Chương II :

ĐỒ THỊ

Phần 1 :

Các Khái Niệm Cơ Bản

I. Định Nghĩa Đồ Thị :

- Đơn đò thị vô hướng G =(V,E) bao gồm V là tập các đỉnh , và E là tập các cặp không có thứ tự bao gồm 2 phần tử khác nhau của V gọi là các cạnh
- Đa đồ thị vô hướng G=(V,E) bao gồm V là tập các đỉnh , E là họ các cặp không có thứ tự gồm 2 phần tử khác nhau của V gọi là các cạnh . Hai cạnh e1 , e2 được gọi là cạnh lặp nếu chúng cùng tương ứng với một cặp đỉnh .
- Giả đồ thị vô hướng G=(V,E) bao gồm V là tập các đỉnh , và E là họ các cặp không có thứ tự gồm 2 phần tử (không nhất thiết phải khác nhau) của V gọi là các cạnh . Cạnh e được gọi là khuyên nếu nó có dạng e=(u,u).
- Ta gọi bậc của đỉnh v trong đồ thị vô hướng là số cạnh liên thuộc với nó và sẽ kí hiệu là deg(v) .
- Trong một đồ thị vô hướng với m cạnh khi đó $2m=\sum deg(v)$. tức là ta sẽ có trong một đồ thị vô hướng , số đỉnh bậc lẻ là một số chẵn .

II. Đường đi , chu trình , liên thông :

- Đường đi độ dài n từ đỉnh u đến đỉnh v , trong đó n là số nguyên dương , trên đò thị vô hướng G=(V,E) là dãy : x0,x1,...xn trong đó u =x0 , v=xn . Đường đi này đi qua các cạnh : (x0,x1),(x1,x2),...(xn-1,xn). u và v gọi là đỉnh đầu và đỉnh cuối .
- Một đường đi có đỉnh đầu trùng với đỉnh cuối thì gọi là chu trình . Nếu trên đường đi đó không có cạnh nào lặp lại hai lần thì gọi là chu trình đơn .
- Đồ thị vô hướng G=(V,E) được gọi là liên thông nếu luôn tìm được đường đi giữa hai đỉnh bất kì của nó .
- Đỉnh v được gọi là đỉnh rẽ nhánh nếu việc loại bỏ v cùng với các cạnh liên thuộc với nó khỏi đồ thị làm tăng số thành phần liên thông của đồ thị . Cạnh e được gọi là cầu nếu loại bỏ nó ra khỏi đồ thị làm tăng số thành phần liên thông của đồ thị .
- Đồ thị có hướng G=(V,E) được gọi là liên thông mạnh nếu luôn tìm được đường đi giữa 2 đỉnh bất kỳ của nó .

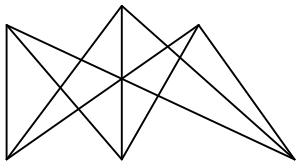
- Đồ thị vô hướng G=(V,E) được gọi là liên thông yếu nếu đồ thị vô hướng tương ứng với nó là đồ thị vô hướng liên thông
- Đồ thị vô hướng liên thông là định hướng được khi và chỉ khi mỗi cạnh của nó nằm trên ít nhất một chu trình
- Đơn đồ thị G=(V,E) được gọi là đồ thị 2 phía nếu như tập đỉnh V của nó có thể phân hoạch thành 2 tập X và Y sao cho mỗi cạnh của đồ thị chỉ nối một đỉnh nào đó trong X với một đỉnh nào đó trong Y . Khi đó ta sẽ sử dụng kí hiệu (X∪Y,E) để chỉ đồ thị 2 phía với tập đỉnh X∪Y .

Đơn đồ thị 2 phía khi và chỉ khi nó không chia chứa chu trình độ dài lẻ.

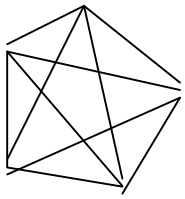
- **Đồ thị Phẳng** là đồ thị nếu ta có thể vẽ nó trên mặt phẳng sao cho các cạnh của nó không cắt nhau ngoài ở đỉnh . Cách vẽ như vậy được gọi là biểu diễn phẳng của đồ thị .

  Ta gọi một phép chia cạnh (u,v) của đồ thị là việc loại bỏ cạnh này khỏi đồ thị và thêm vào đồ thị một đỉnh mới W cùng với 2 cạnh (u,w),(w,u) . Hai đồ thị G=(V,E) và H=(W,F) được gọi là đồng cấu nếu chúng có thể thu được từ cùng 1 đồ thị nào đó nhờ các phép chia cạnh

  Định Lý Kuratovski : Đồ thị là phẳng khi và chỉ khi nó không chứa đồ thị con đồng cấu K3,3 hoặc K5 :



Đồ Thị $K_{3,3}$



Đồ thị K_5

  Công thức euler : Giả sử G là đồ thị phẳng liên thông với N đỉnh , M cạnh . Gọi r là số miền của mặt phẳng bị chia bởi biểu diễn mặt phẳng của G . Khi đó : r = m + 2 – n .

III. Đồ Thị Euler :

- Chu Trình đơn trong G đi qua mỗi cạnh của nó một lần được gọi là chu trình Euler . Đường đi đơn G đi qua mỗi cạnh của nó 1 lần được gọi là đường đi Euler . Đồ thị được gọi là đồ thị Euler nếu nó có chu trình Euler và gọi là đồ thị nửa Euler nếu nó có đường đi Euler .
- Đồ thị vô hướng liên thông G là đồ thị Euler khi và chỉ khi mọi đỉnh của G đều là bậc chẵn .
- Đồ thị nửa Euler là đồ thị có không quá 2 đỉnh bậc lẻ .

Ta sẽ có thuật toán tìm chu trình Euler trong một đồ thị G :

```
Procedure        cycle;
```

```
Var
        dem1 , dem2 :  integer ;
        Ok          :       boolean ;
        i , j :  integer ;
        Begin
                dem1 :=1 ; stack[1]:=1 ;
                dem2 := 0 ;
                while dem1<>0 do
                Begin
                        i := stack [ dem1 ] ;
                        Ok:=true ;
                        For j := 1 to n do
                         if a[i,j]=1 then
                         begin ok := false ; break ; end ;
                        If not ok then
                        begin
                                inc ( dem1 ) ; stack [ dem1]:= j;
                                a[i,j]:=1 ; a[j,i]:=1;
                        end
                        else
                        begin
                                dec ( dem1 ) ;
                                Inc (dem2 ) ; e [ dem2 ]:=stack[dem1+1];
                        end ;
                End ;
        End ;
```

IV. Đồ thị Hamilton :

- Đường đi qua tất cả các đỉnh của đồ thị , mỗi đỉnh đúng 1 lần được gọi là đường đi Hamilton . Chu trình bắt đầu từ 1 đỉnh v nào đó đến tất cả các đỉnh còn lại , mỗi đỉnh đúng 1 lần rồi qua lại đỉnh đó thì gọi là chu trình Hamilton . Đồ thị G được gọi là đồ thị Hamilton nếu nó chứa chu trình hamilton và gọi là nửa chu trình Hamilton nếu nó chứa đường đi Hamilton .
- Đơn đồ thị vô hướng G với N >2 đỉnh , mỗi đỉnh có bậc không nhỏ hơn n/2 là đồ thị Hamilton.
- Không có một thuật toán nào tối ưu mà có thể tìm được chu trình Hamilton . Chính vì thế giải quyết nó chỉ là phương pháp duyệt .

V. ứng Dụng

Bài toán 1 : Đồ Thị Song Liên Thông

Đề Bài :

Cho một đồ thị vô hướng có N đỉnh , đồ thị được gọi là song liên thông nếu như phá bất kì đỉnh nào của đồ thị cũng không đánh mất tính liên thông của nó . Hãy tìm đồ thị con song liên thông với số đỉnh nhiều nhất của đồ thị đã cho .

Dữ Liệu : Vào từ File SLT.TXT có cấu trúc như sau :

- Dòng đầu tiên ghi sô N (N <=100)
- Các dòng tiếp theo , mỗi dòng chứa 2 số (x , y) thể hiện 1 cạnh của đồ thị

Kết Qủa : Ghi vào file SLT.OUT như sau :

- Dòng đầu tiên chứa số K (số đỉnh của đồ thị con)
- Dòng thứ hai chứa K số thể hiện các đỉnh của đồ thị con .

Ví dụ :

Hướng Dẫn :

Chúng ta sẽ giải quyết tổng quát cho đồ thị có đường đi một chiều . Nếu giải được nó thì bài toán trên sẽ hoàn toàn được giải .

Bài toán 2 : Thành phần liên thông mạnh

Đề bài :

Cho đồ thị có hướng có N đỉnh , và M cung một chiều . Một thành phần liên thông mạnh của đồ thị là một tập các đỉnh sao cho hai đỉnh u , v thuộc tập đều có thể đi từ u đến v bằng các đường một chiều , đồng thời nếu thêm vào bất kì một đỉnh nào khác ngoài tập hợp thì điều kiện vừa nêu không còn đúng . Yêu cầu chỉ ra các thành phần liên thông mạnh của đồ thi

Dữ Liệu :

- N , M (N <= 100 , M <= N*N)
- M dòng tiép theo , mỗi dòng ghi hai số u , v cho biết có cung 1 chiều nối u đến v.

Kết Quả :

- K số thành phần song liên thông .
- K dòng , mỗi dòng ghi danh sách các đỉnh thuộc một thành phần liên thông mạnh.

Hướng Dẫn :

Thuật toán Tarjan : Ta duyệt theo chiều sâu các đỉnh , với mỗi đỉnh ta lưu 2 số :

Low(u) và Number(u) , trong đó Number sẽ là thứ tự duyệt đỉnh u còn Low sẽ bằng Number nhỏ nhất trong các đỉnh thuộc thành phần liên thông mạnh với u , tức là đực duyệt đầu tiên trong các đỉnh thuộc liên thông mạnh.

Tại bước duyệt dỉnh u :
Khởi tạo Low (u) = Number(u) = thứ tự duyệt
đẩy u vào stack
For các v mà u có thể tới đực :
{
nếu v chưa thăm thì thăm(v);
Low(u) = min(Low(u) , Low(v))
}
nếu sau khi thăm xong toàn bộ u mà Low(u) = Number(u) thì u là đỉnh tổ tiên của thành phần son liên thông mạnh , ta chỉ việc tìm lại các đỉnh trong stack mà là con của u và chưa được giải phóng . Ngược lại giải phóng stack

Bài toán 3 : Tạo Đồ Thị

Đề Bài :
Chúng ta biết được rằng , bậc của một đỉnh trong đồ thị bằng số cạnh nối đến nó trong đồ thị . Khi biết được đồ thị , thì chúng ta xác định được bảng Deg(i) là bảng bậc của các đỉnh . Yêu cầu của bài toán đặt ra là : Khi chúng ta xác định được một bảng danh sách các Deg(i) thì hãy xác định một đồ thị thoả mãn .
Dữ Liệu : Vào cho từ file : Deg.Inp như sau :

- Dòng đầu tiên ghi số N (N <=100)
- N dòng sau , mỗi dòng ghi 1 số biểu diễn bậc của đỉnh đó

Kết Quả : Ghi vào file : Deg.Out như sau :

- Dòng tiên ghi số 1 hoặc 0 (1 nếu có đồ thị thoả mãn , 0 thì ngược lại)
- Nếu có kết quả thì N dòng sau ,mỗi dòng ghi N số biểu diễn ma trận kề biểu diễn đồ thị đó .

Ví Dụ :
Hướng dẫn :
Sắp xếp thứ tự các đỉnh theo thứ tự giảm dần . Sau đó :

- Chắc chắn tổng số bậc phải là một số chẵn , Nếu không thì không có đồ thị nào thoả mãn .
- Tìm hai đỉnh mà thoả mãn bậc của chúng lớn hơn 1 , và chúng chưa được nối với nhau thì xác định cạnh nối hai đỉnh đó và giảm bậc mỗi đỉnh .
- Nếu quá trình trên hết mà tất cả các đỉnh đều về bậc không thì đò thị có các cạnh chính là đồ thị cần tìm
- Nếu có hai đỉnh i và j mà bậc của chúng lớn hơn không mà đã được nối với thì chúng ta thực hiện như sau : Tìm đường đi từ i đến j sao cho : trên đường đi ấy qua các đỉnh : i=S1,S2,S3..Sk=j . Thì các cạnh : S1S2 chưa được nối , S2S3 phải không được nối , ...$S_{2*l+1}S_{2*l+2}$ phải không được nối , với điều kiện : k=2*l+2 . Sau đó chúng ta đổi lại sự nối các cạnh như sau : Nếu trên đường đi đó , các cạnh nào đã được

nối thì chuyển thành chưa nối , và các cạnh không được nối thì sẽ nối . Và lúc đó chắc chắn số cạnh của đồ thị sẽ tăng lên một , tức là mỗi đỉnh i , j sẽ giảm đi một bậc .

- Quá trình đó cứ tiếp tục cho đến hết không còn đỉnh nào có bậc .

Bài toán 4 :

Trồng Cây

Đề Bài :

Một mảnh sân hình chữ nhật cần trồng hai loại cây là phượng và điệp . Người ta chia mảnh sân thành các ô vuông bởi M hàng và N cột và ghi nhận vị trí những ô vuông mà ở đó phải trồng cây (mỗi ô chỉ có thể trông một cây) . Để cho cân đối giữa hai loại cây , việc trồng cây phải đảm bảo số cây phượng và số cây điệp hoặc bằng nhau hoặc chỉ có thể chênh nhau 1 trong tất cả các tình huống sau :

- Nhìn toàn bộ sân
- Nhìn theo bất cứ hàng nào
- Nhìn theo bất cứ cột nào

Hãy tìm một cách trồng cây thoả mãn yêu cầu đã nêu .

Dữ liệu : vào file Cay.Inp :

- Dòng đầu tiên ghi các giá trị M , N (cách nhau ít nhất một dấu trắng)
- M dòng tiếp theo , mỗi dòng ghi thông tin một hàng của sân dưới dạng một xâu nhị phân độ dài N (các kí tự 0 , 1 liền nhau) , với quy ước ký tự 1 ghi nhận vị trí có cây và ký tự 0 ghi nhận vị trí không có cây .

Kết quả : ghi ra file Cay.Out gồm M dòng , mỗi dòng ghi một độ dài N gồm các ký tự 0 ,1,2 viết liền nhau với quy ước ký tự 1 ghi nhận vị trí trông cây phượng , ký tự 2 ghi nhận vị trí trồng cây điệp và kí tự 0 ghi nhận các vị trí không có cây .

Ví dụ :

CAY.INP
6 11 01010010001 10100001000 01010100001 00000001010 00001000000

CAY.OUT
01020010002 10200001000 02010200001 00000002010 20000100000

Hướng Dẫn :

Ta coi mỗi hàng , mỗi cột là một đỉnh của đồ thị . Nếu ô (i,j) có giá trị <> 0 thì đỉnh hàng i nối với đỉnh cột j . Bài toán trở thành :

Tìm các tô các cạnh của một đồ thị bằng hai màu , sao cho :

- với mỗi đỉnh thì độ chênh lệch hai màu tô các cạnh nối nó chênh lệch không quá 1 .
- Với cả đồ thị chúng cũng chênh lệch nhau không quá 1 .

Chúng ta có phương pháp giải quyết bài toán này như sau :

- Nhận xét 1 : Nếu xuất phát từ một đỉnh bậc lẻ và đi một cách bất kỳ theo các cung của đồ thị , mỗi cung đi qua chỉ một lần thì trạng thái *tắc đường* phải xảy ra tại một đỉnh bậc lẻ khác (số đỉnh bậc lẻ nếu có trong đồ thị là một số chẵn)
- Nhận xét 2 : Nếu xuất phát từ một đỉnh bậc chẵn trong đồ thị không có đỉnh bậc lẻ và đi một cách bất kỳ các cung của đồ thị , mỗi cung đi qua chỉ một lần thì trạng thái *tắc đường* phải xảy ra tại chính đỉnh xuất phát .

Trạng thái *tắc đường* là trạng thái mà tại đỉnh vừa tới không còn cung nào chưa đi qua . Dựa vào hai nhận xét chúng ta có :

- Trường hợp 1 : Khi đồ thị còn đỉnh bậc lẻ

  Chọn một đỉnh lẻ bất kỳ để xuất phát . Bằng một cách đi bất kỳ qua các cung của đồ thị màu chưa được tô , mỗi cung đi qua ta tô xen kẽ bằng hai màu cho đến khi *tắc đường* . Trong trường hợp này ,tại đỉnh bậc lẻ kết thúc đường đi trên ,không còn cung nào chứa nó chưa được tô , đồng thời , tại đỉnh xuất phát , số cung còn lại chưa tô (nếu có) là một số chẵn , còn tại các đỉnh còn lại trên đường đi số cung được tô bằng các màu bằng nhau
- Trường hợp 2 : Khi đồ thị chỉ còn đỉnh bậc chẵn

  Trong trường hợp này , tất cả các cung kề với các đỉnh bậc lẻ (nếu có) của đồ thị ban đầu đều đã được tô . Chọn một đỉnh nào đó còn có cung chưa tô chứa nó làm đỉnh xuất phát và cũng đi một cách bất kỳ theo các cung chưa tô cho đến khi đạt được trạng thái kết thúc (tại đỉnh xuất phát) . Bằng cách tô màu các cung xen kẽ trên lộ trình đã đi qua . Khi đó só lượng các cung được tô hai màu được tô kề với mỗi đỉnh trên lộ trình là bằng nhau .

Bài toán 5 : Hệ Thống Thông Báo Hoàn Thiện

Đề Bài :

Một trường có N học sinh với tên 1..N, N<=10000. Một *hệ thống thông báo* trong trường được tổ chức và hoạt động như sau. Mỗi học sinh chọn *một học sinh duy nhất khác* (được gọi là *người kế tiếp*) để truyền trực tiếp thông báo. Mỗi học sinh khi nhận được thông báo phải truyền cho người kế tiếp của mình.

Hệ thống thông báo được gọi là *hoàn thiện* nếu khi một học sinh bất kỳ phát đi một thông báo nào đó tới người kế tiếp, người đó lại truyền cho người kế tiếp, cứ tiếp tục như vậy, thông báo sẽ được truyền đến mọi người trong trường kể cả người ban đầu đã phát đi thông báo.

Dữ Liệu : Hệ thống thông báo được cho bởi file TB.INP trong đó dòng thứ nhất ghi số nguyên dương N. Trong N dòng tiếp theo, dòng thứ I ghi tên người kế tiếp của người I.

Cần xét xem hệ thống thông báo đã cho có hoàn thiện không. Nếu không, hãy thay đổi người kế tiếp của một số ít nhất người để nhận được một hệ thống thông báo hoàn thiện.

Kết quả: Ghi ra file TB.OUT: dòng thứ nhất ghi số T là số người cần thay đổi người kế tiếp (T = 0 có nghĩa là hệ thống là hoàn thiện). Nếu T>0, trong T dòng tiếp theo mỗi dòng ghi hai số U, V có nghĩa là V là người kế tiếp mới của U.

Ví dụ

TB.INP	TB.OUT
4	1
2	4 1
3	
4	
2	

Hướng Dẫn :

Chúng ta xây dựng đồ thị như sau : Nếu i nối với j (đồ thị có hướng) tức nghĩa là j là người tiếp theo của i trong dây chuyền thông báo . Chúng ta sẽ tìm các chu trình , rồi gỡ rối chu trình đó ra bằng cách nối nó vào đầu một cây khác . Cứ như thế thì chúng ta sẽ giải phóng đồ thị có chu trình (nếu hệ thống không hoàn thiện) thành đồ thị dạng cây (hệ thống thông báo hoàn thiện) . Lời giải các bạn có thể tham khảo ở chương trình mẫu kèm theo .

Bài toán 6 : Hệ Thống Ngân Hàng

Đề Bài :

Một ngân hàng có N chi nhánh có tên từ 1 đến N, mỗi chi nhánh có một hệ thống dữ liệu (từ đây viết tắt là HTDL), hai chi nhánh khác nhau có hai HTDL khác nhau. Trong một lần thay đổi máy tính của toàn bộ N chi nhánh, do sơ xuất, người ta đã cài đặt không đúng vị trí của các HTDL, chẳng hạn, HTDL tại chi nhánh A là của chi nhánh B, HTDL tại B là của chi nhánh C, . . . (có thể có chi nhánh đã có đúng HTDL của nó), *mặc dù hai chi nhánh khác nhau vẫn giữ hai HTDL khác nhau.*

Yêu Cầu : Cần phải tiến hành tráo đổi các HTDL giữa các chi nhánh cho nhau sao cho mỗi chi nhánh có được HTDL của nó. Giữa hai chi nhánh có thể tiến hành tráo đổi HTDL cài trên nó cho nhau và công việc đó diễn ra trong đúng một ngày. Hai cặp chi nhánh khác nhau có thể đồng thời tiến hành công việc này. Hãy tính xem cần ít nhất bao nhiêu ngày để hoàn tất công việc này.

Dữ liệu: vào được cho bởi file NH.DAT trong đó dòng thứ nhất ghi số nguyên dương N<=10000; trong các dòng tiếp theo ghi N số nguyên dương khác nhau từng đôi và không lớn hơn N, số thứ i trong N số này bằng k có nghĩa là chi nhánh i đang giữ HTDL của chi nhánh k, mỗi dòng ghi 30 số.
Kết quả Ghi ra file GP.DAT như sau: dòng thứ nhất ghi số M là số ngày cần thiết (M có thể bằng 0), tiếp theo là M *dòng*, dòng thứ j, 1<=j<=M, ghi N số trong đó số thứ i là số hiệu chi nhánh mà chi nhánh i tráo đổi HTDL.
Ví dụ

NH.DAT				GP.DAT			
4				2			
4	3	1	2	3	4	1	2
				1	2	4	3

Hướng Dẫn :
Chúng ta sẽ xây dựng một đồ thị có hướng Nx N đỉnh mà trong đó mỗi hệ thống dữ liệu như đai diện cho ngân hàng có cùng số thứ tự . A[i,j]=1 nếu HTDL i nằm ở ngân hàng j . Nhưng ta nhận thấy đồ thị này có một đặc điểm là tất cả các đỉnh chỉ có một cung đi vào nó mà thôi. Tức là tập hợp các ngân hàng có HTDL để nhầm sẽ tạo thành một chu trình khép kín (mỗi đỉnh chỉ có một cung vào và ra) . Để đưa hệ thống đỉnh đó trở về nguyên dạng thì ta thấy rằng số lần đổi chỗ như vậy sẽ không thể quá 2 lần .

VI. Tìm Kiếm Trên Đồ thị :

Có hai thuật toán tìm kiếm cơ bản : chiều sâu và chiều rộng . Sau đây là mô tả chương trình của chúng :

1. Tìm kiếm theo chiều sâu :

```
Procedure       DFS ( i : integer ) ;
var
      j     :     integer ;
      Begin
            thăm đỉnh ( i ) ;
            xet [i]:=false ;
            for j := 1 to n do if xet[j]then if a[i,j]=1 then DFS(j);
      End ;
```

2. Tìm kiếm theo chiều rộng :

```
Procedure       BFS (i);
```

```
Var
    dau , cuoi  , j :  integer ;
    Begin
        q [1 ]:=i ;
        dau :=1 ;
        cuoi := 1 ;
        While dau<= cuoi do
        Begin
            i:= q [ dau ] ;
            inc ( dau ) ;
            Thăm đỉnh ( i ) ;
            For j := 1 to n do
            If xet [j] and ( a[i,j]=1) then
            begin
                inc ( cuoi ) ; q [cuoi ] := j ;
                xet [j]:=false;
            end ;
        End ;
    End ;
```

Các Bài toán ở phần này xin các bạn theo dõi ở chương III (phần tìm kiếm)

Phần 2 :

Cây khung và đường đi ngắn nhất

I. Cây :

1. Định Nghĩa :

- Cây là đồ thị vô hướng liên thông không có chun trình
- Ta có các định lý sau :
  + Nếu G là một cây thì G có N-1 cạnh (N là số đỉnh)
  + Nếu G là một cây thì khi ta thêm một cạnh bất kì vào đồ thị thì đồ thị sẽ có một chu trình
  + Hai đỉnh được nối với nhau bằng đúng một đường đi đơn .
- Cây khung của một đồ thị là một đồ thị con của một đồ thị và nó là một cây

2. Bài toán cây khung nhỏ nhất :

Bài toán :

" Cho G=(V,E) là đồ thị vô hướng liên thông với tập đỉnh V={1,2,...N} và tập cạnh E gồm m cạnh . Mỗi cạnh e của đồ thị G được gán với một số không âm C(e) , gọi là độ dài của nó . Giả sử H=(V,T) là cây khung của đồ thị G . Ta gọi là độ dài C(h) của cây khung H là tổng độ dài của các cạnh của nó :

$C(h)=\sum C(e)$

Bài toán dặt ra là trọng số tất cả các cây khung của đồ thị G , hãy tìm cây khung với độ dài nhỏ nhất .”

