MỘT SỐ BÀI TẬP ỨNG DỤNG CỦA TÌM KIẾM ƯU TIÊN THEO CHIỀU SÂU (DFS) TRÊN ĐỒ THỊ

Depth first search (DFS) là một trong những thuật toán tìm kiếm kinh điển trên đồ thị. Thuật toán không chỉ đơn thuần là duyệt tìm kiếm ưu tiên theo chiều sâu, mà ứng dụng của nó cũng rất sâu sắc trong việc tìm cây khung, tìm chu trình, tìm thành phần liên thông mạnh, tìm cầu và khớp, sắp xếp topo.

Trong chuyên đề này, tôi không nhắc lại lý thuyết của DFS (vì đã có rất nhiều tài liệu viết về vấn đề này rồi), mà ở đây tôi chỉ chú trọng vào những bài tập ứng dụng của DFS nhằm giúp người đọc có cái nhìn toàn diện về thuật toán này.

Bài 1. Đường 1 chiều

Một hệ thống giao thông gồm có N nút giao thông đánh số từ 1 đến N và M đường hai chiều nối một số cặp nút, không có hai đường nối cùng một cặp nút. Hệ thống đảm bảo đi lại giữa hai hút bất kì. Để đảm bảo an toàn, người ta quyết định rằng các đường hai chiều trước đây nay sẽ thành một chiều, và vấn đề ở chỗ chọn chiều cho mỗi đường như thế nào.

Hãy tìm cách định hướng các cạnh sao cho hệ thống vẫn đảm bảo đi lại giữa hai cặp nút bất kì.

INPUT: ONEWAY.INP

- Dòng đầu ghi hai số nguyên dương N, M (1 <= N <= 50000 , 1 <= M <= 100000).
- M dòng tiếp theo, mỗi dòng thể hiện một đường hai chiều gồm u, v là chỉ số hai nút mà nó nối tới .

OUTPUT: ONEWAY.OUT

- Dòng đầu ghi 1/0 tương ứng với có tìm được phương án thoả mãn hay không .
- Nếu có, M dòng tiếp theo mỗi dòng thể hiện sự định hướng một cạnh bao gồm hai số u, v với ý nghĩa định hướng cạnh (u,v) thành đường một chiều từ u đến v.

Ví dụ

Test 1		Test 2	
ONEWAY.INP	ONEWAY.OUT	ONEWAY.INP	ONEWAY.OUT
4 5 1 2 2 3 2 4 3 4 1 4	1 1 2 2 3 2 4 3 4 4 1	4 4 1 2 2 3 3 4 3 1	0

* Hướng dẫn giải:

- Sử dụng DFS để định chiều các cạnh của đồ thị. Cạnh nào thuộc cây DFS sẽ được định chiều từ gốc xuống lá (tạm gọi là hướng xuôi), những cạnh không thuộc cây DFS sẽ được định chiều theo hướng ngược lại.

- Sau khi định chiều xong ta được một đồ thị 1 chiều mới, kiểm tra đồ thị mới có liên thông không (sử dụng thuật toán Tajan). Nếu có thì trả lời có phương án, ngược lại thì không có phương án.

* Chương trình mẫu

```
#include <bits/stdc++.h>
#define pii pair<int, int>
#define fi first
#define se second
#define mp make_pair
using namespace std;
const int nmax = 50001;
pii dsc[nmax*2];
vector <int> adj[nmax], adj1[nmax];
int dd[nmax], d[nmax], cha[nmax], num[nmax], low[nmax];
int n, m, id, dem=0;
void doc()
{
    cin >> n >> m;
    for (int i=1; i<=m; i++)
    {
        int u,v;
        cin >> u >> v;
        dsc[i] = mp(u,v);
        adj[u].push_back(v);
        adj[v].push_back(u);
    }
}

void DFS(int u)
{
    dd[u] = 1;
    for (int i=0; i<adj[u].size(); i++)
    {
        int v = adj[u][i];
        if (dd[v] == 0)
        {
            d[v] = d[u] + 1;
            cha[v] = u;
            DFS(v);
        }
    }
}

void dinh_chieu()
{
    for (int i=1; i<=m; i++)
    {
        int u = dsc[i].fi;
        int v = dsc[i].se;
        if (cha[v] == u) adj1[u].push_back(v);
```

```
        else if (cha[u] == v) adj1[v].push_back(u);
             else if (d[u] < d[v]) adj1[v].push_back(u);
                  else adj1[u].push_back(v);
    }
}

void DFS1(int u)
{
    dd[u] = 1;
    id++;
    num[u] = id;
    low[u] = id;
    for (int i=0; i<adj1[u].size();i++)
    {
        int v=adj1[u][i];
        if (dd[v] == 0)
        {
            DFS1(v);
            low[u] = min(low[u],low[v]);
        }
        else if (dd[v] == 1) low[u] = min(num[v],low[u]);
    }
    if (low[u] == num[u]) dem++;
}
void tajan()
{
    memset(dd,0,sizeof(dd));
    for (int i=1; i<=n; i++)
    if (dd[i] == 0) DFS1(i);
}

void ghi()
{
    if (dem != 1) cout << 0;
    else
    {
        cout << 1 << endl;
        for (int u=1; u<=n; u++)
        for (int i=0; i<adj1[u].size(); i++)
        {
            int v = adj1[u][i];
            cout << u << " " << v << endl;
        }
    }
}
int main()
{
    freopen("oneway.inp","r",stdin);
    freopen("oneway.out","w",stdout);
    doc();
    DFS(1);
    dinh_chieu();
    tajan();
```

```
    ghi();
    return 0;
}
```

*** Test:**

- Các thầy cô có thể download bằng link.

http://www.mediafire.com/download/78z32dhfh3znyak/oneWay.rar

Bài 2. Bảo vệ

Một thành phố có N địa điểm chiến lược và M con đường một chiều giữa các địa điểm. Là thị trưởng của thành phố, bạn sẽ phải bảo vệ an toàn cho N địa điểm này.

