CHUYÊN ĐỀ: DFS VÀ CÁC ỨNG DỤNG

1. Giới thiệu

Trong chuyên đề này tôi trình bày về phương pháp duyệt theo chiều sâu (DFS) trên đồ thị và các ứng dụng. Phần lý thuyết về cơ bản đã rất rõ ràng trong cuốn Tài liệu giáo khoa chuyên Tin, tôi trình bày lại theo hướng tiếp cận của cá nhân, tài liệu được dùng để dạy các em học sinh lớp 10 chuyên Tin, mới học đến phần lý thuyết cơ bản về đồ thị.

Đối với học sinh lần đầu tiên tiếp cận với lý thuyết đồ thị sẽ không tránh khỏi những ngỡ ngàng, lạ lẫm, có phần trừu tượng khó hiểu, để giúp đỡ giải quyết vấn đề đó, trong phạm vi chuyên đề chỉ trình bày những ứng dụng cơ bản của DFS và các bài tập cho học sinh luyện tập, các bài tập được sắp xếp theo mức độ từ dễ tới khó, mỗi bài có hướng dẫn sơ lược, có link test, một số bài chỉ để link để bạn đọc tham khảo thêm.

Các bài tập được sưu tầm chủ yếu nguồn trên SPOJ và Uva để thuận tiện cho bạn đọc luyện tập. Code mẫu một phần là của chính tác giả, một phần được tham khảo trên Internet. Do hạn chế về thời gian cũng như trình độ, chuyên đề phản ánh góc tiếp cận của cá nhân trong quá trình dạy học phần DFS nên còn có nhiều thiếu sót, rất mong nhận được ý kiến đóng góp của bạn đọc để hoàn thiện hơn. Xin chân thành cảm ơn.