Có hai thuật toán cơ bản để giải quyết bài toán này :

a. Thuật Toán Kruskal :

```
Procedure               Kruskal ;
Begin
    T:=∅ ;
    While | T| <(N-1) and ( E≠∅)do
    Begin
          Chọn e là cạnh có độ dài nhỏ nhất trong E ;
          E:=E\{e};
          If (T∪{e} không chứa chu trình ) then T:=T∪{e};
    End ;
    If | T|<n-1 then đồ thị không liên thông ;
End ;
```

b. Thuật toán Prim :

```
Procedure       Prim ;
Begin
    Chọn s là một đỉnh nào đó của đò thị ;
V_H:={s };
T:=∅ ;
d[s]:=0;
nears[s]:=s;
for v∈V \V_H do
begin
    d[v]:=c[s,v];
    nears[v]:=s;
end ;
stop:=false;
while not stop do
begin
    tìm u∈V\V_H do
     d[u]=min{d[v]:u∈V\V_H};
    V_H:=V_H∪{u};
    T:=T∪{(u,near[u]};
    If |V_H| =n then
```

```
            begin
                    stop:=true ;
                    H=(V_H,T) là cây khung nhỏ nhất của đồ thị ;
            end
            else
            for v∈V\V_H do
            begin
                    d[v]:=c[u,v];
                    nears[v]:=u;
            end ;
    end ;
    end ;
```

II. Đường đi Ngắn nhất :

1. Thuật toán Ford_Bellman :

Tìm đường đi ngắn nhất từ một đỉnh s đến đỉnh t trong đồ thị:

```
Procedure               Ford_Bellman ;
Begin
    For i := 1 to n do
    begin
            d [i]:=maxint ;
            tr[i]:=maxint ;
    end ;
    d[s]:=0;
    Repeat
            Ok:=true;
            For i:=1 to n do
             if d[i]<>maxint then
            for j:=1 to n do
            if (a[i,j]<>0)and(d[i]+a[i,j]<d[j]) then
            begin
                    ok:=false;
                    d[j]:=d[i]+a[i,j];
                    tr[j]:=i;
            end;
    until ok ;
```

Thực chất của thuật toán này là thuật toán Quy Hoạch Động , ở dạng 2 mà các bạn đã gặp trong chương I . Trong đó , D[i] là mảng độ dài ngắn nhất

đi từ s đến i . vậy nếu t là đỉnh cần thiết thì d[t] là độ dài cần tìm . Còn nếu muốn lưu lại đường đi thì chúng ta dùng mảng Tr [i] để đi ngược lại .

2. Thuật toán DijKstra :
   Dùng cho một ma trận đường đi mà có trọng số không âm :

```
Procedure           Dijsktra ;
begin
    for i:=1 to n do
    begin
            d[i]:=a[s,i];
            tr[i]:=s;
    end ;
    d[s]:=0;
    fillchar ( xet , sizeof ( xet) , true ) ;
    xet[s]:=false ; count := 1 ;
    while count<>n do
    begin
            tìm đỉnh u thoả mãn xet[u]=true và d[u]=min{d[z] };
            xet[u]:=false;
            for i:=1 to n do
            if d[i]>d[u]+a[u,i] then
            begin
                    d[i]:=d[u]+a[u,i];
                    tr[i]:=u;
            end ;
            inc ( count ) ;
    end ;
end ;
```

Tốc độ thuật toán dijsktra nhanh hơn nhiều thuật toán ford_bellman . Nhưng nó chỉ áp dụng cho ma trận trọng số không âm .

3. Thuật toán Ployd :

   là lập bảng danh sách các đường đi ngắn nhất giữa các đỉnh :

```
Procedure           Ployd ;
Begin
    for k := 1 to n do
     for i := 1 to n do
```

```
 if a[i,k]<>0 then
 for j:=1 to n do
  if (i<>j)and(a[k,j]<>0)then
  If (a[i,j]=0)or(a[i,j]>a[i,k]+a[k,j])then
begin
        a[i,j]:=a[i,k]+a[k,j];
        c[i,j]:=k;
end ;
end ;
```

lúc đó a[i,j] là đường đi ngắn nhất giữa các cặp đỉnh (i,j).

III. ứng Dụng :

Bài toán 7: mạng điện

Đề bài :

Trong một lần dự trại hè , các bạn trẻ đã cắm các ngôi trại trên một mảnh đất có hình như một hệ trục toạ độ . Điều bức xúc cần giải quyết gấp đó là : chúng ta cần bắc các đường dây điện nối các trại với nhau (để trao đổi thông tin , và cũng có thể là cung cấp điện) . Bài toán đặt ra là hãy tìm cách nối sao cho ít dây nhất mà trại nào cũng có thể có đường nói đến trại khác .

Dữ liệu : Vào từ file Noidien.Inp :

- Dòng đầu tiên ghi số trại có thể có (N <=100)
- N dòng sau , mỗi dòng ghi toạ độ của các trại (toạ độ nằm trong longint)

Kết quả : Ghi ra file Noidien.Out :

- Dòng đầu tiên ghi tổng số dây cần nối (tính đến sau 2 dấu phẩy)
- N dòng sau biểu diễn ma trận kề để biểu diễn trại i có nối với trại j . Nếu có nối trực tiếp thì giá trị ma trận bằng 1 ngược lại bằng 0 .

Hướng Dẫn :

Bài toán trở thành bài toán tìm cây khung ngắn nhất của đồ thị trên . Nhưng đặc điểm của bài toán đó là dữ liệu có thể rất lớn . Nên cẩn thận trong tính toán .

Bài toán 8: Nâng cấp đường

Đề Bài :

Có một hệ thống đường liên thông nối N nút giao thông có tên 1..N , 3<=N<=2000 . Số đoạn đường nối trực tiếp hai nút không quá 10000 . Hệ thống đường này đã xuống cấp . Cần chọn một số đoạn đường nâng cấp , hai yêu cầu sau được thoả mãn :

- Giữa hai nút bất kỳ chỉ có đúng một đường đi

- Số M các nút giao thông chỉ là đầu mút của một đoạn đường là nhiều nhất có thể được

Dữ liệu : Vào từ file NC.INP gồm N dòng trong đó dòng thứ I ghi tên các nút có đoạn đường nối trực tiếp với nút I

Kết quả : Ghi ra file NC.OUT như sau :

- Dòng thứ nhất ghi số M , tiếp theo là một số dòng , mỗi dòng ghi tên hai nút đầu mút của một đoạn đường cần nâng cấp

Ví Dụ :

NC.INP
2 3 8 1 5 6 8 1 4 7 8 3 5 6 7 2 4 6 2 4 5 7 3 4 6 8 1 2 3 7

NC.OUT
5 2 8 1 8 3 8 7 8 7 6 6 4 5 6

Hướng Dẫn :

Bài toán này không có thuật giải chính xác . Chính vì thế phải sử dụng phương pháp tham lam . Nếu tham lam tốt thì sẽ có kết quả tốt . Nhưng mặt khác phải xử lí dữ liệu tốt . Tốt nhất nên dùng danh sách liên kết để lưu trữ các đỉnh kề với một đỉnh

Bài toán 9 : Nâng cấp đường

Đề bài :

Hệ thống giao thông trong một thành phố bao gồm N nút giao thông và M đoạn đường phố hai chiều , mỗi đoạn nối 2 nút giao thông . Các nút giao thông được đánh số từ 1 đến N và các đoạn đường phố được đánh số từ 1 đến M . Giữa hai nút giao thông có không quá một đoạn đường phố nối chúng . Hệ thống giao thông đảm bảo sự đi lại giữa hai nút giao thông bất kỳ . Ban quản lí hệ thống giao thông được giao nhiệm vụ thực hiện dự án nâng cấp tất cả các đoạn đường phố . Mọi sự đi lại theo đoạn đường phố sẽ bị cấm trong suốt thời gian thực hiện thi công nâng cấp nó . Thời gian cần thiết để hoàn thành việc thi công nâng cấp bất cứ đoạn đường phố nào cũng là 1 ngày và trong một ngày ban quản lí có thể tổ chức thực hiện việc thi công nâng cấp đồng thời không quá K đoạn đường phố . Để đảm bảo sự đi lại giữa hai nút giao thông bất kỳ trong

suốt thời gian thực hiện dự án , Ban quản lí cần tìm lịch thi công các đoạn đường một cách hợp lí .

Yêu cầu : Tìm lịch thi công nâng cấp tất cả các đoạn đương phố đảm bảo sự đi lại giữa hai nút giao thông bất kỳ trong suốt quá trình thực hiện dự án được hoàn thành sau ít ngày nhất .

Dữ liệu : Vào từ văn bản BL1.INP :

- Dòng đầu tiên chứa ba số nguyên dương N ,M,K (2<=N<=500,1<=M<=20000 , 1<=K<=N) .
- Dòng thứ i trong số M dòng tiếp theo chứa cặp hai số hiệu của hai nút giao thông tương ứng là hai đầu mút của đoạn đường phố thứ i

Kết quả : Ghi ra file văn bản BL1.OUT :

- Dòng đầu tiên ghi P là s ngày cần thực hiện theo lịch thi công tìm được (qui ước ghi P=-1 , nếu như không tìm được lịch thoả mãn yêu cầu đặt ra)
- Nếu tìm được lịch thì dòng thứ i trong số M dòng tiếp theo ghi chỉ số của ngày thực hiện thi công nâng cấp đoạn đường thứ i . Các ngày trong lịch thực hiện dự án được đánh số từ 1 đến P theo đúng trình tự thời gian

Ví Dụ :

BL1.INP
5 10 5 1 2 2 3 3 4 4 5 5 1 1 3 2 4 3 5 4 1 5 2

BL1.OUT
2 1 1 1 1 1 2 2 2 2 2

Hướng Dẫn :

Bài toán 10 : Biến đổi bảng

Đề bài :

Một ma trận vuông kích thước N*N gồm các số 0, 1, trong đó N là một số nguyên dương lẻ < 50. Một phép biến đổi ma trận là chọn một tập S gồm N phần tử của ma trận trong đó mỗi hàng và mỗi cột đều có đúng một phần tử thuộc S, sau đó ta biến đổi mỗi phần tử 0 bởi 1 và ngược lại 1 bởi 0.

Yêu Cầu : Cho ma trận N*N như trên, hãy chỉ ra một dãy các phép biến đổi sao cho trên ma trận thu được số phần tử mang giá trị 1 không vượt quá N-1.

Dữ Liệu :

- Dòng đầu tiên ghi số nguyên dương N
- N dòng tiếp theo, dòng thứ N ghi N số trong đó số thứ j của dòng thứ i ghi giá trị của tại vị trí (i,j) của ma trận.

Kết Quả :

- Dòng đầu ghi K là số phép biến đổi hoặc -1 nếu không tìm được phép biến đổi thoả mãn yêu cầu.
- K dòng tiếp theo, mỗi dòng ghi một N số thể hiện một tập S của phép biến đổi ma trận, trong đó số thứ i là vị trí cột của phần tử thuộc tập S trên hàng i của ma trận.

Hướng Dẫn :

Ta xây dựng đồ thị hai phía X,Y tập X gồm các đỉnh tương ứng với N hàng, Y gồm các đỉnh tương ứng N cột, (x,y) ($x \in X$, $y \in Y$) là cạnh khi và chỉ khi vị trí (x,y) của ma trận là 1.

Bước đầu tiên, ta xét tất cả các đỉnh bậc lẻ trên đồ thị.

Nếu có một trong hai tập X, Y mà ta có thể giả sử là X mà mọi $\in$ X đều có bậc lẻ. Vì X là tập lẻ đỉnh (vì N lẻ), nên số đỉnh lẻ của tập Y phải là số lẻ, tức là ít nhất 1 đỉnh lẻ. Ta thực hiện một phép biến đổi bất kì, khi đó mọi đỉnh đều bị thay đổi bậc (chẵn thành lẻ và ngược lại), tức là số đỉnh lẻ trên mỗi tập luôn nhỏ hơn N.

Như vậy, sau bước đầu tiên, mỗi tập X, Y có nhỏ hơn N đỉnh lẻ.

Ta định nghĩa một thủ tục giữ lại cạnh với cặp (x,y) ($x \in X$, $y \in Y$) như sau:

1.Nếu (x,y) đã là cạnh thì ta tưởng tượng không có cạnh này, tức là cuối cùng thực tế trên đồ thị còn lại cạnh này.

2.Nếu (x,y) không là cạnh thì ta tưởng tượng có thêm cạnh này, để khi trong phép biến đổi có cạnh này thì thực tế từ không có cạnh sẽ thành có cạnh.

Thực chất thủ tục giữ lại cạnh chính là giữ lại các cạnh còn lại cuối cùng sau tất cả các phép biến đổi.

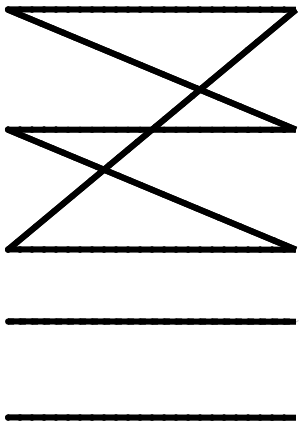
Trở lại bài toán: Sau khi hai tập X,Y có ít hơn N đỉnh lẻ, ta thực hiện tiếp như sau:

+ Nếu còn tồn tại hai đỉnh lẻ $x \in X$, $y \in Y$ thì ta tiến hành giữ lại cạnh (x,y), chú ý sau thủ tục này hai đỉnh u,v sẽ là đỉnh chẵn. Sau thủ tục này, các đỉnh lẻ nếu có chỉ thuộc một tập mà thôi ta giả sử là X.

+ Rõ ràng số đỉnh lẻ của X sau khi thực hiện bước trên phải là số chẵn, ta thực hiện thủ tục giữ lại cạnh (x, 1) với mỗi đỉnh x∈ X mà x lẻ. Do số đỉnh lẻ thuộc X là số chẵn nên bậc của đỉnh 1 ∈ Y là số chẵn.

Sau hai bước trên tất cả các đỉnh đều có bậc chẵn. Ta thực hiện bước cuối cùng như sau:

Nếu trên đồ thị còn cạnh, vì số đỉnh chẵn nên phải tồn tại một chu trình đơn. Vì là đồ thị hai phía nên chu trình phải có chẵn cạnh, ta đánh số các cạnh liên tiếp chẵn lẻ theo chu trình. Khi đó tập B0 các cạnh chẵn của đồ thị tường ứng với tập con S nào đó của một phép biến đổi ma trận nào đó. Gọi A là phần bù của B0 trong S, thực hiện phép biến đổi ma trận với với tập S, sau đó thay S bằng hợp của A với B1 là tập các vị trí tương ứng cạnh lẻ của chu trình tìm được. Rõ ràng B1 và B0 có cùng tập đỉnh X và Y cho nên tập S mới này cũng thoả mãn cho một phép biến đổi hợp lệ, ta thực hiện tiếp phép biến đổi với tập S mới tạo ra



Ta có thể dễ thấy, tập A xuất hiện hai lần trong cả hai phép biến đổi cho nên coi như không biến đổi, như vậy ta đã loại khỏi đồ thị tất cả các cạnh trên chu trình của đồ thị mà không mà ảnh hưởng đến các phần tử khác.

Cú thực hiện các bước tìm chu trình và xoá khỏi đồ thị, cuối cùng ta sẽ thu được đồ thị không có chu trình. Tuy nhiên trên thực tế, ta hãy xét lại định nghĩa thủ tục giữ lại cạnh, ta chú ý rằng tất cả mọi cặp (x,y) trong thủ tục này sẽ là cạnh còn lại trên thực tế. Vì số lần gọi thủ tục này bằng max của số đỉnh lẻ trên hai tập X,Y mà trên hai tập không có tập nào có tất cả đỉnh đều lẻ nên số lần gọi thủ tục không quá N-1. Điều này chỉ ra rằng trên đồ thị còn lại có không quá N-1 cạnh, tức là ma trận nhận được có không quá N-1 vị trí 1.

Bài toán 11 : Guest

Đề Bài :

Công ty trách nghiệm hữu hạn " Vui vẻ " có n cán bộ đánh số từ 1 đến N . Cán bộ i có đánh giá độ vui tính là Hi (i = 1 , 2 ..N) . Ngoại trừ giám giám đốc công ty , mỗi cán bộ có 1 thủ trưởng trực tiếp của mình .

Bạn cần giúp công ty mời một nhóm cán bộ đến dự dạ tiệc " Vui vẻ " sao cho trong số những người được mời không đồng thời có mặt nhân viên và thủ trưởng trực tiếp và đồng thời tổng đánh giá vui tính của những người dự tiệc là lớn nhất .

Giả thiết rằng mỗi một thủ trưởng không có quá 20 cán bộ trực tiếp dưới quyền

Dữ liệu : Vào từ file Gues.Inp :

- Dòng đầu tiên ghi số cán bộ của công ty (1<n<1001)
- Dòng thứ i trong số N đòng tiếp theo ghi hai số nguyên dương Ti , Vi trong đó Ti là số hiệu của thủ trưởng trực tiếp và Vi là độ vui tính của cán bộ i . Quy ước Ti=0 nếu như số hiệu của giám đốc công ti

Kết quả : Ghi ra file Guest.Out :

- Dòng đầu tiên ghi hai số M và V . Trong đó M là tổng số cán bộ được mời còn V là tổng độ vui vẻ của các cán bộ được mời
- Dòng thứ K trong số M dòng tiếp theo , mỗi dòng ghi số hiệu các cán bộ được mời

Ví Dụ :

GUEST.INP		GUEST.OUT
3 0 3 1 6 2 4		2 7 1 3

Hướng Dẫn :

Thực chất là chúng ta tìm các đỉnh của một cây mà sao cho không có đỉnh nào là con của đỉnh kia , và có tổng độ vui vẻ là lớn nhất . Chúng ta sử dụng thuật toán đệ quy có nhớ . Nhưng đó là chúng ta duyệt cận của cây (nhánh) . Nên chương trình sẽ thu gọn về tốc độ rất nhiều .

Bài toán 12 : Salary

Đề Bài :

Một xí nghiệp liên doanh có N nhân viên , đánh số từ 1 tới N . Để xây dựng hệ thống tiền lương hợp lí , người ta quyết định trả lương theo năng suất cá nhân . Kí hiệu X_i là năng suất lao động của người thứ i và S_i là lương của người đó . Nếu giữa hai người thứ i và j năng suất lao động có quan hệ Q_{ij} thế nào thì tiền lương của họ cũng sẽ có quan hệ như vậy . Các quan hệ có thể là < , ≤ , = , ≥ , > và kí hiệu tương ứng bằng các số nguyên từ 1 đến 5 . Theo quy định

, lương tối thiểu không ít hơn S_{min} và lương tối đa không vượt quá S_{max} , các bậc lương liên tiếp chênh lệch nhau d đơn vị (d>0)

Ban giám đốc muốn xây dựng mức lương với chi phí ít nhất, còn công đoàn muốn người lao động phải được trả lương cao nhất có thể . Mức lương thực trả là trung bình của 2 đề án trên . Cho số và các quan hệ Q_{ij} . Hãy xác định xem có thể có hệ thống lương hợp lí hay không . Nếu có , hãy đưa ra mức lương của mỗi người (không tính phần)

Dữ liệu : Vào từ file SALARY.INP :

- Dòng đầu tiên ghi số N (N<=1000)
- Dòng thứ hai ghi 3 số nguyên S_{min} , S_{max} , d
- Các dòng sau , mỗi dòng 3 số nguyên i , j , Q_{ij}
- Dấu hiệu kết thúc là dòng chứa 3 số 0

Kết quả : Ghi ra file SALARY.OUT :

- Dòng đầu tiên thông báo CO hoặc KHONG
- N dòng sau ghi lương S_i , i =1 , 2 , ... N

Hướng Dẫn :

Thực chất bài toán này là bài toán phân lớp có tính chất về lớp tiền lương . Chúng ta sẽ đi từ gốc của cây đó , và phân về lớp cuối cùng , cho đến khi phân được các lớp của cây đó . Tức là ta sẽ tìm lớp của "cành cây" là lớp thứ mấy . (Sử dụng thủ tục numbering trong cuốn sách Toán Rời Rạc – Trang 205)Rồi sau đó tính tiền lương cho mỗi người .

Chú ý : Xử lí dữ liệu của bài toán là rất cẩn thận .Nếu không có thể dẫn đến sai lạc .

Bài toán 13 : Xếp ăn tiệc

Đề Bài :

Tổng giám đốc một công ty muốn tổ chức một buổi liên hoan cho các đồng nghiệp trong công ty. Mỗi cán bộ của công ty ngoại trừ tổng giám đốc đều có đúng một *cấp trên trực tiếp* của mình. Cán bộ P được gọi là cấp trên của cán bộ Q nếu có một dãy cán bộ P1, P2, . ., Pk, k>=2, sao cho P=P1, Pk=Q và với 1<=i<=k-1, Pi là cấp trên trực tiếp của P_{i+1}. Để cho mọi cán bộ được thoải mái, tổng giám đốc không muốn một cán bộ bất kỳ nào ngồi cùng một bàn với cấp trên của mình. Hãy tính xem cần tối thiểu bao nhiêu bàn dùng cho buổi liên hoan theo yêu cầu của TGĐ.

Dữ liệu vào được cho bởi file INP.TXT trong đó dòng thứ nhất ghi số nguyên dương N <=200 là số lượng toàn thể cán bộ của công ty, các cán bộ của công ty có tên từ 1 đến N, TGĐ có tên 1 và không có cấp trên. Dòng thứ hai ghi số nguyên dương K, 2 <= k <= 10, là số người tối đa có thể ngồi trong một bàn (chú ý rằng khi xếp, không nhất thiết mọi bàn phải đủ K người). Dòng thứ ba ghi N số trong đó số thứ nhất là số 0, số thứ i là tên cấp trên trực tiếp của cán bộ i. Dữ liệu đúng như mô tả.

Ghi ra file OUT.TXT số lượng M bàn cần dùng. Trong M dòng tiếp theo, dòng thứ i ghi tên cán bộ ngồi bàn i.

dutiec.inp
7 3
0 1 1 2 2 3 3
dutiec.out
3
1
2 6 7
3 4 5

Hướng Dẫn :

Thuật toán duyệt theo cây . Nhưng phải sử dụng tính chất của cây để làm cận cho chương trình .

Bài toán 14 : D'artagnan

Đề bài :

D'artagnan là một lính ngự lâm cự phách , nhân vật chính trong cuốn tiểu thuyết ba người lính ngự lâm của nhà văn A.Dumas . Thời đó nước pháp có N thành phố , N<=50 ,với tên 1..N . Paris mang tên 1 còn thành phố quê hương của D'artagnan mang tên thành phố N . Có hai lực lượng bảo vệ kình địch nhau : lính ngự lâm quân và lính cảnh vệ . Với mỗi con đường từ thành phố A đến thành phố B , tại lối vào-ra hai thành phố đầu mút có hai vọng gác . Một vọng gác do lính ngự lâm , một do lính cận vệ . Luật phát thời đó quy định khi vào một thành phố bất kỳ bằng một vọng gác do loại lính nào đó đảm nhiệm thì cũng ra qua vọng gác cùng loại đó .

Thông tin về mỗi con đường nối hai thành phố được cho bởi một dòng như sau L

U , V , W1 , W2 , L

Với ý nghĩa có đường trực tiếp từ thành phố U đến thành phố V , thời gian đi từ U đến V bằng W1 , thời gian đi từ V đến U bằng W2 , L là ký tự M/G nêu vọng gác đầu U của đoạn đường do lính ngự lâm /lính cận vệ đảm nhiệm (theo quy định khi đó vọng gác đầu V tương ứng do lính cận vệ / lính ngự lâm đảm nhiệm) , giữa hai mục liên tiếp nêu trên dòng cách nhau đúng một dấu trống . Các số W1,W2 không lớn hơn 60000 .

Một lần D'artagnan cần đi rất gấp từ thành phố quê hương đến Paris để hỗ trợ cho các bạn của mình . D'artagnan đấu kiếm rất giỏi nhưng tính toán hơi yếu . Hãy giúp D'artagnan tìm một con đường để đi nhanh nhất từ quê hương lên Paris theo đúng các quy định về vọng gác .

Dữ liệu : Vào từ file Dart.Inp :

- Dòng thứ nhất ghi số N

- Tiếp theo là nhóm dòng không quá 1000 dòng , dòng thứ I ghi thông tin về đoạn đường thứ I theo cách ghi trên . D'artagnan luôn có thể đi từ quê hương lên Paris .