Để có thể bảo vệ cho các địa điểm, bạn phải xây dựng các đồn cảnh sát tại một vài địa điểm. Đồn cảnh sát tại địa điểm i có thể bảo vệ cho địa điểm j nếu i = j hoặc cảnh sát có thể đi tuần tới địa điểm j từ i và có thể quay trở lại đồn tại địa điểm i.

Để có thể xây dựng được các đồn cảnh sát cần phải mất chi phí, do địa hình mỗi địa điểm là khác nhau nên chi phí xây dựng đồn cũng có thể khác nhau.

Bạn phải xác định số tiền nhỏ nhất để xây dựng các đồn cảnh sát để có thể bảo vệ được tất cả N địa điểm, hơn nữa bạn phải đưa ra có bao nhiêu cách xây dựng để đảm bảo chi phí nhỏ nhất đó.

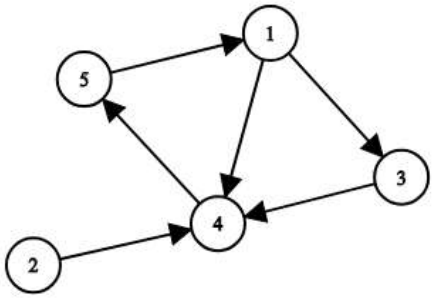
INPUT: SECURITY.INP

- Dòng 1 chứa số nguyên dương N ($1 \le N \le 10^5$)
- Dòng 2 chứa N số nguyên, trong đó số nguyên thứ i là chi phí để xây dựng đồn cảnh sát tại địa điểm i (chi phí $\le 10^9$).
- Dòng 3 chứa số nguyên M ($0 \le M \le 3*10^5$)
- M dòng tiếp theo, mỗi dòng chứa hai số nguyên dương u và v ($1 \le u, v \le n; u \ne v$) biểu diễn một con đường một chiều nối từ địa điểm u tới v. Không có nhiều hơn 1 con đường nối giữa 2 địa điểm.

OUTPUT: SECURITY.OUT

- Một dòng duy nhất chứa hai số, số thứ nhất là chi phí nhỏ nhất để xây dựng các đồn cảnh sát, số thứ hai là số phương án xây dựng (mod (10^9+7)).

Ví dụ:

SECURITY.INP	SECURITY.OUT	
5 2 8 0 6 0 6 1 4 1 3 2 4 3 4 4 5 5 1	8 2	

* Hướng dẫn thuật toán:

- Sử dụng thuật toán Tajan để tìm các thành phần liên thông mạnh, tại mỗi thành phần liên thông mạnh sẽ xây dựng một đồn cảnh sát có chi phí xây dựng là nhỏ nhất, và đếm số lượng các đồn có cùng chi phí nhỏ nhất đó.

- Tổng số tiền xây dựng là tổng số tiền xây dựng các đồn có chi phí nhỏ nhất ở mỗi thành phần liên thông.

- Số cách xây dựng là tích của số lượng các đồn cùng chi phí nhỏ nhất ở mỗi thành phần liên thông.

* Chương trình mẫu:

```
uses    math;
const   fi='security.INP';
        fo='security.OUT';
        nmax = 100000;
        mmax = 300000;
        vc = 1000000010;
        base = 1000000007;
type    cung = record
                 u,v:longint;
        end;
var     dsc:array[1..mmax] of cung;
        dsk:array[1..mmax] of longint;
        head,free,low,number,cost,stack,tsmin:array[1..nmax+1] of longint;
        N,M,top,id,costmin,count,top1:longint;
        kq,ts:int64;
        f:text;
procedure chuanbi;
begin
        top := 0; ts:=1;
        fillchar(dsc,sizeof(dsc),0);
        fillchar(dsk,sizeof(dsk),0);
        fillchar(head,sizeof(head),0);
```

```
        fillchar(free,sizeof(free),0);
        fillchar(low,sizeof(low),0);
        fillchar(number,sizeof(number),0);
end;
procedure doc;
var     i:longint;
begin
        assign(f,fi); reset(f);
        readln(f,N);
        for i:=1 to N do read(f,cost[i]);
        readln(f,M);
        for i:=1 to M do
        begin
                readln(f,dsc[i].u,dsc[i].v);
                inc(head[dsc[i].u]);
        end;
        for i:=2 to N do head[i] := head[i] + head[i-1];
        head[N+1] := head[N];
        for i:=1 to M do
        begin
                dsk[head[dsc[i].u]] := dsc[i].v;
                dec(head[dsc[i].u]);
        end;
        close(f);
end;
procedure DFS(u:longint);
var     i,v:longint;
begin
        inc(top);
        stack[top] := u;
        free[u] := 1;
        inc(id);
        number[u] := id;
        low[u]:=id;
        for i:=head[u]+1 to head[u+1] do
        begin
                v := dsk[i];
                case free[v] of
                0:      Begin
                                DFS(v);
                                low[u]:=min(low[v],low[u]);
                        end;
                1:      begin
                                low[u]:=min(low[u],number[v]);
                        end;
                end;
        end;
        if low[u] = number[u] then
        begin
                costmin := vc;
                inc(count);
                top1 := top;
                repeat
```

