Contents

1. BIT Tree 1D:

```
int test,n,a[maxn+1];
int sum[maxn+1];

void insert_t(int x, int val) {
    while (x <= n) {
        sum[x] += val;
        x += (x&(-x));  //di chuyen toi nut cha
    }
}
int sum_t(int x) {
    int t = 0;
    while (x>0) {
        t += sum[x];
        x &= (x-1); //di chuyen toi nut ke
        //x-=(x&(-x));
    }
    return t;
}
```

2. BIT Tree 2D:

```
int sum[maxn+3][maxn+3],a[maxn+3][maxn+3];

void insert_t(int x,int y, int val) {
    while (x <= maxn) {
        int y1 = y;
        while (y1 <= maxn) {
            sum[x][y1] += val;
            y1 += (y1&(-y1));
        }
        x += (x&(-x));  //di chuyen toi nut cha
    }
}
int sum_t(int x, int y) {
    int t = 0;
    while (x>0) {
        int y1 = y;
        while (y1 > 0) {
            t += sum[x][y1];
            y1 &= (y1-1);
        }
        x &= (x-1); //di chuyen toi nut ke
    }
    return t;
}
void init() {
    FOR (i,1,maxn)
        FOR (j,1,maxn) {
            sum[i][j] = 0;
            a[i][j] = 0;
        }
}
```

3. Segment Tree 1D:

```
node it[maxn*4];

void build_it(int i,int l, int r) {
    if (l > r) return;
    if (l == r) {
        it[i].min_ = it[i].max_ = a[l];
        return;
    }
    int mid = (l+r) >> 1;
    build_it(i*2, l, mid);
    build_it(i*2+1, mid+1, r);
    it[i].min_ = min ( it[i*2].min_, it[i*2+1].min_);
    it[i].max_ = max ( it[i*2].max_, it[i*2+1].max_);
}
int get_min(int i, int u, int v, int l, int r) {
    if (l > v || r < u) return MAX;
    if (l >= u && r <=v) return it[i].min_;
    int mid = (l+r) >> 1;
    return min (get_min(i*2,u,v,l,mid), get_min(i*2+1,u,v,mid+1,r));
}
```

```
void update(int i, int u, int v, int l, int r) {
    if (r < u ||l > v) return;
    if (l >= u && r <= v) {
        it[i]++;
        return;
    }
    int mid = (l+r)>>1;
    update(i*2,u,v,l,mid);
    update(i*2+1,u,v,mid+1,r);
}
```

4. Segment Tree 2D:

```
(1081 - Light OJ) The game is played on a 2D N x N grid
Input
Input starts with an integer T (≤ 3), denoting the number of test
cases.
The first line of a case is a blank line. The next line contains two
integers N (1 ≤ N ≤ 500), Q (0 ≤ Q ≤ 50000).
Each of the next N lines will contain N space separated integers
forming the grid. All the integers will be between 0 and 105.
Each of the next Q lines will contain a query which is in the form I
J S (1 ≤ I, J ≤ N and 1 ≤ I + S, J + S < N and S > 0).
Output
For each test case, print the case number in a single line.
Then for each query you have to print the maximum integer found in
the square whose top left corner is (I, J) and whose bottom right
corner is (I+S-1, J+S-1).
```

```
#define maxn  505
#define MOD 1000000005

int n,a[maxn][maxn];
int it[maxn*maxn*16];

struct point {
    int _x;
    int _y;
    point(int x,int y) {
        _x = x;
        _y = y;
    }
};

int MAX(int a,int b,int c,int d) {
    return max (max(a,b),max(c,d));
}

void build_it_2d(int i,point top_left,point bottom_right) {
    it[i] = -1; /*****IMPORTANT FUCKKKKKKKK******/
    if (top_left._x > bottom_right._x || top_left._y > bottom_right._y) return;
    if (top_left._x == bottom_right._x && top_left._y == bottom_right._y) {
        it[i] = a[top_left._x][top_left._y];
        return;
    }
    int mid_x = (top_left._x + bottom_right._x) / 2;
    int mid_y = (top_left._y + bottom_right._y) / 2;

    build_it_2d(i*4+1, top_left, point(mid_x,mid_y));
    build_it_2d(i*4+2, point(mid_x+1,top_left._y), point(bottom_right._x,mid_y));
    build_it_2d(i*4+3, point(top_left._x,mid_y+1), point(mid_x,bottom_right._y));
    build_it_2d(i*4+4, point(mid_x+1,mid_y+1), bottom_right);

    it[i] = MAX(it[i*4+1],it[i*4+2],it[i*4+3],it[i*4+4]);
}
int get_max_it_2d(int i,point u, point v, point top_left, point bottom_right) { //get u,v,
dang xet top_left, bottom_right
    if (top_left._x > bottom_right._x || top_left._y > bottom_right._y) return -1;
    if (top_left._x > v._x || bottom_right._x < u._x || top_left._y > v._y ||
bottom_right._y < u._y) return -1;
    if (u._x <= top_left._x && bottom_right._x <= v._x && u._y <= top_left._y &&
bottom_right._y <= v._y) {//top,bottom inside u,v
        return it[i];
    }
    int mid_x = (top_left._x + bottom_right._x) / 2;
    int mid_y = (top_left._y + bottom_right._y) / 2;
    return MAX( get_max_it_2d(i*4+1,u,v,top_left, point(mid_x,mid_y)),
                get_max_it_2d(i*4+2,u,v,point(mid_x+1,top_left._y),
point(bottom_right._x,mid_y)),
                get_max_it_2d(i*4+3,u,v,point(top_left._x,mid_y+1),
point(mid_x,bottom_right._y)),
                get_max_it_2d(i*4+4,u,v,point(mid_x+1,mid_y+1), bottom_right)
    );

}
```

5. RMQ:

```
int a[maxn],f[maxn][32],n;

void init_rmq(int n)
{
    FOR (i,1,n) f[i][0]=a[i];
    for (int j=1; (1<<j) <= n; j++)
        for (int i=1; i+(1<<j)-1<=n; i++)
            f[i][j] = min (f[i][j-1],f[i+(1<<(j-1))][j-1]);
}
int get_rmq(int i,int j)
{
    int k = log2(j-i+1);
    return min(f[i][k],f[j-(1<<k)+1][k]);
}
```

6. Deque Min-max đoạn tịnh tiến:

```
/**
Problem MINK - VOJ
Cho day n phan tu va so k.
Voi moi~ doan con co k phan tu, tim phan tu be nhat.
DPT: O(n)
*/
int n,k,a[maxn];;

int main()
{
    int test;
    scanf("%d",&test);
    while (test--) {
        scanf("%d %d",&n,&k);
        /**Deque luu chi? so (KO phai luu gia tri)
        Cac phan tu cua deque luon dk sap xep tang dan*/
        deque<int> D;
        FOR (i,1,n) {
            scanf("%d",&a[i]);
            if (i > k) {
                /**Loai bo cac phan tu KO thuoc [i-k+1, i]*/
                while (!D.empty() && D.front() <= i-k) D.pop_front();
            }
            /**Loai bo cac phan tu >= a[i] */
            while (!D.empty() && a[D.back()] >= a[i]) D.pop_back();
            /**Push i vao Deque*/
            D.push_back(i);
            /**Phan tu dau tien cua deque luon la phan tu be nhat cua doan
[i-k+1,i]*/
            if (i >= k) printf("%d ", a[D.front()]);
        }
        printf("\n");
    }
    return 0;
}
```

7. Stack – Tính mảng left, right

```
/**
Problem: 1083 - Histogram
Input
Input starts with an integer T (≤ 20), denoting the number of test cases.

Each case contains a line with an integer N (1 ≤ N ≤ 30000) denoting the
number of rectangles.
The next line contains N space separated positive integers (≤ 30000) denoting
the heights.

Output
For each case, print the case number and the largest rectangle that can be
made.

Solution: Su dung stack de tim mang left, right
DPT: O(n)
*/

int n,h[maxn],L[maxn],R[maxn];

int main()
{
    #ifndef ONLINE_JUDGE
    freopen("test.inp","r",stdin);
    //freopen("test.out","w",stdout);
    #endif
    int test;
    scanf("%d",&test);
    FOR (te,1,test) {
        scanf("%d",&n);
        FOR (i,1,n) scanf("%d",&h[i]);
        h[n+1] = 0;
        stack <int> S; S.push(0);
        FOR (i,1,n+1) {
            while (h[i] < h[S.top()]) {
                R[S.top()] = i;
                S.pop();
            }
            if (h[i] == h[S.top()]) L[i] = L[S.top()]; else L[i] = S.top();
            S.push(i);
        }
        int ans = 0;
        FOR (i,1,n) {
            //printf("L = %d R = %d\n", L[i],R[i]);
            ans = max (ans, h[i] * (R[i] - L[i] - 1));
        }
        printf("Case %d: %d\n", te, ans);
    }

    return 0;
}
```

8. LCA Problem (Sử dụng QHĐ để jump):

Một trong những cách làm là thế này:
Gọi f[u, k] là cha bậc 2^k của u định nghĩa như sau:
f[u, 0] = nút cha của u,
f[u, k] = f[v, k - 1], với v = f[u, k - 1].
Tức là cha bậc 2^k là cha bậc 2^(k-1) của cha bậc 2^(k-1)

Để tìm cha bậc q của u, thay vì từ u đi theo liên kết parent q lần ta có thể phân tích q thành tổng các lũy thừa của 2 giống như thuật toán đổi nhị phân:
q = 2^x1 + 2^x2 + ...+2^xp
Sau đó nhảy theo các liên kết:
v1 = f[u , x1]
v2 = f[v1, x2]
v3 = f[v2, x3]
v4 = f[v3, x4]
...