Kết quả : Ghi ra file Dart.Out :

- Dòng thứ nhất ghi độ dài đường đi ,
- Dòng thứ hai ghi tên các thành phố lần lượt trên hành trình từ thành phố N đến thành phố 1 .
- Dòng thứ ba ghi tên các số hiệu các đoạn đường lần lượt trên hành trình .

Ví dụ :

Dart.Inp	Dart.Out
5	5
1 2 3 3 M	5 4 1
1 4 4 4 G	6 2
1 3 2 2 G	
2 5 3 3 G	
2 4 10 10 G	
5 4 1 1 G	
4 3 1 1 M	

Hướng Dẫn :

Dùng phương pháp đường đi ngắn nhất . Nhưng chúng ta cần phải lưu ý vì có đến hai cửa đến và ra . Như vậy chúng ta sẽ tìm hai lần đường đi từ s đến t và t đến s . Sau đó chúng ta chọn ra đường đi tối ưu cho cách đi đó .

Bài toán 15 : Hàng Đổi Hàng

Đề Bài :

Hiện nay vẫn có nhiều vùng dân cư buôn bán theo cách đổi hàng lấy hàng theo một bản quy ước gồm nhiều mục , ví dụ : " 2 lợn đổi 1 bò " , " 2 bò đổi 1 ngựa " là hai mục trong bản quy ước . Bản quy ước này do hình thành lâu đời nên có thể có mâu thuẫn , chẳng hạn cùng với hai mục trên có thể có mục " 2 ngựa đổi 3 lợn " . Mặt khác , người dân chỉ biết các số nguyên dương nên các số viết trong các mục là nguyên dương , trong trường hợp cần trao đổi hai loại hàng hoá nào đó mà tương quan không có trong mục , nếu việc suy diễn không mâu thuẫn , kết quả suy diễn từ các mục khác về tương quan giữa hai loại hàng đó phải được quy về các số nguyên dương chọn nhỏ nhất có thể được , ví dụ từ " 2 lợn đổi 1 bò " , " 2 bò đổi 1 ngựa " và trong bản quy ước không có tương quan giữa lợn với ngựa thì phải suy ra " 4 lợn đổi 1 ngựa " mà không thể viết là " 2 lợn đổi ã ngựa " hoặc " 8 lợn đổi 2 ngựa " . Chú ý rằng trong văn bản quy ước có thể có mục " 6 lợn đổi 3 bò " nhưng tương quan suy diễn (nếu có thể và cần có) phải theo quy định trên . Tất nhiên cùng một loại hàng , ngầm định luôn là 1 đổi 1 .

Có một bản quy ước trao đổi được ghi trong file TD1.INP gồm một số dòng , mỗi dòng ghi 4 số nguyên dương U , V , X , Y với ý nghĩa X đơn vị hàng U đổi Y đơn vị hàng V . Các yêu cầu trao đổi được ghi trong file TD2.Inp gồm một số dòng ghi hai số A, B với ý nghĩa cần trao đổi giữa hai loại hàng A và B . Có N loại hàng với tên từ 1 đến N và N không quá 60 , các số trong file TD1.INP không lớn hơn 200 , mọi số phát sinh không vượt Longint .
Hãy cho biết bản quy ước có mâu thuẫn không ?
Nếu mâu thuẫn , ghi ra file TD.OUT như sau :

- Dòng đầu tiên ghi số 1
- Tiếp theo là hai dòng , dòng thứ nhất ghi bốn số nguyên dương A , B , C , D , dòng thứ hai ghi bốn số A , B , E , F với ý nghĩa , từ bản quy ước ta có thể suy ra C đơn vị hàng A đổi được D đơn vị hàng B nhưng cũng có thể suy ra E đơn vị hàng A đổi được F đơn vị hàng B và $CF \neq DE$.

Nếu không mâu thuẫn , ghi ra file TD.OUT như sau :

- Dòng đầu tiên ghi số 0
- Tiếp theo với mỗi dòng ghi hai số U , V đọc trong file TD2.INP , ghi ra trên một dòng hai số X, Y mà X=Y=0 có ý nghĩa là từ văn bản quy ước không suy ra được tương quan giữa hai loại hàng hoá ; U , V , nếu X , $Y \neq 0$ có nghĩa là ta suy ra được X đơn vị hàng U đổi được Y đơn vị hàng V .

Ví dụ :

TD1.INP :
1 2 6 15
3 4 47 9
2 3 2 1

TD2.INP :
2 1
1 4
TD.OUT
0
5 2
188 45

Hướng Dẫn :

Xây dựng một đồ thị mà tỉ lệ đổi là một số nguyên , tức là A[i,j].x và A[i,j].y biểu diễn cho : x đơn vị hàng i đổi được y đơn vị hàng j . Trong quá trình tiếp nhận đồ thị , nếu mâu thuẫn thì loại . Còn nếu không thì chúng ta tìm đường đi từ s đến t của đồ thị sẽ đưa ra mối quan hệ của hàng s và hàng t .

Bài toán 16 : Nâng cấp

Đề bài :

Một mạng giao thông có N nút đánh số từ 1 đến N , giữa một số cặp nút có đường đi hai chiều và mạng liên thông . Hiện nay toàn bộ hệ thống đường rất xấu .

Cần chọn một nút đặt trạm cứu hoả và một số đoạn đường để nâng cấp sao cho với hệ thống chỉ gồm những đoạn đường được nâng cấp , từ trạm cứu hoả đến mỗi nút có đúng một đường đi và khoảng cách từ nút xa trạm nhất đến trạm nhỏ nhất có thể được .

Dữ liệu : Vào từ file CH.INP :

- Dòng đầu tiên ghi số N <=200 .
- Trong một số dòng tiếp theo , mỗi dòng ghi ba số nguyên dương U,V,W với ý nghĩa có đường đi hai chiều nói nút U với nút V dài W , W<=10000 .

Kết quả : Ghi ra file CH.OUT :

- Dòng thứ nhất ghi tên nút đặt trạm cứu hoả
- Dòng thứ hai ghi khoảng cách từ nút xa nhất đến trạm ,
- Tiếp theo là một số dòng , mỗi dòng ghi hai nút đầu mút của một đoạn đường cần nâng cấp .

Ví Dụ :

CH.INP
5 1 2 50 1 3 30 1 4 100 1 5 10 2 3 5 2 4 20 3 4 50 4 5 10

CH.OUT
4 25 1 5 2 3 2 4 4 5

Hướng Dẫn :

Bài toán này không có thuật toán cụ thể , mà chúng ta phải tham lam theo một cách nào đó tối ưu gần đúng mà thôi .

Bài toán 17 : Hội thảo bằng điện thoại

Đề bài :

Cho một cuộc hội thảo tổ chức thông qua điện thoại (vì lí do thời gian và khoảng cách khá xa) . Có N địa điểm , Các địa điểm được nối với nhau thông qua đường dây điện thoại . Nhưng chi phí kết nối trong 1h giữa hai địa điểm i và j là A[i,j] . Chính vì vậy ,cho nên có thê khi người ta cần kết nối i với j thì có thê nối thông qua một số trạm khác . Có ba nhà ngoại giao , họ đang cần bàn kế

hoạch giải quyết hoà bình của một cuộc chiến mới bắt đầu . Các bạn hãy tìm cách kết nối các đường dây sao cho mỗi người phải luôn nói chuyện được với các người khác .

Dữ liệu : Vào từ file Conf.Inp :

- Dòng đầu tiên ghi hai số N và M . N là số nút (N<=100) và M là số đoạn nối các địa điểm với nhau .
- M dòng sau mỗi dòng ghi ba số : I,J,V trong đó biểu diễn đoạn nối I với J thì tốn chi phí kết nối là V trong 1 h
- Dòng cuối cùng ghi ba số X,Y,Z là ba địa điểm của ba nhà ngoại giao .

Kết quả : Ghi ra file conf.Out :

- Dòng đầu tiên ghi 2 sô T và K : T là số tiền kết nối (trong 1 h) và K là số đoạn được dùng
- K dòng sau , mỗi dòng ghi 2 số biểu diễn các trạm được kết nối với nhau :

Ví Dụ :

CONF.INP
8 12 1 2 20 2 3 8 2 4 3 2 5 3 2 6 6 3 5 2 3 6 9 4 7 5 5 6 1 5 7 7 6 8 4 7 8 6 1 4 6

CONF.OUT
27 4 1 2 2 4 2 5 5 6

Hướng Dẫn :

Bài toán tưởng là phức tạp . Thế nhưng chúng ta thấy rằng chắc chắn đoạn nối đó phải là một cây . Tức là sẽ có một cây đồ thị bao lấy ba địa điểm đó . Mà cây đó là cây có độ dài nhỏ nhất . Vì vậy tồn tại một điểm là trung gian T (có thể trùng với 1 trong ba địa điểm đó) . Thì tổng đường truyền từ T đến 3 đỉnh đó phải nhỏ nhất . Tức là ta sẽ dùng thuật toán Ploy . Sau đó tìm đỉnh nào có tổng khoảng cách nhỏ nhất đến ba đỉnh làn nhỏ nhất thì các đường nối đó chính là các đường nối thoả mãn .

Bài toán 18 : **Truyền tin trên mạng**

Đề Bài :

Trong một mạng gồm N máy tính đánh só từ 1 đến N . Sơ đồ nối mạng được cho bởi hệ thống gồm M kênh nối trực tiếp giữa một số cặp máy trong mạng . Biết chi phí truyền một đơn vị thông tin theo mỗi kênh nối của mạng .

Người ta cần chuyển một bức thông điệp từ máy s đến t . Để đảm bảo an toàn , người ta muốn chuyển bức thông điệp này theo hai đường truyền tin khác nhau (tức là không có kênh nào của mạng được sử dụng trong cả hai đường truyền tin) . Chi phí của một đường truyền tin được hiểu là tổng chi phí trên các kênh của nó .

Yêu cầu : Giả sử bức thông điệp có độ dài là 1 đơn vị thông tin ,hãy tìm cách chuyển thông điệp từ s đến t sao cho tổng chi phí chuyển thông điệp (bằng tổng chi phí theo cả hai đường truyền tin)là nhỏ nhất .

Dữ liệu : Vào từ file văn bản Mess.Inp :

- Dòng đầu tiên ghi bốn số M,N,S,T cách nhau bởi dấu cách (N<=100)
- Mỗi dòng thứ i trong số m dòng tiếp theo ghi thông tin về kênh nối thứ i của mạng gồm 3 số di,ci,gi , trong đó di,ci là chỉ số của hai máy tương ứng với kênh này và gi(nguyên dương là chi phí để truyền một đơn vị thông tin từ máy Di đến máy Ci và ngược lại) theo kênh này (i=1,2,..n)

Kết quả : Ghi ra file văn bản Mess.Out :

- Dòng đầu tiên ghi chi phí truyền thông điệp theo cách tìm được
- Dòng thứ hai ghi đường truyền tin thứ nhất dưới dạng dãy có thứ tự các máy bắt đầu từ máy s kết thúc ở máy t .
- Dòng thứ ba ghi đường truyền tin thứ hai dưới dạng dãy có thứ tự các máy bắt đầu từ máy s đến máy t

Ví dụ :

Mess.Inp	Mess.Out
5 7 1 5	24
1 2 3	1 2 3 5
1 4 8	1 4 5
2 3 5	
2 4 4	
3 5 5	
4 3 8	
4 5 3	

Hướng Dẫn :

Chúng ta sẽ thực hiện các bước như sau :

- Đồ thị của bài toán là đồ thị có hướng , ma trận trọng số là C
- Dùng chương trình tìm đường đi ngắn nhất , tìm đường đi ngắn nhất từ s đến t

- Nếu nó không có đường đi tức là không thể truyền tin , hoặc không có hai đường đến t thì sẽ bị loại
- Trên đường đi ngắn nhất mới tìm được , giả sử là : s =a1,a2,...ak=t . thì ta sẽ chuyển trọng số các cung như sau :
  Trọng số của đoạn nối :
  Nếu C[ai,ai+1]<0 thì C[ai,ai+1]:=-C[ai,ai+1] và C[ai+1,ai]:=C[ai,ai+1] .
  Nếu C[ai,ai+1]>0 thì C[ai,ai+1]:=0 và C[ai+1,ai]:=-C[ai+1,ai]
- Sau đó ta lại tìm đường đi ngắn nhất từ s đến t
- Chúng ta sẽ có đường truyền trong mạng như sau : (lấy ra đường truyền)
  + Với đường truyền thứ nhất :

```
i :=s ;
Repeat
        for j:=1 to n do if a[j,i]<0 then break ;
        a[i,j]:=-a[j,i] ; a[j,i]:=a[i,j];inc(tổng số chi phí , a[i,j]);
        i:=j ;(* j sẽ thuộc đường truyền thứ nhất *)
Until i=t ;
```

  + Với đường truyền thứ hai :
  Cũng lặp lại như quá trình ghi ngược của đường đi thứ nhất .

Ta có thể mở rộng cho việc tìm k đường truyền có tổng chi phí nhỏ nhất .

Bài toán 19 : Giải phương trình

Đề bài :

Chúng ta biết hệ phương trình bậc nhất như sau :

```
x1-x2=t1
xi-xj =tk
.....
```

Chúng ta cần giải được một tập nghiệm của phương trình này (nguyên)

Dữ liệu : Vào từ file BPTrinh.Inp :

- Dòng đầu tiên là số N , M : N là số ẩn , M là số hệ phương trình .
- M dòng sau , mỗi dòng ghi 3 số , mỗi dòng ghi 3 số : U,V,K trong đó biểu diễn :
  U-V>K trong đó U,V là hệ số của dãy nghiệm X1,X2..Xn

Dữ liệu cho luôn có nghiệm và các số K luôn nguyên .

Kết quả : Ghi ra file BPTrinh.Out :

- 1 dòng n số X1,X2,..Xn

Ví Dụ :

BPTRINH.INP		BPTRINH.OUT

Hướng dẫn :

Xây dựng đồ thị NxN là đồ thị có hướng . Trong đó i nối với j có giá trị là t khi :

Xi –Xj=t . Ta xây dựng một đỉnh giả S . S nối với tất cả các đỉnh nhưng hệ số cung đó là 0 . (cung này chỉ có chiều từ S đến nó mà thôi) . Sau đó giải bài toán tìm đường đi dài nhất cho đò thị đó từ s đến tất cả các đỉnh khác . Thì nghiệm của độ dài ngắn nhất đó chính là nghiệm của hệ phương trình .

Bài toán 20 : Phân nhóm.

Đề bài :

Một hội nghị có N người và có M quan hệ quen biết giữa một số cặp người. Hãy tìm cách phân N người thành nhóm thoả mãn :

+ Mỗi người phải ở một trong hai nhóm.
+ Mỗi nhóm phải có ít nhất một người.
+ Hai ngừoi trong cùng một nhóm thì quen biết nhau.
+ Chêch lệch số người hai nhóm là nhỏ nhất.

Dữ Liệu :

- Dòng đầu ghi N, M (N <= 100)
- M dòng tiếp theo, mỗi dòng thể hiện một quan hệ quen biết gồm hai số u, v thẻ hiện người u quen người v.

Kết Quả :

Nếu không có cách phân chia thoả mãn đầu bài, thì thông báo vô nghiệm, ngược lại output gồm hai dòn, mỗi dòng là danh sách những người trong môt nhóm.

Hướng Dẫn :

Ta xây dựng đồ thị vô hướng gồm N đỉnh, mỗi đỉnh của đồ thị tương ứng 1 người. Hai đỉnh có cạnh nối nếu hai người tương ứng không quen biết nhau.

Ta tìm các thành phần liên thông của đồ thị. Trên mỗi thành phần liên thông, nếu có tồn tại chu trình lẻ thì bài toán vô nghiệm, ngược lại ta có thể phân tập đỉnh thành 2 lớp mà mọi cạnh chỉ có thể nối 2 đỉnh thuộc 2 lớp khác nhau (xem thêm bài 93). Ta có thể chỉ ra rằng, với mỗi thành phần liên thông thì cách phân chia như thế là duy nhất không kể thứ tự, khi đó hai tập từ phép phân lớp trên sẽ phải thuộc hai nhóm trong phép phân nhóm thoả mãn yêu cầu đề bài.

Vấn đề còn lại là tìm cách chọn tập nào trong từng phân lớp để độ chênh lệch là nhỏ nhất. Ta giải tiếp bằng qui hoạch động. Ta đánh số các thành phần liên thông theo thứ tự từ 1 đến m, giả sử a1(i) , a2(i) là số phần tử trong từng tập trong phép phân lớp của thành phần liên thông i . Hàm F(i,j) cho biết có thể chọn trong các thành phần liên thông từ 1 đến i, mỗi thành phần liên thông chọn một tập trong phân lớp của nó, để có một tập gồm j đỉnh hay không.

Dễ thấy C(i,j) Chỉ cần được tính từ C(i-1.j-a1(i)) và C(i-1,j-a2(i)), tương ứng chọn tập nào trong thành phần liên thông thứ i.

Bài toán chênh lệch nhỏ nhất tương đương tìm k lớn nhất sao cho k <= n div 2 , và C(m,k) nhận giá trị đúng (có cách chọn tập thoả mãn). Khi đó, k là số phần tử của nhóm ít hơn trong hai nhóm của cách phân nhóm tối ưu.

Bài toán 21 : Tìm đường đi dài nhất thứ k của đồ thị

Đề bài

Cho dồ thị vô hướng có trọng số gồm n đỉnh , người ta cần đi từ đỉnh 1 đến đỉnh N . Do nhiều yêu cầu khách quan , người ta không chỉ muốn biết đường đi ngắn nhất giữa hai đỉnh đó mà còn cần biết k đường đi có độ dài ngắn nhất giữa 1 và N . Bạn hãy viết chương trình tìm k đường đi ngắn nhất giữa 1 và N .

Chú ý : đường đi ở dây có thể lặp đỉnh và cạnh.

Dữ Liệu :

- Dòng đầu ghi N là số đỉnh của đồ thị và số nguyên dương k (0 < k <= N <= 100)
- Các dòng tiếp theo , mỗi dòng mô tả một cạnh của đồ thị gồm ba số nguyên dương : u , v , c cho biết đường hai chiều nối với v có độ dài c. Không có hai đường nào cùng nối một cặp điểm.

Kết Quả :

- Dòng đầu ghi d là độ dài đường đi ngắn thứ k .
- Dòng thứ hai mô tả đường đi gồm dãy các đỉnh trên đường đi bắt đầu từ 1 kết thúc ở N .

Hướng Dẫn :

Gọi F(i,j) độ dài đường đi ngắn thứ j từ 1 đến i. Ta áp dụng thuật toán Dijstra :

Khởi tạo các F(i,j) có giá trịvô cùng trừ C(1,1) bằng 0.

Mỗi lần tìm một (i,j) chưa đánh dấu có nhãn nhỏ nhất .

Từ i cập nhật các đỉnh kề vời nó : Mỗi lần thêm một đương mới đến đỉnh u thì danh sách các đường đi lại được cập nhật và ta chỉ lưu k đường đi đầu tiên mà thôi. Sau khi cập nhật các đỉnh kề thì đánh dấu i,j để lần sau không tìm lại nữa. Cứ như thế cho đến khi không còn nhãn chưa đánh dấu .

Phần 3 :

Bài Toán Luồng Cực Đại Trong Mạng

Bài toán luồng cực đại trong mạng là một trong số những bài toán tối ưu trên đồ thị tìm được những ứng dụng rông rãi trong thực tế cũng như ứng dụng

trong lý thuyết tổ hợp .Bài toán được đề xuất vào đầu những năm 1950 ,và gắn liền với tên tuổi của hai nhà toán học Mỹ là Ford và Fulkerson .Trong chương này chúng ta sẽ trình bày thuật toán của Ford và Fulkerson để giải bài toán đặt ra nêu một số ứng dụng của bài toán .

I . Các Định nghĩa cơ bản :

1. Mạng

- Định Nghĩa : Ta gọi mạng là đồ thị có hướng G=(V,E) trong đó có duy nhất một đỉnh s không có cung đi qua vào gọi là điểm phát , duy nhất một đỉnh t không có cung đi ra gọi là điểm thu cà mỗi cung $e=(v,w) \in E$ được gán với một số không âm c(e)=c(v,w) gọi là khả năng thông qua của cung e .

2. Luồng Trong Mạng :

- Định Nghĩa :

Giả sử cho mạng G=(V,E) . Ta gọi luồng f trong mạng G=(V,E) là ánh xạ $f: E \to R_+$ gán cho mỗi cung $e=(v,w) \in E$ một số thực không âm f(e)=f(v,w) , gọi là luồng trên cung e ,thoả mãn các điều kiện sau ;

+ Luồng trên mỗi cung $e \in E$ không vượt qua khả năng thông qua của nó :

$$0 \le f(e) \le c(e)$$

+ Điều kiện cân bằng luồng trên mỗi đỉnh của mạng : Tổng luồng trên các cung đi vào đỉnh v bằng tổng luồng trên các cung đi ra khỏi đỉnh v,nếu $v \neq s, t$:

$$Div_f(v) = \sum_{w \in \Gamma^-(v)} f(w,v) - \sum_{w \in \Gamma^+(v)} f(v,w) = 0$$

Trong đó $\Gamma^-(v)$ - tập các đỉnh của mạng mà từ đó có cung đến v, $\Gamma^+(v)$- tập các đỉnh của mạng mà từ đó có cung đến nó :

$$\Gamma^-(v) = \{ w \in V : (w,v) \in E \} \quad , \Gamma^+(v) = \{ w \in V : (v,w) \in E \}$$

+ Giá trị của luồng là số :

$$Val(f) = \sum_{w \in \Gamma^+(s)} f(s,w) = \sum_{w \in \Gamma^-(v)} f(w,t)$$

3. Bài Toán Luồng Cực Đại Trong Mạng :

" Cho mạng G=(V,E) . Hãy tìm luồng f^* trong mạng với giá trị luồng $val(f^*)$ là lớn nhất . Luồng như vậy ta sẽ gọi là luồng cực đại trong mạng "

4. Lát Cắt

Định Nghĩa :

Ta gọi lát cắt (X,X^*) là một cách phân hoạch tập đỉnh V của mạng ra thành hai tập X và $X^* = V \setminus X$, trong đó $s \in X$ và $t \in X^*$. Khả năng thông qua của lát cắt (X,X^*) là số :

$$c(X,X^*) = \sum_{v \in X, w \in X^*} c(v,w)$$

lát cắt với khả năng thông qua nhỏ nhất được gọi là **lát cắt hẹp nhất.**

- Bổ đề : Giá trị của mọi luồng f trong mạng luôn nhỏ hơn hoặc bằng khả năng thông qua của lát cắt (X,X^*) bất kỳ trong nó : $val(f) \le c(X,X^*)$.
- Hệ Quả : Giá trị luồng cực đại trong mạng không vượt quá khả năng thông qua của lát cắt hẹp nhất trong mạng

Ford và Fulkerson đã chứng minh rằng giá trị luồng cực đại trong mạng đúng bằng khả năng thông qua của lát cắt hẹp nhất .Để có thể phát biểu và chứng minh kết quả này chúng ta sẽ cần thêm một số khái niệm .

Giả sử f là một luồng trong mạng G=(V,E) .Từ mạng G=(V,E) ta xây dựng đồ thị có trọng số trên các cung được xác định theo quy tắc sau :

- Nếu $e=(v,w) \in E$ với $f(v,w)=0$ thì $(v,w) \in E_f$ vơi trọng số c(v,w);
- Nếu $e=(v,w) \in E$ với $f(v,w)=c(v,w)$,thì $(w,v) \in E_f$ với trọng số f(v,w)
- Nếu $e=(v,w) \in E$ với $0< f(v,w)<c(v,w)$,thì $(v,w) \in E_f$ với trọng số c(v,w)-f(v,w) và $(w,v) \in E_f$ với trọng số f(v,w).

Các cung của G_f đồng thời cũng là cung của G được gọi là cung thuận , các cung còn lại gọi là cung nghich .Đồ thị G_f được gọi là đồ thị tăng luồng

Giả sử P = (s=V0,V1,..Vk=t) là một đường đi từ s đến t trên đồ thị tăng luồng G_f .Gọi δ là gí trị nhỏ nhất của các trọng số của các cung trên đường đi P . Xây dựng luồng f ' trên mạng G theo quy tắc sau :

$$f'(u,v) = \begin{cases} f(u,v)+\delta & \text{nếu } (u,v) \in P \text{ là cung thuận ,} \\ f(u,v)-\delta & \text{nếu } (u,v) \in P \text{ là cung nghịch,} \\ f(u,v) & \text{nếu } (u,v) \notin P . \end{cases}$$

Dễ dàng kiểm tra được rằng f ' được xây dựng như trên là luồng trong mạng và $val(f') = val(f)+s$. Ta sẽ gọi thủ tục biến đổi luồng vừa nêu là tăng luồng dọc theo đường P .

