. i64 get(i64 node, i64 st, i64 en, i64 L, i64 R) {
33. if (st > en || st > R || en < L) return -LINF;
34. if (tree[node].lazy > 0) {
35. tree[node].val += tree[node].lazy;
36. if (st != en) {
37. tree[node\*2].lazy += tree[node].lazy;
38. tree[node\*2+1].lazy += tree[node].lazy;
39. }
40. tree[node].lazy = 0;
41. }
42. if (L <= st && en <= R) return tree[node].val;
43. i64 p1 = get(node\*2, st, (st+en)/2, L, R);
44. i64 p2 = get(node\*2+1, (st+en)/2+1, en, L, R);
45. return max(p1, p2);
46. }
47. void VarInput() {
48. cin >> n >> m;
49. }
51. void ProSolve() {
52. while (m--) {
53. i64 cmd, l, r; cin >> cmd >> l >> r;
54. if (cmd == 0) {
55. i64 value; cin >> value;
56. update(1, 1, n, l, r, value);
57. }
58. else cout << get(1, 1, n, l, r) << endl;
59. }
60. }

**24.Nhân ma trận:**

struct matrix

{

long long val[5][5];

};

matrix a,c;

long long t;

long long n,m;

matrix nhan(matrix a,matrix b)

{

matrix c;

for (int i=1;i<=3;i++)

for (int j=1;j<=3;j++)

{

c.val[i][j]=0;

for (int k=1;k<=3;k++)

c.val[i][j]=((c.val[i][j]+(long long)a.val[i][k]\*b.val[k][j])%m)%m;

}

return c;

}

matrix mu (matrix a, long long k)

{

matrix x;

if ( k == 1) return a;

x = mu (a, k/2);

x = nhan (x, x);

if (k % 2) x = nhan (x, a);

return x;

}

int main()

{

cin>>n>>m;

a.val[1][1]=1;

a.val[1][2]=1;

a.val[1][3]=1;

a.val[2][1]=1;

a.val[2][2]=0;

a.val[2][3]=0;

a.val[3][1]=0;

a.val[3][2]=1;

a.val[3][3]=0;

if (n==1) cout<<1;

else if (n==2) cout<<2;

else

{

c=mu(a,n-2);

cout<<(c.val[1][1]\*2+c.val[1][2]\*1+c.val[1][3]\*1)%m;

}

}

**25.nCk modulo:**

const long long Mod = 1e9 + 7;

const int lim = 1e6 ;

long long fact[1000005], inv[1000005]; // size of array = lim + 5

long long pw(long long a, long long b) {

if (b == 1) return a;

if (b == 0) return 1;

long long q = pw(a, b/2) % Mod;

if (b % 2 == 0) return q%Mod \* q%Mod;

else return q%Mod \* q%Mod \* a%Mod;

}

long long combi(long long n, long long k, long long m) {

if (n < k) return 0;

if (vp[n] > vp[n - k] + vp[k]) return 0;

return fact[n] \* inv[k] % m \* inv[n - k] % m;

}

for (int i = 1; i <= 100000; i++) {

int val = i; vp[i] = vp[i - 1];

while (val % p == 0) {

vp[i]++;

val /= p;

}

fact[i] = fact[i - 1] \* val % p;

inv[i] = pw(fact[i], p - 2, p);

}

inv[lim] = pw(fact[lim],Mod-2);

for (long long i = (lim-1); i >= 1; i--) {

inv[i] = inv[i+1] \* (i+1) % Mod;

}

int t;

cin >> t;

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

long long b,a;

cin >> a >> b;

cout << combi(a,b) << endl;

}

return 0;

}

**26.nCk modulo hợp số**

Combi mod p(p là hợp số) > phân tichs p = p0 \* p1 \*...

long long fact[100005];

long long inv[100005];

long long vp[100005][10];

// tính combi(n, k) với mọi p

long long pw(long long a, long long n, long long m) {

if (n == 0) return 1;

long long q = pw(a, n / 2, m);

if (n % 2 == 0) return q \* q % m;

else return q \* q % m \* a % m;

}

// giả sử m = 1e9 + 22071997 = p0(19) \* p1(127) \* p2(467) \* p3(907)

long long combi(long long n, long long k, long long m) {

if (n < k || k < 0) return 0;

if (m == 1019972207) {

return fact[n] \* inv[k] % m \* inv[n - k] % m;