Với 2 nút u và v cần tìm LCA, đầu tiên nếu depth[u] - depth[v] > 0 thì từ v nhảy lên đúng depth[u] - depth[v] bước để cho 2 nút đó có cùng độ sâu.
Nếu u == v thì xong.
Nếu u!=v, nếu cha bậc 2^k của chúng khác nhau thì cả 2 cùng nhảy lên 2^k bước lên cha bậc 2^k của chúng. Ở đây k được xét lần lượt theo dãy [lgn], [lgn] - 1, ..., 4, 3, 2, 1, 0. Cụ thể là nếu f[u, k] != f[v, k] thì đặt u = f[u, k], v = f[v, k]. Sau khi duyệt xong dãy [lgn], [lgn] - 1, ..., 4, 3, 2, 1 thì u và v trở thành 2 nút cùng cha. Đáp số LCA(u, v) là cha (trực tiếp) của chúng.

```
/**Problem RENDEZVOUS - ACM Regional Vietnam 2010*/
#define maxn  2000005
#define MOD 1000000005

int n,k; // number of node & queries
vector <int> adj[maxn];      //danh sach ke
int parent[maxn];            //parent[i]  = parent of node number i
bool Free[maxn];             //Free[i]=1 neu dinh i chua tham, nguoc lai
la 0
int f[maxn][20];             //f[u][k] = cha thu 2^k cua dinh u
                             //f[u][0] = cha truc tiep cua node u
                             //f[u][k] = f[v,k-1] voi v = f[u][k-1]
                             //Tuc la cha bac 2^k la cha bac 2^(k-1) cua
cha bac 2^(k-1)
int h[maxn];                 //chieu cao (height) cua node i

void bfs(int u) {
    queue<int> Q;
    int v;
    Free[u] = 0;
    h[u] = 1;
    Q.push(u);
    while (!Q.empty()) {
        u =  Q.front(); Q.pop();
        FO (i,0,adj[u].size()) {
```

```
                v = adj[u][i];
                if (Free[v]) {
                    parent[v] = u;
                    Free[v] = 0;
                    h[v] = h[u] + 1;
                    Q.push(v);
                }
            }
        }
}
void initParent() {          //tim cha cua cac dinh
    SET(Free,1);
    parent[1]=0;             //dinh 1 la root
    bfs(1);//root is node 1
    //visit(1,1);
    //FOR (i,1,n) cout << h[i] << " ";
}
void initF() {
         /**tao LCA*/
    FOR (u,1,n) f[u][0]=parent[u];//f[u][0] = cha truc tiep cua node u
    int m = log2(n);
    FOR (k,1,m)
        FOR (u,1,n) {
            int v = f[u][k-1];
            f[u][k] = f[v][k-1];
            //printf("%d %d %d\n",u,k,f[u][k]);
        }
}
void jump(int &u,int height) {
         /**Nhay tu u den node co do cao la height*/
    int m = log2(n);
    FORD (k,m,0) {
        int v = f[u][k];
        if (h[v] >= height) {
            u = v;
        }
    }
}
int lca(int u,int v) {
    if (h[u]<h[v]) swap(u,v);
    jump(u,h[v]);
    if (u==v) return u;
    int u1,v1,m=log2(n);
    FORD (i,m,0) {
        u1 = f[u][i];
        v1 = f[v][i];
        if (u1 != v1) {//u1 va v1 ko la cha chung
            u = u1;
            v = v1;
        }
    }
    return parent[u];
}
int main() {
         freopen("RENDEZVOUS.INP","r",stdin);
         freopen("RENDEZVOUS.OUT","w",stdout);
    scanf("%d %d",&n,&k);
    int u,v;
    FO (i,1,n) {
        scanf("%d %d",&u,&v);
        adj[u].pb(v);
        adj[v].pb(u);
    }
    initParent(); //finish init parent
    initF();    //finish init lca
    printf("finish init lca");
    FOR (i,1,k) {
        scanf("%d %d",&u,&v);
        printf("%d\n",lca(u,v));
    }
        return 0;
}
```

9. LCA Problem (Đưa về bài toán RMQ – Dùng DFS visit + IT Tree)

```
#define maxn  400005
#define MOD 1000000005
int n,k,maxn_it;
VI adj[maxn];
int visited[maxn],h[maxn];
int visit[maxn], height[maxn], nVisit, nHeight;
int pos[maxn], parent[maxn];


void dfs(int u) {
    stack<int> S;
    S.push(u);
    h[u] = 1;
    while (!S.empty()) {
        int u = S.top(); S.pop();
        visit[++nVisit] = u;
        height[++nHeight] = h[u];

        if (visited[u]) continue;//da tham cac dinh ke u
        visited[u] = 1;
        pos[u] = nVisit;

        FO (i,0,adj[u].size()) {
            int v = adj[u][i];
            if (visited[v] == 0) {
                S.push(u);      //quay lai dinh cha
                S.push(v);
                h[v] = h[u] + 1;
            }
        }
    }
}
int it_min[maxn*3];
void build_it(int i, int l, int r) {
    if (l>r) return;
    if (l == r) {
        it_min[i] = l;
        return;
    }
    int mid = (l+r)>>1;
    build_it(i*2,l,mid);
    build_it(i*2+1,mid+1,r);
```

```
    if (height[it_min[i*2]] < height[it_min[i*2+1]]) it_min[i] =
it_min[i*2];
    else it_min[i] = it_min[i*2+1];
}
int get_min(int i, int u, int v, int l, int r) {
    if (l > v || r < u) return 0;
    if (l >= u && r <= v) {
        return it_min[i];
    }
    int mid = (l+r)>>1;
    int h1 = get_min(i*2,u,v,l,mid);
    int h2 = get_min(i*2+1,u,v,mid+1,r);
    if (height[h1] < height[h2]) return h1; return h2;
}
int get_lca(int u,int v) {
    int l = pos[u];
    int r = pos[v];
    if (l > r) swap(l,r);
    int pos = get_min(1,l,r,1,maxn_it);
    return visit[pos];
}
void init_lca() {
    SET(visited,0);
    SET(h,0);
    h[1] = 1;
    dfs(1);        //init array visit and height

    maxn_it = nVisit;
    height[0] = infi;
    build_it(1,1,maxn_it);    //build iterval tree
}
int main()
{
    #ifndef ONLINE_JUDGE
    //freopen("RENDEZVOUS.inp","r",stdin);
    freopen("RENDEZVOUS.inp","r",stdin);
    freopen("RENDEZVOUS.out","w",stdout);
    #endif
    int u,v;
    scanf("%d %d",&n,&k);
    FOR (i,1,n-1) {
        scanf("%d%d",&u,&v);
        adj[u].pb(v);
        adj[v].pb(u);
    }
    init_lca();
//    FOR (i,1,nVisit) cout << visit[i] << " "; cout << endl;
//    FOR (i,1,nHeight) cout << height[i] << " ";cout << endl;
//    FOR (i,1,n) cout << pos[i] << " "; cout << endl;

    FOR (i,1,k) {
        scanf("%d %d",&u,&v);
        printf("%d\n",get_lca(u,v));
    }
    return 0;
}
```

10. Suffix Array and LCP Table

```
/**Problem 343 - Le Minh Hoang*/
#define maxn  10000005
#define MOD 1000000005

/*
Suffix array O(n lg^2 n)
LCP table O(n)
*/

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

/**
Before each test case memset "sa" and "lcp"
*/

const int MAXN = 100010;
char S[maxn];
int N, gap;
int sa[MAXN], pos[MAXN], tmp[MAXN], lcp[MAXN];

bool sufCmp(int i, int j)
{
    if (pos[i] != pos[j])
        return pos[i] < pos[j];
    i += gap;
    j += gap;
    return (i < N && j < N) ? pos[i] < pos[j] : i > j;
}

void buildSA()
{
    N = strlen(S);
    REP(i, N) sa[i] = i, pos[i] = S[i];
    for (gap = 1;; gap *= 2)
    {
        sort(sa, sa + N, sufCmp);
        REP(i, N - 1) tmp[i + 1] = tmp[i] + sufCmp(sa[i], sa[i +
1]);
        REP(i, N) pos[sa[i]] = tmp[i];
        if (tmp[N - 1] == N - 1) break;
    }
}

void buildLCP()
{
    for (int i = 0, k = 0; i < N; ++i)
        if (pos[i] != N - 1)
        {
            for (int j = sa[pos[i] + 1]; S[i + k] == S[j + k];)
            ++k;
            lcp[pos[i]] = k;
            if (k) --k;
        }
}
 // end namespace SuffixArray
int test;
int main()
{
```

```
    #ifndef ONLINE_JUDGE
    freopen("SUBSTR_ACM.inp","r",stdin);
    freopen("SUBSTR_ACM.OUT","w",stdout);
    #endif
    scanf("%d\n",&test);
    while (test--) {
        gets(S);
        /**INIT BEFORE TEST CASE*/
        SET(sa,0);
        SET(lcp,0);
        /**----------------------*/
        strcat(S,"@");
        buildSA();
        buildLCP();
    //    FO (i,0,N) cout << sa[i] << endl; cout << endl;
    //    FO (i,0,N) cout << lcp[i] << endl;
        LL ans = 0;
        FO (i,0,N) {
            ans += (N - 1 - sa[i]) - lcp[i];
        }
        cout << ans << endl;
    }
    return 0;
}
/**
Sources: http://codeforces.com/blog/entry/4025
BANANA@
-------
Suffix
0 - BANANA@
1 -  ANANA@
2 -   NANA@
3 -    ANA@
4 -     NA@
5 -      A@
6 -       @
-------
Suffix Array
6 - @           0
5 - A@          1
3 - ANA@        2
1 - ANANA@      3
0 - BANANA@     4
4 - NA@         5
2 - NANA@       6
-------
LCP
lcp[0] = 0
lcp[1] = 1
lcp[2] = 3
lcp[3] = 0
lcp[4] = 0
lcp[5] = 2
lcp[6] = ###
*/
```

