MỤC LỤC

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

Mở đầu

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 đồ thị (vì đã có rất nhiều tài liệu viết về vấn đề này rồi đặc biệt là cuốn tài liệu giáo khoa chuyên tin quyển 1), mà ở đây tôi chỉ chú trọng vào việc xây dựng hệ thống các bài tập có ứng dụng DFS để giải quyết (chủ yếu các bài toán là đồ thị vô hướng).

Trong nhiều năm giảng dạy các lớp chuyên tin tôi nhận thấy phải trang bị cho học sinh kiến thức nền thật chắc chắn thì học sinh mới có khả năng tiếp thu và tự học những kiến thức cao hơn. Nhiều khi thầy cô chúng ta quá chú trọng trang bị kiến thức mới mà quên đi những kiến thức nền tảng giúp học sinh hiểu bản chất vấn đề.

Để học sinh có thể hiểu bản chất và làm thành thạo các bài toán cơ bản thì không còn cách nào khác là các em phải luyện tập thật nhiều. Tuy nhiên không phải lấy một bài làm đi làm lại mà phải làm những bài tương ứng với nhiều dạng phát biểu khác nhau để học sinh nhận diện bài toán. Đặc biệt nếu các bài tập mới mà được phát triển từ những bài cơ bản trước đó, các em chỉ sửa lại một chút code, hoặc thêm một vài kỹ thuật xử lý đơn giản thì việc thành thạo kiến thức cơ bản không những trở nên thú vị mà còn rèn luyện kỹ năng lập trình cho các em.

Với mục tiêu đó tôi đã cố gắng sưu tầm phân loại và sắp xếp để có thể đưa ra một hệ thống các bài tập cơ bản giúp học sinh có thể rèn luyện các kỹ năng của mình. Các bài tập mà tôi đưa ra thường ở mức cơ bản và dành cho đối tượng học sinh lớp chuyên tin từ 10 lên 11, các em mới được tiếp cận với lý thuyết đồ thị nên không tránh khỏi những ngỡ ngàng, lạ lẫm, có phần trừu tượng khó hiểu. Sau khi nắm vững những dạng bài tập này chúng ta có thể mở rộng các dạng toán phức tạp hơn.

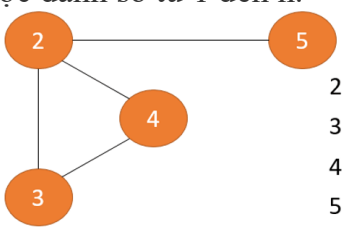
Để học được chuyên đề này học sinh cần có kiến thức và kỹ năng về lập trình đệ quy.

I. Tổng quan về các cách biểu diễn đồ thị và phép duyệt đồ thị theo chiều sâu (DFS).

Yêu cầu đối với việc tiếp cận chuyên đề này: học sinh đã nắm được một số khái niệm cơ bản về đồ thị (đỉnh, cạnh, đồ thị có hướng, vô hướng, có trọng số, không có trọng số, thành phần liên thông, đường đi,…)

I.1 Biểu diễn đồ thị bằng ma trận kề

Giả sử chúng ta có một **đơn đồ thị vô hướng không trọng số**, có n đỉnh. Chúng ta coi như các đỉnh được đánh số từ 1 đến n.



	2	3	4	5
2	0	1	1	1
3	1	0	1	0
4	1	1	0	0
5	1	0	0	0

Tổ chức dữ liệu theo kiểu ma trận kề trong lý thuyết đồ thị bằng cách xây dựng ma trận vuông a[i,j] cấp n. Với:

- A[i][j] = 1 khi đồ thi của chúng ta có cạnh nối từ đỉnh i đến đỉnh j.
- A[i][j] = 0 khi đồ thị không có cạnh nối từ đỉnh i đến đỉnh j.

Nhược điểm khi dùng ma trận kề trong lý thuyết đồ thị:

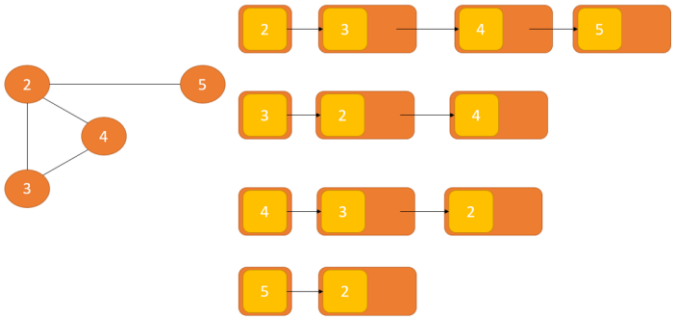
Dễ dàng nhận thấy vì sử dụng mảng 2 chiều nên kích thước dữ liệu chúng ta tốn n^2 và độ phức tạp của dữ liệu này cũng là O(n^2).

Ưu điểm khi dùng ma trận kề trong lý thuyết đồ thị: là khi nhìn vào dữ liệu được tổ chức bằng ma trận kề, thì dễ dàng nhận biết được 2 đỉnh nào kề với nhau, chúng ta dễ cài đặt, và biết được bậc của từng đỉnh nếu đó là đồ thị đơn.

I.2 Biểu diễn đồ thị bằng danh sách kề

Danh sách kề biểu thị một biểu đồ dưới dạng một mảng các danh sách được liên kết. Chỉ số của mảng đại diện cho một đỉnh và mỗi phần tử trong danh sách liên kết của nó đại diện cho các đỉnh khác tạo thành một cạnh với đỉnh đó.

Biểu đồ và biểu diễn danh sách kề tương đương của nó được biểu diễn trong hình bên dưới.



Danh sách kề đơn giản nhất cần có cấu trúc dữ liệu nút để lưu một đỉnh và cấu trúc dữ liệu đồ thị để tổ chức các nút.

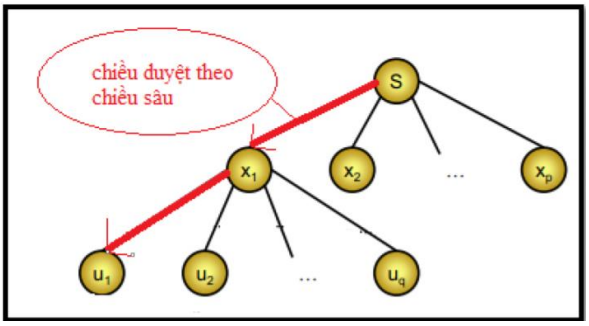
Danh sách kề có hiệu quả về mặt lưu trữ vì chúng ta chỉ cần lưu trữ các giá trị cho các cạnh. Đối với một đồ thị thưa có hàng triệu đỉnh và cạnh, điều này có nghĩa là rất nhiều không gian sẽ được tiết kiệm.

I.3 Phép duyệt đồ thị theo chiều sâu:

Xuất phát từ đỉnh gốc, từ đỉnh đó phát triển xa nhất có thể theo mỗi nhánh

Ý tưởng thuật toán:

Từ một đỉnh S ban đầu ta sẽ có các đỉnh kề là x, từ đỉnh x ta sẽ có các đỉnh kề là u, và nó cũng thuộc nhánh s-x-u… Chúng ta thăm các nhánh đó theo chiều sâu (thăm đến khi không còn đỉnh kề chưa duyệt). Điều đó gợi cho chúng ta viết một thủ tục đệ quy DFS(u) để mô tả việc duyệt từ đỉnh u sang đỉnh kề v chưa được thăm.