}

else {

long long res = fact[n] \* inv[k] % m \* inv[n - k] % m;

if (vp[n][0] > vp[k][0] + vp[n - k][0]) {

res \*= pw(19, vp[n][0] - vp[k][0] - vp[n - k][0], m);

res %= m;

}

if (vp[n][1] > vp[k][1] + vp[n - k][1]) {

res \*= pw(127, vp[n][1] - vp[k][1] - vp[n - k][1], m);

res %= m;

}

if (vp[n][2] > vp[k][2] + vp[n - k][2]) {

res \*= pw(467, vp[n][2] - vp[k][2] - vp[n - k][2], m);

res %= m;

}

if (vp[n][3] > vp[k][3] + vp[n - k][3]) {

res \*= pw(907, vp[n][3] - vp[k][3] - vp[n - k][3], m);

res %= m;

}

return res;

}

}

int main() {

freopen("input.txt", "r", stdin);

ios::sync\_with\_stdio(0);

cin.tie(NULL);

int t, p;

cin >> t >> p;

t = 5, p = 1019972207;

int n = 100000;

fact[0] = 1;

inv[0] = 1;

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

int val = i;

for (int j = 0; j <= 3; j++) {

vp[i][j] = vp[i - 1][j];

}

while (val % 19 == 0) {

vp[i][0]++;

val /= 19;

}

while (val % 127 == 0) {

vp[i][1]++;

val /= 127;

}

while (val % 467 == 0) {

vp[i][2]++;

val /= 467;

}

while (val % 907 == 0) {

vp[i][3]++;

val /= 907;

}

fact[i] = fact[i - 1] \* val % p;

inv[i] = pw(fact[i], p - 2, p);

}

cout << combi(n,k,p);

return 0;

}

**27.BAO LỒI( MONOTONE):**

### const int N = 20000;

### 

### struct Point {

### long long x, y;

### bool operator<(const Point &v) const { return x == v.x ? y < v.y : x < v.x; }

### long long cross(const Point &p, const Point &q) const { return (p.x - x) \* (q.y - y) - (p.y - y) \* (q.x - x); }

### } p[N], poly[N];

### 

### int n;

### 

### void enter() {

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

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

### scanf("%lld%lld", &p[i].x, &p[i].y);

### }

### 

### long long size(Point poly[], int k) {

### long long S = (poly[k - 1].x - poly[0].x) \* (poly[k - 1].y + poly[0].y);

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

### S += (poly[i - 1].x - poly[i].x) \* (poly[i - 1].y + poly[i].y);

### return S;

### printf("%lld\n", S);

### }

### 

### void solve() {

### sort(p, p + n);

### int k = 0;

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

### while (k >= 2 && poly[k - 2].cross(poly[k - 1], p[i]) <= 0) --k;

### poly[k++] = p[i];

### }

### for (int i = n - 2, t = k + 1; i >= 0; --i) {

### while (k >= t && poly[k - 2].cross(poly[k - 1], p[i]) <= 0) --k;

### poly[k++] = p[i];

### }

### printf("%lld\n", size(poly, k));

### }

### 

### int main() {

### enter();

### solve();

### return 0;

### }

### 

**28. 2-sat**

struct twosat{ //~x = 2\*x+1 , x = 2\*x --> ~x = x^1

int n;

vector < vector < int > > in , out;

vector < int > mark , col , topo , ans;

int c = 0;

twosat(int N):

n(N) , in(n\*2+5) , out(n\*2+5) , mark(n\*2+5 , 0) , col(n\*2+5, 0) , ans(n + 5){}

bool operator [] (int x){ return(ans[x]);};

void add\_edge(int v , int u){

in[u].pb(v) , out[v].pb(u);

}

void sfd(int v){

mark[v] = 1;

col[v] = c;

for(int u : in[v]) if(!mark[u])

sfd(u);

}

void dfs(int v){

mark[v] = 1;

for(int u : out[v]) if(!mark[u])

dfs(u);

topo.pb(v);

}

bool validate(){

for(int i = 1 ; i <= 2\*n + 1 ; i ++)if(!mark[i])dfs(i);

reverse(topo.begin() , topo.end());

memset(&mark[0] , 0 , sizeof(mark[0]) \* mark.size());

for(auto v : topo)

if(!mark[v])

c++ , sfd(v);

for(int i = 1 ; i <= n ; i ++)if(col[2\*i] == col[2\*i+1])return(0);

for(int i = 1 ; i <= n ; i ++)ans[i] = (col[2\*i] > col[2\*i + 1]);

return(1);

}

};

;

**29.convex hull trick**

//Convex Hull Trick

//Add a line with slope "m" and y-intercept "b"

//Query the MAXIMUM value at "x".