11. Disjoint Set

```
long findset(long i) {
    if (parent[i]<0) return i;   //Nếu i là gốc
    parent[i]=findset(parent[i]);  // Heuristic Path Compression
    return parent[i];
}
void Union(long x,long y) {
    long u,v;
    u=findset(x);  //Tìm gốc của x
    v=findset(y);  //Tìm gốc của y
    if (u==v) return;  //Cùng gốc

    //Heuristic Union by rank
    if (parent[u]<parent[v]) {  //u nhiều con hơn v
       parent[u]+=parent[v];
       parent[v]=u;  //Nối nhánh gốc v vào u
    } else {
       parent[v]+=parent[u];
       parent[u]=v;
    }
}
Khởi tạo: Tất cả các parent[] = -1.
```

12. Trie

```
struct node {
      bool is_end;
      int maxWord;
      node* child[26];
} *root;
void init() { //init root
      root = new node();
      root->is_end=0;
      root->maxWord=0;
}
void insert(string s) {//insert word
      node* current = root;
      FO (i,0,s.size()) {
            if (current->child[s[i]-'a']==NULL)
                  current->child[s[i]-'a'] = new node();
            //current->maxWord = max(current->maxWord
            current = current->child[s[i]-'a'];
            //di nhanh s[i]
      }
      current->is_end=1;
}
```

13. Dijkstra Pirority Queue

```
/**Solved problem FLOYD VOJ*/
#define maxn  105
#define MOD 1000000005
#define INF 10000010

priority_queue <II, VII, greater<II> > Q;
//Q->first: distance
//Q->second: vertex
vector <pair<int,int> > adj[maxn];
int dist[maxn], dad[maxn];

void init_dijkstra(int n, int start) {
    FOR (i,1,n) {
        dist[i] = INF;
        dad[i] = -1;
    }
    dist[start] = 0;
    while (!Q.empty()) Q.pop();
    Q.push(make_pair(0,start));
}
int dijkstra(int type_query, int n, int start, int finish) {
    init_dijkstra(n,start);
    int u,v,d_u_v;
    while (!Q.empty()) {
        u = Q.top().second; Q.pop();
        if (u == finish) break;
        FO (i,0,adj[u].size()) {
            v = adj[u][i].second;
            d_u_v = adj[u][i].first;
            if (dist[u] + d_u_v < dist[v]) {
                dist[v] = dist[u] + d_u_v;
                dad[v] = u;
                Q.push(make_pair(dist[v], v));
            }
        }
    }
    /****TRACE MIN PATH*****/
    VI path;
    for (int i = finish; i != -1; i = dad[i]) path.push_back(i);
    reverse(path.begin(), path.end());
    /*********************/
    if (type_query == 0) printf("%d\n", dist[finish]);
    else {
        printf("%d",path.size());
        FORV (it,path) printf(" %d", *it);
        printf("\n");
    }
    return dist[finish];
}
```

14. Thành phần liên thông mạnh (SCC)

```
vector <int> a[maxn];  // danh sách kề
stack <int> s;
int n,m,t;
int number[maxn],low[maxn],Free[maxn],Count,ans;
void visit(int u) {
    low[u]=number[u]=++Count;
        s.push(u);
        FO (i,0,a[u].size()) {
                int v =  a[u][i];
                if (Free[v]==0) {
                        if (number[v]==0) {
                                visit(v);
                                low[u]=min(low[u],low[v]);
                        }       else
low[u]=min(low[u],number[v]);
                }
        }
        if (low[u]==number[u]) {
                ans++;  //tăng số thành phần lt mạnh
                set <int> t;  //tập các đỉnh trong thành
phần lt này
                set <int> :: iterator it;
                while (1) {
                        t.insert(s.top());
                        Free[s.top()]=1;
                        if (s.top()==u) {
                                s.pop();
                                break;
                        };
                        s.pop();
                }
                // in ra các đỉnh thuộc tplt
      for (it=t.begin();it!=t.end();it++) cout << *it;
                cout << endl;

        }
}
main() {
        scanf("%d %d",&n,&m);
        int u,v;
        FOR (i,1,m) {
                scanf("%d %d",&u,&v);
                a[u].pb(v);  // danh sách kề
        }
        FOR (i,1,n)
                if (Free[i]==0) visit(i);
        cout << ans;
}
```

15. Khớp và cầu:

```
/*
- set <int> a[u]: là tập các đỉnh kề đỉnh u
- parent[u] = đỉnh cha của đỉnh u trong cây DFS
- num[u] = thứ tự duyệt đến
- low[u] = giá trị num[.] nhỏ nhất của những đỉnh đến được từ nhánh DFS gốc u bằng một cung ngược
- numChild[u] = số nhánh con của cây DFS gốc u
- soCau = số cầu của đồ thị
- soKhop = số khớp của đồ thị
- laKhop[u] = 1 nếu u là khớp, ngược lại là 0.
*/
void dfs(int u) {
    num[u] = low[u] = dem++;
    int v;
    FORV (it,a[u]) {
        v = *it;
        a[v].erase(u);                     //xóa cung v->u
        if (parent[v]==0) {
            parent[v] = u;
            dfs(v);
            low[u] = min(low[u],low[v]);
        }   else
            low[u] = min(low[u],num[v]);
    }
}
void process() {
    Set(parent,0);
    FOR (i,1,n)
        if (parent[i]==0) {
            parent[i] = -1;
            dfs(i);
        }
}
void timCau() {
    FOR (v,1,n) {
        int u = parent[v];
        if (u!=-1 && low[v]>=num[v]) {
            // u-v la cau
            soCau++;
        }
    }
}
void timKhop() {
    //đếm số con của đỉnh u trên cây DFS
    FOR (v,1,n) {
        int u = parent[v];
        if (u!=-1) numChild[u]++;
    }
    //u là gốc của 1  cây DFS thì là khớp nếu nó có nhiều hơn 1 nhánh con
    FOR (u,1,n)
        if (parent[u]==-1 && numChild[u]>1) laKhop[u]=1;
    // u ko pải là gốc là khớp nếu có đỉnh v là con của u và low[v] >= num[u]
    FOR (v,1,n) {
        int u = parent[v];
        if (u!=-1 && parent[u]!=-1 && low[v]>=num[u]) laKhop[u]=1;
    }
    //đánh dấu xong các đỉnh là khớp
    FOR (u,1,n)
        if (laKhop[u]) soKhop++;
}
int main() {
    scanf("%d %d",&n,&m);
    int u,v;
 FOR (i,1,m) {scanf("%d %d",&u,&v);a[u].insert(v);a[v].insert(u); }
    process();timCau();timKhop();
    printf("%d %d",soKhop,soCau);
}
```

16. Floyd – Đường đi ngắn nhất giữa mọi cặp đỉnh

```
for (int k=1;k<=n;k++)
      for (int u=1;u<=n;u++)
          for (int v=1;v<=n;v++)
              if (c[u][v]>c[u][k]+c[k][v]) {
                  c[u][v] = c[u][k]+c[k][v];   // Tối ưu
                  trace[u][v] = trace[u][k];   //Lưu vết
              }
```

17. Sắp xếp tô-pô:

```
void dfs_visit(ll u) {
    FO (i,0,a[u].size()) {  //Thăm các đỉnh kề u
        ll v=a[u][i];
        if (Free[v]==0) {
                Free[v]=1;        // Đánh dấu v đã thăm
                dfs_visit(v);     // Thăm v
        }
        }
//Duyệt xong nhánh DFS gốc u - Thứ tự đỉnh u trong đthị mới là 'dem'
        list[dem]=u;
        dem--;
}
void numbering() {  // đánh số các đỉnh
    FOR (i,1,n) Free[i]=0;
    dem=n;
    FOR (i,1,n)
        if (Free[i]==0) dfs_visit(i);
}
void topo() {
    FOR (i,1,n) {
        ll u=list[i];  //Ánh xạ lại đỉnh u
        FO (j,0,a[u].size()) {  //Duyệt các đỉnh kề u
            ll v=a[u][j];  //Ánh xạ lại
            d[v]=min(d[v],d[u]+c[u][v] ;
        }
    }
}
```