5. Đường Tăng Luồng

Định Nghĩa :

Ta gọi đường tăng luồng là mọi đường đi từ s đến t trên đồ thị tăng luồng G(f).

Định Lý 1 :

Các mệnh đề dưới đây là tương đương :

+ f là luồng cực đại trong mạng

+ Không tìm được đường tăng luồng f
+ Val(f) = c(X,X^*) với một lát cắt (X,X^*) nào đó .

Định Lý 2 :

Luồng cực đại trong mạng bằng khả năng thông qua của lát cắt hẹp nhất .

Định Lý 3 :

Nếu tất cả các khả năng thông qua là các số nguyên thì luôn tìm được luồng cực đại với luồng trên các cung là các số nguyên .

6 . Thuật toán tìm luồng cực đại :

```
Procedure    Max_Flow ;    { Main Procedure  }
{  Thuật toán Ford -Fulkerson  }
Begin
        { Khởi tạo :  Bắt đầu từ luồng với giá trị 0 }
        Fillchar(F,Sizeof(F),0);
        Stop:=False;
        While Not Stop Do
        Begin
                {Thủ Tục Tìm Đường Tăng Luồng }
                If < Tìm Được đường tăng luồng P > Then < Tăng Luồng
                Dọc Theo P> Else Stop:=False;
        End;
End;
```

Trong Đó Ta Có Các Thủ Tục :

```
{P[v] , E[v] là nhãn của đỉnh v  }
{ V_T - Danh sách các đỉnh có nhãn nhưng chưa xét }
{ C[u,v] - Khả năng thông qua của cung (u,v) ,u,v∈ V}
{  F[u,v] -Luồng trên cung (u,v) (u, v ∈ V}
Procedure    Find_Path;
Begin
        p[s]:=s;
        e[s]:=maxint;
        V_T = +∞;
        PathFuond:=True;
        While V_T ≠ ∅ Do
        Begin
                u ⇐ V_T ; { Lấy u từ V_T }
```

```
                For v∈ V\VT Do
                Begin
                        If (C[u,v]>0)and(F[u,v]<C[u,v]) Then
                        Begin
                                P[v]:=u;
                                e[v]:=Min{E[u],C[u,v]-F[u,v]};
                                VT = VT ∪ {v}; {Nạp v vào danh sách đỉnh có
                                nhãn}
                                If v=t then exit;
                        End;
                        If (C[v,u]>0)And(F[v,u]>0) Then
                        Begin
                                p[u]:=-v;
                                e[u]:=Min{e[u],f[v,u]}
                                VT = VT ∪ {v} ; { Nạp v vào danh sách đỉnh có
                                nhãn}
                        End;
                End;
        End;
        PathFound:=False;
End;

Procedure    Inc_Flow;
Begin
        v:=p[t];
        u:=t;
        tang:=e[t];
        While u<>s Do
        Begin
                If v>0 then inc(f[v,u],tang) esle
                begin
                        v:=-v;
                        dec(f[u,v],tang);
                end;
                u:=v;
                v:=p[u];
        End;
End;
```

6. Một Số Bài Toán Luồng Tổng Quát :

a. Mạng với nhiều điểm phát và điểm thu :

Xét mạng G với p điểm phát S1,S2,...Sp và q điểm thu T1,T2,..Tq . Giả sử rằng luồng có thể đi từ một điểm phát bất kỳ đến tất cả các điểm thu .Bài toán tìm luồng cực đại từ các điểm phát đến các điểm thu có thể đưa về bài toán với một điểm phát và một điểm thu bằng cách đưa vào một điểm phát giả s và một điểm thu giả t và các cạnh nối s với tất cả các điểm phát và các cạnh nối các điểm thu với t .

b. Bài toán với khả năng thông qua của các cung và các đỉnh

Giả sử trong đồ thị G, ngoài khả năng thông qua của các cung c(u,v), ở mỗi đỉnh $v \in V$ còn có khả năng thông qua của đỉnh là d(v), và đòi hỏi tổng luồng đi vào đỉnh v không được vượt quá d(v), tức là :

$$\sum_{w \in V} f(w,v) \leq d(v)$$

Cần phải tìm luồng cực đại giữa s và t trong mạng như vậy.

Xây dựng một mạng G' sao cho : mỗi đỉnh v của G tương ứng với 2 đỉnh v^+, v^- trong G' ,mỗi cung (u,v) trong G ứng với cung (u^-,v^+) trong G',mỗi cung (v,w) trong G ứng với cung (v^-,w^+) trong G ' .Ngoài ra ,mỗi cung (v^+,v^-) trong G' có khả năng thông qua là d(v), tức là bằng khả năng thông qua của đỉnh v trong G.

Do luồng đi vào đỉnh v^+ phải đi qua cung (v^+,v^-) với khả năng thông qua d(v), nên luồng cực đại trong G' sẽ bằng luồng cực đại trong G với khả năng thông qua của các cung và các đỉnh .

c. Mạng trong đó khả năng thông qua của mỗi cung bị chắn hai phía

Xét mạng G mà trong đó mỗi cung (u,v) có khả năng thông qua (cận trên của luồng trên cung) c(u,v) và cận dưới của luồng là d(u,v) . Bài toán đặt ra là liệu có tồn tại luồng tương thích từ s đến t , tức là luồng $\{ f(u,v) : u,v \in V\}$ thoả mãn thêm ràng buộc

$$d(u,v) \leq f(u,v) \leq c(u,v) \quad , \forall (u,v) \in E$$, hay không ?

Đưa vào mạng G đỉnh phát giả S_a và đỉnh thu giả T_a và xây dựng mạng G_a theo quy tắc :

Mỗi cung (u,v) mà $d(u,v) \neq 0$ sẽ tương ứng với 2 cung (S_a,v) và (u,T_a) với khả năng thông qua là d(u,v) . Giảm c(u,v) đi d(u,v) tức là thay khả năng thông qua của cung (u,v) bởi c(u,v)-d(u,v) còn cận dưới của nó đặt bằng 0 . Ngoài ra thêm vào cung (t,s) với $c(t,s) = \infty$.

Định Lý :

- Nếu luồng lớn nhất trong mạng G_a từ S_a đến T_a bằng d^* thì tồn tại luồng tương thích trong G.
- Nếu luồng lớn nhất trong mạng G_a từ S_a đến T_a là khác d^* thì không tồn tại luồng tương thích trong G .

8. Một Số ứng Dụng Trong Tổ Hợp:

Bài toán luồng cực đại có rất nhiều ứng dụng trong các bài toán tổ hợp , hay các bài toán tối ưu khác . Chúng ta có thể xét một số ví dụ mà thường gặp nhất của bài toán luồng . Còn các ứng dụng khác thì các bạn có thể xem ở phần 9 của phần này . Cón bây giờ chúng ta bắt đầu xét từ những ví dụ đơn giản :

a. Bài toán đám cưới vùng quê :

- Bài toán : *" có m chàng trai ở một lang quê nọ. Đối với mỗi chàng trai ta biết các cô gái mà anh ta vừa ý . Hỏi khi nào thì có thể tổ chức các đám cưới trong đó các chàng trai nào cũng sánh duyên cới cô gái mà mình vừa ý "*

Ta có thể xét đồ thị biểu diễn các chàng trai và các cô gái bằng các đỉnh ,còng biểu thị sự vừa ý của các chang trai đối với các cô gái . Khi đó ta thu được một đồ thị hai phía . Lúc đó cách giải thì chúng ta đưa về dạng 7.a mà ta vừa nêu .Bài toán này là một trường hợp riêng trong bài toán cặp ghép 2 phía mà các bạn có thể xem rõ trong phần Cặp Ghép

b. Bài toán về hệ thống đại diện chung :

Cho tập m phần tử X= {Z1,Z2,..Zm }.Giả sử < A1,A2,..An> và <B1,B2,..Bn> là hia dãy các tập copn của tập X .Dãy gồm n phần tử khác nhau của X : < a1,a2,..an> được gọi là hệ thống các đại diện chung của hai dãy đã cho nếu như tìm được một hoán vị δ của tập {1,2,..n} sao cho <a1,a2,..an> là hệ thống các đại diện phân biệt của hai dãy <A1,A2,..An> và $<B_{\delta(1)},B_{\delta(2)},..B_{\delta(n)}>$, tức là điều kiện sau được thoả mãn : $a_i \in A_i \cap B_{\delta(i)}$, i=1,2,,..n .Xây dựng mạng G=(V,E) với tập đỉnh :

$$V = \{s,t\} \cup \{X_1,X_2,..X_n\} \cup \{U_1,U_2,..U_m\} \cup \{v_1,v_{2,..}v_m\} \cup \{Y_1,Y_2,..Y_n\}$$

trong đó đỉnh X_i tương ứng với tập A_i đỉnh y_i tương ứng với tập B_i ,các phần tử U_i,V_j tương ứng với phần tử Z_j . Tập các cung của mạng G được xác định như sau:

$$E = \{(s,x_i): 1 \le i \le n\} \cup \{(x_i,u_j): \text{với } Z_j \in A , 1 \le i \le n , 1 \le j \le m \} \cup \{(u_j,v_j): 1 \le j \le m \} \cup \{(v_j,y_j) : \text{với } Z_j \in B_i, 1 \le j \le m \} \cup \{ (y_i,t): 1 \le i \le n\}$$

Khả năng thông qua của tất cả các cung được đặt bằng 1. Dễ thấy rằng hệ thống đại diện chung của hai dãy < A1,A2,..An> và <B1,B2,..Bn> tồn tại khi và chỉ khi trong mạng G =(V,E) tìm được luồng với giá trị n . Để xét sự tồn tại của luồng như vậy có thể sử dụng thuật toán tìm luồng cực đại từ s đến t trong mạng G=(V,E).

c. Một bài toán tối ưu khác mà chúng ta thường gặp :

Nội dung của bài toán này thông thường được biểu diễn dưới các dạng sau :

- Dạng 1 : Bài Toán Phân Nhóm Sinh Hoạt

Có m sinh viên và n nhóm sinh hoạt chuyên đề . Với mỗi sinh viên i , biết :

Ai,j =1 nếu sinh viên i có nguyện vọng tham gia vào nhóm j,

Ai,j=0 ,nếu ngược lại

và P_i là số lượng nhóm chuyên đề mà sinh viên i phải tham dự , i=1,2,..m; j=1,2,..n .

Trong số các cách phân các sinh viên vào các nhóm chuyên đề mà họ có nguyện vọng tham gia và đảm bảo mỗi sinh viên i phải tham gia đúng P_i nhóm , hãy tìm cách phân phối với số người trong nhóm có nhiều sinh viên tham gia nhất là nhỏ nhất có thể được .

- Dạng 2 : Bài Toán Lập Lịch Cho Hội Nghị

Một hội nghị có m tiểu ban , mỗi tiểu ban cần sinh hoạt trong một ngày tại phòng họp phù hợp với nó . Có n phòng họp dành cho việc sinh hoạt của các tiểu ban . Biết :

Ai,j=1 , nếu phòng họp i thích hợp với tiểu ban j

Ai,j=0 , nếu ngược lại . (i=1,2,..m;j=1,2,..n . Hãy bố trí các phòng họp cho các tiểu ban sao cho hội nghị kết thúc sau ít ngày nhất .

với tiểu ban j

Ai,j=0 , nếu ngược lại . (i=1,2,..m;j=1,2,..n . Hãy bố trí các phòng họp cho các tiểu ban sao cho hội nghị kết thúc sau ít ngày nhất .

Cách Giải :

Các bài toán trên có thể được mô hình hoá như sau :

$$f(x1,x2,..xn) = \max_{1 \le i \le m} \sum_{j=1}^{n} Xij \rightarrow \min \qquad (1)$$

Với điều kiện :

$$\sum_{j=1}^{n} Aij.\ Xij = P_i \quad , i=1,2,..m \qquad (2)$$

$$Xij=0 \text{ hoặc } 1 \ , j=1,2,...n \qquad (3)$$

Trong đó $Aij \in \{0,1\}$, i=1,2,..m ; j=1,2,..n , P_i- nguyên dương ,i=1,2,..m

Ta có một số nhận xét sau :

- Bài toán (1)-(3) có phương án tối ưu khi và chỉ khi :

$$\sum_{m} Aij \ge P_i \quad ,i=1,2,..m \qquad (4)$$

Ký hiệu : $\delta = \sum_{m} P_i$

$$\sum_{i=1}$$

Do (4) là điều kiện cần để giải bài toán (1)-(3) có phương án , nên trong phần tiếp theo ta sẽ luôn giả thiết rằng điều kiện này được thực hiện

Bây giờ ta sẽ chỉ ra rằng việc giải bài toán (1)-(3) có thể dẫn về việc giải một số hữu hạn bài toán luồng cực đại trong mạng . Trước hết , với mỗi số nguyên dương k, xây dựng mạng G(k)=(V,E) với tập đỉnh

$V = \{s\} \cup \{U_i, i=1,2,..m\} \cup \{W_j : j=1,2,..n\} \cup \{t\}$

trong đó s là điểm phát , t là điểm thu , và tập cung

$E = \{(s,u_i): i=1,2,..m\} \cup \{(u_i,w_j) : i=1,2,..m; j=1,2,..n\} \cup \{(w_j,t) : j=1,2,..n\}$

Mỗi cung $e \in E$ được gán với khả năng thông qua q(e) theo quy tắc sau :

$q(s,u_i)=Pi$, i=1,2,..m

$q(u_i,w_j) =Aij$, i=1,2,..m ; j=1,2,...n

$q(w_j,t)=k$, j=1,2,...n

Ta có một số bổ đề sau :

Bổ Đề 1: Giả sử đối với số nguyên dương k nào đó , luông cực đại nguyên ξ^* trong mạng G(k) có giá trị là δ . Khi đó $X^*=(x^*_{ij})_{m \times n}$ với các thành phần được xác định theo công thức :

$X^*_{ij} = \xi^*(u_i,w_j)$, i=1,2,...m ; j=1,2,...n

là phương án của bài toán (1)-(3).

Bổ Đề 2 : Giả sử $X^*=(x^*_{ij})$ là phương án tối ưu và k^* là giá trị tối ưu của bài toán (1)-(3) . Khi đó luồng cực đại trong mạng $G(k^*)$ là có giá trị δ

Bổ Đề 3 : nếu k=m thì luồng cực đại trong mạng G(m) có giá trị δ

Từ bổ đề (1) và (2) sau ra việc giải quyết bài toán (1)-(3) dẫn về việc tìm giá trị K^* nguyên dương nhỏ nhất sao cho luồng cực đại trong mạng $G(k^*)$ có giá trị δ . Bổ đề 3 cho thấy giá trị $k^* \in [1,m]$. Vì vậy để giải bài toán tìm luồng cực đại trong mạng có thể sử dụng thuật toán đa thức như nói trên . Từ đó ta đã giải quyết được bài toán (1)-(3) .

9. Một Số ứng Dụng Khác :

Bài toán 22 : Máy phục vụ đổi tiền

Đề Bài :

Có M máy tự động đổi tiền, mỗi một máy chỉ phục vụ thu một số loại thẻ tín dụng. Giả sử cùng một lúc có K khách đến, mỗi khách cần đổi tiền bằng thẻ tín dụng của mình. Hãy chỉ dẫn cho khách đến các máy tương ứng để cùng một lúc, số khách được phục vụ là nhiều nhất.

Giả sử các máy được đánh số từ 1 đến M, các khách được đánh số từ 1 đến K. Các loại thẻ được phân biệt bằng bằng các chữ cái lớn lấy trong bừng chữ cái tiếng Anh (gọi là mã thẻ).

Dữ liệu : Vào cho trong file văn bản, gồm:

- Dòng đầu ghi hai giá trị M và K (viết cách nhau ít nhất một dấu trắng).
- Dòng thứ i trong M dòng tiếp, mỗi dòng ghi một dãy các chữ cái lớn (viết liền nhau), mô tả danh sách các mã thẻ mà máy i phục vụ.
- Dòng cuối ghi dãy gồm K chữ cái lớn (viết liền nhau), chữ cái thứ i là mã thẻ của khách thứ i.

Kết quả : Ghi ra file văn bản gồm:

- Dòng đầu ghi số khách được phục vụ
- Các dòng sau, mỗi dòng ghi một cặp số (viết cách nhau ít nhất một dấu trắng),mô tả một cặp khách - máy, trong đó số đầu là số hiệu khách được phục vụ, số sau là số hiệu máy phục vụ khách.

Giới hạn kích thước M, K ≤ 100

Thí dụ, file dữ liệu vào:

```
4       5
CD
AB
BE
ABC
ACAAD
```

Có file kết quả:

```
3
1       2
2       1
3       4
```

Hướng dẫn :

Xây dựng đồ thị hai phía : Một phía là tập M đỉnh đại diện cho M máy đổi tiền . Một phía là K đỉnh đại diện cho K người đổi . Đỉnh i thuộc phần 2 chỉ nối được đỉnh j của phần 1 khi máy đổi j đổi được tiền của người thứ i . Xây dựng thêm hai đỉnh giả S và T . Trong đó S nối với tất các đỉnh thuộc phần K (người đổi) . Còn tất cả các đỉnh thuộc phần M sẽ nối với T . Tất cả các trọng số trong đồ thị trên là 1 nếu như có đỉnh nối (và 0 nếu không) .

Sau đó tìm luồng cực đại từ S đến T . Thì dòng chảy mạng đó chính là cách phối hợp đổi tiền cần làm .

Bài toán 24 : Hệ đại diện

Đề Bài :

Một khối gồm M học sinh có tên từ 1 đến M. Khối tổ chức N nhóm Tin Học $A_1, A_2, \ldots, A_N$, và N nhóm Thể Thao $B_1, B_2, \ldots, B_N$. Mỗi học sinh phải tham gia ít nhất một nhóm Tin Học và ít nhất một nhóm Thể Thao, và tuỳ theo khả năng, một học sinh có thể tham gia nhiều nhóm Tin Học hoặc Thể Thao. Khối

cần chọn một Ban Đại Diện gồm N người sao cho mỗi nhóm Tin Học và mỗi nhóm Thể Thao đều có ít nhất một thành viên của mình trong Ban Đại Diện này. Hãy tìm cho khối một Ban Đại Diện như vậy.

Dữ liệu : Ghi trong file INP.BL5 với cấu trúc như sau:

- Dòng thứ nhất ghi hai số M và N.
- Dòng thứ i trong M dòng tiếp theo ghi chỉ số các nhóm Tin Học mà học sinh i tham gia.
- Dòng thứ i trong M dòng cuối cùng ghi chỉ số các nhóm Thể Thao mà học sinh i tham gia.

Trên mỗi dòng, hai số liên tiếp cách nhau bởi dấu trống.

Dữ liệu : Ghi ra file OUT.BL5 theo quy cách sau:

- Dòng thứ nhất ghi N tên học sinh được chọn vào Ban Đại Diện,
- Dòng thứ hai ghi chỉ số các nhóm Tin Học mà các học sinh (trong Ban Đại Diện) tham gia theo thứ tự tương ứng với kết quả dòng 1,
- Dòng thứ ba ghi chỉ số các nhóm Thể Thao mà các học sinh (trong Ban Đại Diện) tham gia theo thứ tự tương ứng với kết quả dòng 1.

Hướng Dẫn :

Xây dựng đồ thị nối giữa các học sinh với nhóm mình tham dự . Tức là xây dựng đồ thị 3 phía :

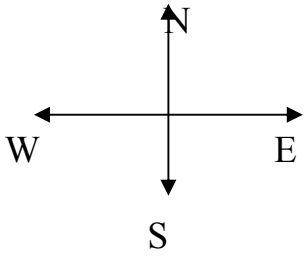
	A1	Học sinh1	B1		
	A2	Học sinh 2	B2		
S	.....	.....		.....	T
	.....	.....		.....	
	An	Học sinh M	Bn		

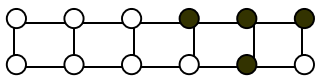
Sau đó dùng hai đỉnh giả s và t nối với hai đầu . Ta tìm luông cực đại nối từ S đến T . Thì sau đó các cặp nối chính là hệ đại diện cho học sinh .

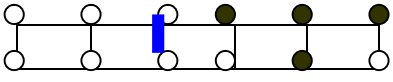
Bài toán 25 : Bảng điện

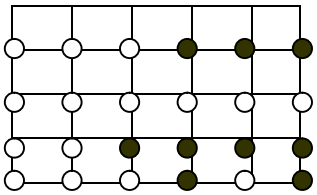
Đề bài :

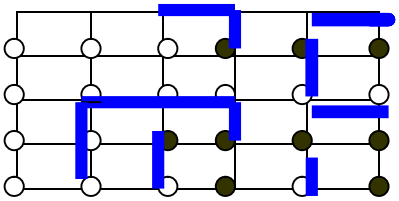
Một lưới ô vuông được phủ trên một bảng điện hình vuông . Vị trí nằm trên giao của 2 đường kẻ của lưới được gọi là nút . Tất cả có NxN nút trên lưới .











(Hình vẽ tuy xấu nhưng hy vọng các bạn hiểu được nội dung bài toán)

Có một số nút chứa tiếp điểm . Nhiệm vụ của bạn là cần nối các tiếp điểm với các nút ở trên biên của bảng bởi các đoạn dây dẫn (gọi là các mạch) . Các mạch chỉ được chạy dọc theo các đường kẻ của lưới (nghĩa là không được chạy theo đường chéo) . Hai mạch không được phép có điểm chung . Vì vậy hai mạch bất kì không được phép chạy qua cùng một đoạn đường kẻ của lưới cũng như không được chạy qua cùng một nút của lưới . Các mạch cung không được chạy dọc theo các đoạn kẻ của lưới ở trên biên (mạch phải kết thúc khi nó gặp biên) và cũng không được chạy qua nút chứa tiếp điểm khác .

Ví Dụ : bảng điện vad các tiếp điểm được cho trong hình 2a . Nút tô đậm trong hình vẽ thể hiện vị trí các tiếp điểm .

Yêu cầu : Viết chương trình cho phép nối được một số nhiều nhất các tiếp điểm với biên . Các tiếp điểm ở trên biên đã thoả mãn đòi hỏi đặt ra , vì thế không nhất thiết phải thực hiện mạch nối chúng . Nếu như có nhiều lời giải thì chỉ cần đưa ra một trong số chúng .

Dữ liệu : Vào từ file ELE.INP :

- Dòng đầu tiên ghi số nguyên N (3<N<=15)
- Mỗi dòng trong số n dòng tiếp theo chứa N kí tự phân cách nhau bởi một dấu cách . Mỗi kí tự chỉ là 0 hoặc 1 . Kí tự 1 thể hiện tiếp điểm ,ký tự 0 thể hiện nút không có tiếp điểm trên vị trí tương ứng của lưới . Các nút được đánh số từ 1 đến N xN theo thứ tự từ trái qua phải , từ trên xuông dưới . Chỉ số của nút chứa tiếp điểm sẽ là chỉ số của tiếp điểm

Kết quả : Ghi ra file ELE.OUT :

- Dòng đầu tiên chứa k là số tiếp điểm lớn nhất có thể nối với biên bởi các mạch .
- Mỗi dòng trong số k dòng tiếp theo mô tả mạch nối một trong k tiếp điểm với biên theo quy cách sau : Đầu tiên là chỉ số của tiếp điểm được nối , tiếp đến là dãy các ký tự mô tả hướng của mạch nối E : đông , W : tây , N : bắc , S : Nam . Giữa chỉ số và dãy ký tự mô tả hướng phải có đúng 1 dấu cách , còn giữa các ký tự trong dãy không được có dấu cách .

Kết quả phải đưa ra theo thứ tự tăng dần của chỉ số tiếp điểm

Ví Dụ :

ELE.INP		ELE.OUT

6 0 0 0 1 1 1 0 0 0 0 1 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 1 0 1		6 11 E 16 NWN 17 SE 27 S 28 NWWSS 29 S

Hướng Dẫn :

Ta xây dựng đồ thị N^2xN^2 đỉnh . Trong đó các đỉnh đại diện cho một nút . Hai đỉnh được nối với nhau khi nút của chúng kề nhau trong bảng (có đường nối trực tiếp với nhau) . Tức là một đỉnh có tối đa là bốn đỉnh kề . Xây dựng hai đỉnh giả S và T . Trong đó S sẽ nối với các đỉnh thuộc nút tiếp điểm (trừ các tiếp điểm ở biên) . Còn T sẽ nối với các đỉnh có nút ở biên . Sau đó Tìm luồng cực đại từ S đến T . Thì đường nối (hay đường đi trong mạng vừa tìm kiếm đó) sẽ là đường nối có nhiều tiếp điểm được nối nhất . (Với điều kiên luồng này là luộng chặn – tức là không có đỉnh nào (trừ hai đầu) là có quá hai cung vào nó) .