Nội dung chính

1. GIỚI THIỆU 1

NỘI DUNG CHÍNH 1

2. TỔNG QUAN VỀ CÁC CÁCH BIỂU DIỄN ĐỒ THỊ VÀ PHÉP DUYỆT ĐỒ THỊ THEO CHIỀU SÂU (DFS). 3

2.1. BIỂU DIỄN ĐỒ THỊ BẰNG MA TRẬN KỀ VÀ DANH SÁCH KỀ. 3

2.2. MÔ HÌNH DUYỆT DFS CƠ BẢN: 3

Bài 1. Bài tập (SGK Chuyên Tin _Q1) 4

Đề bài: 4

Thuật toán: 4

Chương trình mẫu: (SGK Chuyên Tin – Q1 – Trang 144) 4

Nhận xét: 4

3. MỘT SỐ ỨNG DỤNG CỦA DFS 4

3.1. Duyệt qua tất cả các đỉnh thuộc đồ thị 4

Bài 2. Chú bò hư hỏng (BCDAISY) 5

Đề bài: 5

Thuật toán 5

Chương trình mẫu: http://ideone.com/tL8Wrz 5

Test: http://www.spoj.com/PTIT/problems/BCDAISY/ 5

Nhận xét 5

3.2. Tìm, đếm thành phần liên thông trên đồ thị vô hướng 6

Bài 3. Robin C11BC2 6

Đề bài: 6

Thuật toán: 7

Link đề và test: http://vn.spoj.com/problems/C11BC2/ 7
Chương trình mẫu: http://ideone.com/YrAqX5 7
Nhận xét: 7
3.3. Đánh số các thành phần liên thông (Floodfill) 7
Bài 4. Ốc sên ăn rau (OCSE) 8
Đề bài 8
Thuật toán 8
Chương trình mẫu: http://ideone.com/O7BNKC 8
Test: http://laptrinh.ntu.edu.vn/Problem/Details/51 8
Nhận xét 8
Bài 5. Đếm ao (BCLKCOUN) 9
Đề bài 9
Thuật toán: 9
Chương trình mẫu: http://ideone.com/8lD8xw 9
Test: http://www.spoj.com/PTIT/problems/BCLKCOUN/ 9
Nhận xét 9
Bài 6. Bảo vệ nông trang (NKGUARD) 10
Đề bài: 10
Thuật toán 10
Chương trình mẫu: http://ideone.com/qfyDb5 10
Test: http://vn.spoj.com/problems/NKGUARD/ 10
Nhận xét 10
Bài 7. (Tự giải) Trò chơi Lines (LINES) 11
Đề bài 11
Test: http://laptrinh.ntu.edu.vn/Problem/Details/107 12
Bài 8. Kết nối (CONNECT) – Trại hè HV 2015 – K11 12
Đề bài 12
Code mẫu 12
Bài 9. Bãi cỏ ngon nhất (VBGRASS) 14
Đề bài: http://vn.spoj.com/problems/VBGRASS/ 14
Code mẫu: http://ideone.com/KJEqYQ 14
Thuật toán: Dùng DFS đếm số lượng TPLT. Bài này cơ bản dễ 14
Bài 10. NƯỚC BIỂN (BCISLAND) 14
Đề bài: 14
Thuật toán: 15
Code mẫu: http://ideone.com/3xsDwK 15
Test: http://www.spoj.com/PTIT/problems/BCISLAND/ 15
Nhận xét: 15
Bài 11. Tính toán lượng nước (PBCWATER) 15
Đề bài: 15
Thuật toán: 16
Code mẫu: http://ideone.com/bHmvZP 17
Test: http://vn.spoj.com/problems/PBCWATER/ 17
Nhận xét: dạng khác của việc dùng DFS: loang theo lớp; đóng băng các vùng đã xử lý. 17
Bài 12. (Tham khảo) UVa 00260 - Il Gioco dell'X 17
Bài 13. (Tham khảo) UVa 00469 - Wetlands of Florida 17
Bài 14. (Tham khảo) UVa 00572 - Oil Deposits 17
Bài 15. (Tham khảo) UVa 00785 - Grid Colouring 17
3.4. Liệt kê khớp, cầu của đồ thị vô hướng 17
Bài 16. Khớp và cầu cơ bản (GRAPH_) 17
Đề bài 17
Thuật toán: tìm khớp cầu cơ bản 19
Code mẫu: http://ideone.com/gtnfCv 19

Test: http://vn.spoj.com/problems/GRAPH_/ 19
Nhận xét: tìm khớp cầu cơ bản 19
Bài 17. Điều kiện thời tiết (WEATHER) 19
Đề bài + Test: http://vn.spoj.com/problems/WEATHER/ 19
Code mẫu: http://ideone.com/DeUfCw 19
Bài 18. Thành phố trọng yếu (CRITICAL) 19
Đề bài + Test: http://vn.spoj.com/problems/CRITICAL/ 19
Code mẫu: http://ideone.com/CmciaP 19
Bài 19. Phương án bắn pháo (BCACM11E) 19
Đề bài + Test: http://www.spoj.com/PTIT/problems/BCACM11E/ 19
Code mẫu: http://ideone.com/Ge9ggu 19
Bài 20. Mạng máy tính an toàn (SAFENET2) 19
Đề bài + Test: http://vn.spoj.com/problems/SAFENET2/ 19
Code mẫu: http://ideone.com/zOc7zp 19
Bài 21. (Tham khảo) UVa 00315 – Network 19
Bài 22. (Tham khảo) UVa 00610 - Street Directions (finding bridges) 19
Bài 23. (Tham khảo) UVa 00796 - Critical Links * (finding bridges) 19
Bài 24. (Tham khảo) UVa 10199 - Tourist Guide (finding articulation points) 19
Bài 25. (Tham khảo) UVa 10765 - Doves and Bombs * (finding articulation points) 19
3.5. Tìm thành phần liên thông mạnh trên đồ thị có hướng 19
Bài 26. (Tham khảo) Tìm thành phần liên thông mạnh (TJALG) 19
Bài 27. (Tham khảo) Truyền tin (MESSAGE) 19
Bài 28. (Tham khảo) Biến đổi số (NUMBER) 19
Bài 29. (Tham khảo) Cho kẹo hay bị phá nào (TREAT) 19
Bài 30. (Tham khảo) UVa 11838 - Come and Go (check if graph is strongly connected) 19
Bài 31. (Tham khảo) UVa 00247 - Calling Circles * (SCC + printing solution) 20
Bài 32. (Tham khảo) UVa 11709 - Trust Groups (find number of SCC) 20

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

2.1. Biểu diễn đồ thị bằng ma trận kề và danh sách kề.

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,…) đồng thời biết cách biểu diễn đồ thị cơ bản (Biểu diễn bằng ma trận kề/ ma trận trọng số, danh sách kề)