//Source: https://github.com/kth-competitive-programming/kactl/blob/main/content/data-structures/LineContainer.h

struct Line {

mutable ll m, b, p;

bool operator<(const Line& o) const { return m < o.m; }

bool operator<(ll x) const { return p < x; }

};

struct ConvexHullTrick : multiset<Line, less<>> {

// (for doubles, use inf = 1/.0, div(a,b) = a/b)

static const ll inf = LLONG\_MAX;

ll div(ll a, ll b) { // floored division

return a / b - ((a ^ b) < 0 && a % b); }

bool isect(iterator x, iterator y) {

if (y == end()) return x->p = inf, 0;

if (x->m == y->m) x->p = x->b > y->b ? inf : -inf;

else x->p = div(y->b - x->b, x->m - y->m);

return x->p >= y->p;

}

void add(ll m, ll b) {

auto z = insert({m, b, 0}), y = z++, x = y;

while (isect(y, z)) z = erase(z);

if (x != begin() && isect(--x, y)) isect(x, y = erase(y));

while ((y = x) != begin() && (--x)->p >= y->p)

isect(x, erase(y));

}

ll query(ll x) {

assert(!empty());

auto l = \*lower\_bound(x);

return l.m \* x + l.b;

}

};

//ConvexHullTrick cht;

//cht.add(0,0);

**30.FLOW:**

int get\_index() {

static int id = -1;

id += 1;

return id;

}

struct FlowEdge {

int v, u;

long long cap, flow = 0;

FlowEdge(int v, int u, long long cap) : v(v), u(u), cap(cap) {}

};

struct Dinic {

const long long flow\_inf = 1e18;

vector<FlowEdge> edges;

vector<vector<int>> adj;

int n, m = 0;

int s, t;

vector<int> level, ptr;

queue<int> q;

Dinic(int n, int s, int t) : n(n), s(s), t(t) {

adj.resize(n);

level.resize(n);

ptr.resize(n);

}

void add\_edge(int v, int u, long long cap) {

edges.emplace\_back(v, u, cap);

edges.emplace\_back(u, v, 0);

adj[v].push\_back(m);

adj[u].push\_back(m + 1);

m += 2;

}

bool bfs() {

while (!q.empty()) {

int v = q.front();

q.pop();

for (int id : adj[v]) {

if (edges[id].cap - edges[id].flow < 1)

continue;

if (level[edges[id].u] != -1)

continue;

level[edges[id].u] = level[v] + 1;

q.push(edges[id].u);

}

}

return level[t] != -1;

}

long long dfs(int v, long long pushed) {

if (pushed == 0)

return 0;

if (v == t)

return pushed;

for (int& cid = ptr[v]; cid < (int)adj[v].size(); cid++) {

int id = adj[v][cid];

int u = edges[id].u;

if (level[v] + 1 != level[u] || edges[id].cap - edges[id].flow < 1)

continue;

long long tr = dfs(u, min(pushed, edges[id].cap - edges[id].flow));

if (tr == 0)

continue;

edges[id].flow += tr;

edges[id ^ 1].flow -= tr;

return tr;

}

return 0;

}

long long flow() {

long long f = 0;

while (true) {

fill(level.begin(), level.end(), -1);

level[s] = 0;

q.push(s);

if (!bfs())

break;

fill(ptr.begin(), ptr.end(), 0);

while (long long pushed = dfs(s, flow\_inf)) {

f += pushed;

}

}

return f;

}

};