Bài toán 26 : Bảo vệ thành phố

Đề bài :

Một mạng lưới gồm N thành phố, và một số đường một chiều nối các cặp thành phố (giữa hai thành phố có thể có nhiều đường nối một chiều). Quân địch đang tập trung ở thành phố N, định tiến công ta ở thành phố 1, và chúng sẽ tiến công trên tất cả các con đường chưa được bảo vệ để tiến vào thành phố 1. Bộ chỉ huy ta cần xác định số quân ít nhất trên các con đường để chặn địch tiến về thành phố 1.

Dữ liệu : Vào từ File text **Baove.Inp** gồm:

- Dòng đầu ghi số N (N <= 5000).
- Các dòng tiếp theo cho đến hết file, mỗi dòng mô tả một đường gồm u, v, s cho biết có đoạn đường một chiều từ u đến v, và phải cần ít nhất s quân để chặn đic h trên đường này (s <= 65000). Có không quá 10000 đường.

Kết quả : Ghi ra File text **Baove.out** cấu trúc như sau:

- Dòng thứ nhất ghi số P là số quân ít nhất cần huy động.
- Các dòng tiếp theo mỗi dòng mô tả việc bố trí quân bao gồm u, v, s với ý nghĩa trên các đường từ u đến v ta bố trí s quân.

Ví Dụ :

baove.inp		baove.out
7 3 1 7 6 1 47 7 2 12 7 2 20 2 4 11 7 3 15 5 3 20 7 4 10 4 2 21 4 6 16 6 5 23		23 3 1 7 4 6 16

Hướng Dẫn :

Chúng ta sẽ tìm luồng cực đại từ S đến T . Thì số lính cần bố trí chính bằng luồng cực đại đó . Sau đó tìm trên đường chảy của luồng , ta tìm các cầu của đò thị đó . Thì những đoạn đường đó chính là nơi cần bố trí . Bài toán khó ở các điểm sau :

- Gải bài toán luồng cho một đồ thị lớn
- Tìm các con đường cần mai phục (bài toán này thì chúng ta dựa vào giá trị luồng tại mỗi đỉnh sẽ ra kết quả cần tìm)

Bài toán 27 : Lrec

Đề bài :

Cho một bảng số gồm m dòng n cột (1<=m<n<=100) có tính chất thú vị sau : " *mối dòng của bảng ghi các số tự nhiên từ 1 đến n sao cho không có cột nào chứa hai số giống nhau* " Bạn cần bổ sung vào bảng số này k dòng nữa (K<=n-m) sao cho bảng thu được vân có tính chất giống như bảng ban đầu .

Dữ liệu :

Vào từ file văm bản : Lrec.Inp :

- Dòng đầu tiên chứa ba số m , n , k
- Dòng thứ i trong số m dòng tiếp theo chứa các phần tử của dòng thứ i của bảng đã cho

Kết quả : ghi ra file văn bản Lrec.out : mỗi dòng ghi các phần tử của một trong k dòng bổ sung .

ví dụ :
lrec.inp :
4 2 1
1 2 3 4
4 3 2 1
lrec.out
2 1 4 3
Hướng Dẫn :

Chúng ta thấy rằng khi thêm một số vào thì số đó phải thoả mãn điều kiện bài toán ra . Tức là chúng ta sẽ xây dựng một đồ thị hai phía mà các đỉnh một bên là đại diện cho các cột và một bên đại diện cho các số từ 1 đến N . Việc thêm vào một hàng tứclà chúng ta tìm được một luồng đi từ đỉnh s đến đỉnh t (trong đó s và t là hai đỉnh giả nối với N cột và N số đó) .

Cứ như vậy việc thêm nếu thoả mãn thì có luồng đi qua nó . Hơn thế nữa ma trận này là ma trận đặc biệt nên bao giờ cũng tìm được một luồng thoả mãn .

Bài toán 28 : Chiếu Kho Xăng

Đề Bài :

Khu vực đặt các bể xăng của một tổng đại lý Xăng dầu có dạng một hình chữ nhật được chia thành m × n ô vuông .Các ô vuông đánh toạ đọ từ trên xuống dưới , từ trái qua phải theo thứ tự hàng ,cột . Tại k ô của lưới có đặt các bể xăng .Người ta cần xây dựng một hệ thống đèn pha chiếu dọc theo hàng hoặc là cột của lưới ô vuông sao cho mỗi bể chứa phải được chiếu sáng bởi ít nhất một đèn pha chiếu dọc theo hàng hoặc theo cột chứa nó. Biết:

- Ai là chi phí xây dựng đèn chiếu sáng dọc theo hàng i (i=1,2...m)
- Bj là chi phí xây dựng đèn chiếu sáng dọc theo cột j (j=1,2,..n)

Yêu Cầu : Tìm cách xây dựng hệ thống đèn chiếu với tổng chi phí xây dựng là nhỏ nhất

Dữ Liệu: Vào từ file Khoxang.Inp :

- Dòng đầu tiên chứa ba số nguyên dương m,n,k (m,n< 100) ;
- Dòng thứ hai chứa m số nguyên dương A1,A2,...Am
- Dòng thứ ba chứa n số nguyên dương : B1,B2,..Bn
- Dòng thứ i trong số k dòng còn lại chứa toạ độ của bể xăng thứ i (i=1,2,..k)

Kết Quả : ghi ra file Khoxang.Out như sau :

- Dòng đầu tiên ghi tổng chi phí theo cách xây dựng tìm được
- Dòng thứ hai ghi hai số P và Q theo thứ tự là số lượng đèn chiếu dọc theo hàng và cột (ghi số 0 nếu không có)
- P+Q dòng tiếp theo lần lượt các toạ độ của các hàng rồi đến các cột có đặt đèn chiếu ,mỗi dòng ghi 1 số .

Ví Dụ:

Khoxang.Inp	Khoxang.Out	Khoxang.Inp	Khoxang.Out
2 3 4 10 5 12 4 2 1 2 1 3 2 1 2 3	11 1 2 2 2 3	2 3 4 15 17 2 4 6 1 1 2 2 2 3 2 1	12 0 3 1 2 3

Hướng Dẫn :

Bài toán có thể đưa về bài toán luồng cực đại . Gọi X là tập các hàng, Y tập đại diện cho các côt.

Từ đỉnh phát giả s có cung đến các đỉnh thuộc X với giá trị bằng chi phí lắp đèn trên hàng tương ứng, từ mỗi đỉnh thuộc Y có một cung nối tới đỉnh thu t với trọng số bằng chi phí lắp đèn trên hàng tương ứng .

Ngoài ra, nếu hàng x, cột y mà ô (x,y) có kho xăng thì trọng số x,y = +vô cùng ngược lại bằng 0.

Gọi A, B là lát cắt nhỏ nhất, ta chọn tập các hàng không thuộc A, các cột thuộc A đề chọn mắc đèn.

Bài toán 29 : Chọn ô Vuông

Đề bài :

Cho một hình chữ nhật kích thước m x n ô vuông , trong đó có một số ô được tô màu đen (gọi là ô đen) , các ô còn lại tô màu trắng (gọi là ô trắng) . Giả thiết rằng mỗi dòng có ít nhất hai ô đen và m ,n<100 .

Yêu cầu : Cần phải chọn ra 2*m ô đen của lưới sao cho :

- Trong mỗi dòng của lưới có đúng hai ô được chọn
- Số ô được chọn trong cột có nhiều ô được chọn nhất là nhỏ nhất

Dữ liệu : Vào được cho trong file :BL1.INP :

- Dòng đầu tiên ghi các giá trị m,n (cách nhau bởi dấu cách)
- M dòng tiếp theo , mỗi dòng ghi thông tin về mầu các ô trong một dòng của lưới với quy ước : 1 là ô màu đen , 0 là ô màu trắng . Các số viết liền nhau tạo thành một xâu .

Kết quả : Đưa ra file BL1.OUT :

- Dòng đầu tiên số ô được chọn trong cột có nhiều ô được chọn nhất
- M dòng tiếp theo , mỗi dòng thứ i ghi một xâu độ dài n gồm các kí tự 0,1 viết liên nhau với quy ước ký tự 1 ghi nhận vị trí ô được chọn , kí tự 0 ghi nhận vị trí ô không được chọn

Ví Dụ :

BL1.INP
6 8 11011011 01101101 10101010 01101101 11100111 00011100

BL1.OUT
2 10000001 01000001 00101000 01000100 00100100 00011000

Hướng Dẫn :

Chúng ta xây dựng đồ thị hai phía như sau :

Một tập đỉnh là đại diện cho hàng và một tập đỉnh đại diện cho cột .

	Các đỉnh hàng	Các đỉnh cột	
	A1	B1	
	A2	B2	
S	.....	.....	T
	.....	.....	
	Am	Bn	

Đỉnh thứ i của tập 1 sẽ nối với đỉnh thứ j của tập hai khi tại ô[i,j] có màu đen . Ta xây dựng hai đỉnh giả S vả T . S sẽ nối với Tập Ai . Trong đó giá trị trọng số là 2 (còn giá trị trọng số của Ai nối với Bj là bằng 1) . Còn tất cả các đỉnh Bj sẽ nối với T . Nhưng vấn đề trọng số lại cần tính . Do yêu cầu hai của bài toán , nên chúng ta sẽ thử tất cả các giá trị trọng số đó , chọn trọng số nào nhỏ nhất mà đồ thị vẫn có luồng đầy đủ chạy qua . (cách giải như giả bài toán (1)-(3) ở phần ứng dụng của luồng mà ta vừa nêu trên) .Thì cách chọn đó chính là cách chọn ra các ô cần trong bảng (của để bài) .

Phần 4 : Lý Thuyết Ghép Cặp

I. Các Định Nghĩa Cơ Bản:

1. Cặp Ghép:

Xét hai tập hữu hạn X,Y gồm N phần tử :

$X=\{x_1, x_2,....x_n\}$

$Y=\{y1,y2,...y_n\}$

Cặp phần tử (x,y) với $x\in X$, $y\in Y$ được gọi là một cặp ghép . Hai cặp ghép (x,y) và (x',y') được gọi là rời nhau nếu $x \neq x'$ và $y\neq y'$. Tập M gồm các cặp ghép rời nhau được gọi là một tập cặp ghép .

Thông thường bài toán xây dựng các cặp ghép được tiếp cận theo 2 hướng : hoặc thoả mãn một điều kiện ghép cặp nào đấy , khi đó người ta quan tâm đến khả năng ghép cặp tối đa , hoặc lượng hoá việc ghép cặp , khi đó người ta quan tâm đến phương án ghép cặp tối ưu theo các giá trị đã thoả lượng hoá .

Ví số tập ghép cặp là hữu hạn , nên có một phương pháp xây dựng tàm thường là thử tất cả các khả năng thông . Tất nhiên ,số khả năng như vậy là cỡ rất lớn (N!) . Vì thế ,người ta quan tâm đến việc tìm kiếm những thuật giải hữu hiệu ,dễ cài đặt chương trình và có tính khả thi cao . Bài này nhằm giới thiệu một số mô hình ghép cặp như vậy.

2. Cặp Ghép Đầy Đủ:

Một tập cặp ghép sao cho tất cả các phần tử của X và Y đều được ghép cặp (nghĩa là có đủ n cặp với n là số phần tử của X và Y) ,được gọi là một tập cặp ghép đầy đủ.

Rõ ràng mỗi song ánh p: $X \rightarrow Y$ xác định một tập cặp ghép đầy đủ , trong đó mỗi cặp ghép được viết dưới dạng (x,P(x)) ,$x \in X$. Từu đó suy ra có tất cả n! cách xây dựng tập cặp ghép đầy đủ khác nhau .

Với các tập cặp ghép đầy đủ ,một cách tự nhiên ,người ta quan tâm đến tập cặp ghép " tốt nhất " thep một nghĩa nào đó đã được lượng hoá . Tập cặp ghpé này được gọi là tập cặp ghép đầy đủ tối ưu.

3. Bài Toán Ghép Cặp :

Bài toán tìm cặp ghép đầy đủ tối ưu có nhiều mô hình ứng dụng thực tế .Một trong những mô hình này là người ta quan tâm đến việc ghép cặp sao cho có hiệu quả nhất . Để lượng hoá việc ghép mỗi phần tử $x \in X$ với một phần tử $y \in Y$, người ta đưa vào ma trận trọng số C[i,j] (i,j=1,2,..n) với ý nghĩa C[i,j] mô tả hiệu quả của việc ghép x_i với Yj . Bài toán được đặt ra là xây dựng một tập cặp ghép đầy đủ có tổng hiệu quả lớn nhất . Bài toán vừa nêu thương được phát biểu dưới dạng một mô hình thực tế là bài toán phân công dưới đây:

Bài Toán Phân Công Việc : " *Có n người và n công việc . Biết C[i,j] là số tiền làm ra nếu giao công việc j cho người i thực hiện .Hãy tìm cách phân công mỗi người mỗi việc để tổng số tiền làm ra là lớn nhất* " .

II. Giải Bằng Phương Pháp Đồ Thị:

1. Định Lý Cơ Sở:

Việc xây dựng tập cặp ghép đầy đủ tối ưu dựa vào dấu hiệu nhận biết một tập cặp ghép đầy đủ khi nào là tối ưu .Dĩ nhiên việc thử dấu hiệu này không phải là việc so sánh với tất cả các cặp ghép ,mà phải được xây dựng mang tính khả thi .Để làm điều này ,người ta xây dựng hàm số F , xác định trên tập các phần tử $Xi \in X$, $Yj \in Y$ .mà ta gọi là nhãn của các phần tử . Nhãn F được gọi là chấp nhận được nếu thoả mãn bất đẳng thức $F(Xi)+F(Yj)>=C[i,j]$ với mọi $Xi \in X$,$Yj \in Y$ .Tập cặp ghép M và nhãn F được gọi là tương thích với nhau nếu thoả mãn đẳng thức $F(Xi)+F(Yj)=C[i,j]$ với mọi $(Xi,Yj) \in M$ .Nói riêng ,tập cặp ghép rỗng được xem như tương thích với mọi nhãn .

Định Lý : Tập cặp ghép đầy đủ M* là tối ưu khi tồn tại nhãn F chấp nhận được là tương thích với nó .

Chứng Minh :

Giả Sử M là một tập cặp ghép đầy đủ nào đó ,gọi C(M) là tổng hiệu qủa của M :

$$C(M)= \sum_{(Xi,Yj)\in M} (C[i,j])$$

Vì mỗi cặp $(Xi,Yj) \in M$ đi qua các phần tử của X và Y ,mỗi phần tử đúng một lần ,và nhãn F thoả mãn bất đẳng thức $F(Xi)+F(Yj) \geq C[i,j]$ nên :

$$C(M)=\sum_{(Xi,Yj)\in M} C[i,j] \leq \sum_{v\in X\cup Y} F(v)$$

Riêng đối với mọi cặp $(Xi,Yj) \in M^*$,nhãn thoả mãn đẳng thức $F(Xi)+F(Yj)=C[i,j]$ nên :

$$\sum_{v\in X\cup Y} F(v) = \sum_{(Xi,Yj)\in M^*} C[i,j] = C(M^*)$$

từ đó nhận được : $C(M) \leq C(M^*)$ nghĩa là M^* là tập cặp ghép đầy đủ tối ưu .

Dực vào định lý vừa chứng minh ,người ta có 2 hướng tiếp cận cặp ghép đầy đủ tối ưu :

- Một là , xuất phát từ một cặp ghép đầy đủ M nào đó ,người ta xây dựng một nhãn F tương thích với M .Nếu F là chấp nhận được , thì M là tối ưu . Trái lại , người ta điều chỉnh M cho đến khi F tương thích là chấp nhận được và khi đó M là tối ưu .
- Hai là, xuất phát từ một nhãn F chấp nhận được và một tập cặp ghép M bất kỳ tương ứng với F (có thể rỗng) ,người ta tăng dần số cặp ghép của M sao cho vẫn đảm bảo tìm được nhãn F tương thích với M là chấp nhận được . Quá trình tăng sẽ kết thúc khi M đầy đủ và khi đó M là tối ưu.

Dưới đây trình bày một thuật toán tìm cặp ghép đầy đủ tối ưu theo hướng hai :

2. Thuật Toán Kunh-Munkers :

Nội dung chủ yếu của phương pháp là xuất phát từ một tập cặp ghép nào đó chưa đầy đủ (có thể là rỗng) , ta tăng dần số cặp ghép sao cho khi trở thành đầy đủ , các cặp ghép thu được cũng đồng thời thoả mãn tính tối ưu. Có nhiều hình thức trình bày phương pháp này . Dưới đây là cách trình bày trên ngôn ngữ đồ thị kèm với việc dùng thuật toán tìm đường đi . Cách này có nhiều ưu điểm : trực giác ,dễ phát biểu ,dễ chứng minh và đặc biệt ,dễ cài đặt chương trình vì việc tìm đường đi trên đồ thị là một thao tác cơ bản và quen thuộc .

Giả sử F là một nhãn chấp nhận được và M là một tập cặp ghép tương thích với F . Xem các Phần tử của X và Y như những đỉnh của một đồ thị có hướng hai phía (một phía X và một phía Y) . Các cạnh của đồ thị này được xác định tuỳ thuộc nội dung của nhãn F và tập cặp ghép M như sau :

- Mỗi cặp phần tử $Xi \in X$, $Yj \in Y$ thoả mãn đẳng thức $F(Xi)+F(Yj)=C[i,j]$,sẽ xác định một cạnh của đồ thị .
- Cạnh này sẽ có hướng từ X sang Y nếu cặp (Xi,Yj) không thuộc M (gọi là cạnh thuận) và hướng lại, có hướng từ Y sang X nếu cặp (Xi,Yj) thuộc M (gọi là cạnh nghịch).

Đồ thị xây dựng theo quy tắc vừa nêu được gọi là đồ thị cân bằng tương ứng với F,M và được ký hiệu là G(F,M) .

Bước 1 . **Khởi Tạo** : Xây dựng nhãn F chấp nhận được như sau :

$F(Xi) := \text{Max} \{ C[i,j] , Yj \in Y \} , Xi \in X .$

$F(Yj) := 0 \quad , Yj \in Y$

M là tập cặp ghép rỗng . Chú ý rằng , có thể xuất phát từ bất kỳ một nhãn F nào chấp nhận được và bất kỳ một tập cặp ghép nào tương ứng với F .

Bước 2 . **Tìm Đỉnh Tự Do Thuộc X** : tìm đỉnh $u \in X$ chưa được ghép cặp . Nếu không còn đỉnh nào của X chưa ghép cặp thì kết thúc : tập cặp ghép M hiện hành là tập cặp ghép đầy đủ tối ưu . Trái lại sang bước kế tiếp .

Bước 3. **Tìm Đường Tăng Cặp Ghép** : Xuất phát từ u, thực hiện việc tìm kiếm trên đồ thị G(F,M). Kết quả tìm kiếm có hai trường hợp :

- Nếu đến được một đỉnh $z \in Y$ chưa ghép cặp thì ghi nhận đường đi từ u đến z (gọi là đường tăng cặp ghép) và chuyển sang bước tăng cặp ghép trên đường đi này.
- Nếu không tồn tại đường đi như vậy thì chuyển sang bước sửa nhãn F .

Bước 4. **Tăng Cặp Ghép** : Điều chỉnh M như sau :

- Giữ nguyên những cặp ghép của M nằm ngoài đường tăng cặp ghép
- Trên đường tăng cặp ghép , bỏ đi những cặp ghép của M là cạnh ngược và thêm vào M những cặp ghép là cạnh thuận .

Sau bước này ,số cặp ghép thuocc M được tăng thêm 1 và đỉnh u trở thành đã ghép cặp , ngoìa ra , tính tương thích giữa F và M vẫn được bảo toàn . Sau đó quay về bước 2 để lặp lại với đỉnh tự do khác .

Bước 5. **Sửa Nhãn** : Gọi S là tập các đỉnh thuộc X và T là tập các đỉnh thuộc Y đã được đi đến trong quá trình tìm kiếm ở bước 3. Việ sửa nhãn F được tiến hành như sau :

- Tìm lượng sửa nhãn : d:= Min{F(Xi)+F(Yj)-C[i,j] , Xi∈ S, Yj ∉T }
- Gán lại nhãn : F(Xi):=F(Xi)-d với Xi ∈ S ; F(Yj):=F(Yj)+d với Yj∈ T .

Sau đó , quay về bước 3 để lặp lại việc tìm đường tăng cặp ghép (với đỉnh xuất phát u cũ và nhãn F mới).

Chú ý : Sau khi thay đổi ,nhãn F vẫn giữ nguyên tính chấp nhận được và tính tương thích với M .Ngoài ra có thêm ít nhất một cặp (Xi,Yj) thoả mãn F(Xi)+F(Yj)=C[i,j] , vì thế ,sau một số lần sửa nhãn , chắc chắn sẽ tăng được cặp ghép .

3. Chương Trình:

```
(*  ************************************    *)
 (*     Le Van Hung K28 A2 Phan Boi Chau  *)
 (*  ************************************    *)

program      cap_ghep_toi_uu ;
uses         crt;
const
    fi      =        'CAP_GHEP.INP' ;
    fo      =        'CAP_GHEP.OUT' ;
type
   trongso   =        array[ 1..170 , 1..170 ] of integer ;
   arr1      =        array[ 1..170 ] of longint ;
   arr2      =        array[ 1..340 ] of longint ;
var
  c                :        trongso ;
  n,u,z,total      :        longint ;
  ff               :        text ;
  px , py          :        arr1 ;
  f,q,queue        :        arr2 ;

  procedure          readfile ;
  var
    t, i , j      :      integer ;
    begin
        assign ( ff , fi ) ;
        reset ( ff ) ;
        readln ( ff , n ) ;
        for i := 1 to n do
         for j := 1 to n do
```

```
        read ( ff , c [ i , j ] ) ;
      close ( ff ) ;
  end;

procedure          khoi_tao ;
var
  x , y ,  xx      :      integer ;
  begin
      fillchar ( f  ,   sizeof( f )  , 0 ) ;
      for x := 1 to n do
       begin
          xx := c [ x , 1 ] ;
          for y := 2 to n do
           if  xx < c [ x , y ] then xx := c [ x , y ] ;
          f [ x ] := xx ;
       end ;
       fillchar ( px  ,  sizeof( px )  ,  0 ) ;
       fillchar ( py  ,  sizeof( py )  ,  0 ) ;
       for x := 1 to n do
        for y := 1 to n do
         if ( py [ y ] = 0 ) and ( f [ x ] + f [ y+n ] = c [ x , y ] ) then
         begin
            px [ x ] := y ;
            py [ y ] := x ;
            break ;
         end ;
  end ;

function           timthaydinhtudo      :     boolean ;
var
    x      :      integer ;
    begin
        for x := 1 to n do
         if px [ x ] = 0 then
         begin
            u := x ;
            timthaydinhtudo := true ;
            exit ;
         end ;
        timthaydinhtudo := false ;
    end ;

function    timthayduongtangcapghep    :     boolean ;
var
```

```
  dau , cuoi , v , w :       integer ;
  begin
     fillchar ( q   ,   sizeof( q )  ,  0  ) ;
     dau := 1 ;
     cuoi := 1 ;
     queue [ 1 ] := u ;
     q [ u ] := u ;
     while cuoi >= dau do
      begin
          v := queue [ dau ] ;
          inc ( dau ) ;
          if v < n+1 then
          begin
              for w := n+1 to 2*n do
               if ( f [ v ] + f [ w ] = c [ v , w-n ] )
               and ( q [ w] = 0 ) then
                begin
                    inc ( cuoi ) ;
                    queue [ cuoi ] := w ;
                    q [ w ] := v ;
                end ;
          end
          else
          if py [ v-n ] = 0 then
           begin
              timthayduongtangcapghep := true ;
              z := v ;
              exit ;
           end
           else
           begin
              w := py [ v-n ] ;
              inc ( cuoi ) ;
              queue [ cuoi ] := w ;
              q [ w ] := v ;
           end ;
      end ;
     timthayduongtangcapghep := false ;
  end ;

procedure          tangcapghep ;
var
  x , y     :      integer ;
  thuocy    :      boolean ;
```

```
  begin
      y := z ;
      thuocy := true ;
      while y <> u do
       begin
          x := q [ y ] ;
          if thuocy then
           begin
              px [ x ] := y-n ;
              py [ y-n ] := x ;
           end ;
          y := x ;
          thuocy := not thuocy ;
       end ;
  end ;

procedure          suanhan ;
var
  x , y , d , h       :      integer ;
  begin
      d := maxint ;
      for x := 1 to n do
       if q [ x ] > 0 then
        for y := n+1 to 2*n do
         if q [ y ] = 0 then
          begin
              h := f [ x ] + f [ y ] - c [ x , y-n ] ;
              if h < d then d := h ;
          end ;
       for x := 1 to n do
        if q [ x ] > 0 then dec ( f [ x ] , d ) ;
       for y := n+1 to n*2 do
        if q [ y ] > 0 then inc ( f [ y ] , d ) ;
  end ;

procedure          writefile ;
var
  x       :      integer ;
  begin
      assign( ff , fo ) ;
      rewrite( ff ) ;
      total := 0 ;
      for x := 1 to n do
       inc ( total , c [ x , px [ x ] ] ) ;
```

```
        writeln ( ff , total ) ;
        for x := 1 to n do
         write ( ff , px [ x ] :4 , ' ' ) ;
        close( ff ) ;
    end ;

  procedure          process ;
  begin
      khoi_tao ;
      while timthaydinhtudo do
       begin
          while not timthayduongtangcapghep do suanhan ;
          tangcapghep ;
       end ;
  end ;

BEGIN
    readfile ;
    process ;
    writefile ;
END .
```

4. Cặp Ghép Đầy Đủ Với Chi Phí Nhỏ Nhất :

Trong một số tình huống ,các cặp ghép đầy đủ được tìm sao cho có tổng trọng số là nhỏ nhất . Trường hợp này có thể xem C[i,j] mô tả chi phí của việc Ghép Xi với Yj và lời giải của bài toán cho ta cách ghép cặp đầy đủ ít tốn kém nhất . Vì thuật toán vừa nêu không phụ thuộc vào dấu của các C[i,j] nên bài toán tìm Min được dẫn về bài toán tìm Max với việc đổi dấu tất cả các trọng số C[i.j] ở đầu vào .