2.2. Mô hình duyệt DFS cơ bản:

```
Procedure DFS(u:longint); // Đỉnh xuất phát là u
// mảng D dùng đánh dấu các đỉnh 0-chưa thăm; 1-đã thăm
Begin
      D[u]:=1; // Đánh dấu u đã thăm
      Write(u,' '); // In ra đỉnh u
      For (mọi v kề u) do
            If d[v]=0 then DFS(v); // Nếu v đến được từ u mà v chưa thăm thì Thăm u
End;
```

Bài 1. Bài tập (SGK Chuyên Tin _Q1)

Đề bài:

Cho một đồ thị có hướng G(V,E), N đỉnh, M cạnh ($(N \leq 10^5, M \leq 10^6)$ các đỉnh được đánh số từ 1 tới n và được đồng nhất với số hiệu của chúng. Khuôn dạng Input, Output được quy định như sau:

Input:

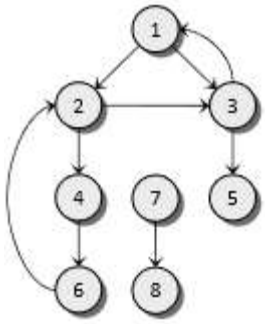
- Dòng 1: chứa số đỉnh N, đỉnh xuất phát S
- N dòng tiếp theo, dòng thứ i chứa một danh sách các đỉnh, mỗi đỉnh j trong danh sách tương ứng với một cung (i,j) của đồ thị, ngoài ra còn có thêm số 0 ở cuối dòng để báo hiệu kết thúc.

Ouput:

- Danh sách các đỉnh có thể đến được từ s

Ví dụ:

Input	Output
8 1 5 2 3 0 3 4 0 1 5 0 6 0 0 2 0 8 0 0	1 2 3 4 5 6



Thuật toán:

Tiến hành duyệt DFS từ đỉnh S, trong quá trình duyệt, tại mỗi đỉnh in ra số hiệu của đỉnh hiện tại.

Chương trình mẫu: (SGK Chuyên Tin – Q1 – Trang 144)

Nhận xét:

Bài này hết sức cơ bản, mục đích để cho Hs hiểu được hoạt động của thủ tục DFS. Tiến hành thăm các đỉnh đến được từ một đỉnh.

3. Một số ứng dụng của DFS

3.1. Duyệt qua tất cả các đỉnh thuộc đồ thị

Với thủ tục duyệt DFS như trên, muốn duyệt qua tất cả các đỉnh của đồ thị ta tiến hành như sau trong chương trình chính:

```
Begin
      For i:=1 to N d[i]:=0; // Đánh dấu tất cả các đỉnh đều chưa thăm
      For i:=1 to N do
            If d[i]=0 then DFS(i);
End.
```

Bài 2. Chú bò hư hỏng (BCDAISY)

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

Chương trình mẫu: http://ideone.com/tL8Wrz

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

Nhận xét

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

3.2. Tìm, đếm thành phần liên thông trên đồ thị vô hướng

Với thủ tục DFS như trên, để đếm số thành phần liên thông trên đồ thị vô hướng, ta duy trì biến đếm DemTPLT để lưu số lượng TPLT, mỗi lần gọi thủ tục DFS ta sẽ tăng DemTPLT

```
Begin
      Demtplt:=0;
      For i:=1 to N d[i]:=0; // Đánh dấu tất cả các đỉnh đều chưa thăm
      For i:=1 to N do
            If d[i]=0 then begin
Inc(demtplt);
Write('TPLT thu ',i,' gom cac dinh: ');
DFS(i);
Writeln;
End.
```

Bài 3. 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.

Input:

- 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)).

Output: 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	YES YES NO NO NO

5 2 1 3 5 1 2 1 2 1 1 2 2 4 1 2	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 thuật toán BFS hoặc 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.

Link đề và test: http://vn.spoj.com/problems/C11BC2/

Chương trình mẫu: http://ideone.com/YrAqX5

Nhận xét:

Bài cơ bản áp dụng DFS để đánh dấu các thành phần liên thông, có sử dụng thêm một chút khéo léo trong quá trình xử lý.

3.3. Đánh số các thành phần liên thông (Floodfill)

Thủ tục DFS còn được sử dụng trong nhiều bài toán gán nhãn các thành phần (floodfill) như sau: Cho đồ thị, tô màu giống nhau cho các đỉnh cùng thành phần liên thông (TPLT 1 được gán nhãn 1 – màu 1; TPLT 2 được gán nhãn 2 – màu 2....). Thủ tục DFS sẽ được cải tiến thành thủ tục **tomau** dưới đây. Sau khi thực hiện chương trình; ở mảng đánh dấu d, các đỉnh thuộc TPLT 1 sẽ có giá trị d[i]=1; các TPLT 2 sẽ có giá trị d[i]=2, ...