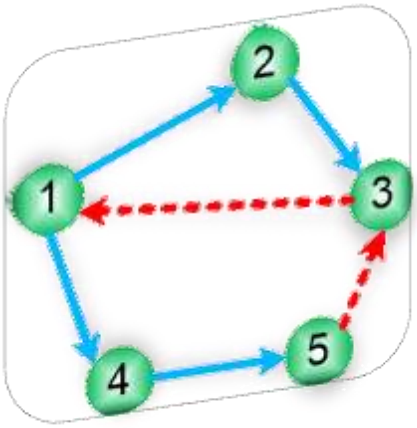
```
                        v := stack[top1];
                        if costmin > cost[v] then costmin := cost[v];
                        dec(top1);
                        free[v] := 2;
                until v = u;
                kq := kq + int64(costmin);
                repeat
                        v := stack[top];
                        if costmin = cost[v] then inc(tsmin[count]);
                        dec(top);
                until v = u;
                ts := (ts*int64(tsmin[count])) mod int64(base);
        end;
end;
procedure xuly;
var     i:longint;
begin
        for i:=1 to N do
        if free[i]=0 then DFS(i);
end;
procedure ghi;
begin
        assign(f,fo); rewrite(f);
        writeln(f,kq,' ',ts);
        close(f);
end;
BEGIN
        chuanbi;
        doc;
        xuly;
        ghi;
END.
```

*** Test:**

- Các thầy cô có thể download bằng link.

http://www.mediafire.com/download/se5imtp18887kke/security.rar

Bài 3. Tàu cao tốc.



Có ***n*** điểm tập trung dân cư đông đúc. Các điểm này được đánh số từ 1 đến ***n*** ($1 \le n \le 10^4$). Mạng lưới giao thông công cộng là ***m*** đường xe lửa cao tốc một ray, mỗi đường nối một cặp điểm dân cư và chạy hai chiều ($0 \le m \le 10^5$), và mọi cặp điểm đều có thể đi đến được với nhau. Để tránh sự va chạm giữa các con tàu cao tốc khi chúng có thể đi ngược chiều trên cùng một đường, chính quyền thành phố quyết định sửa lại các con đường đó thành một chiều. Tuy nhiên, sau khi thay đổi thì lại có một vấn đề bất cập sảy ra, đó là: tồn tại các cặp điểm tập trung dân cư không thể đi đến được nhau.

Chính vì vậy, chính quyền lại thêm một quyết định nữa, đó là sẽ xây dựng thêm một số ít nhất các tuyến đường mới để đảm bảo từ một điểm bất kỳ có thể đi tới điểm bất kỳ khác bằng tàu cao tốc.

Ví dụ, với ***n*** = 5 và hiện có 4 đường: $1 \to 2$, $2 \to 3$, $1 \to 4$ và $4 \to 5$. Để đảm bảo yêu cầu đã nêu, người ta cần xây dựng ít nhất 2 đường mới, chẳng hạn $5 \to 3$ và $3 \to 1$.

Yêu cầu: Cho ***n***, ***m*** và các cặp (***a***, ***b***) mô tả mạng giao thông sau khi đã sửa thành đường 1 chiều. Mỗi cặp (***a***, ***b***) xác định tồn tại đường tàu $a \to b$. Hãy xác định số lượng tối thiểu các đường cần xây dựng thêm.

Dữ liệu: Vào từ file văn bản MONORAIL.INP:

- Dòng đầu tiên chứa 2 số nguyên ***n*** và ***m***,
- Mỗi dòng trong ***m*** dòng tiếp theo chứa 2 số nguyên ***a*** và ***b***.

Kết quả: Đưa ra file văn bản MONORAIL.OUT một số nguyên – số đường mới.

Ví dụ:

MONORAIL.INP
5 4 **1 2** **2 3** **1 4** **4 5**

MONORAIL.OUT
2

*** Hướng dẫn thuật toán:**

- Sử dụng Tajan để tìm các thành phần liên thông mạnh

- Mỗi thành phần liên thông mạnh thuộc 1 trong 3 loại sau:

+ Loại 1: Thành phần liên thông chỉ có cung đi ra mà không có cung đi vào (ví dụ đỉnh 1 trong hình trên)

+ Loại 2: Thành phần liên thông có cả cung đi ra và cả cung đi vào (ví dụ đỉnh 2 và 4 trong hình trên)

+ Loại 3: Thành phần liên thông chỉ có cung đi vào mà không có cung đi ra (ví dụ đỉnh 3, 5 trong hình trên)

- Để tạo thành 1 vùng liên thông mạnh thì cần phải bổ sung cung nối từ thành phần liên thông loại 3 sang thành phần liên thông loại 1.

Từ đó hình thành cách giải như sau:

- Đếm số thành phần liên thông loại 1 (gọi là d1) và loại 3 (gọi là d3)

- Kết quả của bài toán chính là max(d1,d3).