18. Xử lý số lớn:

```
class bigNum {
private: vector <int> num;
/* Lưu ý: Các chữ số được xếp theo thứ tự ngược (tức là: chữ số đầu      tiên là chữ số hàng đơn vị, rồi đến hàng chục,…)
*/
public:
    bigNum() {  //khởi tạo
        num.clear();    }
    bigNum(string s) {  //khởi tạo số bigNum từ xâu C++
        FORD (i,s.size()-1,0)
            num.pb(s[i]-'0');   }
    bigNum (int x) {  //tạo số bigNum từ số nguyên
        if(x == 0) num.pb(0);
        while (x>0) {
            num.pb(x%10);
            x/=10;
        } }
    void push_back(int x) {  //thêm vào sau số bigNum
        num.pb(x);}
    void del_0() {  //xóa bỏ các chữ số 0 vô nghĩa
        while (num.size()>1 && num.back()==0) num.pop_back();
    }
    int size() {  // số các số của bigNum
        return num.size();
    }
    friend void add_0(bigNum &a,bigNum &b) {//cho a,b có cùng số chữ số
        if (a.size()<b.size())
            while (a.size()!=b.size()) a.pb(0);
        if (a.size()>b.size())
            while (a.size()!=b.size()) b.pb(0);
    }
   friend istream &operator>>(istream &in,bigNum &x) { //nhập
        string s;
        in >> s;
        x = bigNum(s);
        x.del_0();
        return in;
    }
   friend ostream &operator<<(ostream &out,bigNum x) { //xuất
        FORD (i,x.size()-1,0)
            out << x.num[i];
        return out;
    }
    friend bool operator>(bigNum a,bigNum b){
        if(a.size() > b.size()) return true;
        if(a.size() < b.size()) return false;
        if(a.size() == b.size()){
            FORD(i,a.size()-1,0){
                if(a.num[i] > b.num[i]) return true;
                if(a.num[i] < b.num[i]) return false;
                if(a.num[i] == b.num[i]) continue;
            }
        }
    }
    friend bool operator==(bigNum a,bigNum b){
        if(a.size() == b.size()){
            FORD(i,a.size()-1,0)
                if(a.num[i] != b.num[i]) return false;
            return true;
        }
        return true;
    }
    friend bigNum operator+(bigNum a,bigNum b) { //cộng
        bigNum ans;
        add_0(a,b);
        int memo=0;
        FO (i,0,a.size()) {
            ans.pb((a.num[i]+b.num[i]+memo)%10);
            memo = (a.num[i]+b.num[i]+memo)/10;
        }
        if (memo) ans.pb(memo);
        return ans;
    }
    friend bigNum operator-(bigNum a,bigNum b) {  //trừ
        bigNum ans;
        int memo=0;
        add_0(a,b);
        FO (i,0,a.size()) {
            ans.pb((a.num[i]-b.num[i]-memo+10)%10);
            if (a.num[i]-b.num[i]-memo<0) memo=1; else memo=0;
        }
        ans.del_0();
        return ans;
    }
    friend bigNum operator*(bigNum a,bigNum b) {  //nhân
        bigNum ans;
        int memo=0,tmp,jj;
        FO (i,0,b.size()) a.pb(0);
        FO (i,0,a.size()) {
            tmp=memo;
            jj=i;
            FO (j,0,b.size()) {
                tmp+=a.num[jj--]*b.num[j];
                if (jj<0) break;
            }
            ans.pb(tmp%10);
            memo=tmp/10;
        }
        ans.del_0();
        return ans;
    }
};
```

19. Chu trình Euler:

```
stack := (1); //Ngăn xếp ban đầu chỉ chứa một đỉnh bất kỳ, chẳng hạn đỉnh 1
repeat
  u := Get; //Đọc phần tử ở đỉnh ngăn xếp
  if ∃(u, v) ∈E then //Từ u còn đi tiếp được
    begin
      Push(v);
      E := E - {(u, v)}; //Xóa cạnh (u, v) khỏi đồ thị
    end;
  else //Từ u không đi đâu được nữa
    begin
      u := Pop; //Lấy u khỏi ngăn xếp
      Output ← u; //In ra u
    end;
until stack = Ø; //Lặp tới khi ngăn xếp rỗng
```

20. KMP:

```
int next[maxn];
int kq[maxn],dem=0;
char T[maxn],P[maxn];
int M,N;

void initKMP() {
    next[0] = -1;
    int j = -1;
    FO (i,1,M) {
        while (j>-1 && P[i]!=P[j+1]) j=next[j];
        if (P[i] == P[j+1]) j++;
        next[i] =j;
    }
}

void process() {  //T=Text; P=Pattern
    int j = -1;
    FO (i,0,N) {
        while (j>-1 && T[i]!=P[j+1]) j = next[j];
    //tim xau P[1..j] la suffix cuaT[1..i-1] va P[j+1] ==
T[i]
        if (T[i] == P[j+1]) j++;
        if (j>=M-1) {
            kq[++dem] = i-j+1;
            j = next[j];  //khi tim thay roi thi dich luon
        }
    }
}

int main() {
    gets(T);gets(P);
    N = strlen(T);
    M = strlen(P);
    initKMP();
    process();
    FOR (i,1,dem) printf("%d ",kq[i]);
```

21. Lower bound:

```
//Tìm phần tử đầu tiên trong dãy thỏa mãn hàm check
int l = 1, r = 1000000001, mid, c = r - l;
    while (c > 0) {
        int step = c/2;
        int it;
        it = l + step;
        if (!check(it)) {
            l = ++it;
            c -= step + 1;
        }   else c = step;
    }
    cout << l << endl;//return l
```

```
template <class ForwardIterator, class T>
  ForwardIterator lower_bound (ForwardIterator first,
ForwardIterator last, const T& val)
{
  ForwardIterator it;
  iterator_traits<ForwardIterator>::difference_type count, step;
  count = distance(first,last);
  while (count>0)
  {
    it = first; step=count/2; advance (it,step);
    if (*it<val) {                 // or: if (comp(*it,val)), for
version (2)
      first=++it;
      count-=step+1;
    }
    else count=step;
  }
  return first;
}
```

22. Template:

```
#include <set>
#include <map>
#include <list>
#include <cmath>
#include <queue>
#include <stack>
#include <cstdio>
#include <string>
#include <vector>
#include <cstdlib>
#include <cstring>
#include <sstream>
#include <iomanip>
#include <iostream>
#include <algorithm>
#include <ctime>
#include <deque>
#include <bitset>
```

```
#include <cctype>
#include <utility>

#define ULL unsigned long long
#define LL long long
#define FOR(i,a,b) for(int i = (a); i <= (b); i++)
#define FO(i,a,b) for(int i = (a); i < (b); i++)
#define FORD(i,a,b) for(int i= (a); i >= (b); i--)
#define FOD(i,a,b) for(int i= (a); i > (b); i--)
#define FORV(i,a) for(typeof(a.begin()) i = a.begin(); i != a.end();
i++)
#define SET(a,c) memset(a, c, sizeof(a))
#define fi first
#define se second
#define pb push_back
#define mp make_pair
#define eps 1e-5
#define infi 1e9
#define PI 2*acos(0.0)
using namespace std;

typedef pair<int,int>II;
typedef pair<int,II>PII;
typedef vector<int> VI;
typedef vector<II> VII;
typedef set<int> SI;
typedef map<string,int> MSI;
typedef map<int,int> MII;

template<class T> T gcd(T a, T b) {
    T r;
    while (b != 0) {
        r = a % b;
        a = b;
        b = r;
    }
    return a;
}
template<class T> T lcm(T a, T b) {
    return a / gcd(a, b) * b;
}
template<class T> T sqr(T x) {
    return x * x;
}
template<class T> T cube(T x) {
    return x * x * x;
}
template<class T> int getbit(T s, int i) {
    return (s >> i) & 1;
}
template<class T> T onbit(T s, int i) {
    return s | (T(1) << i);
}
template<class T> T offbit(T s, int i) {
    return s & (~(T(1) << i));
}
template<class T> T togglebit(T s, int i) {
    return s ^ (T(1) << i);
}
template<class T> int cntbit(T s) {
    return s == 0 ? 0 : cntbit(s >> 1) + (s & 1);
}
#define maxn  10000005
#define MOD 1000000005

int main()
{
    #ifndef ONLINE_JUDGE
    freopen("test.inp","r",stdin);
    //freopen("test.out","w",stdout);
    #endif

    return 0;
}
```

23. Hash:

```
#define base 1000000007LL
LL POW[maxn], hashT[maxn];
void Init(){
    POW[0] = 1;
    FOR(i,1,maxn) POW[i] = (POW[i-1] * 10) % base;
}
LL getHashT(int i, int j){
    return (hashT[j] - hashT[i-1]*POW[j-i+1]+base*base) % base;
}
int process(){
    m = T.size(); T = " " + T;
    n = P.size(); P = " " + P;
    LL hashP = 0; hashT[0] = 0;
    FOR(i,1,n) hashP = (hashP*10+P[i]-'0') % base;
    FOR(i,1,m) hashT[i] = (hashT[i-1]*10+T[i]-'0') % base;
    FOR(i,1,m-n+1)if(hashP == getHashT(i,i+n-1)) return i;
    return 0;
}
```