Cũng có thể trực tiếp giải bài toán tìm Min từ các C[i,j] đã cho mà không phải đổi dấu chúng bằng cách đối ngẫu lại ý nghĩa của các bất đẳng thức :

- Nhãn F được gọi là chấp nhận được nếu $F(Xi)+F(Yj) \le C[i,j]$
- Nhãn F được xây dựng ban đầu : $F(Xi):=0$ với $Xi \in X$; $F(Yj):=Min\{C[i,j], Xi \in X\}$ với $Yj \in Y$
- Công thức sửa nhãn : $d:=Min\{C[i,j]-F(Xi)-F(Yj) , Xi \in S, Yj \notin T\}$
  $F(Xi):=F(Xi)+d$ với $Xi \in S$ và $F(Yj):=F(Yj)-d$ với $Yj \in T$

III. Giải Bằng Phương Pháp Hungarian:

Đối với thuật toán này , chúng tôi xin được đề nghị về bài toán với giá trị nhỏ nhất của cặp ghép để các bạn tiện theo dõi ở hai phương pháp , mỗi phương pháp có một đặc điểm riêng . Và để cho các bạn nhìn thấy được thuật toán nào là đơn giản hơn

1. Các Định Lý Cơ Sở :

Định Lý 1:

Giả sử ma trận chi phí của bài toán cặp ghép không âm và có ít nhất n phần tử 0 . Hơn nữa ,nếu n phần tử 0 này nằm ở n hàng khác nhau và n cột khác nhau thì phương án phân cho người i thực hiện công việc tương ứng với số 0 này ở hàng i sẽ là phương án tối ưu (lời giải) của bài toán ghép cặp .

Định Lý 2 :

Cho $C=[C_{i,j}]$ là ma trận chhi phí của bài toán , và $x^*=\{x^*_{ij}\}$ là một lời giải của bài toán này . Giả sử C' là ma trận nhận được từ C' bằng cách thêm số a (dương hay âm) vào mỗi phần tử ở hàng i' của C . Khi đó , x^* cũng là lời giải của bài toán ghép cặp có ma trận chi phí C' .

Từ các kết qủa trên ta có thuật giải như sau :

- **Bước 1 :**

  Trừ các phần tử trên mỗi hàng của Ccho phần tử nhỏ nhất trên hàng đó , tiếp đó trừ các phần tử trên mỗi cột cho phần tử nhỏ nhất trên cột đó . Kết quả ta được ma trận C' có tính chất : trên mỗi hàng , cột có ít nhất một phần tử 0 và bài toán phân việc với ma trận C' có cùng lời giải như bài toán với ma trận C .

- **Bước 2 :**

  Với mỗi hàng , lần lượt từ hàng 1 tới hàng n , đánh dấu * cho phần tử 0 đầu tiên trên hàng đó không nằm trên cột đã có phần tử 0* (phần tử 0 được đánh dấu *)

  Nếu sau khi đánh dấu thấy có đủ n phần tử 0* thì dùng các phần tử o* sẽ cho lời giải cần tìm . Cụ thể : người i được phân thực hiện công việc tương ứng với phần tử 0* trên hàng i . Nếu số phần tử 0 nhỏ hơn n thì chuyển sang bước 3.

- **Bước 3 :**

  Lần lượt từ hàng 1 tới hàng n , tìm hàng đầu tiên không chứa phần tử 0* . Hàng như thế phải có vì lúc này chưa có đủ n phần tử 0* . Giả sử đó là hàng i_0. Vì trên mỗi hàng đều có ít nhất một phần tử 0 , nên trên hàng i_0 phải có phần tử 0 ,chẳng hạn ở cột j_0 . Xuất phát từ ô (i_0,j_0) , ta sẽ xây dựng một dây chuyền các ô kế tiếp nhau theo chiều ngang , dọc nối các phần tử 0 với 0* ,0* với 0 như sau :

  (A):

  Giả sử ta đang ở phần tử 0 trong ô (i_k,j_k) với $k \geq 0$, tìm phầntử 0* trong cột j_k . Nếu tìm thấy thì thêm vào dây chuyền đang xét ô chứa phần tử 0* này , rồi thực hiện thao tác (B) dưới đây . Nếu trái lại , ta đổi mỗi phần tử 0 trên đây chuyền này thành 0* và đổi 0* thành 0 . Sau đó, nếu

có đủ n phần tử 0* thì dừng . Nếu trái lại ,ta chuyển tới xét hàng tiếp theo không chứa phần tử 0* .

(B):

Giả sử ta đang ở phần tử 0* trong ô (i_{k+1},j_k) với $k \geq 0$. Trên hàng i_{k+1} tìm phần tử 0 không nằm trên cột đã vào dây chuyền đang xét ô chứa phần tử 0 này , rồi thực hiện thao tác (A) nêu trên . Nếu trái lại ,thì gọi cột j_k là cột thiết yếu và loại khỏi dây chuyền các ô (i_k,j_k) và (i_{k+1},j_k) .

Nếu vẫn còn ô trên dây chuyền đang xét ($k\geq 1$) , ta lại xuất phát từ phần tử 0* trong ô (i_k,j_{k+1}) và lặp lại thao tác (B) với hang i_k thay cho hang i_{k+1} . Nghĩa là trên hàng i_k tìm phần tử 0 , không nằm trên cột j_k (cột thiết yếu) và trên các cột trước đó đã có mặt trong dây chuyền đang xét .

Nếu không còn ô nào trên dây chuyền nữa (k=0) , thì ta lại tìm phần tử 0 trong dòng i_0 ở cột không phải là thiết yếu .Nếu thấy phần tử 0 như thế , chẳng hạn trong cột j_0' ,ta lặp lại thao tác (A) ,xuất phát từ ô(i_0,j_0') .Nếu không tìm thấy ta chuyển sang bước 4 .

- **Bước 4 :**

Lúc này chưa có đủ n phần tử 0* và bước 3 ta đã xác định được các cột thiết yếu . Bây giờ ta cần xác định các hàng thiết yếu . Một hàng gọi là thiết yếu nếu hàng đó chứa phần tử 0* ở cột không phải là thiết yếu .

Đến đây ta cần có thêm một định lý để thuật toán tiếp tục :

Định Lý 3 :

Số tối đa các phần tử 0 được đánh dấu * bằng số tối thiểu các hàng và các cột thiết yếu ,và số các hàng ,cột này chứa trọn mọi phần tử 0 của C' .

Bước 5 :

Giả sử a là số nhỏ nhất trong số các phần tử của ma trận C' thuộc hàng và cột không thiết yếu .Từ định lý 3 , suy ra a>0. vì mọi phần tử 0 đều nằm trên các hàng và cột thiết yếu . Biến đổi các phần tử 0 trong C' bằng cách trừ a vào mỗi phần tử thuộc hàng không thiết yếu và thêm a vào mọi phần tử thuộc hàng và cột đều là thiết yếu . Quay trở lại bước 2 để đánh dấu lại các phần tử 0 trong ma trận thu được .

Phương pháp này do nhà bác học người Hung tên là Kõnig và Egeváry . Tuy nhiên các bạn có thể thấy rõ về tính đơn giản về ngôn ngữ và diễn đạt thì thuật toán này không thể bằng thuật toán cặp ghép bằng ngôn ngữ đồ thị . Cho nên chúng ta nên theo phương pháp 1 mà chúng ta đã làm thì hay hơn . Chính vì thế phương pháp này chúng tôi không nêu lời giải mẫu . Nếu các bạn nào muốn mình có kỹ năng lập trình và gọn chương trình thì nên hãy thử sức vào làm chương trình cho thuật toán này ! .

IV. Các Bài toán Ghép cặp khác :

1. Bài toán cặp ghép cho đồ thị chung :

Bài toán :

" Cho một đồ thị 2*M đỉnh . các đỉnh được nối với nhau bằng các cung . Nhưng chúng ta cần tìm ra M cặp nối với nhau : một đỉnh chỉ nối với một đỉnh khác , và đỉnh nào cũng phải được nối . Sao cho hệ số nối là ít nhất "

Đây thật sự là một bài toán tổng quát cho các bài toán cặp ghép cho đồ thị . Đã có những công trình nghiên cứu bài toán này . Nhưng vì chưa có được lời giải , vì thế tôi chỉ nêu ra cho các bạn tham khảo .

2. Phương pháp ghép cặp có chặn :

Hoàn toàn tương tự như phần cuối ở chương Luồng Cực Đại . Các bài toán cặp ghép cũng có bài toán chặn trên(hoặc dưới) .

Chúng ta xét Các bài toán sau :

Bài toán 29 :

BÁN HÀNG

Đề Bài :

Một công ty kinh doanh có N cửa hàng (đánh số từ 1 đến N), mỗi cửa hàng phụ trách bán một mặt hàng (đánh số từ 1 đến N) của Công ty. Sau một thời gian thử nghiệm, người ta tính được doanh số trung bình Cij của cửa hàng i nếu bán mặt hàng j. Hãy tìm phương án phân công cửa hàng nào bán mặt hàng nào để tổng doanh số của Công ty là lớn nhất.

Dữ liệu: Vào cho trong file văn bản có dạng:

N

$C_{1\,1}$ $C_{1\,2}$ ... $C_{1\,N}$

$C_{2\,1}$ $C_{2\,2}$ ... $C_{2\,N}$

.....

$C_{N\,1}$ $C_{N\,2}$ ... $C_{N\,N}$

Các C_{ij} được giả thiết là nguyên, $N \le 100$.

Kết quả ghi ra file văn bản, gồm 1 dòng ghi N số nguyên:

H_1 H_2 ... C_{NN}

Trong đó H_1 là số hiệu mặt ahàng phân công cho của bảng i.

Thí dụ, file dữ liệu vào :

4

2 5 1 6

8 7 6 4

6 9 3 5

5 1 2 7

Có file kết quả:

4 3 2 1

Chú ý:

Các số trên cùng một dòng tróng các file được ghi cách nhau ít nhất một dấu trắng .

Hướng Dẫn :

Giải bài toán cặp giống như N thợ làm N việc (thay bằng N cửa hàng bán N mặt hàng) .

Bài toán 30 : Deltail

Đề Bài :

Một sản phẩm được lắp ghép từ n chi tiết được đánh số từ 1 đến N . Có N máy để thực hiện việc gia công các chi tiết này . Các máy được đánh số từ 1 đến N . Mỗi một trong số N máy có thể thực hiện việc gia công bất kỳ chi tiết nào m nhưng thời gian hoàn thành việc gia công các chi tiết khác nhau trên nó có thể khác nhau . Biết T[j,k] là thời gian cần thực hiện để hoàn thành việc gia công chi tiết k trên máy j (k ,j = 1,2..n) .

Yêu Cầu : Hãy tìm cách bố trí việc thực hiện gia công mỗi chi tiết trên một máy sao cho có thể bắt đầu thực hiện việc gia công tất cả các chi tiết tại cùng một thời điểm

0 , đồng thời việc hoàn thành gia công tất cả các chi tiết được thực hiện tại thời điểm sớm nhất .

Dữ liệu : Vào từ file văn bản : Details.Inp :

- Dòng đầu tiên ghi só nguyên dương N (N <=200) .
- Dòng thứ j trong số N dòng tiếp theo ghi N số nguyên dương T[j,1],T[j,2],...T[j,n] , hai số liên tiếp nhau cách nhau ít nhất một dấu cách . j= 1 ,2 ..n

Kết quả : Ghi ra file văn bản Details.Out :

- Dòng đâuf tiên ghi thời điểm hoàn thành việc gia công tất cả các chi tiết theo cách bố trí tìm được .
- Dòng thứ hai ghi dãy gồm n số , số thứ i trong dãy cho biết chỉ số của chi tiết được bố trí gia công trên máy i .

Ví Dụ :

DETAILS.INP	ĐETAILS.OUT
2 3 2 1 2	2 2 1

DETAILS.INP	DETAILS.OUT
3 3 11 7 9 3 30 4 15 3	3 1 2 3

Hướng Dẫn :

Đây là một trong những bài toán tiêu biểu cho bài toán chặn của cặp ghép . Ta sẽ xây dựng tập đồ thị hai phía : Tập 1 gồm các đỉnh đại diện cho các máy , Tập hai gồm các đỉnh đại diện cho các chi tiết . Trọng số của các cung bằng thời gian thực hiện các chi tiết trên các máy tương ứng .

Chúng ta sẽ giải bài toán cặp ghép cho đồ thị này . Nhưng chúng ta sẽ sử dụng hệ số như sau : ta sẽ tìm hệ số Max của ma trận . Sau đó ta gọi T là hệ số Max tạm thời của ma trận . Với mỗi T $\in$[1,Max] thì ta sẽ loại các cung có trọng số lớn hơn T . Giải bài toán cặp ghép cho các trường hợp của T . Thì trường hợp nào có ghép cặp thoả mãn T nhỏ nhất có thể thì đó là trường hợp cần lấy(tức là cách ghép ở ghép cặp này chính là các bố thực hiện máy) .

Bài toán 31 : Thực hiện song song

Đề bài :

Có N chương trình (đánh số từ 1 đến N) cần được thực hiện song song trên M máy (đánh số từ 1 đến M , M>=N) . Do đặc tính khác nhau của mỗi chương trình và cấu hình khác nhau của mỗi máy nên mỗi máy chỉ có thể thực hiện được một số chương trình vỡi những thời gian hoàn thành khác nhau . Cần phân công mỗi máy chạy một chương trình (bắt đầu cùng một thời điểm) để tất cả các chương trình đều được thực hiện với thời gian hoàn thành sớm nhất .

Dữ liệu : Vào trong file SN.INP :

- Dòng đầu tiên là số M , N
- M dòng sau biểu diễn T[i,j] là những số nguyên ,ghi nhận thời gian máy i hoàn thành chương trình j với quy ước T[i,j] =-1 nếu không thực hiện được . Các số ghi trên cùng một dòng cách nhau ít nhất một dấu trắng . Giới hạn M<=100 .

Kết quả : Ghi ra file SN.OUT :

- Ghi N số : X[1],X[2],...X[N] , trong đó X[j] là số hiệu của máy được phân chạy chương trình j . Nếu không có phương án phân công thì file kết quả ghi một số –1

Ví Dụ :

SN.INP
4 3 -1 5 6 4 3 –1 7 5 –1 6 –1 3

SN.OUT
2 1 4

Hướng Dẫn :

Hoàn toàn tương tự bài toán vừa rồi , nhưng đây là trường hợp cắp ghép không cân bằng đỉnh . Nhưng chúng ta sẽ thêm cho phía nào đó (nếu thiếu đỉnh) các đỉnh giả . Sau đó thực hiện bài toán ghép cặp chặn như trên .

Bài toán 32 : Block

Đề Bài :

Trên mặt phẳng cho N^2 khối lập phương (đánh số tù 1 đến N^2) có cạnh độ dài đơn vị . Mỗi mặt của khối lập phương có một màu nào đó lấy trong 6 mầu , mô tả bởi 6 chữ cái A,B,C,D,E,F . Trạng thái của khối lập phương được xác định bởi xâu 6 chữ cài (không nhất thiết khác nhau) , mô tả của 6 mặt theo trình tự : bắt đầu từ mặt trước ,sau đó là các mặt bên theo chiều ngược kim đồng hồ và cuối cùnd là các mặt trên , mặt dưới . Thí dụ ở hình sau , khối lập phương có trạng thái ABCDAB :

Trạng thái của khối lập phương sẽ được thay đổi khi lật qua các cạnh của nó theo 4 phía . Ký hiệu tương ứng các phép biến đổi khi lật qua các cạnh của nó theo 4 phía . Ký hiệu tương ứng các phép biến đổi này là : R(phải) , L (trái) , U (trên) , D (dưới) . Chẳng hạn , khi thực hiện phép lật phải (R) , trạng thái của khối ở hình trên là : AAACBDB .

Bằng cách lật một số lần thích hợp các khối lập phương , sau đó di chuyển chúng (không thay đổi trạng thái) , người ta có thể xếp các khối đã cho cạnh nhau thành một khối chữ nhật kích thước N * N * 1 *(chiều cao 1) có mầu của mặt trên và các mặt xung quanh trùng với một mẫu cho trước .

Yêu cầu : Tìm một phương ámn lật các lập phương đã cho (không tính đến các phép di chuyển) để có thể xếp các khối này thoả mãn điều khiển đã nêu , sao cho tổng số các phép lật càng ít càng tốt . Dữ liệu vào và kết quả ra được ghi trong các file văn bản theo các mô tả dưới đây . Giuới hạn $N \le 15$.

Dữ liệu : Block.inp

- Dòng đầu tiên ghi số N .
- Mỗi dòng thứ i trong N^2 dòng tiếp theo là xâu 6 chữ cái (trong phạm vi từ A đến F) viết liền nhau , mô tả trạng thái của khối lập phương thứ i ,
- N dòng tiếp theo , mỗi dòng chứa N chữ cái (viết liền nhau) mô tả mầu của mặt trên của khối mẫu theo đúng thứ hàng , cột (các hàng tính từ trên xuống dưới , các cột tính từ trái sang phải).
- Dòng cuối là xâu gồm 4*n chữa cái (viết liền nhau) , mô tả mầu xung quanh của khối mẫu theo chiều ngược kim đồng hồ (bắt đầu từ khối ở hàng N , cột 1) .

Kết Quả : Block.Out

- Dòng đầu ghi YES hoặc NO cho biết có xếp được các khối hay không , nếu xếp được thì ghi tiếp .

- Mỗi dòng thứ i trong số N^2 dòng tiếp theo là dãy gồm các ký tự viết liền nhau lấy từ tập hợp R,L,U,D , mô tả các phép lật khối lập phương thứ i theo đúng thứ tự đã thực hiện , nếu khối này không được lật thì dòng tương ứng ghi một ký tự # .
- Cuối cùng là N dòng , mỗi dòng gồm N số ghi cách nhau ít nhất một dấu trắng , trong đó mỗi số là số hiệu của khối lập phương theo đúng vị trí hàng , cột đã được xếp .

Ví Dụ :

BLOCK.INP	BLOCK.OUT
3	YES
ACFACE	DL
BFCBCB	L
ADBBDD	U
CFDBBD	DLD
BBAEEF	#
EEACDB	U
AEAEFD	DL
FFDDBA	L
DBFDEB	#
DCE	4 1 5
EFF	6 2 8
EAE	9 3 7
DDFAABAEBDCD	

Hướng Dẫn :

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Trước tiên ta đánh số các ô của khối cần ghép thành thứ tự : tăng dần của hàng , cột (ví dụ trên cho N=4) . Tức là chúng ta có N^2 ô , chúng ta coi N^2ô này như là N^2 đỉnh của một đồ thị , mà chúng được nối với N^2 đỉnh khác mà các đỉnh đó là coi như N^2 đỉnh của một đồ thị . Và ma trận giá trị của đồ thị hai phía này được biểu diễn :

A[i,j]=∞ nếu như hình hộp thứ i cho dù có tất cả các cách quay đi nữa thì cũng không thể để vào ô thứ j trong khối . Nếu có thể đặt được thì A[i,j]=Số lần quay ít nhất để đặt vào ô thứ j đó . Nên nhớ rằng điều kiện để khối thứ i có thể đặt được vào vị trí thứ j là : Mặt trên của khối và mặt trên của vị trí đó phải

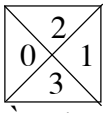
giống nhau , thêm vào đó nếu vị trí đó thuộc biên thì các mặt biên có thể có của khối đó cũng phải giống .

Sau khi xây dựng ma trận chi phí đó , chúng ta giải bài toán ghép cặp để tìm được cách đặt khối sao cho chúng có số lần xoay tối ưu . (Vì $N^2<=15^2$ nên số đỉnh của đồ thị hai phía đó là 15^2*2) .

Bài toán 33 : Xếp Hình

Đề bài :

Một trò chơi xếp hình trên máy tính từ những ô hình vuông kích thước giống nhau. Mỗi ô hình vuông này được chia thành 4 tam giác, mỗi tam giác có



thể nhận một trong 4 mầu đỏ, xanh, vàng, trắng (hai tam giác khác nhau không nhất thiết có mầu khác nhau) với quy ước 0-mầu đỏ, 1-mầu xanh, 2-mầu vàng, 3-mầu trắng như hình vẽ dưới đây:

Người chơi cần dùng chuột kéo những ô vuông đã cho vào một lưới ô vuông M hàng, N cột (mỗi ô vuông vào một ô của lưới, số lượng các ô vuông đã cho bằng số lượng các ô của lưới) sau đó có thể nhấn phím phải chuột để quay ô. Mỗi lần nhấn phím phải chuột, ô quay một góc 90 độ theo chiều kim đồng hồ. Giả thiết mỗi lần kéo một ô vuông vào một vị trí của lưới mất 1 giây và mỗi lần nhấn phím phải chuột mất 1 giây. Bằng những thao tác như vậy, người chơi cần tạo ra một hình giống như hình mẫu cho trước với thời gian nhanh nhất.

Các ô vuông đã cho được đánh số từ 1. Trạng thái của một ô vuông được mã hóa bằng một byte (8 bit) như sau: 2 bit trái nhất mô tả giá trị mầu của tam giác phía bắc, lần lượt các nhóm 2 bit tiếp theo mô tả giá trị mầu của các tam giác phía đông, nam, tây của ô vuông. Hình vẽ trên mô tả ô vuông có trạng thái là 156.

Dữ liệu: Vào được cho trong file văn bản XEP.INP gồm:

- dòng đầu là 2 giá trị M và N,
- dòng tiếp theo lần lượt là các trạng thái của MìN ô vuông đã cho theo thứ tự số hiệu ô,
- M dòng tiếp, mỗi dòng gồm N số nguyên mô tả trạng thái của dòng tương ứng của hình mẫu.

Kết quả: Ghi ra file văn bản XEP.OUT, nếu không có cách xếp thì ghi một số -1 và kết thúc, trái lại ghi như dưới đây:

- M dòng đầu, mỗi dòng gồm N số nguyên, ghi số hiệu của ô vuông được xếp vào vị trí tương ứng của lưới,
- M dòng tiếp, mỗi dòng gồm N số nguyên trong phạm vi từ 0 đến 3, ghi số lần nhấn phím phải chuột để quay ô tương ứng.

Ghi chú:

- Giới hạn kích thước M, N không quá 50.
- Các số nguyên trên cùng một dòng trong các file vào/ra ghi cách nhau ít nhất một dấu trắng.

Thí dụ:

XEP.INP
2 3 **183 237 183 108 27 198** **198 123 108** **237 222 177**

XEP.OUT
5 1 4 **2 3 6** **1 2 0** **0 3 1**

Hướng Dẫn :

Chúng ta thấy sự xếp hình bằng cách xoay được vào bảng là tương ứng 1 – 1 . Chúng ta tạo ma trận A[1..M*N,1..M*N] trong đó A[i,j] là số lần kích chuột tác động lên ô i khi đặt vào ô thứ j trên bảng (giả sử ta đánh vị trí các ô vào bảng theo thứ tự từ trên xuống , trái sang phải của hình đó) . Nếu không thể kích chuột mà tạo thành thì A[i,j]:=Maxint . Chúng ta sẽ giải bài toán cặp ghép để tìm ra cách tác động để xếp các hình . Nhưng các bạn hãy thử lại khi M=50 , N=50 thì mảng A [1..2500,1..2500] thì kích cỡ này là không thể trong Pascal , chưa thể nói đến tốc độ chương trình . Chúng ta lại có một điều hết sức đặc biệt , đó là đây là một bài toán ghép cặp hết sức đặc biệt .