**31.MO:**

|  |  |
| --- | --- |
| #define N 30010 |  |
|  | #define A 1000010 |
|  | #define BLOCK 174 |
|  | #define M 200010 |
|  |  |
|  | int answer; |
|  | struct query |
|  | { |
|  | int l,r,ind; |
|  | }Q[M]; |
|  | int ans[M]; |
|  | int a[N]; |
|  | int cnt[A]; |
|  |  |
|  | void add(int index) |
|  | { |
|  | cnt[a[index]]++; |
|  | if(cnt[a[index]]==1) answer++; |
|  | } |
|  |  |
|  | void remove(int index) |
|  | { |
|  | cnt[a[index]]--; |
|  | if(cnt[a[index]]==0) answer--; |
|  | } |
|  |  |
|  | bool cmp(query f,query s) |
|  | { |
|  | if((f.l/BLOCK)!=(s.l/BLOCK)) |
|  | return (f.l/BLOCK)<(s.l/BLOCK); |
|  | else |
|  | return f.r<s.r; |
|  | } |
|  |  |
|  | int main() { |
|  | ios\_base::sync\_with\_stdio(false); |
|  | int n,q; |
|  | scanf("%d",&n); |
|  | REP(i,0,n) |
|  | { |
|  | scanf("%d",&a[i]); |
|  | } |
|  | scanf("%d",&q); |
|  | REP(i,0,q) |
|  | { |
|  | scanf("%d%d",&Q[i].l,&Q[i].r); |
|  | Q[i].ind=i; |
|  | Q[i].l--; |
|  | Q[i].r--; |
|  | } |
|  | sort(Q,Q+q,cmp); |
|  | int cl=0,cr=0; |
|  | REP(i,0,q) |
|  | { |
|  | int left=Q[i].l,right=Q[i].r; |
|  | while(cl<left) |
|  | remove(cl),cl++; |
|  | while(cl>left) |
|  | add(cl-1),cl--; |
|  | while(cr<=right) |
|  | add(cr),cr++; |
|  | while(cr>(right+1)) |
|  | remove(cr-1),cr--; |
|  | ans[Q[i].ind]=answer; |
|  | } |
|  | REP(i,0,q) |
|  | printf("%d\n",ans[i]); |
|  | return 0; |
|  | }  **32.TERNARY SEARCH**  double ternary\_search(double l, double r) {  double eps = 1e-9; //set the error limit here  while (r - l > eps) {  double m1 = l + (r - l) / 3;  double m2 = r - (r - l) / 3;  double f1 = f(m1); //evaluates the function at m1  double f2 = f(m2); //evaluates the function at m2  if (f1 < f2)  l = m1;  else  r = m2;  }  return f(l); //return the maximum of f(x) in [l, r]  } |

**33.[Chinese remainder theorem](https://en.wikipedia.org/wiki/Chinese_remainder_theorem)**

**ll euclid(ll a, ll b, ll &x, ll &y) {**

**if (!b) return x = 1, y = 0, a;**

**ll d = euclid(b, a % b, y, x);**

**return y -= a/b \* x, d;**

**}**

**ll crt(ll a, ll m, ll b, ll n) {**

**if (n > m) swap(a, b), swap(m, n);**

**ll x, y, g = euclid(m, n, x, y);**

**assert((a - b) % g == 0); // else no solution**

**x = (b - a) % n \* x % n / g \* m + a;**

**return x < 0 ? x + m\*n/g : x;**

**}**

**34 Index Set**

#include <ext/pb\_ds/assoc\_container.hpp>

using namespace \_\_gnu\_pbds;

typedef tree<int,null\_type,less<int>,rb\_tree\_tag,tree\_order\_statistics\_node\_update> IndexTree;

// Ex;

s={2,5,6,10}

\*s.find\_by\_order(2)=6;

s.order\_of\_key(x) == lower\_bound(x)-s.begin();

**35. COMPRESS**

// Usage:

// Compressor<int> comp;

// Adding an element : comp.add(value)

// After having all : comp.compress()

// Find index of value v : comp.find(v) (1-based index)

// Get the original value of index i: comp.orig[i] (1-based index)

template <class T>

struct Compressor {

vector<T> values, orig;

void add(T x) {

values.push\_back(x);

}

void compress() {

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

values.erase(unique(values.begin(), values.end()), values.end());

orig.resize(values.size() + 1);

for (int i = 0; i < values.size(); i++) {

orig[i + 1] = values[i];