24. Hash + BIT - :

```
/**
Cho xâu chữ cái in thường s độ dài ko quá 10^5. Có 2 loại truy vấn:
- đổi kí tự thứ i thành một kí tự a
- kiểm tra xâu con s[i]...s[j] có phải xâu đối xứng hay ko
Số truy vấn ko quá 10^5.
*/
int n, m, L, R;
string s, x;
char c[maxn];
LL tree1[maxn], tree2[maxn], a[maxn], b[maxn];
LL POW[maxn];

void update(int x, int val, LL bit[]){
    while(x < maxn){
        bit[x] = (bit[x] + val) % MOD;
        x += (x&(-x));
    }
```

```
}

LL get(int x, LL bit[]){
    LL ans = 0;
    while(x > 0){
          ans = (ans + bit[x]) % MOD;
          x -= (x&(-x));
    }
    return ans;
}

void Init(){
    POW[0] = 1;
    FOR(i,1,100005) POW[i] = (POW[i-1]*10) % MOD;
    FOR(i,1,n) a[i] = (s[i]*POW[i]) % MOD;
    FOR(i,1,n) b[i] = (s[i]*POW[n+1-i]) % MOD;

    memset(tree1,0,sizeof(tree1));
    memset(tree2,0,sizeof(tree2));
    FOR(i,1,n) update(i,a[i],tree1);
    FOR(i,1,n) update(i,b[i],tree2);
}

int main(){
    scanf("%s", &c);
    s = string(c);
    n = s.size();
    s = '0' + s;
    Init();
    cin >> m;
    FOR(i,1,m)
    {
          cin >> x;
          if(x == "palindrome?"){
          scanf("%d %d", &L, &R);
          LL tong1 = get(R,tree1) - get(L-1,tree1) + MOD; tong1 %=
MOD;
          LL tong2 = get(R,tree2) - get(L-1,tree2) + MOD; tong2 %=
MOD;
          if((tong1*POW[n-R]) % MOD == (tong2*POW[L-1]) % MOD)
printf("Yes\n");
          else printf("No\n");
          //cout << tong1 << " " << tong2 << endl;
         }
          else{
              int u;
              char cc;
              scanf("%d %c", &u, &cc);
              LL tmp = ((int)cc - (int)s[u]) + MOD;
              s[u] = cc; //cout << s << endl;
              LL tang1 = (tmp*POW[u]) % MOD;
              update(u,tang1,tree1);
              LL tang2 = (tmp*POW[n+1-u]) % MOD;
              update(u,tang2,tree2);
          }
    }

    return 0;
}
```

25. Sàng nguyên tố + Phi hàm Euler:

```
// sàng Eratosthenes: O(n loglog n)
    memset(d,0,sizeof(d));
    FOR(i,2,sqrt(n)){
        if(d[i] == 0){
            for(int j = i*i; j <= n; j += i)
                d[j] = i;
        }
    }
    FOR(i,2,n) if(d[i] == 0) d[i] = i;

// Euler phi function
    phi[1] = 1;
    FOR(i,2,n){
        int j = i;
        int k = d[j];
        while(j % k == 0) j /= k;
        phi[i] = phi[j] * (i/j - i/(j*k));
    }

int phi(int n){
    int res = n;
    for(int i = 2; i*i <= n; i++){
        if(n % i == 0) res -= res/i;
        while(n % i == 0) n /= i;
    }
    if(n > 1) res -= res/n;
    return res;
}
```

26. Kiểm tra nguyên tố Miller – Rabin

```
ULL power_mod(ULL a,ULL b,ULL c){
    long long x=1,y=a; // long long is taken to avoid overflow of
intermediate results
    while(b > 0){
        if(b%2 == 1){
            x = (x%c) * (y%c); x%= c;
        }
        y = (y%c) * (y%c); // squaring the base
        y %= c;
        b /= 2;
    }
    return x%c;
}
ULL mulmod(ULL a,ULL b,ULL c){
    ULL x = 0,y = a%c;
    while(b > 0){
        if(b%2 == 1){
            x = (x+y)%c;
        }
        y = (y*2)%c;
        b /= 2;
    }
    return x%c;
}

/* Nếu a^(p-1) == 1 (mod p) thì p là số nguyên tố (Fermat nhỏ)
   phân tích p-1 = 2^s * d
```

```
   Để p là số nguyên tố thì a^d == 1 mod p hoặc tồn tại giá trị r (0<= r
<= s-1) sao cho a^(d*2^r) == -1 mod p.
Iteration is about 8 - 15
*/

int Miller(ULL p, int iteration){
    if(p == 2) return 1;
    if(p != 2 && p%2 == 0) return 0;
    LL d = p-1;
    while(d%2 == 0) d/= 2;
    LL s = (p-1)/d;
    FOR(i,1,iteration){
        LL a = rand()%(p-1) + 1;
        LL mod = power_mod(a,d,p);  /// mod = a^d % p;
        if(mod == 1) continue;
        FOR(i,0,s-1){
             if(mod == p-1) break;
            mod = (mod%p)*(mod%p);
            mod %= p;   /// phai ton tai mod == p-1; mod = mod^2;
        }
        if(mod != p-1) return 0;
    }
    return 1;
}
```

27. Các kiến thức cơ bản của số học:

```
// This is a collection of useful code for solving problems that
// involve modular linear equations.  Note that all of the
// algorithms described here work on nonnegative integers.

typedef vector<int> VI;
typedef pair<int,int> PII;

int mod(int a, int b) {
  return ((a%b)+b)%b;
}

LL GCD(LL a, LL b) {
    return b == 0 ? a : GCD(b, a % b);
}

int gcd(int a, int b) {
  int tmp;
  while(b){a%=b; tmp=a; a=b; b=tmp;}
  return a;
}

int lcm(int a, int b) {
  return a/gcd(a,b)*b;
}

// returns d = gcd(a,b); finds x,y such that d = ax + by
int extended_euclid(int a, int b, int &x, int &y) {
  int xx = y = 0;
  int yy = x = 1;
  while (b) {
    int q = a/b;
    int t = b; b = a%b; a = t;
    t = xx; xx = x-q*xx; x = t;
    t = yy; yy = y-q*yy; y = t;
  }
  return a;
}
// finds all solutions to ax = b (mod n)
VI modular_linear_equation_solver(int a, int b, int n) {
  int x, y;
  VI solutions;
  int d = extended_euclid(a, n, x, y);
  if (!(b%d)) {
    x = mod (x*(b/d), n);
    for (int i = 0; i < d; i++)
      solutions.push_back(mod(x + i*(n/d), n));
  }
  return solutions;
}

// computes b such that ab = 1 (mod n), returns -1 on failure
int mod_inverse(int a, int n) {
  int x, y;
  int d = extended_euclid(a, n, x, y);
  if (d > 1) return -1;
  return mod(x,n);
}

int mod_inverse2(int a, int base){
    a %= base;
    int mu = phi[base] - 1;
    return powerMod(a, mu, base);
}
// Chinese remainder theorem (special case): find z such that
// z % x = a, z % y = b.  Here, z is unique modulo M = lcm(x,y).
// Return (z,M).  On failure, M = -1.
PII chinese_remainder_theorem(int x, int a, int y, int b) {
  int s, t;
  int d = extended_euclid(x, y, s, t);
  if (a%d != b%d) return make_pair(0, -1);
  return make_pair(mod(s*b*x+t*a*y,x*y)/d, x*y/d);
}
// Chinese remainder theorem: find z such that
// z % x[i] = a[i] for all i.  Note that the solution is
// unique modulo M = lcm_i (x[i]).  Return (z,M).  On
// failure, M = -1.  Note that we do not require the a[i]'s
// to be relatively prime.
PII chinese_remainder_theorem(const VI &x, const VI &a) {
  PII ret = make_pair(a[0], x[0]);
  for (int i = 1; i < x.size(); i++) {
    ret = chinese_remainder_theorem(ret.first, ret.second, x[i],
a[i]);
    if (ret.second == -1) break;
  }
  return ret;
}

// computes x and y such that ax + by = c; on failure, x = y =-1
void linear_diophantine(int a, int b, int c, int &x, int &y) {
  int d = gcd(a,b);
  if (c%d) {
```

```
    x = y = -1;
  } else {
    x = c/d * mod_inverse(a/d, b/d);
    y = (c-a*x)/b;
  }
}
Họ nghiệm: x = x_0 + k*b/gcd(a,b), y = y_0 - k*a/gcd(a,b)
```

28. Số nhỏ nhất có N ước:

```
LL ans = 1e18 + 5; int n;
int p[] = { 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41 };
void dfs(int i, LL x, int c){
    if (c > n ) return;
    if (c == n && x < ans) ans = x;
    FOR(j, 1, 60){
        if (ans / p[i] < x) break;
        x *= p[i];
        if ((n % (j + 1)) == 0){
            c *= (j+1);
            dfs(i + 1, x, c );
            c /= (j+1);
        }
    }
}
cin >> n;  dfs(0, 1, 1);  cout << ans;
```