Mô hình thuật toán:

```
void dfs(int u)
{
    free[u]=false; // đánh dấu đỉnh u đã được thăm
    for (int v=1; v<=n; v++)
        if ((tồn tại cạnh u, v) và (free[u][v]==true))
      // tồn tại đỉnh kề với u, chưa được thăm
            dfs(v); //duyệt đỉnh v
}
```

Đề bài ví dụ:

Viết chương trình ghi ra thứ tự duyệt DFS xuất phát từ đỉnh s. Đồ thi gồm n đỉnh, m cạnh 2 chiều, các thành phần trên đồ thị liên thông với nhau.

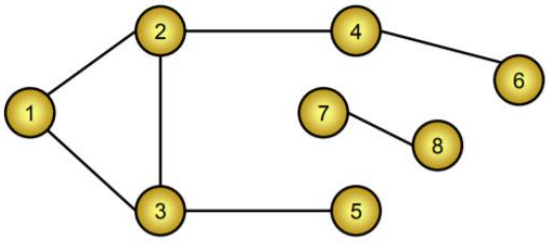
Dữ liệu vào:

- Dòng đầu: gồm 3 số nguyên n, m, s (1<=n, m<=100, 1<=s<=n)
- M dòng tiếp theo: mỗi dòng gồm 2 số u, v, mô tả 1 cạnh trong đồ thị

Dữ liệu ra:

- Gồm nhiều dòng, là thứ tự duyệt DFS.

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



```
#include <bits/stdc++.h>
using namespace std;
 int a[101][101];
 int n, m, Free[101], u, v, s;
 void DFS(int u)
{
    cout << u << endl;
    Free[u] = false;
    for (int v = 1; v <= n; v++)
        if (a[u][v] == 1 && Free[v])
            DFS(v);
}
 int main()
{
    cin >> n >> m >> s;
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= n; j++)
            a[i][j] = 0;

    for (int i = 1; i <= m; i++)
    {
        cin >> u >> v;
        a[u][v] = 1;
        a[v][u] = 1;
    }

    for (int i = 1; i <= n; i++)
        Free[i] = 1;
    DFS(s);
    return 0;
}
```

II. Một số ứng dụng của DFS

II.1. Duyệt qua các đỉnh của đồ thị:

Bài 1: Mạng máy tính.

Đề bài:

Trong một phòng máy, có n máy tính được đánh số từ 1 đến n và kết nối với nhau thông qua hệ thống dây cáp truyền dữ liệu. Hai máy tính truyền được dữ liệu cho nhau nếu được nối cáp trực tiếp hoặc thông qua các máy tính trung gian mà có kết nối với nhau (theo cả 2 chiều). Trong quá trình phục vụ học tập, do học sinh đi lại nhiều nên một số đường cáp đã bị đứt làm cho kết nối không thực hiện được. Để phục vụ cho buổi thi học kỳ tới, thầy giáo Nam quan tâm đến vấn đề rằng: liệu từ máy giáo viên (có số thứ tự 1), thầy có thể thu bài của học sinh ở máy số k thông qua đường truyền dữ liệu được không? Bạn hãy lập trình trả lời câu hỏi trên giúp thầy Nam nhé.

Dữ liệu vào: từ file bai1.inp bao gồm các dòng:

- Dòng đầu ghi các số n, m, k, mỗi số cách nhau một dấu cách. (0< n ≤100)
- m dòng tiếp theo, mỗi dòng ghi 2 số u, v cách nhau một dấu cách cho biết có đường cáp nối trực tiếp từ máy u đến máy v.

Dữ liệu ra: file bai1.out ghi ra thông báo "yes" nếu thầy Nam có thể thu bài được của học sinh ở máy số k, nếu không thì đưa ra thông báo là "no".

Ví dụ:

Bai1.inp	Bai1.out
6 6 3 1 2 1 5 2 5 3 4 4 5 4 6	Yes

Thuật toán:

Bài này là một bài duyệt DFS hết sức cơ bản, có thể coi là bài ở mức độ nhận biết. Ở bài tập này chỉ yêu cầu học sinh nhận diện được bài toán cơ sở và có chỉnh sửa một chút code ví dụ để phù hợp với yêu cầu của bài.

Chương trình tham khảo:

```
#include <bits/stdc++.h>
using namespace std;
int a[101][101],dd[101];
int n,m,k,u,v;
void ktao()
{
    freopen("bai1.inp","r",stdin);
    freopen("bai1.out","w",stdout);
    cin>>n>>m>>k;
    for(int i=1;i<=n;i++)
        for(int j=1;j<=n;j++)a[i][j]=0;
    for(int i=1;i<=m;i++)
    {
        cin>>u>>v;
        a[u][v]=a[v][u]=1;
    }
    for(int i=1;i<=n;i++)dd[i]=0;
    }
void dfs(int x)
{
    dd[x]=1;
    for(int t=1;t<=n;t++)
        if(a[x][t]==1&&dd[t]==0)
    {
        dd[t]=1;
        dfs(t);
    }
}
void kqua()
{
    if(dd[k]==1)
```

```
        cout<<"yes";
        else cout<<"no";
    }
    int main()
    {
       ktao();
       dfs(1);
       kqua();
        return 0;
    }
```

Test:

https://drive.google.com/drive/folders/1e6o5X5k0_KY5U8uYdSAuh3OzvIN-8kpM

Bài 2: Chú bò hư hỏng(BVDAISY)

Đề bài:

Nông dân John có N (1<=N<=250) con bò đánh số từ 1..N chơi trên bãi cỏ. Để tránh bị lạc mất các con bò, mỗi con bò có thể được nối với một số con bò khác bằng dây thừng. Có tất cả M (1 <= M <= N*(N-1)/2) dây thừng nối các con bò. Tất nhiên, không có 2 con bò mà có nhiều hơn 1 dây thừng nối giữa chúng. Dữ liệu cho biết mỗi cặp con bò c1 và c2 là nối với nhau (1 <= c1 <= N; 1 <= c2 <= N; c1 != c2).

Nông dân John buộc cố định con bò 1 bằng sợi dây xích. Các con bò khác phải nối với con bò 1 bằng một số sợi dây thừng. Tuy nhiên, một số con bò hư hỏng không như vậy. Hãy giúp nông dân John tìm các con bò hư hỏng đó (không kết nối tới bò 1). Dĩ nhiên, con bò thứ 1 luôn nối tới chính nó.

Input:

Dòng 1: 2 số nguyên cách nhau bởi dấu cách: N và M

Dòng 2 đến dòng M+1: Dòng i+1 cho biết 2 con bò nối với nhau bằng sợi dây thứ i là c1 và c2 cách nhau bởi dấu cách.

Output:

Nếu không có con bò hư hỏng, in ra 0. Ngược lại, in ra trên mỗi dòng 1 số nguyên là thứ tự con bò hư hỏng theo thứ tự tăng dần.