*** Chương trình mẫu:**

```
program MONORAIL;
const   fi='MONORAIL.INP';
        fo='MONORAIL.OUT';
        Nmax = 10000;
        mmax = 100000;
type    canh = record
            x,y:longint;
        end;
Var     dsc:array[1..mmax] of canh;
        Ke:array[1..2*mmax] of longint;
        Head,chot,stack,Low,Num:array[1..nmax+1] of longint;
        free,cs:array[1..nmax] of integer;
        N,M,Top,dem:longint;
        f:text;
procedure doc;
var     i:longint;
Begin
        assign(f,fi); reset(f);
        readln(f,N,M);
        for i:=1 to M do
        Begin
                readln(f,dsc[i].x,dsc[i].y);
                inc(Head[dsc[i].x]);
```

```
        end;
        for i:=2 to N do Head[i] := Head[i-1] + Head[i];
        Head[N+1] := Head[N];
        for i:=1 to M do
        Begin
                ke[Head[dsc[i].x]] := dsc[i].y;
                dec(Head[dsc[i].x]);
        end;
        close(f);
end;
procedure chuanbi;
Begin
        fillchar(free,sizeof(free),0);
        fillchar(chot,sizeof(chot),0);
end;
function min(x,y:longint):longint;
begin
        if x > y then min := y
        else min := x;
end;
procedure DFS(u:longint);
var     i,v:longint;
Begin
        inc(dem);
        Num[u] := dem;
        Low[u] := dem;
        free[u] := 1;
        inc(top);
        Stack[top] := u;
        for i := Head[u]+1 to Head[u+1] do
        Begin
                v := ke[i];
                if free[v] = 0 then
                Begin
                        DFS(v);
                        Low[u] := Min(Low[u],Low[v]);
                end
                else  if free[v] = 1 then Low[u] := min(Low[u],Num[v]);
        end;
        if Low[u] = Num[u] then
        Begin
                repeat
                        v := Stack[top];
                        dec(top);
                        chot[v] := u;
                        free[v] := 2;
                until v = u;
        end;
end;
procedure xuly;
var     i,u,v:longint;
Begin
        chuanbi;
```

```
        for i :=1 to N do
        if free[i] = 0 then
        Begin
                DFS(i);
        end;
        for u := 1 to N do
        for i := Head[u] + 1 to Head[u+1] do
        Begin
                v := ke[i];
                if chot[u] <> chot[v] then
                Begin
                        if cs[chot[u]] = 0 then cs[chot[u]] := 1
                        else if cs[chot[u]] = 3 then cs[chot[u]] := 2;
                        if cs[chot[v]] = 0 then cs[chot[v]] := 3
                        else if cs[chot[v]] = 1 then cs[chot[v]] := 2;
                end;
        end;
end;
procedure ghi;
var     d1,d2,i:longint;
begin
        d1 := 0; d2 := 0;
        for i:=1 to N do
        if cs[i] = 1 then inc(d1)
        else if cs[i] = 3 then inc(d2);
        assign(f,fo); rewrite(f);
        if d1 > d2 then write(f,d1)
        else write(f,d2);
        close(f);
end;
BEGIN
        doc;
        xuly;
        ghi;
END.
```

*** Test:**

- Các thầy cô có thể download bằng link.

http://www.mediafire.com/download/2g2f6l88cv2fx1n/MONORAIL.rar

Bài 4. Dựng đồ thị

Người ta khởi tạo một đồ thị có hướng gồm 10^9 đỉnh các đỉnh được đánh số từ 1 tới 10^9, ban đầu đồ thị không có cung nào. Người ta thêm lần lượt các cung vào đồ thị bởi câu lệnh dạng **add(u, v)**, nghĩa là thêm một cung nối từ đỉnh u tới đỉnh v trên đồ thị.

Cho trước 2 đỉnh s và t, hãy cho biết số thứ tự của lệnh **add** đầu tiên mà sau thời điểm thực hiện lệnh **add** đó đỉnh s có thể đi tới được đỉnh t theo một số cung của đồ thị.

Input

- Dòng đầu chứa 3 số nguyên $m \leq 10^5$, s, t ($s \neq t$)
- M dòng tiếp theo, mỗi dòng chứa 2 số nguyên u, v thế hiện lệnh **add(u, v).**

Output

- Ghi trên 1 dòng là kết quả của bài toán. Nếu không tồn tại kết quả thỏa mãn, in ra một số 0.

Ví dụ

GRAPH.INP	GRAPH.OUT
5 1 5 1 2 3 5 3 1 2 3 2 4	4

* Hướng dẫn thuật toán

- Đề bài cho tối đa 10^9 đỉnh nhưng số lượng cạnh chỉ tốt đa là 10^5, nên thực tế số đỉnh chỉ cần dùng đến 10^5 đỉnh mà thôi. Từ đó trước hết ta sẽ rời rạc hóa các đỉnh <=10^9 về các đỉnh <= 10^5.

- Tiếp theo, sử dụng thuật toán tìm kiếm nhị phân kết hợp DFS để giải quyết bài toán. Mỗi lần chặt nhị phân số cung để tìm số lần thêm cung tối thiểu, ta sử dụng thuật toán DFS để tìm đường đi từ s đến t.

* Chương trình mẫu:

```
#include <bits/stdc++.h>
#define pii pair<int,int>
#define fi first
#define se second
#define mp make_pair
using namespace std;
const int nmax = 1e5+5;
vector <int> adj[nmax];
map <int,int> M;
int dd[nmax];
```

```
int n, s, t, res=0;
pii dsc[nmax];
void doc()
{
    map <ll, ll> :: iterator it;
    int dem=0;
    cin >> n >> s >> t;
    for (int i=1; i<=n; i++)
    {
        int u, v;
        cin >> u >> v;
        it = M.find(u);
        if (it == M.end())
        {
            dem++;
            M.insert(mp(u,dem));
            dsc[i].fi = dem;
        }
        else dsc[i].fi = it->se;
        it = M.find(v);
        if (it == M.end())
        {
            dem++;
            M.insert(mp(v,dem));
            dsc[i].se = dem;
        }
        else dsc[i].se = it->se;
    }
    it = M.find(s);
    s = it->se;
    it = M.find(t);
    t = it->se;
}
void DFS(int u)
{
    dd[u] = 1;
    for (int i=0; i<adj[u].size();i++)
    {
        int v = adj[u][i];
        if (dd[v] == 0) DFS(v);
    }
}
bool check(int k)
{
    memset(dd,0,sizeof(dd));
    for (int i=1; i<=n; i++) adj[i].clear();
    for (int i=1; i<=k; i++)
    {
        int u = dsc[i].fi;
        int v = dsc[i].se;
        adj[u].push_back(v);
    }
    DFS(s);
```

```
    return (dd[t] == 1);
}
void xuly()
{
    int d=1,c=n;
    while (d <= c)
    {
        int g = (d + c)/2;
        if (check(g))
        {
            res = g;
            c = g - 1;
        }
        else d = g + 1;
    }
    cout << res;
}
int main()
{
    freopen("GRAPH.INP","r",stdin);
    freopen("GRAPH.OUT","w",stdout);
    doc();
    xuly();
    return 0;
}
```