29. Bài toán lũy thừa tầng:

Áp dụng định lý Ferma nhỏ và định lý số dư Trung Hoa, giải hệ phương trình đồng dư:

$$\begin{cases} x \equiv a_1 \pmod{m_1} \\ x \equiv a_2 \pmod{m_2} \\ \dots \\ x \equiv a_k \pmod{m_k} \end{cases}$$

```
int limited(int heso, int uoc, int start, int finish, int base){
    //kiem tra xem luy thua tang co chia het cho base không? Yes ->
-1; No -> reminder
    double tu = base, mau = uoc;
    FOR(i,start+1,finish){
        if(heso > (double)log(tu)/log(mau)) {
            return -1;
        }
        else{
             tu = (double)log(tu)/log(mau);
             mau = heso;
        }
    }
    int tmp = heso;
    FORD(i,finish-1,start+1){
        tmp = Power(heso,tmp);
    }
    return powerMod(heso,tmp,base);
}
```

```
void phantich(vector<int> &V, int n){
    phân tích thành các thừa số nguyên tố + số mũ
}
LL Chinese_systems_equation(vector<int> B, vector<int> R, int base){
    /// sub-bases and reminders
    LL ans = 0;
    int length = B.size();
    vector<int> M, Y; M.resize(length); Y.resize(length);
    FOR(i,0,length-1) M[i] = base/B[i];
    FOR(i,0,length-1) Y[i] = mod_inverse(M[i], B[i]);

    FOR(i,0,length-1) {
        LL tmp = 1LL*R[i]*Y[i]*M[i];
        ans += tmp;
    }
    ans %= base;
    return ans;
}

LL dequy(int a, int start, int finish, int base){
    if(start == finish+1) return 1;
    if(start == finish) return a;
    vector<int> V;
    vector<int> ans;
    phantich(V, base);
    /// base = product of all V[i].fi ^ V[i].se
    FOR(i,0,V.size()-1){
        int tmp;
        int uoc = GCD(V[i], a);
        if(uoc != 1){
            tmp = limited(a, uoc, start, finish, V[i]);
            if(tmp == -1) ans.pb(0);
            else ans.pb(tmp);
        }
        else{
            if(phi[V[i]] == 1) ans.pb(1);
            else{
                tmp = dequy(a, start+1, finish, phi[V[i]]);
                ans.pb(powerMod(a, tmp, V[i]));
            }
        }
    }
    LL res = 0;
    res = Chinese_systems_equation(V, ans, base);
    return res;
}
```

30. Tổ hợp chập k của n:

Đệ quy:

```
LL tohop(LL n, LL k){
    LL ans = 1;
    FOR(i,1,k) {
        if(ans > x) return infi;
        ans = ans * (n-i+1)/i;
    }
    return ans;
}
```

Xây dựng bằng QHĐ: $\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}, \quad \text{for } 0 < k < n,$

31. Tính chất dãy Fibonacci:

A 2-dimensional system of linear difference equations that describes the Fibonacci sequence is

$$\begin{pmatrix} F_{k+2} \\ F_{k+1} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} F_{k+1} \\ F_k \end{pmatrix}$$

$$\vec{F}_{k+1} = \mathbf{A}\vec{F}_k .$$

$$F_n = \frac{1}{\sqrt{5}} \cdot \left(\frac{1+\sqrt{5}}{2}\right)^n - \frac{1}{\sqrt{5}} \cdot \left(\frac{1-\sqrt{5}}{2}\right)^n .$$

$$\gcd(F_m, F_n) = F_{\gcd(m,n)} .$$

Any three consecutive Fibonacci numbers are pairwise coprime, which means that, for every *n*.

Faster formular:

```
LL fib(int n) {
    if (n <= 2) return (n ? 1 : 0);
    if (n & 1) {
        return sqr(fib(n >> 1)) + sqr(fib((n >> 1) + 1));
    }
    LL x = fib(n >> 1);
    return (2 * fib((n >> 1) - 1) + x) * x;
}
```

32. Định lý Lucas

Let p be a prime number, and let r and c be written in p-ary notation

$$r = r_k \cdots r_2 r_1 r_0 = r_0 + r_1 p + r_2 p^2 + \cdots + r_k p^k \quad (0 \le r_i \le p)$$

$$c = c_k \cdots c_2 c_1 c_0 = c_0 + c_1 p + c_2 p^2 + \cdots + c_k p^k \quad (0 \le c_i \le p)$$

Then

$$\binom{r}{c} = \binom{r_0}{c_0}\binom{r_1}{c_1}\binom{r_2}{c_2} \dots \binom{r_k}{c_k} \ (mod\ p)$$

We take here the standard convention that the binomial coefficient (r,c)=0 if c > r.

33. Hình học cơ bản:

32.1. Các kiến thức cơ bản:

```
int cmp(double a, double b){
    if(abs(a-b) < eps) return 0;
    return (a > b ? 1 : -1);
}
struct point{
    double x;
    double y;
    point(double _x = 0, double _y = 0){
        x = _x; y = _y;
    }
    bool operator == (const point& that) const{
        return (cmp(x, that.x) == 0 && cmp(y, that.y) == 0);
    }
    bool operator < (const point& that) const{
        if(cmp(x, that.x) != 0) return cmp(x, that.x) < 0;
        return cmp(y, that.y) < 0;
    }
};
int ccw(point p0, point p1, point p2){ /// vector p0p1, p0p2
    double dx1 = p1.x - p0.x, dy1 = p1.y - p0.y;
    double dx2 = p2.x - p0.x, dy2 = p2.y - p0.y;
    return cmp(dx1*dy2 - dx2*dy1, 0);
    ///0 thẳng hàng      1 huong duong      -1 huong am
}
```

32.2: Hai đoạn thẳng thực sự cắt nhau

```
int isRealCut(point p0, point p1, point p2, point p3){
    return ccw(p0,p1,p2)*ccw(p0,p1,p3)<0 &&
ccw(p2,p3,p0)*ccw(p2,p3,p1)<0;
}
```

32.3: Vị trí của p0 so với đoạn thẳng p1p2

```
int segmentPos(point p0, point p1, point p2){
    if(p1 == p2) return -2;
    else if(ccw(p0,p1,p2) != 0) return -1; /// khong thang hang
    else if (p2 < p1) swap(p1,p2);
    if(p0 < p1) return 1; /// nam ngoai gan phia p1
    else if(p2 < p0) return 2; /// nam ngoai gan phia p2
    return 0; /// nam trong doan
}
```

32.4: Hai đoạn thẳng cắt nhau

```
int isSegmentCut(point p0, point p1, point p2, point p3){
    if(isRealCut(p0,p1,p2,p3)) return 1;
    if(segmentPos(p0,p2,p3) || segmentPos(p1,p2,p3)) return 1;
    if(segmentPos(p2,p0,p1) || segmentPos(p3,p0,p1)) return 1;
    return 0;
}
```

32.5: Phương trình đường thẳng

```
int getLine(point p0, point p1, double &a, double &b, double &c){
    if(p0 == p1) return 0;
    a = p1.y - p0.y;
    b = p0.x - p1.x;
    c = -(a*p0.x + b*p0.y);
    return 1;
}
```

32.6: Khoảng cách giữa hai điểm

```
double dist(point p0, point p1){
    double dx = p1.x - p0.x, dy = p1.y - p0.y;
    return sqrt(dx * dx + dy * dy);
}
```

32.7: Giao điểm của hai đoạn thẳng

```
int getIntersection(point p0, point p1, point p2, point p3, point &p4){
    double a0,b0,c0,a1,b1,c1;
    getLine(p0,p1,a0,b0,c0);
    getLine(p2,p3,a1,b1,c1);
    double d = a0*b1 - a1*b0;
    double dx = b0*c1 - b1*c0;
    double dy = -a0*b1 + a1*c0;
    if(cmp(d,0) == 0){
        if(cmp(dx,0) == 0 && cmp(dy,0) == 0) return -1; /// trung nhau
        else return 0; /// song song
    }
    p4.x = dx/d; p4.y = dy/d;
    return 1;
}
```

34. Đa giác:

34.1. Diện tích đa giác

```
double areaPolygon(point P[], int n){
    P[n+1] = P[1];
    double ans = 0;
    FOR(i,1,n) ans += P[i].x*P[i+1].y - P[i+1].x*P[i].y;
    return ans/2;
}
```

34.2.Kiểm tra một điểm nằm trong đa giác O(n)

```
int insidePolygon(point P[], int n, point p0){
    P[n+1] = P[1];
    /** truong hop da giac khong phai da giac loi
    FOR(i,1,n){
        if(segmentPos(p0, P[i], P[i+1]) == 0) return 1;
    }
    int dem = 0;
    point Z; Z.x = 1000000007; Z.y = 1000000008;
    /*while(1){
         int ok = 1;
        FOR(i,1,n) if(ccw(Z,P[i],P[i+1]) == 0) {
            ok = 0;
            Z.y++;
        }
        if(ok == 1) break;
    }
    FOR(i,1,n) if(isRealCut(p0,Z,P[i],P[i+1])) dem++;
    return (dem%2);
    */
    int x1 = 0, x2 = 0;
    FOR(i,1,n){
        if(ccw(P[i], P[i + 1], p0) == 0) return 0;
        else if(ccw(P[i], P[i + 1], p0) == -1) x1++;
        else x2++;
    }
    return (!x1 || !x2);
}
```