}

}

// return index of value (lower\_bound)

T find(T x) {

return lower\_bound(values.begin(), values.end(), x) - values.begin() + 1;

}

};

int main() {

int n;

Compressor<int> comp;

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

cin >> a[i];

comp.add(a[i]);

}

comp.compress();

int maxcs = 0;

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

a[i] = comp.find(a[i]);

maxcs = max(maxcs, a[i]);

// last\_pos[a[i]].push\_back(i);

}

}

**36** **RANDOM**

mt19937\_64 gen(chrono::steady\_clock::now().time\_since\_epoch().count());

ll Rand(ll l, ll r){

uniform\_int\_distribution<ll> rnd(l,r); return rnd(gen);

}

**37 POLLARD RHO**

/\* POLLARD RHO algorithm \*/

// prime factorization algorithm for n = 1e18

// dpt can bac 4

mt19937\_64 gen(chrono::steady\_clock::now().time\_since\_epoch().count());

long long Rand(long long l, long long r){

uniform\_int\_distribution<long long> rnd(l,r);

return rnd(gen);

}

long long mul(long long a, long long b, long long m){

if (b == 1) return a % m;

if (b == 0) return 1 % m;

a %= m; b %= m;

long long q = mul(a, b / 2, m);

if (b % 2 == 0) return ((q + q) % m + m) % m;

else return ((q + q + a) % m + m) % m;

}

long long pw(long long a, long long n, long long m){

if (n == 0) return 1 % m;

long long q = pw(a, n / 2, m);

if (n % 2 == 0) return mul(q, q, m);

return mul(mul(q, q, m), a, m);

}

// Milner-Rabin algorithm

bool checkprime(long long n){

if (n == 2) return 1;

if (n % 2 == 0 || n == 1) return 0;

long long m = n - 1;

long long s = 0;

while (m % 2 == 0){

m /= 2; s++;

}

long long dem = 0, q = 0, a, b;

while (dem <= 3){

dem++;

a = Rand(2, n - 2);

b = pw(a, m, n);

if ((b + 1) % n == 0 || (b - 1) % n == 0) {q++;continue;}

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

b = mul(b, b, n);

if ((b + 1) % n == 0) {q++; break;}

}

}

if (q == 4) return 1;

return 0;

}

map<long long, long long> alpha;

void brute(int n){

if (n == 1) return;

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

if (n % i == 0) {

while (n % i == 0) {alpha[i]++; n /= i;}

}

}

if (n > 1){

alpha[n]++;

}

}

long long f(long long x, long long m){

return (mul(x, x, m) + 1) % m;

}

long long findFactor(long long n){

long long x = Rand(2, n - 1);

long long y = x;

long long p = 1;

while (p == 1){

x = f(x, n);

y = f(f(y, n), n);

p = \_\_gcd(abs(x - y), n);

}

return p;

}

void fact(long long n){

if (n == 1) return ;

if (checkprime(n)) {alpha[n]++; return;}

if (n <= 10000) {

brute(n); return;

}

long long p = 0;

while (p == 0 || p == n){

p = findFactor(n);

}

fact(p); fact(n / p);

}

int main() {

freopen("input.txt", "r", stdin);

long long n;

while (cin >> n){

alpha.clear();

fact(n);

for (auto i: alpha){

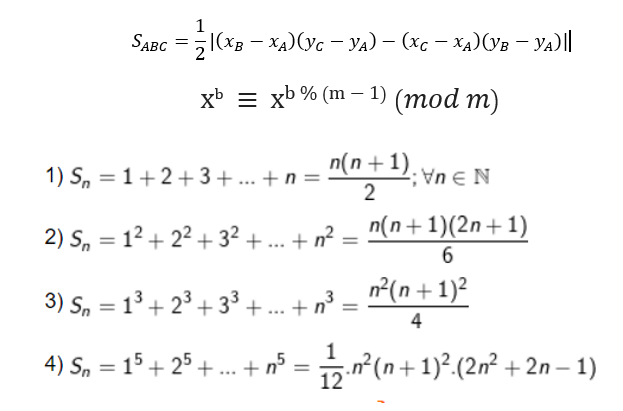
cout << i.first << "^" << i.second << "\n";