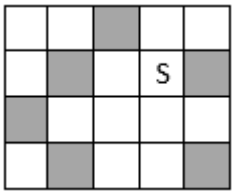
```
Procedure tomau(u:longint; color: longint); // Tô đỉnh u và các đỉnh cùng TPT với u màu color
Begin
      D[u]:=color;
      For (mọi v kề u) do
            If d[v]=0 then tomau(v,color);
End;
BEGIN
      Demtplt:=0;
      For i:=1 to N do d[i]:=0;
      For i:=1 to N do
            If d[i]=0 then begin
                  Inc(demtplt);
                  Tomau(i,demtplt);
END.
```

Đây là một kỹ thuật rất quen thuộc và gặp ở nhiều bài toán.

Bài 4. 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 ≤ y ≤ n ≤ 100,1 ≤ x ≤ m ≤ 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 xuất:

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

Ví dụ:

input
4 5 2 4 0 0 1 0 0 0 1 0 0 1 1 0 0 0 0 0 1 0 0 1
output
10

Thuật toán

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 mẫu: http://ideone.com/O7BNKC

Test: http://laptrinh.ntu.edu.vn/Problem/Details/51

Nhận xét

Bài OCSE là dạng đơn giản, áp dụng DFS để loang trên ma trận.

Bài 5. 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 sàu đó vài ngày, trận lụt khủng khiếp đã diễn rà 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 chià 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 W……..WW. .WWW…..WWW ….WW…WW. ……..WW. ……..W.. ..W……W.. .W.W….WW. W.W.W…..W. .W.W……W. ..W……..W.	3

Thuật toán:

Tiến hành dùng DFS để đếm số lượng các vùng chứa kí tự 'W'

Chương trình mẫu: http://ideone.com/8lD8xw

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

Nhận xét

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

Bài 6. 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

Từ 1 ô chưa duyệt, ta đi đến những ô có cùng độ cao với nó và đánh dấu lại. Nếu gặp phải một ô có độ cao lớn hơn thì đánh dấu đây không phải là đỉnh đồi.

Chương trình mẫu: http://ideone.com/qfyDb5

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

Nhận xét

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

Bài 7. (Tự giải) Trò chơi Lines (LINES)

Đề bài

Trò chơi Line là trò chơi di chuyển các viên bi trong một hình vuông 9 x 9 ô. Bi được ăn khi tạo thành các hàng, cột, đường chéo gồm 5 viên bi liên tiếp cùng màu.

Một thuật toán được sử dụng trong trò chơi là tìm đường đi để di chuyển một viên bi. Giả sử trò chơi Line tổng quát có n dòng, n cột. Đánh số các dòng từ 1 đến n theo thứ tự từ trên xuống dưới, đánh số các cột từ 1 đến n theo thứ tự từ trái sang phải. Giả sử có một viên bi tại ô (y, x) - dòng y cột x, bi có thể di chuyển đến 1 trong 4 ô (y+1, x), (y-1, x), (y, x+1), (y, x-1), nếu ô đó đang trống. Cho vị trí bắt đầu và vị trí kết thúc của viên bi, hãy viết chương trình xác định xem có tồn tại đường đi để di chuyển viên bi hay không.

Input: gồm các dòng sau

- Dòng thứ nhất gồm năm số *n*, *sy*, *sx*, *dy*, *dx*, mỗi số cách nhau một khoảng trắng ($2 \le n \le 9$; $1 \le sy, sx, dy, dx \le n$). *sy* là chỉ số dòng, *sx* là chỉ số cột của viên bi cần di chuyển. *dy* là chỉ số dòng, *dx* là chỉ số cột của vị trí cần di chuyển viên bi đến.
- Trong *n* dòng tiếp theo, mỗi dòng gồm *n* số nguyên 0 hoặc 1, mỗi số cách nhau một khoảng trắng, biểu thị tình trạng của trò chơi. Số 1 nghĩa là vị trí ô đó có bi, số 0 nghĩa là vị trí ô đó đang trống.