Ví dụ:

Input	Output
6 4 1 3 2 3 1 2 4 5	4 5 6

Thuật toán:

Dùng DFS liệt kê các đỉnh không thuộc thành phần liên thông với đỉnh có số hiệu 1

Mục đích của bài này là giúp HS luyện cách nhận biết và sử dụng DFS. Bài cơ bản và rất dễ code.

Chương trình tham khảo:

```
#include <bits/stdc++.h>
using namespace std;
 int a[251][251];
 int n, m, Free[251], u, v, ok=0;
 void DFS(int u)
{
    //cout << u << endl;
    Free[u] = false;
    for (int v = 1; v <= n; v++)
        if (a[u][v] == 1 && Free[v])
            DFS(v);
}
 int main()
{
    cin >> n >> m;
    for (int i = 1; i <= n; i++)
        for (int j = 1; j <= n; j++)
            a[i][j] = 0;

    for (int i = 1; i <= m; i++)
    {
        cin >> u >> v;
        a[u][v] = 1;
        a[v][u] = 1;
    }

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

    for(int i=1;i<=n;i++)
    if (Free[i]==1) {ok=1;cout<<i<<endl;}
    if (ok==0) cout<<0;
    return 0;
}
```

Test:

http://www.spoj.com/PTIT/problems/BCDAISY/

II.2. Tìm đếm các thành phần liên thông trong đồ thị vô hướng:

Vùng liên thông trong đồ thị là tập hợp các đỉnh mà từ một đỉnh bất kỳ có đường đi trực tiếp hoặc gián tiếp đến các đỉnh khác trong tập hợp đó.

Bài 3: Đếm thành phần liên thông

Đề bài:

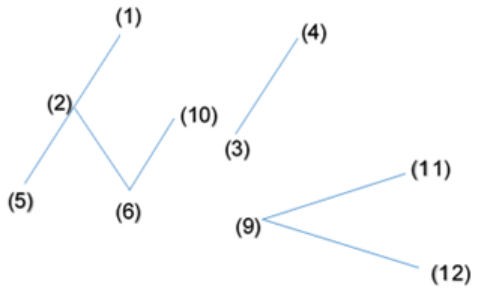
Cho đồ thị vô hướng có N đỉnh, M cạnh. Hãy đếm số lượng vùng liên thông trong đồ thị.

Dữ liệu: Vào từ file văn bản BAI3.INP gồm:

- Dòng 1: Ghi số nguyên N và M (M, N ≤ 3000).
- M dòng tiếp theo, mỗi dòng ghi số nguyên dương u và v thể hiện có đường đi giữa hai đỉnh u và v (u, v ≤ N).

Kết quả: Ghi ra file văn bản BAI3.OUT số lượng vùng liên thông.

BAI3.INP	BAI3.OUT
12 7 1 2 2 5 2 6 6 10 3 4 9 11 9 12	5 (gồm 3 vùng bên, đỉnh 7, đỉnh 8)



Thuật toán:

Tư tưởng giải thuật Duyệt theo chiều sâu:
Ban đầu các đỉnh fre[i] = True.
Duyệt đến đỉnh thứ i, đánh dấu i đã đi đến (fre[i] = False). Ta thực hiện duyệt tất cả các đỉnh j đang free và có đường đi trực tiếp từ i đến j thì ta Duyệt (j).
Tới đỉnh j, ta lặp lại việc duyệt như ở bước trên.
Lưu ý: Với mỗi đỉnh i, ta sẽ đi đến được tất cả các đỉnh có đường đi trực tiếp hoặc gián tiếp tới i. Như vậy, mỗi lần duyệt (i) xong ta sẽ thu được một vùng liên thông. Ở bài này, ta tăng biến đếm số lượng vùng liên thông lên.
Độ phức tạp thuật toán: O(M + N).

Chương trình tham khảo:

```
#include <bits/stdc++.h>
using namespace std;
int a[3001][3001],dd[3001];
int n,m,u,v,dem;
void ktao()
{
    freopen("Bai3.inp","r",stdin);
    freopen("Bai3.out","w",stdout);
    cin>>n>>m;
    for(int i=1;i<=n;i++)
        for(int j=1;j<=n;j++)a[i][j]=0;
    for(int i=1;i<=m;i++)
    {
        cin>>u>>v;
```

```
            a[u][v]=a[v][u]=1;
        }
        for(int i=1;i<=n;i++)dd[i]=0;
        }
    void dfs(int x)
    {
        dd[x]=1;
        for(int t=1;t<=n;t++)
            if(a[x][t]==1&&dd[t]==0)
        {
            dd[t]=1;
            dfs(t);
        }
    }
    void kqua()
    {   dem=0;
        for(int i=1;i<=n;i++)
          if(dd[i]==0)
            {
                dem++;
                dfs(i);
            }
        cout<<dem;
    }
    int main()
    {
       ktao();
       kqua();
       return 0;
    }
```

Test:

https://drive.google.com/drive/folders/1e6o5X5k0_KY5U8uYdSAuh3OzvIN-8kpM

Bài 4: Tin nhắn

Đề bài:

Một lớp gồm N học sinh, mỗi học sinh cho biết những bạn mà học sinh đó có thể liên lạc được bằng điện thoại (liên lạc được 2 chiều). Thầy chủ nhiệm đang có một thông tin rất quan trọng cần thông báo tới tất cả các học sinh. Để tiết kiệm thời gian, thầy chỉ nhắn tin tới 1 số học sinh rồi sau đó nhờ các học sinh này nhắn lại cho tất cả các bạn mà các học sinh đó có thể liên lạc được và cứ lần lượt như thế làm sao cho tất cả các học sinh trong lớp đều nhận được tin. Hãy tìm một số ít nhất các học sinh mà thầy chủ nhiệm cần nhắn.

Dữ liệu vào: từ file Bai4.inp bao gồm các dòng:

– Dòng đầu là N, M (N ≤30, M là số lượng liên lạc điện thoại 2 chiều)

– M dòng tiếp theo mỗi dòng gồm 2 số u,v cho biết học sinh u và v có thể gửi tin nhắn tới nhau.

Dữ liệu ra: từ file Bai4.out ghi số học sinh ít nhất mà thầy cần nhắn tin.

Bai4.inp	Bai4.out
7 6 1 2 1 3 2 3 4 5 4 6 5 6	3

Thuật toán:

Bài toán này thực chất là bài toán đếm thành phần liên thông trong đồ thị vô hướng.

Thuật toán và code giống hệt bài 3. Bài này nhằm mục đích chỉ ra liên hệ thực tế của bài toán đếm thành phần liên thông. Giúp học sinh bước đầu nhận ra bài toán đồ thị ở một phát biểu dưới dạng khác thông lệ.

Chương trình tham khảo: Giống bài 3

Test : Trình sinh test giống bài 3(để ý n<=30 cho phù hợp với thực tế) (đổi tên file input, output cho phù hợp).