Chúng ta sẽ ưu tiên từng ô một , từ (1,1) ...(m,n) thì ta sẽ tìm cho mỗi ô một ô tương ứng ở ngoài mà có số lần kích tới nó là ít nhất, cứ như vậy cho đến hết . Sở dĩ chúng ta làm được là , giá trị A[i,j] nếu khác Maxint thì chỉ có thể là 1 , 2 , 3 ,4 (kể cả kéo chuột) . Như vậy hệ số này sẽ không quan trọng khi chúng ta cho chúng có vai trò bình đẳng nhau .

Bài toán này tuy là bài toán cặp ghép , nhưng trên cơ sở ghép cặp ta có thể tìm ra một cách giải hoàn toàn tốt hơn .

Bài toán 34 : Xếp hàng

Đề bài :

Một bàn cờ điện tử kích thước NxN , trong đó đánh dấu một số ô cấm . Trên bàn cờ có K quân mã đang đứng ở những vị trí nào đó . Người ta muốn lập trình cho các quân mã này để chúng có thể di chuyển đến cạnh nhau (mỗi quân mã một vị trí) , xếp thành một đội ngũ có dạng hình chữ nhật (các cạnh song song với các cạnh bàn cờ) . Giả thiết rằng , các quân mã đồng thời di chuyển bắt đầu từ một thời điểm nào đó với vận tốc như nhau (mỗi nước đi là một đơn vị thời gian) . Trong quá trình di chuyển , mã không được nhảy đến các ô cấm , nhưng có thể nhảy đến ô đã có những quân mã khác đang đứng .

Yêu cầu : Xác định cách đi các quân mã sao cho thời điểm hoàn thành việc xếp hàng các quân mã là sớm nhất .

Dữ Liệu : Vào từ file văn bản XH.Inp :

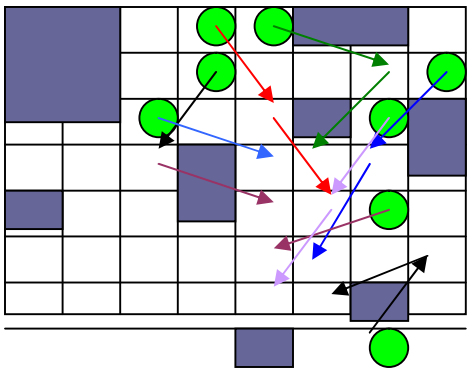
- Dòng đầu tiên ghi số N ,
- Tiếp theo là N dòng , mỗi dòng gồm N số (ghi cách nhau ít nhất một dấu trắng) mô tả trạng thái các ô tương ứng của bàn cờ cới quy ước : Ghi số 1 là ô cấm , ghi số 2 là ô xuất phát của một quân mã , các ô khác ghi 0.

Kết quả : Đưa ra file XH.Out :

- Dòng đầu tiên ghi thời gian hoàn thành việc xếp hàng , nếu không có cách xếp thì ghi –1 . Trong trường hợp xếp được :
- Ghi tiếp K dòng , mỗi dòng mô tả một cách đi quân mã , gồm một dãy cặp số , mỗi cặp số là toạ độ dòng , cột của bàn cờ , cặp số thứ nhất là toạ độ ô xuất phát , các cặp tiếp là toạ độ lần lượt của các ô mà mã nhảy đến , cặp cuối cùng là toạ độ ô tập kết .

  Các số trên cùng một dòng ghi cách nhau ít nhất một dấu trắng .

Ví dụ :



XH.INP	XH.OUT
8 1 1 0 2 2 1 1 0 1 1 0 2 0 0 0 2 1 1 2 0 0 1 2 1	2 3 3 4 5 1 5 2 7 4 6 2 4 4 3 5 5

0 0 0 1 0 0 0 1 1 0 0 1 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 2 0	1 4 3 5 5 6 5 7 6 5 2 8 4 7 6 6 3 7 5 6 7 5 8 7 6 8 7 6

Hướng Dẫn :

Trước tiên chúng ta tìm các miền hình chữ nhật có thể có trong bảng sao cho các vị trí của hình chữ nhật không có vật cản và hình có diện tích bằng số mã có trên bàn cờ .

Với mỗi hình tìm được chúng ta đánh số thứ tự các ô của hình chữ nhật này theo thứ tự trái sang phải , trên xuống dưới . Sau đó ta xây dựng đồ thị hai phía , mỗi phía có K đỉnh (K là số con mã) . Và A[i,j] là số bước để con mã thứ i nhảy tới ô thứ j trong hình chữ nhật đó . A[i,j]=∝ nếu con mã i không thể nhảy tới vị trí ô j . Sau đó chúng ta giải bài toán ghép cặp của đồ thị xây dựng được . Rồi tính tổng giá trị ghép cặp nhỏ nhất . Trong các hình chũ nhật thì hình nào có tổng số lần nhảy ít nhất thì lấy .

```
Procedure    main ;
begin
        while có hình chữ nhật thoả mãn do
        begin
                Xây dựng ma trận chi phí ;
                giải bài toán ghép cặp cho ma trận đó ;
                kiểm tra tính tối ưu nếu thoả mãn thì ghi nhận ;
        end ;
end ;
```

Bài toán 35 : Thằng bờm và con mực

Đề bài :

Hàng ngày thằng Bờm thường dạo chơi với con chó của nó có tên là Mực. Bờm đi với tốc độ không đổi và đường đi của nó là một đường gấp khúc (có thể tự cắt) có N điểm gãy. Mỗi điểm gãy được cho bởi cặp hai số nguyên (x_1, y_1) là toạ độ của nó trên mặt phẳng toạ độ. Con Mực chạy chơi với chủ theo con đường riêng của nó nhưng luôn gặp ông chủ tại N điểm gãy nói trên. Con Mực cùng ông chủ của nó cùng bắt đầu dạo chơi từ điểm (x_1, y_1) và kết thúc dạo chơi đồng thời tại điểm (x_N, y_N). Con Mực có thể chạy với tốc độ không vượt quá 2 lần tốc độ của Bờm. Khi Bờm đi theo đường thẳng từ điểm gãy này đến điểm gãy tiếp theo, con Mực có thể chạy đến các điểm hấp dẫn nó được cho bởi M cặp số nguyên (x_j, y_j). Tuy nhiên, sau khi tách rời ông chủ của nó tại

điểm (x_i, y_i) (trong đó $1 \le i < N$) con Mực có thể thăm không quá 1 điểm hấp dẫn nó để rồi gặp lại chủ của nó tại điểm (x_{i+1}, y_{i+1}).

Yêu cầu: Tìm cách đi của con Mực thoả mãn các điều kiện nêu trên và sao cho số điểm hấp dẫn nó thăm được là nhiều nhất.

Ví dụ, đường đi của thằng Bờm (đường đậm nét), tập các điểm hấp dẫn (các nốt đen) và một trong những đường đi tốt nhất của con Mực (đường đứt nét) được cho trong hình vẽ dưới đây:

Hình

Dữ liệu: Vào từ file văn bản DOG.INP:

- Dòng đầu tiên chứa hai số nguyên dương N và M được ghi cách nhau bởi dấu cách ($2 \le N \le 100$, $0 \le M \le 100$);
- Dòng thứ hai chứa N cặp số nguyên $x_1, y_1, \ldots x_N, y_N$ được ghi cách nhau bởi dầu cách.
- Dòng thứ ba chứa M cặp số nguyên $x_1, y_1, \ldots x_M, y_M$

Các điểm trong file dữ liệu là khác nhau từng đôi và toạ độ của chúng là các số nguyên có trị tuyệt đối không vượt quá 1000.

Kết quả: Ghi ra file văn bản DOG.OUT

- Dòng đầu tiên ghi số nguyên dương K là số điểm gãy trên đường đi của con Mực;
- Dòng thứ hai ghi K cặp toạ độ $u_1, v_1 \ldots, u_K, v_K$ (các toạ độ ghi cách nhau bởi dấu cách) biểu diễn đường đi tìm được.

Ví dụ:

DOG.JNP	**DOG.OUT**
4 5 1 4 5 7 5 2 -2 4 - - 3 9 1 2 - 3 8 - 4 2 1 3	6 1 4 3 9 5 7 5 2 1 2 -2 4

Hướng Dẫn :

Ta xây dựng đồ thị hai phía :

Tập đỉnh thứ nhất là tập đỉnh các đỉnh thuộc hành trình của bờm . Còn tập đỉnh thứ hai gồm các đỉnh thuộc tập các điểm hấp dẫn . Chúng ta có trọng số của đồ thị hai phía này là bằng khoảng cách giữa các điểm đó . (tức là giữa một điểm i nào đó trên hành trình với 1 điểm j nào đó trong các điểm hấp dẫn). Chúng ta cần chọn ra N cặp đỉnh mà có tổng số đường đi là ngắn nhất . Cho nên

ta xây dựng thêm M-N đỉnh giả ở phía tập 1 . Mà độ dài trọng số của nó nối với bất kỳ đỉnh nào thuộc tập 2 là bằng Maxint . Giải bài toán cặp ghép M đỉnh thì ta sẽ chọn ra được cách bố trí đi của chú Mực .

Bài toán 36 : Chess

Đề bài :

Một bàn cờ có dạng bảng chữ nhật kích thước mxn, trên đó có một số quân cờ. Một quân cờ chỉ có thể di chuyển sang một ô kề cạnh còn trống, mỗi di chuyển như vậy được gọi là một bước di chuyển . Cho hai trạng thái của bàn cờ, hãy chỉ ra một dãy các bước di chuyển để đưa bảng từ trạng thái ban đầu đến trạng thái đích. Mỗi trạng thái được mô tả là một ma trận mxn trong đó số ở hàng i cột j là 1 nếu tại vị trí i,j tương ứng có quân cờ đang đứng hoặc bằng 0 nếu không có.

Dữ liệu : Vào từ file : Chess.inp ;

- Dòng đầu ghi hai số nguyên dương m , n (1 < m , n <= 10)
- Tiếp theo là 2m dòng thể hiện ma trận mô tả trạng thái xuất phát và trạng thái đích. m dòng đầu tiên thể hiện ma trận xuất phát, m dòng tiếp theo là ma trận đích.
- Input cho đảm bảo luôn có nghiệm.

Kết quả : Ghi Ra file Chess.Out

- Dòng đầu ghi K là số ít nhất các phép biến đổi tìm đực.
- K dòng tiếp theo, mỗi dòng mô tả một phép biến đổi, theo đúng thứ tự biến đổi, gồm 4 số nguyên dương u, v , x, y thể hiện di chuyển quân cờ ở vị trí (u,v) sang vị trí (x,y).

Hướng Dẫn :

Ta gọi X là tập các ô tại trạng thái xuất phát có quân cờ, còn đích thì không, Y là tập các ô ma trạng thái xuất phát không có mà trạng thái đích có. Dễ thấy bài toán có nghiệm khi và chỉ khi X, Y có cùng số phần từ. Ta xây dựng đô thị hai phía (X,Y) trong đó với mỗi x thuộc X, y thuộc Y thì cạnh (x,y) có trọng số là số các bước di chuyển nhỏ nhất để đi chuyển từ x đến y trên bàn cờ không có vật cản.

Bài toán tìm số bước ít nhất tương đương với cặp ghép tổng trọng số nhỏ nhất trên đồ thị hai phía, như đã xây dựng ở trên. Với cặp ghép tối ưu tìm được, ta cho di chuyển quân cờ tại x đến vị trí cặp ghép tương ứng của x. Lời giải bài toán này thực chất giống bài toán Xếp Hình ở trên .

Mở rộng :

Bài toán có thể mở rộng cho bảng kích thước lớn hơn với điều kiện tập X , Y không lớn (có không quá 200 phần tử), và bước quân cờ có thể di chuyển giống quân mã hay quân vua…

Với bài toán này các bạn còn có một thuật giải khác khá tự nhiên hơn : Tìm kiếm đường đi theo chiều rộng . (Xem thêm bài toán này ở bài toán 2 – Chương III)

Phần 5 : Sau Đây là một số bài toán thuộc phần đồ thị

Bài toán 37 : Mạng liên thông đơn

Đề bài :

Một hệ thống gồm N máy tính (đánh số từ 1 đến N) được kết nối thành mạng bởi các kênh truyền tin một chiều giữa một số cặp máy tính . Biết rằng từ một đỉnh không quá 3 kênh truyền tin đến các máy còn lại . Ta hiểu một đường truyền tin từ máy u đến máy v trên mạng là dãy :

u=w1 ,w2,..wi=v

Trong đó , kênh truyền tin từ máy wi-1 đến wi . Đường truyền tin được gọi là đơn liên thông nếu như không có máy nào bị lặp lại .

Mạng được gọi là liên thông đơn nếu như có đường truyền tin từ máy u đến máy v thì có không quá một đường truyền tin từ máy u đến máy v

Yêu cầu : Xác định mạng đã cho có phải là liên thông đơn hay không

Dữ liệu : Vào từ file Single.Inp :

- Dòng đầu tiên chứa số N (N<=300)
- Mỗi dòng thứ i trong số N dòng tiếp theo chứa chỉ s của các máy mà từ máy i kênh truyền tìm đến chúng hoặc chứa số 0 nếu từ máy i không có kênh truyền tin đến bất kể máy nào (i=1,2,..N)

Kết quả : Ghi ra file Single.Out :

- Dòng đầu tiên ghi YES hoặc NO tương ứng với mạng là liên thông đơn hoặc không là liên thông đơn
- Nếu dòng đầu ghi :
  + YES : thì dòng thứ i trong n-1 dòng tiếp theo ghi độ dài đường truyền tin đơn từ máy 1 đến máy i (ghi số 0 bếu không có đường truyền tin từ máy 1 đến máy i)
  + No : GHi chỉ số của hai máy u,v mà từ u đến v có ít ra là hai đường truyền tin đơn .

Ví Dụ :

SINGLE.INP
6 2 3 4 5 6

SINGLE.OUT
NO 1 2

1

Hướng Dẫn :

Thực chất bài toán này là bài toán tìm các chu trình có thể có của đồ thị . Nếu tồn tại chu trình thì chúng ta chỉ cần lấy hai đỉnh thuộc chu trình đó ra làm hai đỉnh yêu cầu cần lấy . Nếu không tồn tại chu trình thì chúng ta sẽ ghi YES .

Công việc tìm chu trình thì các bạn có thể dùng đệ quy (như thủ tục trong sách toán rời rạc – thủ tục cycle trang 181) .

Bài toán 38 : ngôi nhà của newton

Đề Bài :

Trong toà nhà của Newton gồm N phòng đánh số từ 1 đến N, N<=100 có một con mèo và một con chuột, mỗi con chiếm một phòng riêng làm nhà của mình, mèo phòng M, chuột phòng C, không có cửa từ phòng M sang phòng C. Với hai phòng A và B bất kỳ, từ A sang B, có thể có/không có cửa, nhưng nếu có, Newton quy định rất rõ cửa là của mèo hay của chuột, nếu là của mèo/chuột thì chuột/mèo không được đi. Từ A sang B có thể có cả hai loại cửa hoặc chỉ có một loại cửa cho một trong hai con hoặc không có cửa, các cửa đều chỉ đi một chiều, cửa từ phòng A sang phòng B chỉ có thể đi được từ A sang B.

Yêu Cầu : Hãy trả lời cho chuột hai vấn đề sau (ai trả lời tốt nhất thì họ hàng chuột không bao giờ đến nhà quấy nhiễu):

1. Trong quá trình dạo chơi trong nhà, tại những phòng nào chuột có thể gặp mèo (các cuộc dạo chơi của cả hai con đều xuất phát từ nhà của mình).

2. Có hay không hành trình của chuột qua ít nhất hai phòng khác nhau từ nhà nó và quay về nhà nó sao cho trên hành trình không bao giờ gặp mèo. Nếu có, hãy chọn một trong các hành trình qua nhiều phòng nhất.

Dữ liệu: Vào được cho bởi file NEWTON.INP trong đó dòng thứ nhất ghi ba số N, M, C, tiếp theo là một số dòng, mỗi dòng ghi hai số A, B, có đúng một dòng A=B=-1 là dòng ngăn cách giữa nhóm cửa của mèo ở phía trên với nhóm cửa của chuột ở dưới dòng đó, cặp số A, B có nghĩa là có cửa (một chiều) từ phòng A sang phòng B.

Kết quả : Ghi ra file NEWTON.OUT như sau: dòng thứ nhất ghi số NH là số phòng mà chuột có thể gặp mèo, nếu NH>0, tiếp theo là NH dòng, mỗi dòng ghi số hiệu một phòng; sau đó là một dòng ghi số AT là số phòng *khác nhau* trên hành trình theo yêu cầu ở phần 2, nếu không có thì ghi số 1; nếu có, trong AT dòng tiếp theo, mỗi dòng ghi số hiệu một phòng theo thứ tự trên hành trình bắt đầu từ nhà của chuột.

Ví dụ

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

2	1	2
3	1	3
4	3	2
5	2	4
-1	-1	3
1	3	
2	5	
3	4	
4	2	
4	5	
5	4	

Hướng Dẫn :

Chúng ta sẽ loang các ô mà con mèo có thể đến . Sau đó chọn những ô đó mà mèo không đến thì chuột sẽ đến .

Trong những ô đó , tìm các hành trình dài nhất rồi trở lại nhà chuột .

Bài toán 39 : Đường một chiều

Đề bài **:**

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 giưa 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.

Dữ Liệu :

- Dòng đầu ghi hai số nguyên dương N , M (1 < N < 500 , 1 < M < 10000)
- M dòng tiếp theo , mỗi dòng thể hiện một đường hai chiều gồm u , v là chỉ số hai nút mà nó nối tới .

Kết quả :

- 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.

Hướng Dẫn :

Ta loang theo chiều sâu để xây dựng một cây DFS , các cạnh trên cây sẽ đượoc định hướnh theo chiều từ gốc đến lá. Các cạnh không thuộc cây sẽ được định hướng ngược lại (tức là theo chiều lá về gốc).

Sau khi định hướng như trên , ta kiểm tra xem đồ thị có hướng sau khi định chiều cạnh có còn liên thông hay không , nếu không thì bài toán vô nghiệm ,

ngược là hiển nhiên đó là nghiệm của bài toán . Để kiểm tra tính liên thông , ta có thể dùng thuật toán Tarjan với độ phức tạp O(n+m) (xem thuật toán Tarjan)

Bài toán 40 : Quân mã

Đề Bài :

Cho một bàn cờ kích thước n*n (n <= 200) trên đó có một số ô đã bị đánh dấu. Hãy tìm cách đặt các quân mã lên các ô trống của bàn cờ sao cho:

- ✓ Không có hai con mã nào đặt trên cùng một ô
- ✓ Không có hai quân mã nào khống chế nhau
- ✓ Số quân mã đặt được là lớn nhất có thể

Dữ Liệu : Cho trong file text **Knight.dat**:

- Dòng đầu ghi hai số n, m - số ô bị đánh dấu.
- M dòng tiếp theo mỗi dòng ghi hai số thể hiện một ô bị đánh dấu gồm hai số x, y là vị trí của ô hàng x, cột y.

Kết Quả : File **Knight.out**

- K số quân mã nhiều nhất.
- K dòng tiếp theo mỗi dòng mô tả một vị trí quân mã gồm hai số x, y giống như ở Input

Hướng Dẫn :

Tô mầu các ô của bàn cờ giống như bàn cờ vua, ta phân chia các ô chưa bị đánh dấu thành 2 lớp : lớp các ô đen, lớp các ô trắng, dễ thấy nếu hai ô mà quân mã ở hai ô khống chế nhau thì hai ô đó phải thuộc hai lớp khác nhau. Ta đưa bàn cờ về đồ thị hai phía có V = (X, Y) trong đó hai đỉnh có cạnh nối nếu quân mã ở hai ô tương ứng đó khống chế được nhau.

Tập hợp các vị trí quân mã thoả mãn bài toán chính là tập ổn định cực đại của đồ thị hai phía . Bài toán tìm tập ổn trên đồ thị hai phía có thể giải bằng tìm lát cắt hẹp nhất nếu ta cho nối một đỉnh phát ảo s với mỗi đỉnh thuộc X, đồng thời mỗi đỉnh thuộc Y nối với một đỉnh thu ảo t, Tất cả các cung đặt giá trị thông qua là 1 .

Khi đó từ lát cắt hẹp nhất (A, B) cho ta tập ổn định cực đại gồm các phần tử X thuộc A, và các phần tử thuộc Y nhưng không thuộc A.

Bài toán 41 : Mạng Giao Thông

Đề Bài :

Theo thiết kế, một mạng giao thông gồm N nút có tên từ 1 đến N. Chi phí để xây dựng đường hai chiều trực tiếp từ nút I đến nút J bằng A[I,J]=A[J,I], với mọi I, A[I,I]=0. Hai tuyến đường khác nhau không cắt nhau tại điểm không là đầu mút. Hiện đã xây dựng được K tuyến đường.

Bài toán đặt ra như sau: Hệ thống đường đã xây dựng đã bảo đảm sự đi lại giữa hai nút bất kỳ chưa? Nếu chưa, hãy chọn một số tuyến đường cần xây dựng thêm sao cho:

1. Các tuyến đường sẽ xây dựng thêm cùng với các tuyến đường đã xây dựng bảo đảm sự đi lại giữa hai nút bất kỳ.
2. Tổng kinh phí xây dựng các tuyến đường thêm là ít nhất.

Dữ liệu: Vào được cho bởi file MGT.DAT trong đó :

- Dòng thứ nhất ghi hai số N và K, N<=100.
- Trong K dòng tiếp theo, mỗi dòng ghi hai số là tên của hai nút là hai đầu của một tuyến đường đã xây dựng.
- Cuối cùng là N dòng, dòng thứ I ghi N số A[I,1], . . , A[I,N].

Kết quả : Ghi ra file GP.OUT như sau:

- Dòng thứ nhất ghi số CP là chi phí xây dựng thêm.
- Nếu CP>0, trong một số dòng tiếp theo, mỗi dòng ghi hai số là tên của hai nút là hai đầu của một tuyến đường cần xây dựng thêm.

Ví dụ

MGT.DAT	GP.DAT
5 4	1
1 2	3 4
2 3	
3 1	
4 5	
0 1 1 1 1	
1 0 1 1 1	
1 1 0 1 1	
1 1 1 0 1	
1 1 1 1 0	

Hướng Dẫn :

Đầu tiên dùng thuật toán loang để phân các thành phần của đồ thị ra số các thành phần liên thông . Lúc này ta coi mỗi thành phần liên thông là một đỉnh mới của đồ thị . (số đỉnh của đồ thị mới này bằng số thành phần liên thông có trong đồ thị đã cho) .Hai thành phần liên thồn được nối với nhau bằng cạnh có độ dài ngắn nhất (mà mỗi đỉnh trong cạnh là hai đỉnh của hai thành phần liên thông) . Thì ta sẽ dùng thuật toán người láng giềng gần nhất để xây dựng cách nối các đỉnh của đồ thị mới này bằng các trọng số ít nhất .

Bài toán 42 : Hệ thống đường cao tốc

Hệ thống đường cao tốc hiện tại ở thành phố A mới đảm bảo việc đi lại giữa một số nút giao thông và cũn nhiều nỳt giao thụng chưa có đường cao tốc đi qua. Để giải toả tỡnh trạng ỏch tắc giao thụng, chớnh quyền thành phố quyết định phát triển hệ thống đường cao tốc của thành phố sao cho có thể đi lại giữa hai nút giao thông bất kỡ. Cú N nỳt giao thụng, được đánh số từ 1 đến N. Nút giao thông i được cho bởi toạ độ (x_i, y_i) trong hệ tạo độ Đề các. Mỗi tuyến đường cao tốc nối hai nút giao thông. Tất cả các tuyến đường hiện có cũng như sẽ được phát triển được xây dựng theo đường thẳng nối chúng, vỡ thế độ dài của mỗi tuyến đường chính là khoảng cách giữa hai điểm tương ứng với hai nút giao thông. Tất cả các tuyến đường đều là hai chiều. Các tuyến đường có thể cắt nhau nhưng người sử dụng phương tiện giao thông chỉ được đổi tuyến đi ở các nút giao thông là đầu mút của các tuyến đường.

Chớnh quyền thành phố muốn tỡm cỏch xõy dựng bổ sung một số tuyến đường cao tốc nối các nút giao thông với chi phí nhỏ nhất nhưng vẫn đảm bảo sự đi lại giữa mọi nút giao thông. Chi phí xây dựng tỷ lệ thuận với đội dài tuyến đường.

Dữ liệu : highway.inp

Dũng đầu tiên là số N (N<=750), dũng thứ i trong số N dũng tiếp theo, mỗi dũng chứa hai số x_i, y_i là toạ độ của nỳt giao thụng thứ i. Dũng tiếp theo chứa số M là số tuyến đường cao tốc hiện có (0<=M<=1000). Dũng thứ j trong số M dũng tiếp theo chứa hai chỉ số của hai nỳt giao thụng là đầu mút của tuyến đường j.