(Dữ liệu cho bảo đảm tại ô (*sy*, *sx*) có giá trị là 1, tại ô (*dy*, *dx*) có giá trị là 0)

Output:

- Nếu tìm được đường đi, in ra YES.
- Nếu không tìm được đường đi, in ra NO.

Ví dụ:

Input	**Output**
2 1 1 2 2 1 0 1 0	YES
2 1 1 2 2 1 1 1 0	NO
3 1 1 3 3 1 0 1 1 0 0 1 1 0	YES

Test: http://laptrinh.ntu.edu.vn/Problem/Details/107

Bài 8. Kết nối (CONNECT) – Trại hè HV 2015 – K11

Đề bài

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 ≤ i, j ≤ 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:

- 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**.

Kết quả: 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 ≤ N ≤ 1000, 0 ≤ M ≤ 10000, 1 ≤ i, j ≤ N

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

Code mẫu

```
const
  fi='connect.inp';
  fo='connect.out';
  maxn=100000;

Type  banghi=record
    d,c:longint;
    end;

var adj:array[1..2*maxn] of longint;
    lt,head:array[1..maxn+1] of longint;
    e:array[1..2*maxn] of banghi;
    kt:array[1..maxn] of boolean;
    m,n,kq,max1,max2,stplt:longint;

procedure docdl;
var f:text;
    i:longint;
```

```
Begin
  Assign(f,fi);
  Reset(f);
  Readln(f,n,m);
  for i:=1 to m do
    begin
      readln(f,e[i].d,e[i].c);
      e[m+i].d:=e[i].c;
      e[m+i].c:=e[i].d;
    end;
  m:=2*m;
  for i:=1 to n do head[i]:=0;
  for i:=1 to m do
    with e[i] do
        head[d]:=head[d]+1;
  for i:=2 to n do  head[i]:=head[i-1]+head[i];
  for i:=m downto 1 do
    with e[i] do
    Begin
      adj[head[d]]:=c;
      dec(head[d]);
    End;
  head[n+1]:=m;
  close(f);
End;

function dfs(u:longint):longint;
var v:longint;
Begin
  inc(lt[stplt]);
  kt[u]:=false;
  for v:=head[u]+1 to head[u+1] do
    if kt[adj[v]] then dfs(adj[v]);
End;

procedure xuli;
var i:longint;
Begin
  fillchar(kt,sizeof(kt),true);
  fillchar(lt,sizeof(lt),0);
  stplt:=0;
  for i:=1 to n do
    if kt[i] then
    Begin
      inc(stplt);
      dfs(i)
    end;

    max1:=0;
    max2:=0;
    for i:=1 to stplt do
      if lt[i]>max1 then
        begin
          max2:=max1;
          max1:=lt[i];
        end
      else
        if max2<lt[i] then max2:=lt[i];
End;

procedure ghikq;
var    f:text;
Begin
  Assign(f,fo);
  Rewrite(f);
  Write(f,max1+max2);
```

```
  close(f);
End;

BEGIN
  Docdl;
  Xuli;
  Ghikq;
END.
```

Bài 9. Bãi cỏ ngon nhất (VBGRASS)

Đề bài: http://vn.spoj.com/problems/VBGRASS/

Code mẫu: http://ideone.com/KJEqYQ

Thuật toán: Dùng DFS đếm số lượng TPLT. Bài này cơ bản dễ.

Bài 10. 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 họa này. Cho trước 1 lưới tọa độ thể hiện cào độ của đảo, hãy giúp họ tính toán xem nước biển dâng cào bào nhiêu thì hòn đảo sẽ bị chia 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 của 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 nhưng có độ cao ngang mặt nước biển. Hòn đảo lúc đầu chưa 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 chia 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.

Thuật toán:

Dùng DFS đánh dấu các ô có độ cao <=L ta được mảng đánh dấu d; với mỗi độ cao x, đếm số lượng TPLT là các đỉnh có độ cao >x, nếu số lượng TLPT >1 thì đảo bị chia cắt, lúc này in ra kết quả và thoát khỏi chương trình.