*** Test:**

- Các thầy cô có thể download bằng link.

http://www.mediafire.com/download/olbelxm6ojmymfa/GRAPH.rar

Bài 5. Liên thông – đề thi thử QG 2012 – Hà Nam

Cho một đồ thị vô hướng gồm n đỉnh đánh số từ 1 tới n và m cạnh đánh số từ 1 tới m. Cạnh thứ i nối giữa hai đỉnh u_i, v_i. Nếu ta xoá đi một đỉnh nào đó của đồ thị, số thành phần liên thông của đồ thị có thể tăng lên. Nhiệm vụ của bạn là với mỗi đỉnh, hãy tính xem nếu ta xoá đỉnh đó đi thì đồ thị mới nhận được có bao nhiêu thành phần liên thông.

Dữ liệu: Vào từ file văn bản GRAPH_.INP

- Dòng đầu chứa hai số nguyên dương n, m ($n \leq 20000; m \leq 50000$)
- m dòng sau, dòng thứ i chứa hai số nguyên dương u_i, v_i.

Kết quả: Ghi ra file văn bản GRAPH_.OUT

- n dòng, dòng thứ j cho biết số thành phần liên thông của đồ thị nếu ta xóa đi đỉnh j.

Ví dụ

GRAPH_.INP	GRAPH_.OUT
4 3 1 2 2 3 2 4	1 3 1 1

Chú ý: Ít nhất 60% số điểm ứng với các test có $n \leq 1000; m \leq 2000$

* Hướng dẫn thuật toán

- Đây là bài toán điển hình về tìm khớp trên đồ thị. Nếu đỉnh u không phải là khớp thì số lượng thành phần liên thông không thay đổi, nếu đỉnh u là khớp thì số lượng thành phần liên thông sẽ được tăng lên.

- Vấn đề ở đây là làm thế nào đếm được số lượng thành phần liên thông sau khi đã loại bỏ một khớp u ra khỏi đồ thị. Có thể giải quyết vấn đề này như sau: trong quá trình DFS sử dụng thêm một mảng để lưu số lượng đỉnh con của đỉnh u là slcon[u]. Khi đó nếu số lượng thành phần liên thông ban đầu của đồ thì là k thì tiếp theo sẽ có hai khả năng như sau:

+ Khả năng 1: u là khớp nhưng không phải đỉnh gốc của DFS thì số lượng thành phần liên thông sau khi xóa đỉnh khớp u là: k + slcon[u].

+ Khả năng 2: u là khớp nhưng lại là đỉnh gốc của DFS thì số lượng thành phần liên thông sau khi xóa đỉnh khớp u là: k + slcon[u] – 1.

* Chương trình mẫu:

```
#include <cstdio>
#include <vector>
using namespace std;
int n, m, dem, sotp;
vector<int> ke[20020];
int fa[20020], low[20020], num[20020], sc[20020], add[20020];
void dfs(int i)
{
      ++dem;
      num[i] = dem;
      low[i] = num[i];
      for(int k=0;k<ke[i].size();++k)
```

```
	{
		int j = ke[i][k];
		if(fa[j] == -1)
		{
			++sc[i];
			fa[j] = i;
			dfs(j);
			low[i] = min(low[i], low[j]);
		}
		else if(j != fa[i])
			low[i] = min(low[i], num[j]);
	}
}
int main()
{
	scanf("%d%d", &n, &m);
	for(int i=0;i<m;++i)
	{
		int u, v;
		scanf("%d%d", &u, &v);
		ke[u].push_back(v);
		ke[v].push_back(u);
	}
	memset( fa, -1, sizeof(fa));
	for(int i=1;i<=n;++i)
	if(fa[i]==-1)
	{
		++sotp;
		fa[i] = 0;
		dfs(i);
	}

	for(int i=1;i<=n;++i)
	if(fa[i] != 0)
	{
		int fi = fa[i];
		if(low[i] >= num[fi]) ++add[fi];
	}

	for(int i=1;i<=n;++i)
	{
		if(fa[i] == 0)
			printf("%d\n", sotp + sc[i] - 1);
		else
			printf("%d\n", sotp + add[i]);
	}
	return 0;
}
```

*** Test:**

- Các thầy cô có thể download bằng link.

http://www.mediafire.com/download/15cyq49nmkftobu/Graph_.rar

Bài 6. Cảnh sát (spoj)

Để giúp nắm bắt bọn tội phạm trên chạy, cảnh sát được giới thiệu một hệ thống máy vi tính. Khu vực được quản lý bởi cảnh sát thành phố có chứa N thành phố liên thông bằng E tuyến đường hai chiều kết nối chúng. Các thành phố được dán nhãn từ 1 đến N.

Để xử lý trong trường hợp khẩn cấp, cảnh sát nhờ bạn xây dựng một chương trình phần mềm trả lời hai câu hỏi sau truy vấn:

1. Xem xét việc hai thành phố A và B, và một đường kết nối giữa thành phố G1 và G2. Bọn tội phạm có thể đi từ thành phố A đến thành phố B, nếu con đường đó bị chặn và bọn tội phạm không thể sử dụng nó?
2. Xem xét ba thành phố A, B và C. bọn tội phạm có thể đi được từ thành phố A đến thành phố B nếu toàn bộ thành phố C là được phong tỏa và bọn tội phạm có thể không được nhập vào thành phố này?

Viết một chương trình để thực hiện các mô tả hệ thống.