Bài 5: Robin C11BC2

Đề bài:

Một ngày đẹp trời nọ, trên vương quốc của các Coders 2011, bỗng xuất hiện 1 lão phù thủy độc ác, lão phù thủy sirDat_LS đã có âm mưu thôn tính đất nước của đức vua vodanh9x. Lão phù thủy này rất yêu con gái của đức vua là Rose và đã bắt Rose về nơi ở của lão ta. Đức vua vodanh9x liền tìm hiệp sĩ Robin và sẽ hứa gả con gái cho Robin nếu chàng cứu được công chúa Rose trở về. Lão phù thủy sirDat_LS độc ác với khuôn mặt rất ghê tởm khiến công chúa mỗi khi nhìn thấy hắn thì công chúa lại ngất đi. Và rồi, chàng Robin của chúng ta đã tìm được đến nơi ở của lão phù thủy. Nơi ở của lão là 1 mê cung có N phòng, và N phòng này liên thông với nhau và có đúng N-1 đường đi (coi mỗi đường đi là 1 cạnh). Nhưng khó khăn thay, lão phù thủy đã đánh số mỗi đường đi là 1 hoặc 2. Nếu chàng Robin muốn đến cứu công chúa, thì từ nơi xuất phát đến nơi có công chúa phải có ít nhất một đường đi được đánh số 2, nếu không chàng Robin sẽ chết. Yêu cầu: Cho m truy vấn (m <= 10^5) mỗi truy vấn có dạng (x,y), trong đó x là nơi xuất phát của Robin và y là nơi nhốt công chúa. Xác định đường đi ngắn nhất từ x đến y có cạnh có trọng số 2 hay không.

Dữ liệu vào:

- Dòng đầu là số nguyên N (N <= 10^4) – số đỉnh của đồ thị và M – số truy vấn.
- Từ dòng 2 đến dòng N: dòng thứ i chứa 2 số nguyên dương x (x < i) và k (k <= 2) nghĩa là có cạnh nối giữa i và x và được đánh số là k.
- M dòng sau: mỗi dòng chứa 2 số x và y (Biểu thị cho truy vấn (x,y)).

Dữ liệu ra: Với mỗi truy vấn, xuất ra "YES" nếu tồn tại đường đi có ít nhất 1 cạnh có trọng số 2, ngược lại xuất ra "NO".

Ví dụ:

Input	Ouput
6 7 1 1 1 2 3 1 1 2 5 2 1 3 5 1 2 1 2 1 1 2 2 4 1 2	YES YES NO NO NO YES NO

Thuật toán:

– Chỉ đọc các cạnh 1 vào đồ thị, còn các cạnh 2 ta sẽ bỏ.

– Xây dựng mảng DD[i] là chỉ số vùng liên thông của đỉnh i. Xây dựng mảng này bằng DFS.

– Trả lời các truy vấn: Nếu DD[x]<>DD[y] thì kết quả là YES (bởi vì khi ta bỏ cạnh 2 ra, mà ko thể đi từ x đến y thì dễ dàng suy ra đồ thị ban đầu nếu đi từ x đến y sẽ đi qua cạnh 2). Ngược lại là NO.

Chương trình tham khảo:

```
#include <bits/stdc++.h>
using namespace std;
int64_t d;
bool a[10001][10001];
int64_t u,v,n,m,i,x;
int64_t kt[10001];
int64_t trace[10001];
void DFS(int64_t u)
{
    int64_t v;
    kt[u]=d;
    for(v=1;v<=n;v++)
        if (a[u][v] && trace[v]==0)
    {
        trace[v]=u;
        DFS(v);
    }
}
```

```
void scan()
{
    int64_t u,v;
    for(v=1;v<=n;v++) kt[v]=0;
    d=0;
    for(u=1;u<=n;u++)
        if(kt[u]==0)
    {
        d=d+1;
        DFS(u);
    }
}
int main()
{
    //freopen("dfs.inp","r",stdin);
    //freopen("dfs.out","w",stdout);
    cin>>n>>m;
    for(i=2;i<=n;i++)
    {
        cin>>v>>x;
        if (x!=2)
        {
            a[i][v]=true;
            a[v][i]=true;
        }
    }
    scan();
    for(i=1;i<=m;i++)
    {
        cin>>u>>v;
        if(kt[u]!=kt[v]) cout<<"YES"<<endl; else
cout<<"NO"<<endl;
    }
    return 0;
}
```

Test:

http://vn.spoj.com/problems/C11BC2/

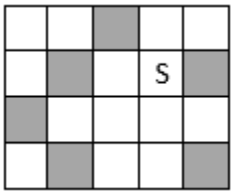
II.3 Đánh số các thành phần liên thông

Bài 6: Ốc sên ăn rau (OCSE)

Đề bài

Có một khu vườn hình chữ nhật kích thước n x m ô vuông (n dòng, m cột). Ta đánh số các dòng từ 1 đến n theo chiều từ trên xuống dưới, các cột từ 1 đến m theo chiều từ trái qua phải. Tại những ô vuông là đất bình thường người ta trồng rau. Tuy nhiên có một số ô là đá nên không trồng rau được. Có một chú ốc sên tại ô (y, x), y là vị trí dòng, x là vị trí cột.

Từ một ô, chú ốc sên chỉ có thể di chuyển sang 4 ô liền kề (y-1, x), (y+1, x), (y, x-1), (y, x+1). Nếu gặp ô đá thì ốc sên không đi vào được.



Ốc sên đang rất đói. Bạn hãy xác định xem chú có thể ăn được số lượng rau nhiều nhất là bao nhiêu.

Dữ liệu vào: gồm các dòng sau:

- Dòng thứ nhất gồm bốn số nguyên n, m, y, x, mỗi số các nhau một khoảng trắng ($1 \le y \le n \le 100, 1 \le x \le m \le 100$).

- Trong n dòng tiếp theo, mỗi dòng gồm m số nguyên 0 hoặc 1 biểu thị vườn rau, mỗi số cách nhau một khoảng trắng. Số 0 nghĩa là ô rau, còn số 1 nghĩa là ô đá.

(Dữ liệu cho đảm bảo ô (y, x) là ô rau)

Dữ liệu ra:

- Là một số nguyên xác định số lượng ô lớn nhất mà ốc sên có thể di chuyển đến.

Ví dụ:

Input	Output
4 5 2 4 0 0 1 0 0 0 1 0 0 1 1 0 0 0 0 0 1 0 0 1	10

Thuật toán:

Bài OCSE là dạng đơn giản, áp dụng DFS để loang trên ma trận.
Dùng DFS loang từ vị trí đứng của con sên, trong quá trình loang đếm số lượng ô là rau mà con sên đi qua.