34.3.Thuật toán bao lồi Graham Scan

```
point O;
int degreeCmp( point p1, point p2){
    int d = ccw(O,p1,p2);
    if(d != 0) return (d > 0);
    return cmp(dist(O, p1), dist(O, p2) < 0);
}
///Tim bao loi cua tap p[] dpt O(n log n)
/// Sap xep lai tap diem p[] va tra ve so diem thuoc bao loi, cap nhat
lai n

void grahamScan(point P[], int &n){
    int j = 1;
    FOR(i,2,n){
    if(cmp(P[j].y, P[i].y) > 0 || (cmp(P[j].y, P[i].y) == 0 &&
cmp(P[j].x, P[i].x) < 0))
           j = i;
    }
    swap(P[1],P[j]);
    ///tim diem P[j] co hoanh do lon nhat trong nhung diem co tung do
nho nhat
    O = P[1];
    sort(P+2, P+n+1,degreeCmp);
    int k = 3;
    FOR(i,3,n){
        while(k > 2 && ccw(P[i], P[k-1], P[k-2]) >= 0)
            k--;
        swap(P[k++], P[i]);
    }
    n = k - 1;
}
```

34.4. Kiểm tra hai đa giác có điểm chung

```
int intersectionPolygon(point P[], int n1, point Q[], int n2){
    P[n1+1] = P[1]; Q[n2+1] = Q[1];
    /// check a point in a polygon
    FOR(i,1,n1) if(insidePolygon(Q,n2,P[i])) return 1;
    FOR(i,1,n2) if(insidePolygon(P,n1,Q[i])) return 1;
    /// check line intersect
    FOR(i,1,n1)
    FOR(j,1,n2)
     if(isRealCut(P[i],P[i+1],Q[j],Q[j+1]) == 1) return 1;
    /// in case same point
    FOR(i,1,n1)
    FOR(j,1,n2){
        if(isSegmentCut(P[i],P[i+1],Q[j],Q[j+1])) return 1;
    }
    return 0;
}
```

34.5.Tìm giao của hai đa giác lồi

```
void get2ConvexPolygon(point P[], int n1, point Q[], int n2, point
ans[], int &dem){
    dem = 1;
    P[n1+1] = P[1]; Q[n2+1] = Q[1];
    FOR(i,1,n1)
```

```
        if(insidePolygon(Q,n2,P[i])) ans[dem++] = P[i];
    FOR(i,1,n2)
        if(insidePolygon(P,n1,Q[i])) ans[dem++] = Q[i];
    FOR(i,1,n1){
        FOR(j,1,n2){
            point p3;
            getIntersection(P[i],P[i+1],Q[i],Q[i+1],p3);
            if(segmentPos(p3,P[i],P[i+1]) == 0)
                if(segmentPos(p3,Q[j],Q[j+1]) == 0)
                    ans[dem++] = p3;
        }
    }
    dem--;
    grahamScan(ans,dem);
}
```

35. Tam giác:

35.1. Diện tích tam giác

```
double areaTriangle(point A, point B, point C){
    double ans = abs(A.x*(B.y-C.y) + B.x*(C.y-A.y) + C.x*(A.y-B.y));
    return ans/2;
}
```

35.2. Tính góc BAC theo radian

```
double getAngle(point A, point B, point C){
    double a = dist(B,C);
    double b = dist(C,A);
    double c = dist(A,B);
    double agoc = (b*b + c*c - a*a) / (2*b*c);
    return acos(agoc);
}
```

35.3. Kiểm tra điểm p0 nằm trong tam giác ABC

```
int insideTriangle(point p0, point A, point B, point C){
    double S1 = areaTriangle(p0,A,B);
    double S2 = areaTriangle(p0,B,C);
    double S3 = areaTriangle(p0,C,A);
    double Sum = areaTriangle(A,B,C);
    if(cmp(Sum, S1+S2+S3) == 0) return 1;
    return 0;
}
```

35.4.Kiểm tra điểm p0 nằm trong góc tạo bởi tia AB, AC

```
int insideAngle(point p0, point A, point B, point C){
    if(p0 == A) return 1;
    if(ccw(p0,A,B) * ccw(p0,A,C) > 0) return 0;
    //return (getAngle(A,p0,B) + getAngle(A,p0,C) < PI+eps);
    if(cmp(getAngle(A,B,C), getAngle(A,p0,B) + getAngle(A,p0,C)) == 0)
return 1;
    return 0;
}
```

36. Một số thuật toán tối ưu:

36.1.Kiểm tra điểm p0 nằm trong đa giác O(log n)

```
int insideConvexPolygonLogN(point P[], int n, point p0){
    if(insideAngle(p0,P[1],P[2],P[n]) == 0) return 0;
    int low = 2, high = n;
    while(high - low > 1){
        int mid = (low+high) / 2;
        if(insideAngle(p0,P[1],P[low],P[mid]))
            high = mid;
        else low = mid;
    }
    return insideTriangle(p0,P[1],P[low],P[high]);
}
```

36.2. Bài toán cặp điểm gần nhất

```
Dữ liệu: chạy từ i = 0 → n-1
struct toCompare{
    bool operator() (point A, point B){
        if(A.y != B.y) return A.y < B.y;
        return A.x < B.x;
    }
};
/// chia de tri
double closestPairDist(point P[], int n){
    sort(P,P+n);
    set<point, toCompare> b;
    set<point, toCompare> ::iterator lowerIt, upperIt, it;
    int j = 0;
    double ans = infi;

    FO(i,0,n){
        while(P[i].x - P[j].x > ans){
            b.erase(P[j++]);
        }
        point c = P[i];
        c.y -= ans;
        lowerIt = b.lower_bound(c);
        c.y += 2* ans;
        upperIt = b.upper_bound(c);
        for(it = lowerIt; it != upperIt; it++){
            ans = min(ans, dist(P[i], *it));
        }
        b.insert(P[i]);
    }
     return ans;
}
```

36.3. Đường tròn nhỏ nhất phủ kín n điểm.

```
// Lay duong trung truc cua p1p2
int getCenterLine(point p0, point p1, double &a, double &b, double &c){
    if(p0 == p1) return 0;
    a = p1.x - p0.x;
    b = p1.y - p0.y;
    point p2;
    p2.x = (p0.x + p1.x) / 2;
    p2.y = (p0.y + p1.y) / 2;
    c = -(p2.x * a + p2.y * b);
    return 1;
}
// Lay duong tron di qua 3 diem
double getCircle(point p0, point p1, point p2, point &p3){
    double r;
    if(ccw(p0, p1, p2) == 0) return 0;
    double a0, b0, c0, a1, b1, c1;
```

```
    getCenterLine(p0, p1, a0, b0, c0);
    getCenterLine(p0, p2, a1, b1, c1);
    double d = a0 * b1 - a1 * b0;
    double dx = b0 * c1 - b1 * c0;
    double dy = -(c1 * a0 - c0 * a1);
    p3.x = dx / d;
    p3.y = dy / d;
    r = dist(p3, p0);
    return r;
}

void thuchien(){
    point OO, Q; /// tâm đường tròn ngoại tiếp
    vector<point> V;
    int ok = 0, vitri;
    double maxRadius, maxAngle, tmpR, tmpA;
    grahamScan(P,n); m = n;
    FOR(i,1,n) P[i].pos = i;

    V.pb(P[n]);
    FOR(i,1,n) V.pb(P[i]); V.pb(P[1]);

    while(1){
        tmpR = 0.0; tmpA = 0.0; maxRadius = 0.0; maxAngle = 0.0;
        if(V.size() == 4) {
            //cout << V[1].x << " " << V[2].y << " FINAL" << endl;
            maxRadius = dist(V[1],V[2]) / 2;
            break;
        }
        FOR(i,1,V.size()-2){
            tmpR = getCircle(V[i-1], V[i], V[i+1], Q);
            tmpA = getAngle(V[i], V[i-1], V[i+1]);
            if(cmp(tmpR, maxRadius) == 1){
                maxRadius = tmpR;
                  maxAngle = tmpA;
                vitri = i;
                OO = Q;
            }
            else if(cmp(tmpR, maxRadius) == 0){
                if(cmp(tmpA, maxAngle) > 0){
                    maxAngle = tmpA;
                       vitri = i;
                    OO = Q;
                }
            }
        }

        if(cmp(getAngle(V[vitri], V[vitri-1], V[vitri+1]), PI/2) <= 0)
break;
        else {
            if(vitri == 1) V[n+1] = V[2];
            else if(vitri == n) V[0] = V[n-1];
            V.erase(V.begin()+vitri);
            n--;
        }

    }
    //cout << OO.x << " " << OO.y << "  ";
    printf("%.6f\n",maxRadius);
```

```
}
```

37. Một số công thức hình học:

37.1.Phép quay góc alpha

```
X = x cos a - y sin a
Y = x sin a + y cos a
```

37.2. Tâm đường tròn nội tiếp

```
xI = (a.xA + b.xB + c.xC)/(a + b + c);
yI = (a.yA + b.yB + c.yC)/(a + b + c);
```

37.3. Khoảng cách 1 điểm tới đường thẳng.

$$\text{distance}(M,d) = \left|\overrightarrow{M'M}\right| = |k||\vec{\eta}| = \left|\frac{ax+by+c}{\sqrt{a^2+b^2}}\right|$$