Dữ liệu vào từ tệp: POLICIJA.INP

- Dòng đầu tiên chứa hai số nguyên N và E ($2 \le N \le 100\ 000, 1 \le E \le 500\ 000$), là số lượng các thành phố và tuyến đường.
- E dòng tiếp theo mỗi dòng ghi hai số thể hiện một tuyến đường
- Dòng tiếp theo ghi số K là số câu hỏi cần truy vấn
- K dòng tiếp theo mỗi dòng ghi một lại câu hỏi:
  - ❖ 1 A B G1 G2: cho loại câu hỏi thứ nhất ($1 \le A, B, G1, G2 \le N$), A,B, G1,G2 khác nhau
  - ❖ 2 A B C: cho loại câu hỏi thứ hai ($1 \le A, B, C \le N$), A,B,C khác nhau.

Dữ liệu ra vào tệp: POLICIJA.OUT

- Tương ứng với mỗi câu truy vấn có một câu trả lời ghi trên một dòng của tệp kết quả. Câu trả lời cho một truy vấn có thể là "no" hay "yes".

Ví dụ

POLICIJA.INP	POLICIJA.OUT

13 15 1 2 2 3 3 5 2 4 4 6 2 6 1 4 1 7 7 8 7 9 7 10 8 11 8 12 9 12 12 13 5 1 5 13 1 2 1 6 2 1 4 1 13 6 7 8 2 13 6 7 2 13 6 8	yes yes yes no yes

*** Hướng dẫn thuật toán:**

Cách 1:O(N * Q):

Với mỗi truy vấn:

- A B G1 G2: bỏ cạnh (G1, G2) rồi DFS lại xem A có đến được B hay không.
- A B C : bỏ đỉnh C rồi DFS lại xem A có đến được B hay không.

Cách 2: O(Q log N):

Ta dựng mảng p[i, j] = tổ tiên bậc 2^j của i. Công thức : p[i, j] = p[p[i, j – 1], j – 1].

- mảng num[i] = thời gian đỉnh i được thăm trong thủ tục DFS.
- mảng fin[i] = thời gian đỉnh i thoát ra khỏi thủ tục DFS.

  ⇨ u thuộc nhánh DFS gốc v ⇔(num[u] >= num[v]) and (fin[u] <= fin[v]) (hàm Kt(u, v)).

Với truy vấn (A, B, G1, G2), giả sử G2 thuộc nhánh DFS gốc G1:

- Nếu G1 không là cha trực tiếp của G2 => in ‘yes’.
- Nếu (G1, G2) không là câu => in ‘yes’.

- Nếu (G1, G2) là câu:
    - Nếu Kt(u, G2) = Kt(v, G2) => in ‘yes’.
    - Nếu không in ‘no’.

Với truy vấn (A, B, C)

- Nếu A, B cùng không thuộc nhánh DFS gốc C ⇔ not Kt(A, C) and not Kt(B, C) => in ‘yes’.
- Nếu Kt(A, C) = Kt(B, C) = true: Gọi u, v lần lượt là tổ tiên xa nhất nhưng vẫn là thuộc nhánh DFS gốc C của A và B.
    - Nếu (u = v) hoặc từ u và v có thể lên được tổ tiên của C ⇔ (low[u] < num[C]) and (low[v] < num[C]) thì in ‘yes’.
    - Nếu không in ‘no’.
- Nếu C, B thuộc nhánh DFS gốc A: Gọi x là tổ tiên xa nhất nhưng vẫn là thuộc nhánh DFS gốc C của B.
    - Nếu x có thể lên được tổ tiên của C ⇔ low[x] < num[C] thì in ‘yes’.
    - Nếu không in ‘no’.

* Chương trình mẫu

```
#include <algorithm>
#include <cstdio>
#include <vector>

using namespace std;

int n, m;
struct edge {
   int u, v;
   edge( int U, int V ) { u = U; v = V; }
};
bool operator < ( const edge &A, const edge &B ) { return A.u < B.u; }

struct sparse_graph {
   vector<edge> E;
   vector< vector<edge>::iterator > V;

   void insert_edge( const edge &e ) {
      E.push_back( e );
   }
```

```
   void init() {
      V.resize(n+1);
      sort( E.begin(), E.end() );
      V[0] = E.begin();
      for( int i = 1; i <= n; ++i )
         for( V[i] = V[i-1]; V[i] != E.end() && V[i]->u < i; ++V[i] );
   }

   inline vector<edge>::iterator begin( int u ) { return V[u]; }
   inline vector<edge>::iterator end( int u ) { return V[u+1]; }
} graph;

vector<int> discover, finish, lowlink, depth;
int Time = 0;

vector< vector<int> > children;

void dfs( int u, int dad, int d ) {
   discover[u] = lowlink[u] = Time++;
   depth[u] = d;

   for( vector<edge>::iterator it = graph.begin(u); it != graph.end(u); ++it
) {
      if( it->v == dad ) continue;

      if( discover[it->v] == -1 ) {
         dfs( it->v, u, d+1 );
         lowlink[u] <?= lowlink[it->v];
         children[u].push_back( it->v );
      } else {
         lowlink[u] <?= discover[it->v];
      }
   }

   finish[u] = Time++;
}

int is_descendant( int a, int b ) {
   return discover[b] <= discover[a] && finish[a] <= finish[b];
}

int find_related_child( int me, int descendant ) {
   int lo = 0, hi = children[me].size() - 1;
   while( lo != hi ) {
      int mid = (lo+hi) / 2;

      if( discover[descendant] > finish[ children[me][mid] ] ) lo = mid+1;
      else if( finish[descendant] < discover[ children[me][mid] ] ) hi =
mid-1;
      else lo = hi = mid;
   }
   return children[me][lo];
```

```
}

int main( void ) {
   scanf( "%d%d", &n, &m );
   for( int i = 0; i < m; ++i ) {
      int u, v;
      scanf( "%d%d", &u, &v ); --u; --v;
      graph.insert_edge( edge( u, v ) );
      graph.insert_edge( edge( v, u ) );
   }
   graph.init();
   discover = finish = lowlink = depth = vector<int> (n, -1);
   children.resize( n );
   dfs( 0, -1, 0 );
   scanf( "%d", &m );
   for( int i = 0; i < m; ++i ) {
      int tip, a, b, c, d;
      scanf( "%d%d%d%d", &tip, &a, &b, &c ); --a; --b; --c;
      if( tip == 1 ) {
         scanf( "%d", &d ); --d;
         if( is_descendant( c, d ) ) swap( c, d );

         if( depth[d] != depth[c]+1 ) printf( "yes\n" );
         else if( lowlink[d] < discover[d] ) printf( "yes\n" );
         else if( is_descendant( a, d ) == is_descendant( b, d ) ) printf(
"yes\n" );
         else printf( "no\n" );

      } else {
         if( !is_descendant( a, c ) && !is_descendant( b, c ) ) printf(
"yes\n" );
         else if( is_descendant( a, c ) && is_descendant( b, c ) ) {
            int e = find_related_child( c, a );
            int f = find_related_child( c, b );
            if( e == f ) printf( "yes\n" );
            else if( lowlink[e] < discover[c] && lowlink[f] < discover[c] )
printf( "yes\n" );
            else printf( "no\n" );
         } else {
            if( is_descendant( a, c ) ) swap( a, b );
            int e = find_related_child( c, b );
            if( lowlink[e] < discover[c] ) printf( "yes\n" );
            else printf( "no\n" );
         }
      }
   }
   return 0;
}
```