Chương trình tham khảo:

```
#include <bits/stdc++.h>
using namespace std;
int dx[]={0,0,-1,0,1};
int dy[]={0,-1,0,1,0};
int64_t a[102][102];
int64_t n,x,y,dem,i,j,m;
void DFS(int64_t xx,int64_t yy)
{
    int64_t i;
    for(i=1;i<=4;i++)
        if (a[xx+dx[i]][yy+dy[i]]==0)
    {
        dem++;
        a[xx+dx[i]][yy+dy[i]]=1;
```

```
            DFS(xx+dx[i],yy+dy[i]);
        }
}
int main()
{
   // freopen("DFS.inp","r",stdin);
   // freopen("DFS.out","w",stdout);
    cin>>n>>m>>x>>y;
    memset(a,1,sizeof(a));
    dem=1;
    for(i=1;i<=n;i++)
    {
        for(j=1;j<=m;j++) cin>>a[i][j];
    }
    a[x][y]=1;
    DFS(x,y);
    cout<<dem;
return 0;
}
```

Test:
http://ntucoder.net/Problem/Details/51

Bài 6: VBGRASS spoj

Đề bài:

Bessie dự định cả ngày sẽ nhai cỏ xuân và ngắm nhìn cảnh xuân trên cánh đồng của nông dân John, cánh đồng này được chia thành các ô vuông nhỏ với R (1 <= R <= 100) hàng và C (1 <= C <= 100) cột. Bessie ước gì có thể đếm được số khóm cỏ trên cánh đồng. Mỗi khóm cỏ trên bản đồ được đánh dấu bằng một ký tự '#' hoặc là 2 ký tự '#' nằm kề nhau (trên đường chéo thì không phải). Cho bản đồ của cánh đồng, hãy nói cho Bessie biết có bao nhiêu khóm cỏ trên cánh đồng.

Ví dụ như cánh đồng dưới dây với R=5 và C=6:

Input	Output	Giải thích
5 6 .#.... ..#... ..#..# ...##. .#....	5	Cánh đồng này có 5 khóm cỏ: một khóm ở hàng đầu tiên, một khóm tạo bởi hàng thứ 2 và thứ 3 ở cột thứ 2, một khóm là 1 ký tự nằm riêng rẽ ở hàng 3, một khóm tạo bởi cột thứ 4 và thứ 5 ở hàng 4, và một khóm cuối cùng ở hàng 5.

Dữ liệu vào:

Dòng 1: 2 số nguyên cách nhau bởi dấu cách: R và C

Dòng 2..R+1: Dòng i+1 mô tả hàng i của cánh đồng với C ký tự, các ký tự là '#' hoặc '.' .

Dữ liệu ra:

Dòng 1: Một số nguyên cho biết số lượng khóm cỏ trên cánh đồng.

Thuật toán: Dùng DFS đếm số lượng thành phần liên thông. Chỉ có điều phải khởi tạo ban đầu:
- Tạo dinh[R][C]: với lần lượt các đỉnh từ 1->R*C;
- Duyệt theo 4 hướng (lên, trái, xuống, phải) và đánh dấu;

Chương trình tham khảo:

```
#include <bits/stdc++.h>
using namespace std;
int chuaxet[10004];
void khoitao ()
{
     for (int i=1; i<=10000; i++)
     {
          chuaxet[i]=1;
     }
}
char grass[102][102];
int dinh[102][102];
int R, C;
void nhap ()
{
     int stt=0;
     cin>>R>>C;
     for (int i=1; i<=R; i++)
     {
          for (int j=1; j<=C; j++)
          {
               cin>>grass[i][j];
               stt++;
               dinh[i][j]=stt;    //khoi tao dinh
          }
     }
}
int xqX[]={0, -1, 0, 1};
int xqY[]={1, 0, -1, 0};
void DFS_RA (int r, int c)
{
     for (int i=0; i<4; i++)
     {
          int X=c+xqX[i];
          int Y=r+xqY[i];
          if ((X>=1 && X<=C) && (Y>=1 && Y<=R) &&
chuaxet[dinh[Y][X]]==1 && grass[Y][X]=='#')
          {
               chuaxet[dinh[Y][X]]=0; //Dinh da duyet
               DFS_RA (Y, X);
          }
```

```
     }
}
int main ()
{
     nhap ();
     khoitao();
     int dem=0;
     for (int i=1; i<=R; i++)
     {
          for (int j=1; j<=C; j++)
          {
               if (grass[i][j]=='#' &&
chuaxet[dinh[i][j]]==1)
               {
                    dem++;
                    chuaxet[dinh[i][j]]=0; //Dinh da duyet
                    DFS_RA (i, j);
               }
          }
     }
     cout<<dem;
     return 0;
}
```

Test:
https://vn.spoj.com/problems/VBGRASS/

Bài 7: Đếm ao (BCLKCOUN)

Đề bài

Sau khi kết thúc OLP Tin Học SV, một số OLP-er quyết định đầu tư thuê đất để trồng rau. Mảnh đất thuê là một hình chữ nhật N x M (1<=N<=100; 1<=M<=100) ô đất hình vuông. Nhưng chỉ sau đó vài ngày, trận lụt khủng khiếp đã diễn ra làm một số ô đất bị ngập. Mảnh đất biến thành một số các ào. Các OLP-er quyết định chuyển sàng nuôi cá. Vấn đề lại nảy sinh, các OLP-er muốn biết mảnh đất chia thành bào nhiêu cái ào để có thể tính toán nuôi trồng hợp lý. Bạn hãy giúp các bạn ý nhé.

Chú ý: Ao là gồm một số ô đất bị ngập có chung đỉnh. Dễ nhận thấy là một ô đất có thể có tối đà 8 ô chung đỉnh.

Input:

- Dòng1: 2 số nguyên cách nhàu bởi dấu cách: N và M
- Dòng 2..N+1: M kí tự liên tiếp nhàu mỗi dòng đại diện cho 1 hàng các ô đất. Mỗi kí tự là 'W' hoặc '.' tương ứng với ô đất đã bị ngập và ô đất vẫn còn nguyên.

Output:

- 1 dòng chứà 1 số nguyên duy nhất là số ao tạo thành.

Ví dụ:

Input	Output
10 12	3

W……..WW. .WWW…..WWW ….WW…WW. ……WW. ……W.. ..W……W.. .W.W…..WW. W.W.W…..W. .W.W……W. ..W…….W.	

Thuật toán:

Tiến hành dùng DFS để đếm số lượng các vùng chứa kí tự ‘W’.

Bài này tương tự bài OCSE, chú ý các đỉnh thuộc cùng TPLT với 1 ô là 8 ô kề cạnh, chung đỉnh.

Chương trình tham khảo:

```
#include <bits/stdc++.h>
using namespace std;

int chuaxet[10004];
void khoitao ()
{
     for (int i=1; i<=10000; i++)
     {
          chuaxet[i]=1;
     }
}
char pond[102][102];
int dinh[102][102];
int R, C;
void nhap ()
{
     int stt=0;
     cin>>R>>C;
     for (int i=1; i<=R; i++)
     {
          for (int j=1; j<=C; j++)
          {
               cin>>pond[i][j];
               stt++;
               dinh[i][j]=stt;     //khoi tao dinh
          }
     }
}
int xqX[]={0, -1, -1, -1, 0, 1, 1, 1};
int xqY[]={1, 1, 0, -1, -1, -1, 0, 1};
void DFS_RA (int r, int c)
{
     for (int i=0; i<8; i++)
```

```
      {
            int X=c+xqX[i];
            int Y=r+xqY[i];
            if ((X>=1 && X<=C) && (Y>=1 && Y<=R) &&
chuaxet[dinh[Y][X]]==1 && pond[Y][X]=='W')
            {
                  chuaxet[dinh[Y][X]]=0;   //Dinh da duyet
                  DFS_RA (Y, X);
            }
      }
}
int main ()
{
      nhap ();
      khoitao();
      int dem=0;
      for (int i=1; i<=R; i++)
      {
            for (int j=1; j<=C; j++)
            {
                  if (pond[i][j]=='W' && chuaxet[dinh[i][j]]==1)
                  {
                        dem++;
                        chuaxet[dinh[i][j]]=0;   //Dinh da duyet
                        DFS_RA (i, j);
                  }
            }
      }
            cout<<dem;
            return 0;
      }
```