37.4. Các hằng số:

```
E = 2.718281828
Thư viện Cmath
Sin, cos, tan, asin, acos, atan, atan2(góc giữa 2 véc tơ).
Log = ln
Log10 = lg
```

37.5.Các công thức trong tam giác:

$$S = \frac{|x_A(y_B-y_C)+x_B(y_C-y_A)+x_C(y_A-y_B)|}{2}$$

$$S=\sqrt{p(p-a)(p-b)(p-c)}$$

Độ dài đường trung tuyến: $m_a = \sqrt{\frac{2b^2+2c^2-a^2}{4}}$

Độ dài đường phân giác: $l_a = \frac{2bc\cos\frac{A}{2}}{b+c}$

Bán kính đương tròn nội tiếp: $r = \frac{2S}{a+b+c} = \frac{S}{p} = (p-a)\tan\frac{A}{2}$

Bán kính đường tròn ngoại tiếp: $R = \frac{a}{\sin A} = \frac{b}{\sin B} = \frac{c}{\sin C} = \frac{abc}{4S}$

Bán kính đường tròn bàng tiếp: $r = \frac{2S}{a+b+c} = \frac{S}{p} = (p-a)\tan\frac{A}{2}$

38. Ma trận:

1. Struct:

```
struct matrix{
    int size;
    LL a[21][21];
    friend ostream &operator<<(ostream &in, matrix m){
        FOR (i, 1, m.size) {
            FOR(j,1, m.size)
            cout << m.a[i][j] << " ";
            cout << endl;
        }
     }
};
```

2. Nhân ma trận:

```
    FOR(i,1,n) FOR(j,1,n){
        LL dem = 0;
        FOR(k,1,n) dem += A.a[i][k] * B.a[k][j];
    }
```

3. Lũy thừa ma trận:

```
matrix powMatrix(matrix A, int p){
    matrix ans; /// ans = I;
    ans.size = A.size;
    FOR(i,1,n) FOR(j,1,n) ans.a[i][j] = 0;
    FOR(i,1,n) ans.a[i][i] = 1;
    while(p > 0){
        if(p % 2 == 1) ans = mulMatrix(A,ans);
        A = mulMatrix(A,A);
        p /= 2;
    }
    return ans;
}
```

4. Định thức:

```
LL lamtronso(double x){
    LL ans = (floor)(x);
    if(abs(ans-x) < eps) return ans;
    else return (ceil)(x);
}

LL determinant(matrix A,int u, int v){
    double a[22][22];
    int demi = 1, demj = 1 ;
    FOR(i,1,n){
        if(i != u){
            FOR(j,1,n){
                if(j != v){
                    a[demi][demj] = A.a[i][j];
                    demj++;
                }
            }
            demi++; demj = 1;
        }
    }

    int ok = 1,h;
    double det = 1;
    FOR(i,1,n-1){
        h = i;
        FOR(k,i+1,n)
            if(abs(a[k][i]) > abs(a[h][i])) h = k;
        if(a[h][i] == 0) return 0;
        if(h != i) {
            FOR(j,1,n) swap(a[i][j],a[h][j]);
            ok = -ok;
        }
        FOR(k,i+1,n){
            double p = 1.0*a[k][i] / (1.0*a[i][i]);
             FOR(j,i+1,n) a[k][j] -= p*a[i][j];
        }
    }
    ///evaluate det
     FOR(i,1,demi-1) det *= a[i][i];
    LL ans = lamtronso(det);
    return ans*ok;
}
```

6. Ma trận nghịch đảo:

```
matrix inverseMatrix(matrix A){    // using Gauss elimation
    matrix ans;
    ans.size = A.size;
    FOR(i,1,n)
    FOR(j,1,n) {
        ans.a[i][j] = 0;
    }
    double aa[21][50];
    FOR(i,1,n) FOR(j,1,n) aa[i][j] = A.a[i][j];
    /// them ma tran don vi vao sau A
    FOR(i,1,n) FOR(j,1,n) aa[i][n+j] = 0;
    FOR(i,1,n) aa[i][n+i] = 1;

    FOR(i,1,n){
        int h = i;
        FOR(k,i+1,n) if(fabs(aa[k][i]) > fabs(aa[h][i])) h = k;
        if(aa[h][i] == 0) cout << "Ma tran suy bien";
        if(h != i){ /// doi hang i va hang h vi a[h][i] > a[i][i]
            FOR(j,1,2*n) swap(aa[i][j],aa[h][j]);
        }
        /// chuyen he so a[i][i] = 1
        double tmp = aa[i][i];
        FOR(j,i,2*n) aa[i][j] /= tmp; /// chia phan con lai cua hang cho
a[i][i]
        /// update lai cac hang
        FOR(k,1,n){
            if(k == i) continue;
            double p = aa[k][i];
            FOR(j,i+1,2*n) aa[k][j] -= p*aa[i][j];
        }
    }
    /// ma tran nghiem
    FOR(i,1,n)
    FOR(j,1,n) ans.a[i][j] = aa[i][j+n];
    return ans;
}
```

7. Giải hệ phương trình Ax = B:

```
void process(){
    /// ma tran bo sung A~
    FOR(i,1,n) a[i][n+1] = b[i];
    double aa[100][100];
       FOR(i,1,n){
        int h = i;
        FOR(k,i+1,n) if(fabs(a[k][i]) > fabs(a[h][i])) h = k;
        if(a[h][i] == 0) cout << "Ma tran suy bien";
        if(h != i){ /// doi hang i va hang h vi a[h][i] > a[i][i]
            FOR(j,1,n+1) swap(a[i][j],a[h][j]);
        }
        /// chuyen he so a[i][i] = 1
        double tmp = a[i][i];
         FOR(j,i,n+1) a[i][j] /= tmp; /// chia phan con lai cua hang cho
a[i][i]
        /// update lai cac hang
        FOR(k,1,n){
            if(k == i) continue;
            double p = a[k][i];
            FOR(j,i+1,n+1) a[k][j] -= p*a[i][j];
        }
    }
    /// bieu dien cac phan tu con lai cua ma tran = 0
    FOR(i,1,n) FOR(j,1,n) if(i != j) a[i][j] = 0;
    /// vector nghiem
    FOR(i,1,n) x[i] = a[i][n+1];
}
// 2 3 5 7 11 13 17 19 23 29 31 37
// 41 43 47 53 59 61 67 71 73 79 83 89
// 97 101 103 107 109 113 127 131 137 139 149 151
// 157 163 167 173 179 181 191 193 197 199 211 223
// 227 229 233 239 241 251 257 263 269 271 277 281
// 283 293 307 311 313 317 331 337 347 349 353 359
// 367 373 379 383 389 397 401 409 419 421 431 433
// 439 443 449 457 461 463 467 479 487 491 499 503
// 509 521 523 541 547 557 563 569 571 577 587 593
// 599 601 607 613 617 619 631 641 643 647 653 659
// 661 673 677 683 691 701 709 719 727 733 739 743
// 751 757 761 769 773 787 797 809 811 821 823 827
// 829 839 853 857 859 863 877 881 883 887 907 911
// 919 929 937 941 947 953 967 971 977 983 991 997
// Other primes:
// The largest prime smaller than 10 is 7.
// The largest prime smaller than 100 is 97.
// The largest prime smaller than 1000 is 997.
// The largest prime smaller than 10000 is 9973.
// The largest prime smaller than 100000 is 99991.
// The largest prime smaller than 1000000 is 999983.
// The largest prime smaller than 10000000 is 9999991.
// The largest prime smaller than 100000000 is 99999989.
// The largest prime smaller than 1000000000 is 999999937.
// The largest prime smaller than 10000000000 is 9999999967.
// The largest prime smaller than 100000000000 is 99999999977.
// The largest prime smaller than 1000000000000 is 999999999989.
// The largest prime smaller than 10000000000000 is 9999999999971.
// The largest prime smaller than 100000000000000 is 99999999999973.
// The largest prime smaller than 1000000000000000 is 999999999999989.
// The largest prime smaller than 10000000000000000 is 9999999999999937.
// The largest prime smaller than 100000000000000000 is 99999999999999997.
// The largest prime smaller than 1000000000000000000 is 999999999999999989.
```

Tọa độ cầu của một điểm có thể tính được từ tọa độ Cartesian bằng công thức sau

$$r = \sqrt{x^2 + y^2 + z^2}$$
$$\varphi = \operatorname{atan2}(y, x)$$
$$\theta = \arccos\left(\frac{z}{\sqrt{x^2 + y^2 + z^2}}\right)$$

trong đó atan2(y,x) là một biến thể của hàm arctan trả ra góc tính từ trục *x* của vectơ (*x*,*y*) trong toàn miền $(-\pi, \pi]$. (Ta không thể dùng hàm arctan thông thường, $\varphi = \arctan(y/x)$, vì nó sẽ trả ra cùng một góc cho (*x*,*y*) và (−*x*,−*y*)).

Ngược lại tọa độ Cartesian có thể tính được từ tọa độ cầu bằng công thức:

$$x = r\,\cos\varphi\,\sin\theta$$
$$y = r\,\sin\varphi\,\sin\theta$$
$$z = r\,\cos\theta$$