*** Test:**

- Các thầy cô có thể download bằng link:

http://www.mediafire.com/download/k2qbna3z4u1a5gt/POLICIJA.rar

Bài 7. Đường đua dài nhất

Mạng giao thông của thành phố ByteLand có N nút giao thông. Giữa hai nút giao thông có tối đa một đường phố hai chiều nối trực tiếp giữa chúng. Nhân dịp chào mừng kỷ niệm 100 năm ngày thành lập, Lãnh đạo thành phố quyết định tổ chức một cuộc đua xe đạp. Đường đua xe đạp sẽ xuất phát từ một nút bất kỳ, qua một số nút khác và trở lại nút ban đầu sao cho không có nút nào (trừ nút xuất phát) đường đua qua đó hai lần. Thật ngạc nhiên, mạng lưới giao thông của thành phố cho phép lập nhiều đường đua xe đạp như vậy tuy nhiên: ***mỗi một đường phố sẽ thuộc không quá một đường đua thỏa mãn điều kiện nêu trên***.

Hãy tìm đường đua có số đường phố khác nhau đi qua nhiều nhất.

INPUT: MAXCYCLE.INP

- Dòng đầu ghi hai số nguyên N, M là số nút giao thông và số đường phố trong thành phố (N≤5.000, M≤100000)
- M dòng tiếp theo, mỗi dòng ghi hai số nguyên u, v thể hiện hai nút của một đường phố

OUTPUT: MAXCYCLE.OUT

- Ghi một số nguyên duy nhất là độ dài (số lượng đường phố khác nhau) của đường đua dài nhất.

Ví dụ:

MAXCYCLE.INP	MAXCYCLE.OUT

7 8 3 4 1 4 1 3 7 1 2 7 7 5 5 6 6 2	4

* Hướng dẫn thuật toán

- Đây là một bài toán tìm thành phần song liên thông với số lượng đỉnh lớn nhất và >=3.

* Chương trình mẫu

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

using namespace std;

#define mp make_pair
#define pii pair<int,int>

const int nmax = 5000;
int n,m,dd[nmax],num[nmax],low[nmax],id = 0,maxx = 0,cha[nmax],dem = 0;
vector <int> adj[nmax],adj1[nmax];
set <pii> se;

struct canh
{
    int u,v;
};
stack <canh> s;

void doc()
{
    freopen("MAXCYCLE.inp","r",stdin);
    freopen("MAXCYCLE.out","w",stdout);
    cin >> n >> m;
    for(int i=1; i<=m; i++)
    {
        int u,v;
        cin >> u >> v;
        adj[u].push_back(v);
        adj[v].push_back(u);
    }
}
```

```
void DFS(int u)
{
    id++;
    num[u] = low[u] = id;
    dd[u] = 1;
    for(int i=0;i<adj[u].size();i++)
    {
        int v = adj[u][i];
        if(dd[v] == 0)
        {
            canh e;
            e.u = u;
            e.v = v;
            s.push(e);
            cha[v] = u;
            DFS(v);
            low[u] = min(low[u],low[v]);
            if(low[v] >= num[u])
            {
                int d = 1;
                do{
                    e = s.top();
                    s.pop();
                    d++;
                }while(e.u != u || e.v != v);
                if (d>2)
                maxx = max(maxx,d);
            }
        }
        else low[u] = min(low[u],num[v]);
    }
}