Code mẫu: http://ideone.com/3xsDwK

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

Nhận xét:

Bài này khá hay, yêu cầu HS nhận biết được dùng DFS để đánh dấu, đếm số lượng TPLT trên mảng đánh dấu. Chú ý : xử lý khéo để có được mảng đánh dấu

Bài 11. Tính toán lượng nước (PBCWATER)

Đề bài:

Nền phẳng của 1 công trình xây dựng được chia thành lưới ô vuông đơn vị kích thước MxN ô. Trên mỗi ô (i,j) của lưới, người ta dựng 1 cột bê tông hình hộp có đáy là ô (i,j) và chiều cao là h[i,j] đơn vị. Sau khi dựng xong thì trời đổ mưa to và đủ lâu. Nhà thầu xây dựng muốn tính lượng nước đọng lại giữa các cột để có kế hoạch thi công tiếp theo. Giả thiết, nước ko thẩm thấu qua các cột bê tông cũng như ko rò rỉ qua các đường ghép giữa chúng.

Nhiệm vụ của bạn là giúp nhà thầu tính toán lượng nước đọng lại giữa các cột.

Input:

- Dòng đầu tiên ghi 2 số nguyên dương M và N
- Dòng thứ i trong M dòng tiếp theo, ghi N số nguyên dương h[i,1],h[i,2]…h[i,N].

Output

- 1 dòng duy nhất chứa số đơn vị khối nước đọng lại.

Ví dụ:

Input:
5 5
9 9 9 9 9
9 2 2 2 9
9 2 5 2 9
9 2 2 2 9
9 9 9 9 9

Output:
60

Giới hạn:
1<=M,N<=100,1<=H[i,j]<=1000

Thuật toán:

Gọi a là tấm bảng nền mô hình với kích thước n*m và a(i,j) là chiều cao của cột nhựa vuông đặt trên ô (i,j). Thoạt tiên ta giả sử chiều cao tối đa của các cột là 1, như vậy các ô trên nền nhà chỉ chứa các giá trị 0 hoặc 1.

1	1	1	1	1	1	1
1	1	0	0	0	0	0
1	1	1	1	0	1	1
1	0	1	1	0	1	1
1	0	1	1	1	1	1
1	0	1	0	0	1	1
1	1	1	1	1	1	1

Nền tầng 1 với các ô đặc mang giá trị 1, ô chứa nước đọng màu trắng (0) và ô thoát nước màu xám (0).
Có 5 ô trắng bị giam nên thể tích nước đọng sẽ là 5.
Các ô liên thông với ô trống trên biên sẽ tạo thành nột cống thoát nước ra ngoài mô hình. Ô mang giá trị 0 chính là mặt sàn, còn ô mang giá trị 1 chính là cột. Bây giờ bạn thử rót nước vào mô hình đơn giản này cho ngập các ô và quan sát điều gì sẽ xảy ra. Dễ thấy là chỉ có các ô trống bị giam tại giữa mô hình là còn chứa nước. Có 5 ô trống bị giam nên thể tích nước đọng sẽ là 5. Các ô trống (mang giá trị 0) liên thông cạnh với một ô trống trên biên sẽ tạo thành một cống thoát nước ra ngoài mô hình. Nhận xét này cho ta thuật toán tính lượng nước đọng tại tầng nền (tầng 0) của mô hình như sau: Duyệt đường biên, nếu gặp ô trống (trên biên và chứa trị 0) thì gọi thủ tục loang đánh dấu các ô này bằng giá trị 1. Duyệt các ô lọt trong mô hình, đếm các ô trống (chứa giá trị 0) và đồng thời đánh dấu các ô này bằng giá trị 1. Đó là thể tich nước đọng. Tiếp theo, sau khi đã

duyệt và tính lượng nước đọng tại tầng nền (tầng 0), bạn dễ dàng suy ra cách tính lượng nước đọng tại tầng 1 nếu như coi các ô trống đã duyệt tại tầng 0 đã được lấp bằng các khối nhựa chiều cao 1. Ta gọi phương thức này là đóng băng các ô đã xử lí. Tổng quát, tại tầng h ta thực hiện thủ tục tương tự như tại tầng nền với chút điều chỉnh là thay giá trị đánh dấu bằng h+1 thay vì bằng 1. Gọi chiều cao tối đa của các cột là hmax, cho h biến thiên từ 0 đến hmax – 1 ta tính được tổng khối nước đọng của mô hình. Để thực hiên thuật toán loang bạn nhớ khởi trị mọi ô của nền nhà là -1.

Thủ tục **Loang(u, v, h)**

1. Xét giá trị của ô (i,j):
Nếu a(i,j) = h thì1.1 Thay a(i,j) bằng h+1;1.2 Loang tiếp sang 4 ô kề: (i,j+1), (i,j-1), (i+1,j) và (i-1,j) 2. end.