Test:
http://www.spoj.com/PTIT/problems/BCLKCOUN/

Bài 8: Bảo vệ nông trang (NKGUARD)

Đề bài:

Nông trang có rất nhiều ngọn đồi núi, để bảo vệ nông trang nông dân John muốn đặt người canh gác trên các ngọn đồi này. Anh ta băn khoăn không biết sẽ cần bào nhiêu người canh gác nếu như anh ta muốn đặt 1 người canh gác trên đỉnh của mỗi đồi. Anh ta có bản đồ của nông trang là một ma trận gồm N (1 < N <= 700) hàng và M (1 < M <= 700) cột. Mỗi phần tử của ma trận là độ cào H_ij so với mặt nước biển (0 <= H_ij <= 10,000) của ô (i, j). Hãy giúp anh ta xác định số lượng đỉnh đồi trên bản đồ. Đỉnh đồi là 1 hoặc nhiều ô nằm kề nhau của ma trận có cùng độ cao được bao quanh bởi cạnh của bản đồ hoặc bởi các ô có độ cao nhỏ hơn. Hai ô gọi là kề nhau nếu độ chênh lệch giữa tọa độ X không quá 1 và chênh lệch tọa độ Y không quá 1.

Input:

- Dòng 1: Hài số nguyên cách nhau bởi dấu cách: N và M

- Dòng 2..N+1: Dòng i+1 mô tả hàng i của mà trận với M số nguyên cách nhau bởi dấu cách: H_ij

Output:

- Dòng 1: Một số nguyên duy nhất là số lượng đỉnh đồi.

Ví dụ:

Input	Output
8 7 4 3 2 2 1 0 1 3 3 3 2 1 0 1 2 2 2 2 1 0 0 2 1 1 1 1 0 0 1 1 0 0 0 1 0 0 0 0 1 1 1 0 0 1 2 2 1 1 0 0 1 1 1 2 1 0	3

Thuật toán

Ý tưởng giải thuật: Ta sẽ làm 2 bước:

Bước 1: Với mỗi đỉnh [i,j] chưa thăm, ta dfs đánh dấu các đỉnh có chiều cao < a[i,j], ta sẽ đảm bảo rằng từ đỉnh có chiều cao a[u,v] nào đó, thủ tục dfs1 sẽ đánh dấu những đỉnh có chiều cao ≤ a[u,v] lận cận;

Như vậy chỉ có các đỉnh có chiều cao "đỉnh" còn lại;

Bước 2: Dfs để tìm các nhóm đỉnh, thực chất Dfs lần này là để đếm số lượng thành phần liên thông.

Bài toán sử dụng DFS trên ma trận 2 chiều như 2 bài trên nhưng mức độ khó hơn.

Chương trình tham khảo:

```
#include <bits/stdc++.h>
using namespace std;
const int Nmax=1010;
const int d[9]={0,-1,-1,-1, 1,1,1, 0,0};
const int c[9]={0,-1, 0, 1,-1,0,1,-1,1};
int n,m,rest=0;
int a[Nmax][Nmax],b[Nmax][Nmax];
void get_data()
{
    int i,j;
    cin>>n>>m;
    for(i=1;i<=n;i++)
        for(j=1; j<=m;j++) cin>>a[i][j];
    for(int u=1;u<=n;u++)
        for(int v=1;v<=m;v++) b[u][v]=1;
}
void DFS1(int x,int y,int s)
{
    for(int i=1;i<=8;i++)
    {
        int u=x+d[i];
        int v=y+c[i];
```

```
            if(0<u && u<=n && 0<v && v<=m &&
               b[u][v] && a[u][v]<=a[x][y] && a[u][v]<s)
            {
                b[u][v]=0;
                DFS1(u,v,s);
            }
        }
    }
    //============
    void DFS2(int x,int y)
    {
        for(int i=1;i<=8;i++)
        {
            int u=x+d[i];
            int v=y+c[i];
            if(0<u && u<=n && 0<v && v<=m && b[u][v])
            {
                b[u][v]=0;
                DFS2(u,v);
            }
        }
    }
    //=================
    void output()
    {
        for(int u=1;u<=n;u++)
            for(int v=1;v<=m;v++)
            if(a[u][v]!=-1) DFS1(u,v,a[u][v]);
        for(int u=1;u<=n;u++)
        for(int v=1;v<=m;v++)
        if(b[u][v])
            {
                DFS2(u,v);
                rest++;
            }
        cout<<rest<<endl;
    }
    int main()
    {
        get_data();
        output();
    }
```

Test:
http://vn.spoj.com/problems/NKGUARD/

II.4 Liệt kê thành phần liên thông

Vùng liên thông trong đồ thị là tập hợp các đỉnh mà từ một đỉnh bất kỳ có đường đi trực tiếp hoặc gián tiếp đến các đỉnh khác trong tập hợp đó.

Bài 9: Các vùng liên thông

Đề bài:

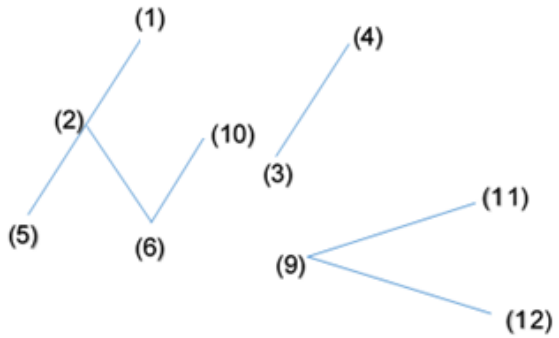
Cho đồ thị vô hướng có N đỉnh, M cạnh. Hãy liệt kê các vùng liên thông trong đồ thị. Trong mỗi vùng các đỉnh được sắp xếp thành dãy tăng. Mỗi vùng liên thông trên một hàng. Mỗi số cách nhau một dấu cách.

Dữ liệu: Vào từ file văn bản Bai9.INP gồm:

- Dòng 1: Ghi số nguyên N và M (M, N ≤ 3000).
- M dòng tiếp theo, mỗi dòng ghi số nguyên dương u và v thể hiện có đường đi giữa hai đỉnh u và v (u, v ≤ N).

Kết quả: Ghi ra file văn bản Bai9.OUT gồm nhiều dòng là các vùng liên thông trong đồ thị. Trong mỗi vùng các đỉnh được sắp xếp thành dãy tăng. Mỗi số cách nhau một dấu cách, mỗi vùng liên thông trên một hàng.

Bai9.INP	Bai9.OUT
12 7 1 2 2 5 2 6 6 10 3 4 9 11 9 12	1 2 5 6 10 3 4 7 8 9 11 12



Thuật toán:

Ta duyệt qua tất cả các đỉnh, nếu i đang free thì ta Duyệt (i).