}

}

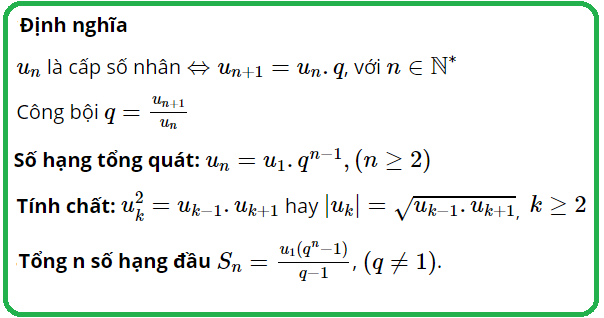
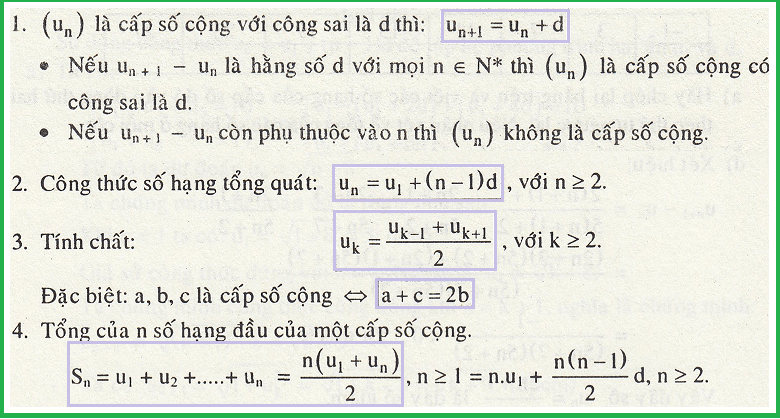
}

**Công thức toán**



**cấp số :**

tổng csn lùi vô hạn: S = u1 / (1-q)



**Fermat nhỏ**: a^(p-1) đồng dư 1 (mod p) (p là snt)

a ^ (phi(n) ) đồng dư 1 ( mod n ) (gcd(a,n) = 1)

a ^ (phi(n) -1) đồng dư a^(-1) (mod n và gcd(a,n = 1)

**39.CÂY KHUNG NHỎ NHẤT**

int par[230997];

int anc(int p) {

    if (par[p] == p)

        return p;

    else

        return par[p] = anc(par[p]);

}

void join(int p, int q) { par[anc(p)] = anc(q); }

typedef pair<int, int> ii;

typedef pair<int, ii> iii;

#define X first

#define Y second

vector<iii> edge;

int n, m;

main() {

    scanf("%d%d", &n, &m);

    for (int i = 1; i <= n; i++)

        par[i] = i;

    while (m--) {

        int p, q, w;

        scanf("%d%d%d", &p, &q, &w);

        edge.push\_back(iii(w, ii(p, q)));

    }

    sort(edge.begin(), edge.end());

    vector<iii>::iterator it;

    int r = 0;

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

        if (anc(it->Y.X) != anc(it->Y.Y)) {

            join(it->Y.X, it->Y.Y);

            r += it->X;

        }

    }

    printf("%d\n", r);

}

**40.ĐƯỜNG ĐI NGẮN NHẤT**

const int oo = 1000111000;

typedef pair<int, int> ii;

vector<ii> a[2309];

int n, m;

int d[2309];

void dijkstra() {

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

    for (int i = 1; i <= n; i++)

        d[i] = oo;

    d[1] = 0;

    pq.push(ii(0, 1));

    while (pq.size()) {

        int u = pq.top().second;

        int du = pq.top().first;

        pq.pop();

        if (du != d[u])

            continue;

        for (int i = 0; i < a[u].size(); i++) {

            int v = a[u][i].second;

            int uv = a[u][i].first;

            if (d[v] > du + uv) {

                d[v] = du + uv;

                pq.push(ii(d[v], v));

            }

        }

    }

}

int main() {

    int p, q, m, w;

    scanf("%d%d", &n, &m);

    while (m--) {

        scanf("%d%d%d", &p, &q, &w);

        a[p].push\_back(ii(w, q));

        a[q].push\_back(ii(w, p));

    }

    dijkstra();

    for (int i = 1; i <= n; i++)

        printf("d( 1 -> %d ) = %d\n", i, d[i]);

}