Code mẫu: http://ideone.com/bHmvZP

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

Nhận xét: dạng khác của việc dùng DFS: loang theo lớp; đóng băng các vùng đã xử lý.

Bài 12. (Tham khảo) UVa 00260 - Il Gioco dell'X

Bài 13. (Tham khảo) UVa 00469 - Wetlands of Florida

Bài 14. (Tham khảo) UVa 00572 - Oil Deposits

Bài 15. (Tham khảo) UVa 00785 - Grid Colouring

3.4. Liệt kê khớp, cầu của đồ thị vô hướng

Một ứng dụng khác của DFS là kiểm tra đỉnh khớp, cạnh cầu của đồ thị bằng cách khảo sát giá trị Low, num của các đỉnh trong quá trình duyệt DFS. Chương trình dưới đây sẽ cho phép liệt kê các khớp, cầu của đồ thị nhập vào – bài GRAPH_

Bài 16. Khớp và cầu cơ bản (GRAPH_)

Đề bài

Xét đơn đồ thị vô hướng G = (V, E) có n(1<=n<=10000) đỉnh và m(1<=m<=50000) cạnh. Người ta định nghĩa một đỉnh gọi là khớp nếu như xoá đỉnh đó sẽ làm tăng số thành phần liên thông của đồ thị. Tương tự như vậy, một cạnh được gọi là cầu nếu xoá cạnh đó sẽ làm tăng số thành phần liên thông của đồ thị.

Vấn đề đặt ra là cần phải đếm tất cả các khớp và cầu của đồ thị G.

Input:

- Dòng đầu: chứa hai số tự nhiên n,m.

- M dòng sau mỗi dòng chứa một cặp số (u,v) (u<>v, 1<=u<=n, 1<=v<n) mô tả một cạnh của G.

Output:

- Gồm một dòng duy nhất ghi hai số, số thứ nhất là số khớp, số thứ hai là số cầu của G

Ví dụ:

> **Input:**
>
> 10 12
> 1 10
> 10 2
> 10 3
> 2 4
> 4 5
> 5 2
> 3 6
> 6 7
> 7 3
> 7 8
> 8 9
> 9 7
>
> **Output:**
> 4 3

Thuật toán: tìm khớp cầu cơ bản

Code mẫu: http://ideone.com/gtnfCv

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

Nhận xét: tìm khớp cầu cơ bản

Bài 17. Điều kiện thời tiết (WEATHER)

Đề bài + Test: http://vn.spoj.com/problems/WEATHER/

Code mẫu: http://ideone.com/DeUfCw

Bài 18. Thành phố trọng yếu (CRITICAL)

Đề bài + Test: http://vn.spoj.com/problems/CRITICAL/

Code mẫu: http://ideone.com/CmciaP

Bài 19. Phương án bắn pháo (BCACM11E)

Đề bài + Test: http://www.spoj.com/PTIT/problems/BCACM11E/

Code mẫu: http://ideone.com/Ge9ggu

Bài 20. Mạng máy tính an toàn (SAFENET2)

Đề bài + Test: http://vn.spoj.com/problems/SAFENET2/

Code mẫu: http://ideone.com/zOc7zp

Bài 21. (Tham khảo) UVa 00315 – Network

Bài 22. (Tham khảo) UVa 00610 - Street Directions (finding bridges)

Bài 23. (Tham khảo) UVa 00796 - Critical Links * (finding bridges)

Bài 24. (Tham khảo) UVa 10199 - Tourist Guide (finding articulation points)

Bài 25. (Tham khảo) UVa 10765 - Doves and Bombs * (finding articulation points)

3.5. Tìm thành phần liên thông mạnh trên đồ thị có hướng

Bài 26. (Tham khảo) Tìm thành phần liên thông mạnh (TJALG)

Bài 27. (Tham khảo) Truyền tin (MESSAGE)

Bài 28. (Tham khảo) Biến đổi số (NUMBER)

Bài 29. (Tham khảo) Cho kẹo hay bị phá nào (TREAT)

Bài 30. (Tham khảo) UVa 11838 - Come and Go (check if graph is strongly connected)

Bài 31. (Tham khảo) UVa 00247 - Calling Circles * (SCC + printing solution)

Bài 32. (Tham khảo) UVa 11709 - Trust Groups (find number of SCC)