Thủ tục Duyệt (i) ngoài việc đánh dấu i đã đi đến, ta lưu i vào mảng C có K phần tử.

Mỗi lần Duyệt xong một vùng liên thông, ta đưa vùng đó ra và khởi tạo lại K=0; Duyệt theo chiều sâu luôn tạo ra các vùng liên thông theo thứ tự từ điển. Do đó cần phải sắp xếp lại mảng C. Độ phức tạp thuật toán: O(M + N).

Chương trình tham khảo:

```
#include<bits/stdc++.h>
using namespace std;
int a[3001][3001];
int n,m, Free[3001], u,v;
vector <int> g;
void DFS(int u)
{
    g.push_back(u);
    Free[u]=false;
    for(int v=1;v<=n;v++)
        if (a[u][v]==1&&Free[v])
        DFS(v);
}
int main()
{
    freopen("Bai9.INP","r",stdin);
    freopen("Bai9.out","w",stdout);
    cin>>n>>m;
    for(int i=1;i<=n;i++)
        for(int j=1;j<=n;j++)a[i][j]=0;
    for (int i=1;i<=m;i++)
    {
        cin>>u>>v;
        a[u][v]=1;
        a[v][u]=1;
    }
for(int i=1;i<=n;i++) Free[i]=1;
for (int i=1;i<=n;i++)
{
    g.clear();
    if (Free[i]==1)
    {
        DFS(i);if(i!=1) cout<<endl;
    }
    sort(g.begin(),g.end());
    for (auto i:g) cout<<i<<" ";
}
    return 0;
}
```

Test:
https://drive.google.com/drive/folders/1e6o5X5k0_KY5U8uYdSAuh3OzvIN-8kpM

Bài 10: Tìm vùng liên thông có nhiều đỉnh nhất

Đề bài:

Cho đồ thị vô hướng có N đỉnh, M cạnh. Hãy liệt kê vùng liên thông có nhiều đỉnh nhất trong đồ thị. Mỗi số cách nhau một dấu cách, nếu có nhiều vùng liên thông có cùng số lượng đỉnh nhiều nhất thì đưa ra vùng liên thông có thứ tự từ điển nhỏ nhất.

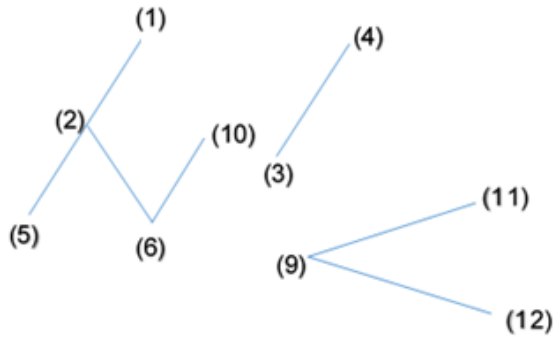
Dữ liệu: Vào từ file văn bản Bai10.INP gồm:

- Dòng 1: Ghi số nguyên N và M (M, N ≤ 3000).
- M dòng tiếp theo, mỗi dòng ghi số nguyên dương u và v thể hiện có đường đi giữa hai đỉnh u và v (u, v ≤ N).

Kết quả: Ghi ra file văn bản Bai10.OUT gồm:

- Dòng 1: Ghi số lượng đỉnh trong vùng liên thông tìm được.
- Dòng 2: Các đỉnh trong vùng liên thông tìm được, được sắp xếp thành dãy tăng. Mỗi số cách nhau một dấu cách.

Bai10.INP	Bai10.OUT
12 7 1 2 2 5 2 6 6 10 3 4 9 11 9 12	5 1 2 5 6 10



Thuật toán:

- Ta duyệt qua tất cả các đỉnh, nếu i đang free thì ta Duyệt (i).
- Thủ tục Duyệt (i) ngoài việc đánh dấu i đã đi đến, ta lưu i vào mảng C có K phần tử.
- Mỗi lần Duyệt xong một vùng liên thông, ta cập nhật lại kmax và lưu mảng C sang mảng D và khởi tạo lại K = 0.
- Ta đưa kmax và mảng D sau khi sắp xếp ra.

```
#include <bits/stdc++.h>
using namespace std;
#define nmax 3001
int n,m,a[3001][3001],u,v,b[100],t=0,maxx=0,s;
bool visit[nmax];
bool check[nmax];
int res=0;
void dfs(int i)
{
    visit[i]=true;
    check[i]=true;
    for(int v=1;v<=n;v++)
    {
        if(a[i][v]==true&&visit[v]==false)
            dfs(v);
```

```
        }
    }
    int main()
    {
        freopen("bai10.inp","r",stdin);
        freopen("bai10.out","w",stdout);
        cin>>n>>m;
        for (int i=1;i<=m;i++)
        {
                cin>>u>>v;
                a[u][v]=1;
                a[v][u]=1;
        }
            vector<int> c;
            for (int i=1;i<=n;i++)
            {
                if (visit[i]==false)
                {
                    res=0;
                    memset(check,0,sizeof(check));
                    dfs(i);
                    for(int j=1;j<=n;j++)
                        if(check[j]) res++;
                    if(maxx<res)
                    {
                        c.clear();
                        maxx=res;
                        for(int f=1;f<=n;f++)
                        {
                            if (check[f])
                            c.push_back(f);
                        }
                    }
                }
            }
            cout<<maxx<<'\n';
            for(int i=0;i<c.size();i++)
                cout<<c[i]<<' ';
            return 0;
    }
```

Test:
https://drive.google.com/drive/folders/1e6o5X5k0_KY5U8uYdSAuh3OzvIN-8kpM

III. Bài tập tự luyện:

Bài 11: Xây cầu.. DFSBRIGE.*

Tại quần đảo ZXY có N hòn đảo, một số hòn đảo đã có cầu nối với nhau. Từ đảo này ta có thể đi sang đảo khác bằng đường đi trực tiếp hoặc đi gián tiếp qua các đảo khác.

Để thuận tiện cho các phương tiện đi lại, ban quản lý sẽ xây thêm một số cầu để từ một đảo ta có thể đi đến các đảo còn lại trong quần đảo.

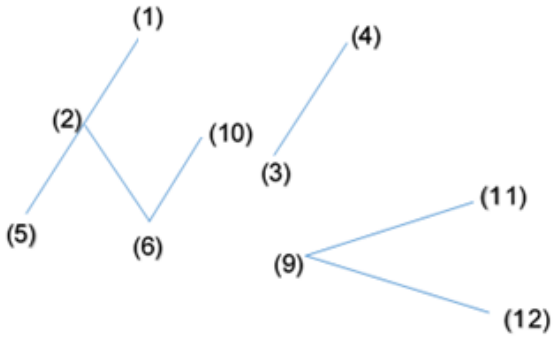
Hãy cho biết, ban quản lý cần xây ít nhất bao nhiêu cầu?

Dữ liệu: Vào từ file văn bản DFSBRIGE.INP gồm:

- Dòng 1: Ghi số nguyên N và M (M, N ≤ 3000).
- M dòng tiếp theo, mỗi dòng ghi số nguyên dương u và v thể hiện có đường đi giữa hai đỉnh u và v (u, v ≤ N).