int main()
{
    doc();
    for(int i=1; i<=n; i++)if(dd[i] == 0)DFS(i);
    cout << maxx;
    return 0;
}
```

*** Test:**

- Các thầy cô có thể download bằng link.

http://www.mediafire.com/download/fvxw2sel1iuig1h/maxcycle.rar

Bài 8. Đường đi dài nhất

Cho đồ thị có hướng không có chu trình (DAG). Mỗi cạnh của đồ thị được gán trọng số là một số nguyên.

Giải các bài toán sau:

a) Tìm đường đi dài nhất từ đỉnh *s* đến đỉnh *t*

b) Đếm số lượng đường đi từ *s* đến đỉnh *t*

Input: DAG.INP

- Dòng đầu tiên ghi hai số nguyên dương *n*, *m* ($1 \le n \le 3.10^5, 1 \le m \le 5.10^5$) lần lượt là số đỉnh và số cạnh của đồ thị. Các đỉnh đánh số từ 1 đến *n*
- Dòng thứ hai ghi hai số nguyên dương *s*, *t* ($1 \le s,t \le n$, $s \ne t$)
- *m* dòng tiếp theo, mỗi dòng ghi 3 số nguyên dương *a*, *b*, *c* ($1 \le a, b \le n$, $a \ne b$, $|c| \le 10^3$) thể hiện có một cung một chiều nối từ đỉnh *a* đến đỉnh *b* có trọng số là *c*.
- Dữ liệu đảm bảo rằng đồ thị không có chu trình

Output: DAG.OUT

- Dòng thứ nhất ghi độ dài của đường đi dài nhất từ *s* đến *t*. Nếu không tồn tại đường đi như vậy thì ghi "NO PATH" (không có dấu nháy kép)
- Dòng thứ hai ghi số lượng đường đi khác nhau có thể có từ *s* đến *t*. Vì con số này có thể rất lớn nên chỉ cần lấy phần dư của chúng khi chia cho 10^9+7

Ví dụ:

DAG.INP	DAG.OUT
5 6 1 5 1 4 5 4 5 -4 1 2 1 2 3 2 3 5 3 1 3 -7	6 3

* Hướng dẫn thuật toán

- Gọi F[u], C[u] là đường đi dài nhất vầ số lượng đường đi từ đỉnh s đến đỉnh u, và khi đó F[t] và C[t] là hai kết quả của bài toán.

- Xét cung (u,v): Dễ dàng suy ra công thức:

$$F[v] = \max(F[u]) + w(u,v) \quad (1)$$

$$C[v] = C[v] + C[u] \quad (2)$$

- Tuy nhiên quá trình duyệt DFS nếu không cẩn thận thì một đỉnh sẽ được duyệt nhiều lần và như vậy sẽ không đáp ứng được yêu cầu của bài toán về mặt thời gian.

- Có thể giải quyết vấn đề trên như sau:

- Sử dụng DFS để đánh lại chỉ số thứ tự các đỉnh của đồ thị gọi là sắp xếp topo.

- Từ sắp xếp topo khi đó sử dụng công thứ (1) và (2) ở trên.

* Chương trình mẫu

```
#include <bits/stdc++.h>
#define maxn 300005
#define tr(i,c) for(typeof((c).begin()) i=(c).begin();i!=(c).end();i++)
#define oo 2000000000
#define modulo 1000000007

using namespace std;
typedef pair<int,int> II;

int n, m, xp, kt;
vector<II> g[maxn];
int x[maxn], slx, cl[maxn];
int kc[maxn], f[maxn];

void dfs(int u)
{
    cl[u]=1;
    tr(i,g[u])
    {
        int v=(*i).first;
        if (cl[v]==0) dfs(v);
    }
    x[--slx]=u;
}


int main()
{
```

```
    freopen("dag.inp","r",stdin);
    freopen("dag.out","w",stdout);
    // Doc du lieu
    scanf("%d%d",&n,&m);
    scanf("%d%d",&xp,&kt);
    for(int i=1; i<=m; i++)
    {
        int u,v,w;
        scanf("%d%d%d",&u,&v,&w);
        g[u].push_back(II(v,w));
    }
    // Sap xep topo tren do thi
    memset(cl,0,sizeof(cl));
    slx=n+1;
    for(int i=1; i<=n; i++)
        if (cl[i]==0) dfs(i);
    // Tim duong di dai nhat tu xp den ket thuc
    for(int i=1; i<=n; i++) kc[i]=-oo;
    for(int i=1; i<=n; i++)
    {
        int u=x[i];
        if (u==xp) kc[u]=0;
        if (f[u]!=-oo)
            tr(i,g[u])
        {
            int v=(*i).first, L=(*i).second;
            kc[v]=max(kc[v],kc[u]+L);
        }
    }
    if (kc[kt]==-oo) printf("NO PATH\n");
    else printf("%d\n",kc[kt]);
    // Dem so luong duong di tu s den t
    for(int i=1; i<=n; i++) f[i]=0;
    for(int i=1; i<=n; i++)
    {
        int u=x[i];
        if (u==xp) f[u]=1;
        tr(i,g[u])
        {
            int v=(*i).first;
            f[v]=(f[v]+f[u]) % modulo;
        }
    }
    printf("%d\n",f[kt]);
}
```

*** Test:**

- Các thầy cô có thể download bằng link.

http://www.mediafire.com/download/o6odd1o7gi7ly8e/dag.rar

Hiểu rõ được cơ chế hoạt động thuật toán tìm kiếm theo chiều sâu (DFS) bằng đệ quy cho ta cách cài đặt rất ngắn gọn, rõ ràng. Những cải tiến nhỏ trong thuật toán có thể đem lai nhiều điều thú vị, giải quyết được nhiều lớp bài toán khác nhau. Trong phạm vi chuyên đề này tôi không thể trình bày hết được những ứng dụng của DFS, nhưng phần nào cho thấy được tầm quan trọng của DFS.

Với khả năng, năng lực còn hạn chế tôi chưa thể hiểu biết được hết tất cả những ứng dụng của DFS, nhưng hy vọng thông qua chuyên đề này tôi đã truyển tải đến đồng nghiệp một phần nào đó cách sử dụng thuật toán DFS để giải quyết các bài tập. Rất mong nhận được sự góp ý của các đồng nghiệp