Kết quả: Ghi ra file văn bản DFSBRIGE.OUT số cầu ít nhất cần xây thêm.

DFSBRIGE.INP	DFSBRIGE.OUT
12 7 1 2 2 5 2 6 6 10 3 4 9 11 9 12	4



Bài 12: Đường đi từ S qua nhiều đảo nhất.. DFSLMAX.*

Tại quần đảo ZXY có N hòn đảo, một số hòn đảo đã có cầu nối với nhau, không có chu trình. Từ đảo này ta có thể đi sang đảo khác bằng đường đi trực tiếp hoặc đi gián tiếp qua các đảo khác.

Jame vừa đến hòn đảo S, anh muốn đi bằng đường bộ đến các đảo khác. Khi đến một đảo, anh ấy có thể nghỉ ngơi ở đảo ấy hoặc đi tiếp sang các đảo khác. Tất nhiên, anh ấy chỉ muốn tham quan ***mỗi đảo một lần.***

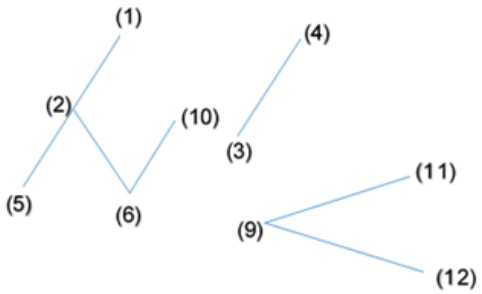
Hãy tìm số đảo nhiều nhất mà Jame có thể tham quan bằng đường bộ.

Dữ liệu: Vào từ file văn bản DFSLMAX.INP gồm:

- Dòng 1: Ghi số nguyên dương N, M và S (M, N ≤ 3000, S ≤ N).
- M dòng tiếp theo, mỗi dòng ghi số nguyên dương u và v thể hiện có đường đi giữa hai đỉnh u và v (u, v ≤ N).

Kết quả: Ghi ra file văn bản DFSLMAX.OUT số đảo nhiều nhất mà Jame có thể tham quan bằng đường bộ.

DFSLMAX.INP	DFSLMAX.OUT
12 7 2 1 2 2 5 2 6 6 10 3 4 9 11 9 12	3 (gồm các đảo 10 6 2)



Bài 13: Kết nối (CONNECT) – Trại hè HV 2015 – K11

Lên LS tham dự trại hè HV, Sơn Tùng gặp lại cô bạn cùng ôn thi đội tuyển năm ngoái. Sau khi hàn huyên đủ thứ, cô bạn muốn Sơn Tùng trợ giúp về vấn đề đang gặp phải.

Tỉnh LS có N thành phố, được đánh số từ 1 đến N. Hai thành phố i và j ($1 \le i, j \le N$) có thể có nhiều nhất một con đường tỉnh lộ hai chiều nối với nhau. Ủy ban nhân dân tỉnh LS quyết định mở thêm một con đường mới nối trực tiếp giữa hai thành phố bất kỳ nào đó trong N thành phố và xây dựng một sân vận động tại một thành phố nào đó với tiêu chuẩn Olympic để tạo điều kiện cho nhân dân luyện tập và thi đấu thể thao.

Cô bạn nhờ Sơn Tùng tính toán xem sân vận động này có thể kết nối nhiều nhất là bao nhiêu thành phố với nhau, biết rằng thành phố định xây sân vận động và những thành phố khác đều có đường đi (trực tiếp hoặc gián tiếp) đến để luyện tập và thi đấu thể thao.

Dữ liệu vào:

- Dòng đầu ghi hai số nguyên N – số thành phố và M – số đường tỉnh lộ nối giữa hai thành phố với nhau.
- M dòng sau, mỗi dòng ghi hai số nguyên dương i và j thể hiện thành phố i có đường tỉnh lộ nối với thành phố j.

Dữ liệu ra: Ghi số nguyên dương là số thành phố lớn nhất mà người dân tại đó có thể tới luyện tập và thi đấu thể thao.

Ví dụ:

Input	output
10 6 1 2 5 4 6 7 10 8 7 8 3 4	7

Ràng buộc: $1 \le N \le 1000$, $0 \le M \le 10000$, $1 \le i, j \le N$

Các số trên cùng một dòng cách nhau bởi một khoảng trắng (space).

Bài 14: NƯỚC BIỂN (BCISLAND)

Đề bài:

Trái đất nóng lên kéo theo mực nước biển dâng. Hòn đảo nhỏ Gonnàsinkà thuê bạn để dự báo trước hiểm hoạ này. Cho trước 1 lưới tọa độ thể hiện cào độ củà đảo, hãy giúp họ tính toán xem nước biển dâng cào bào nhiêu thì hòn đảo sẽ bị chià cắt.

Input:

Input gồm nhiều bộ test, mỗi bộ test bào gồm:

- Dòng đầu ghi 2 số n, m là chiều dài và chiều rộng.
- Sau đó là n dòng, mỗi dòng gồm m số, mỗi số cho biết độ cao củà ô đó, giá trị 0 chỉ mực nước biển. Những ô giá trị 0 dọc theo đường viền và những ô số 0 liền kề những ô này chỉ mặt biển. Những ô có giá trị 0 còn lại (được bao bọc bởi các số > 0) là đất liền bên trong đảo nhứng có độ cao ngang mặt nước biển. Hòn đảo lúc đầu chứà bị chia cắt. Số n và m không lớn hơn 100 và độ cao không lớn hơn 1000.
- Dòng cuối cùng của input chứa 2 số 0

Output:

Với mỗi bộ test, in ra:

Case n: Island splits when ocean raises f feet. (Đảo bị chia khi nước biển dâng cao f feet)

Hoặc:

Case n: Island never splits. (Đảo không bào giờ bị chià cắt)

Ví dụ:

Input	Output
5 5 3 4 3 0 0 3 5 5 4 3 2 5 4 4 3 1 3 0 0 0 1 2 1 0 0 5 5 5 5 5 5 7 4 1 1 1 4 4 1 2 1 3 7 1 0 0 4 7 3 4 4 4 0 0	Case 1: Island never splits. Case 2: Island splits when ocean rises 3 feet.

IV.KẾT LUẬN

DFS còn có khá nhiều ứng dụng khác như liệt kê khớp và cầu đồ thị…Song thời gian và năng lực có hạn nên tôi chưa thể trình bày hết được. Với những nội dung trong tài liệu này tôi hy vọng sẽ giúp thầy cô dễ dàng hơn trong việc giúp học trò tiếp cận với khái niệm duyệt đồ thị_ một khái niệm đầy trừu tượng với học sinh.

Do kinh nghiệm còn chưa nhiều, nên chuyên đề còn có thiếu sót, rất mong các thầy cô góp ý thêm. Tôi xin chân thành cảm ơn.

V.TÀI LIỆU THAM KHẢO

Tài liệu giáo khoa chuyên tin quyển 1.

http://vnoi.info

http://vn.spoj.com

http://ntucoder.net/