Kết quả : highway.out

Dũng đầu tiên là số k là số lượng tuyến đường cần xây dựng bổ sung. Dũng thứ i trong số K dũng tiếp theo chứa hai chỉ số của hai nỳt giao thụng là đầu mút của tuyến đường cần xây dựng bổ sung.

Vớ dụ :

highway.inp	highway.out
9	5
1 5	1 6
0 0	3 7
3 2	4 9
4 5	5 7
5 1	8 3
0 4	
5 2	
1 2	
5 3	
4	
1 3	
9 7	

1 2
2 3

Hướng Dẫn :

Giống như bài toán MGT ở trên .

Ta xây dựng đồ thị có trọng số gồm N đỉnh tương ứng N nút giao thông, gọi trọng số giữa hai đỉnh bất kì bằng 0 nếu có đường cao tốc nối hai nút tương ứng hoặc bằng khoảng cách giữa hai nút tương ứng. Bài toán đưa về tìm cây khung nhỏ nhất trên đồ thị, ta dùng thuật toán Prim, hoặc Kruskal.

Bài toán 43: mạng truyền thông đơn tuyến

Đề bài :

Một mạng truyền thông gồm N trạm có tên từ 1 đến N, 6<=N<=100. Với hai trạm I, J bất kỳ của mạng có không quá một kênh truyền *một chiều* trực tiếp từ I đến J. Từ một trạm bất kỳ, có không quá ba kênh truyền tin trực tiếp đến các trạm khác. Một *đường truyền tin* từ trạm U đến trạm V là một dãy trạm $U=T_1, T_2, \ldots, T_k=V$ sao cho với 1<=h<k, có kênh truyền trực tiếp từ T_h đến T_{h+1}.

Một đường truyền tin được gọi là *đơn* nếu các trạm trên đường truyền tin khác nhau từng đôi.

Mạng truyền thông được gọi là *đơn tuyến* nếu với hai trạm U, V bất kỳ, có không quá một đường truyền tin đơn từ U đến V.

Bài toán đặt ra là cho một mạng truyền thông. Hãy cho biết mạng đó có đơn tuyến không?

Dữ liệu : Vào được cho bởi file COMNET.INP trong đó dòng thứ nhất ghi số N. Trong N dòng tiếp theo, dòng thứ I ghi số T[I] là số trạm có kênh truyền tin trực tiếp từ I đến, nếu T[I]>0, tiếp theo là T[I] số là tên các trạm có kênh truyền tin trực tiếp từ I đến.

Kết quả: Ghi ra file COMNET.OUT như sau: dòng thứ nhất ghi số 1/0 tuỳ theo mạng là/không là đơn tuyến. Nếu dòng thứ nhất số 1, trong N-1 dòng tiếp theo, dòng thứ I ghi độ dài đường truyền tin đơn từ trạm N đến trạm I nếu có đường truyền tin từ N đến, nếu không có đường truyền tin từ trạm N đến, ghi số 101. Nếu dòng thứ nhất ghi số 0, trong dòng thứ hai ghi hai số U, V là tên hai trạm mà từ U đến V có hơn một đường truyền tin đơn, tiếp theo là hai dòng, mỗi dòng ghi lần lượt các trạm trên một đường truyền tin đơn từ U đến V, bắt đầu từ U và kết thúc tại V. Có thể có nhiều cặp trạm vi phạm điều kiện đơn tuyến nhưng chỉ cần chỉ ra một cặp trạm và với cặp trạm đó, chỉ ra hai đường truyền tin đơn.

Trong các file dữ liệu, hai số liên tiếp trên một dòng ngăn cách nhau bởi ít nhất một dấu trống.

Ví dụ

COMNET.INP	COMNET.OUT	COMNET.INP	COMNET.OUT
6	1	6	0
1 3	3	1 2	1 2
1 1	2	1 3	1 2
2 2 4	1	1 4	1 2 3 4 5 6 2
2 5 6	2	1 5	
1 4	3	1 6	
1 3		2 1 2	

Hướng Dẫn :

Bài toán 44 : Thứ tự đỉnh

Đề bài :

Cho một đa giác lồi N đỉnh . Các đỉnh được đánh số một cách ngẫu nhiên từ 1 đến N . Không có hai đỉnh nào trùng số . Người ta kẻ M đường chéo không cắt nhau (nhưng có thể có điểm chung là đỉnh của đa giác) . Cho biết cặp điểm đầu – cuối của các cạnh và các đường chéo đã kẻ .

Yêu cầu : Hãy xác định cách đánh số các đỉnh đa giác theo chiều kim đồng hồ hoặc ngược chiều kim đồng hồ .

Ví Dụ : với N= 4 và thông tin về 4 cạnh và 1 đường chéo (1,3) , (4,2) , (1,2) , (4,1) ,(2,3) . Thứ tự đánh số các đỉnh là 1 , 3 ,2 ,4

Dữ liệu : Vào từ file Order.Inp :

- Dòng đầu tiên là 2 số nguyên N , M (4<=N<=10000 , 0<=M<=N-3)
- N+M dòng tiếp theo mỗi dòng chứa 2 số nguyên cho biết cặp điểm đầu và cuối của một cạnh hay đường chéo . Không có hai dòng nào giống nhay

Kết quả : Ghi ra file Order.Out một dòng N số nguyên số hiệu các đỉnh đa giác bắt đầu từ đỉnh 1 theo trình tự từ 1 đi tới đỉnh có số bé hơn trong hai đỉnh nối với 1 .

Ví Dụ :

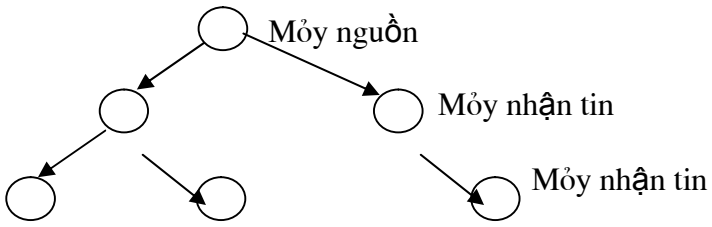
ORDER.INP		ORDER.OUT
4 1 1 3 4 2 1 2 4 1 2 3		1 3 2 4

Hướng Dẫn :

Bài toán 45: Truyền thông trên mạng

Đề Bài :

Cùng với giao thức TCP/IP, mà trong đó thông tin được trao đổi giữa hai máy tính thông qua việc xác lập kết nối, trong mạng còn có một kiểu giao thức gọi là UDP (user datagram protocol) mà trong đó không cần xác lập kết nối lôgic trực tiếp giữa máy nguồn và máy nhận tin. Điều đó cho phép thực hiện việc truyền thông rộng rãi trong mạng mà không cần truyền các gói dữ liệu cho từng máy nhận tin một. Việc truyền tin được tiến hành theo sơ đồ dạng cây như chỉ ra trong hình sau đây:



Giả sử đã biết sơ đồ liên kết của các máy trong mạng dưới dạng bảng các kết nối giữa các máy tính trong mạng, trong đó cho biết máy nguồn và các máy nhận tin. Cần tìm tập gồm một số ít nhất máy tính (kể cả máy nguồn và các máy nhận tin) cần tham gia vào việc truyền thông để toàn bộ các máy nhận tin đều có thể tiếp nhận được thông tin từ máy nguồn.

Dữ liệu: Vào từ file UDP.INP:

- Dòng đầu tiên chứa 3 số N, M, b; trong đó N là sốmáy trong mạng, M - là số máy khác trong mạng; Số đầu tiên K_i là số máy tính trong mạng được kết nối với máy i, tiếp theo là K_i chỉ số của các máy nối với máy i.
- Dòng cuối cùng ghi chỉ số của các máy nhận tin.

Kết quả: Ghi ra file UDP.OUT:

- Dòng đầu tiên ghi số lượng máy cần chọn P (quy ước P = 0, nếu không tìm được tập các máy thoả mãn yêu cầu đặt ra);
- Nếu P > 0 thì ghi chỉ số của các máy được chọn.

Ví dụ:

UDP.INP	UDP.OUT
10 3 4 3 2 5 6 3 1 3 5 4 2 4 8 9 3 3 5 8 3 1 2 4 2 1 8	6 2 4 5 6 7 8

1 8 6 3 4 6 7 9 10 2 3 8 1 8 2 6 7	

Hướng dẫn :

Phần 6 : **các bài toán lập lịch**

A. Lý thuyết :

1. Thuật toán Johnson

Bài toán 1:

" Mỗi một chi tiết D1,D2,...Dn cần phải được lần lượt gia công trên 2 máy A,B . Thời gian gia công chi tiết Di trên máy A là ai trên máy B là bi (i=1,2..n) . Hãy tìm lịch (trình tự gia công) các chi tiết trên hai máy sao cho việc hoàn thành gia công tất cả các chi tiết là sớm nhất " .

Thuật toán Johnson :

- Chia các chi tiết thành 2 nhóm : nhóm N1, gồm các chi tiết Di thoả mãn a1<b1 , tức là min(ai,bi)=ai và nhóm N2 gồm các chi tiết Di thoả mãn ai>bi tức là min(ai,bi)=bi . Các chi tiết Di thoả mãn ai=bi xếp vào nhóm nào cũng được
- Sắp xếp các chi tiết trong N1 theo chiều tăng của các ai và sắp xếp các chi tiết trong N2 theo chiều giảm của các bi
- Nối N2 vào đuôi N1 . dãy thu được (đọc từ trái sang pải) sẽ là lịch gia công .

Bài toán 2 :

" Xét bài toán gia công N chi tiết trên 3 máy theo thứ tự A,B,C với bảng thời gian ai ,bi,ci ,i=1,2,..n thoả mãn :

max bi<=min ai hoặc max bi <= min ci "

Thuật toán :

Lịch gia công tối ưu trên 3 máy sẽ trùng với lịch gia công tối ưu trên 2 máy : máy thứ nhất với thời gian ai+bi và máy thứ hai với thời gian bi+ci .

2. Thuật toán More

3. Các phương pháp khác :

Thông thường thì các bài toán dạng này thường có rất nhiều cách giải . Nhưng thông dụng nhất là thuật toán quy hoạch động và duyệt nhánh cận . Ngoài ra một số còn đưa về đồ thị .

B. Bài toán :

Bài toán 46 : **Lập lịch ưu tiên đúng hạn**

Đề bài :

Có n công việc đánh số từ 1 đến n và một máy để thực hiện chúng. Biết:

- p_1 là thời gian cần thiết để hoàn thành công việc j;
- d_i là thời hạn hoàn thành công việc i.

Mỗi công việc cần được thực hiện liên tục từ lúc bắt đầu cho tới khi kết thúc, không cho phép ngắt quãng. Khoảng thời gian thực hiện hai công việc bất kỳ chỉ được có nhiều nhất 1 điểm chung. Giả sử C_1 là thời điểm hoàn thành trễ hạn, còn nếu $C_i \le d_i$ thì ta nói công việc i được hoàn thành đúng hạn.

Yêu cầu: Tìm trình tự thực hiện các công việc sao cho số công việc được hoàn thành đúng hạn là lớn nhất.

Dữ liệu: Vào từ file văn bản LICHD.JNP:

- Dòng đầu tiên chứa số nguyên dương n ($0 \le n \le 100$)
- Dòng thứ 2 chứa n số nguyên dương $p_1, p_2, ... p_n$.
- Dòng thứ 3 chứa n số nguyên dương $d_1, d_2, ... d_n$.

Kết quả: Ghi ra file văn bản LICHD.OUT

- Dòng đầu tiên ghi số lượng công việc được hoàn thành đúng hạn theo trình tự thu được.
- Dòng tiếp theo ghi trình tự thực hiện các công việc đã cho.

Ví dụ:

LICHD.INP
6 2 4 1 2 3 1 3 5 6 6 7 8

LICHD.OUT
4 1 3 4 6 2 5

Hướng Dẫn :

- Sắp Xếp theo thứ tự tăng dần của D[i]
- Chúng ta xét như sau :

  Đầu tiên cho :

  time :=0 ;

  + Lấy thực hiện các việc theo trật tự tăng dần . Nhưng đến công việc nào mà Thời gian thực hiện nó sẽ vượt qua giới hạn D[i] của nó thì :

  Tìm các công việc đã được xếp trước nó , công việc nào khi bị thay thế bởi công việc này mà thời gian thực hiện xong nó là nhỏ nhất thì ta sẽ thay thế .

Các thủ tục của bài toán :

- Qsort (tăng theo giá trị D[i])
- Thủ tục Xep(i:Integer) là thủ tục xếp việc :

```
procedure xep(x : integer);
var i,t,cs : integer;
   min    : integer;
```

```
begin
  inc(top);
  st[top]:=x;
  if time+p[x]<=d[x] then
    begin
      time:=time+p[x];
      exit;
    end;
  min:=time;
  cs:=top;
  for i:=1 to top-1 do
    begin
      t:=time+p[x]-p[st[i]];
      if (t<min) and (t<=d[x]) then
        begin
          min:=t;
          cs:=i;
        end;
    end;
  dec(top);
  for i:=cs to top do
    st[i]:=st[i+1];
  time:=min;
end;
```

Trong đó Top là số hiệu đỉnh các công việc được thực hiện . Còn mảng St là mảng nêu tuần tự các công việc được thực hiện .

Bài toán 47 : Lập lịch trên m máy

Đề bài :

Có n công việc và một hệ thống gồm m máy có tính năng hoàn toàn giống nhau để thực hiện chúng. Các công việc được đánh số từ 1 đến n. Các máy được đánh số từ 1 đến m. Với mỗi công việc i biết:

- Thời gian hoàn thành p_i (là thời gian cần thiết để hoàn thành công việc).
- Thời điểm sẵn sàng r_i (công việc i không được tiến hành sớm hơn thời điểm r_i);
- Thời hạn hoàn thành d_i (tức là công việc i phải hoàn thành không muộn hơn thời điểm d_i);

Giả thiết là $d_i - r_i \geq p_i$, i = 1, 2, ..., n. Quá trình thực hiện công việc cho phép được ngắt quãng, nghĩa là trong quá trình thực hiện, một máy có thể ngắt việc thực hiện công việc đang tiến hành trên nó để chuyển sang thực hiện một công việc khác và công việc đang thực hiện dở có thể lại tiếp tục được thực hiện trên bất kỳ máy nào của hệ thống tại bất kỳ thời điểm nào sau đấy. Giả thiết rằng:

- Tại mỗi thời diểm mỗi máy chỉ thực hiện không quá một công việc và mỗi công việc được thực hiện trên không quá một máy.
- Thời gian để máy chuyển sang thực hiện công việc khác giả thiết bằng 0.
- Thời điểm bắt đầu thực hiện các công việc là 0.

Yêu cầu: Hãy lập lịch để tất cả các công việc được hoàn thành đúng hạn.

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

- Dòng đầu tiên ghi 2 số n, m ($1 \le n, m \le 100$);
- Dòng thứ i trong số n dòng tiếp theo ghi ba số nguyên dương p_i, r_i, d_i (i = 1, 2, ..., n).

Kết quả: Ghi ra file văn bản PARLICH.OUT.

- Dòng đầu tiên ghi YES (NO) nếu như có (không có) lịch thoả mãn yêu cầu đặt ra.
- Nếu có lịch thì n nhóm dòng tiếp theo sẽ mô tả lịch tìm được, trong đó nhóm dòng thứ i mô tả lịch thực hiện công việc thứ i:

1) Dòng đầu tiên ghi k_i ($k_i \ge 0$) là số lần việc thực hiện công việc i bị ngắt quãng;

2) Dòng thứ j trong số $k_i + 1$ dòng tiếp theo mô tả khoảng thời gian thứ j trong quá trình thực hiện công việc i bao gồm 3 số nguyên dương theo thứ tự là thời điểm bắt đầu, thời điểm dừng và chỉ số của máy thực hiện công việc i.

Trong các file dữ liệu và kết quả các giá trị số ghi trên một dòng cách nhau ít nhất một dấu trắng.

Ví dụ:

PARLICH.INP
4 2 4 2 7 3 3 8 3 3 7 5 1 10

PARLICH.OUT
YES 0 2 6 1 1 3 4 2 6 8 1 0 4 7 2 1 1 3 2 7 10 2

Hướng Dẫn :

Bài toán 48 : **Lập lịch**

Đề bài :

Có n chương trình, đánh số từ 1 đến n (n ≤ 1000) cần tuần tự thực hiện trên một máy tính bắt đầu từ thời điểm 0 (nghĩa là tại mỗi thời điểm máy chỉ thực hiện một chương trình và một chương trình cần được thực hiện liên tục từ khi bắt đầu cho đến khi hoàn thành).

Với mỗi chương trình i , biết các thông tin:

- thời điểm thực hiện p[i]
- thời điểm sẵn sàng thực hiện r[i]
- thời điểm phải hoàn thành d[i].

Các giá trị được giả thiết là nguyên dương.

Chương trình i chỉ có thể bắt đầu thực hiện ở thời điểm không sớm hơn r[i]. Giả sử t[i] là thời điểm hoàn thành chương trình i. Nếu t[i] ≤ d[i] thì ta nói chương trình i được hoàn thành đúng hạn, còn nếu trái lại ta nói chương trình i bị trễ hạn.

Giả thiết rằng đối với hai chương trình i và j bất kỳ, nếu chương trình i có thời điểm sẵn sàng không sớm hơn thời điểm sẵn sàng của chương trình j (tức là r[i] ≥ r[j]) thì thời điểm phải hoàn thành của nó cũng không sớm hơn thời điểm phải hoàn thành của chương trình j (tức là d[i] ≥ d[j]). Thời điểm để máy chuyển từ việc thực hiện chương trình này sang thực hiện chương trình khác giả thiết bằng 0.

Yêu cầu: tìm thứ tự thực hiện các chương trình để tổng số chương trình bị trễ hạn là nhỏ nhất.

Dữ liệu vào: file văn bản BL4.INP gồm:

- Dòng thứ nhất ghi số n,
- Dòng thứ i trong số n dòng tiếp theo ghi 3 số p[i], r[i] và d[i] cách nhau ít nhất một dấu trắng.

Kết quả ghi ra file văn bản BL3.OUT:

- Dòng đầu tiên ghi tổng số công việc bị trễ hạn theo thứ tự thực hiện tìm được,
- Dòng thứ i trong số n dòng tiếp theo ghi số hiệu của chương trình cần thực hiện thứ i.

Ví dụ : các file dữ liệu - kết quả có thể

BL4.INP	BL4.OUT
8	4
4 8 20	1
25 16 41	4
25 22 47	5
3 22 47	7

16	27	47	2
24	39	63	3
9	45	63	6
24	47	71	8

Hướng Dẫn :

Bài toán 49 : xếp lịch học

Đề bài :

Có N thày giáo với tên là 1, 2, . ., N, N môn học cũng có tên tương ứng là 1, 2, . ., N, N<=50, thày i dạy môn học i cho hai lớp học A, B. Thày giáo i phải dạy môn i cho lớp A/B trong A[i]/B[i] ngày. Mỗi thày khi dạy môn của mình cho một lớp thì phải dạy liên tục cho tới khi xong. Trong mỗi ngày, mỗi thày chỉ dạy được không quá một lớp và mỗi lớp chỉ học được không quá một thày. Trong số N môn học, có K môn $i_1, i_2, . . i_K$, mà môn i_{j-1} phải học trước môn i_j, 1<j<=K, 0<=K<=N. Việc học bắt đầu từ ngày thứ nhất.

Hãy thu xếp việc học cho hai lớp sao cho ngày kết thúc việc học của cả hai lớp là sớm nhất có thể được (có thể một trong hai lớp kết thúc trước ngày đó).

Dữ liệu vào được cho bởi file LICH.INP trong đó dòng thứ nhất ghi N số nguyên dương A[1], . ., A[N], dòng thứ hai ghi N số nguyên dương B[1], . ., B[N], dòng thứ ba ghi tên K môn học $i_1, i_2, . . i_K$, nếu K=0 thì dòng này ghi số 0.

Kết quả ghi ra file LICH.OUT như sau: dòng thứ nhất ghi ngày kết thúc việc học của cả hai lớp, tiếp theo là N dòng, dòng thứ I ghi hai số C[I] và D[I] với ý nghĩa thày I dạy cho lớp A ngày C[I] và dạy cho lớp B ngày D[I].

Ví dụ

LICH.INP				LICH.OUT	
1	2	3	4	10	
4	3	2	1	10	1
3	2			4	7
				1	5
				6	10

Hướng Dẫn :

Sử dụng thuật toán JonhSon .

Bài toán 50 : Etime

Đề bài :

Có N công việc đánh số từ 1 đến N cần được bố trí thực hiện trên một máy . Biết :

- Pi – thời gian cần thiết để thực hiện công việc i ;
- Ri – thời điểm sẵn sàng của công việc i (nghĩa là công việc không được thực hiện ở thời điểm sớm hơn Rj)

Trong số các công việc đã cho có một số công việc chỉ được tiến hành sau khi một số công việc nào đó đã hoàn thành . Giả sử thời gian để máy chuyển từ việc thực hiện công việc này sang công việc khác là không đáng kể , và thời gian bắt đầu thực hiện các công việc là 0 .

Yêu cầu : Tìm trình tự thực hiện các công việc sao cho việc hoàn thành tất cả các công việc xảy ra ở thời điểm sớm nhất .

Dữ liệu : Vào từ file Etime.Inp :

- Dòng đầu tiên chứa số nguyên dương N (0<N<=100)
- Dòng thứ 2 chứa N số nguyên dương P1,P2,..Pn
- Dòng thứ 3 chứa N số nguyên dương R1,R2,..Rn
- Dòng thứ i trong số N dòng tiếp theo chứa chỉ số các công việc phải hoàn thành trước khi thực hiện công việc (qui ước ghi số 0 nếu công việc i có thể thực hiện độc lập)

Kết quả : Ghi ra file Etime.Out :

- Dòng đầu tiên ghi thời điểm hoàn thành tất cả các công việc
- Dòng tiếp theo ghi trình tự thực hiện các công việc

Ví Dụ :

ETIME.INP		ETIME.OUT
6 5 9 7 6 14 4 2 1 26 20 0 0 1 6 1 1 3 1 3 0		45 6 1 2 3 5 4

Hướng Dẫn :

Chúng ta xây dựng đồ thị A[i,j] trong đó biểu diễn : Việc i sẽ làm trước việc j . Dùng thuật toán NumBering (Toán rời rạc – Trang 205) . Để tìm thứ tự thực hiện các công việc . Sau đó ta sẽ tính được thời gian chung cho cả công việc .

Bài toán 51 : Day 2. Probtem 1. Sightseeing trip (Chuyển tham quan)

Đề bài :

Một chi nhánh dịch vụ du lịch ở một thành phố nọ quyết định mời chào các khách hàng cua rmình nogài những dịch vụ hấp dẫn khác, một chuyền tham quan miễn phí thành phố. Để có thể thu được càng nhiều lợi nhuận càng tốt, chi nánh đưa ra quyết định tinh ranh sau: Cần tìm một hành trình ngắn nhất bắt đầu và kết thúc tại cùng một địa điểm nào đó. Nhiệm vụ của bạn là viết chương trình tìm ra hành trình như vậy.

Trong thành phố có N núi giao thông đánh số từ 1 đến N và M đoạn đường phố hai chiều đánh số từ 1 đến M. Hai nút giao thông có thể được nối với nhau bởi một vài đoạn đường phố, nhưng không có đoạn đường phố nào nối một nút giao thông với chính nó. Mỗi một hành trình tham quan là một dãy các chỉ số các đoạn phố y_1, ..., y_k, k > 2. Đoạn đường phố y_i (1 < = i < = k-1) nối hai nút giao thông x_i và x_{i+1}, đoạn đường phố y_k nối hai nút giao thông x_k và x_1. Các chỉ số x_1, ..., x_k khác nhau từng đôi.

Độ dài của hành trình tham quan là tổng độ dài của các đoạn đường phố trên hành trinh, tức là số $L(y_1) + L(y_2) + ... + L(y_k)$, trong đó $L(y_i)$ là độ dài của đoạn đường phố y_i (1 < = i < = k). Chương trình của bạn cần tìm hành trình tham quan có độ dài nhỏ nhất, hoặc thông báo rằng không thể tìm được, bởi vì không tồn tại hành trình tham quan trong thành phố.

Dữ liệu: Dòng đầu tiên trong file TRIP.IN chứa hai số nguyên dương: số nút giao thông N <=100 và số đoạn đường phố M <=10000. Mỗi một trong số M dòng tiếp theo mô tả một đoạn đường phóo bao gồm 3 số nguyên dương: Chỉ số của nút giao thông đầu, chỉ số của nút giao thông cuối và độ dài của đoạn đường phố (độ dài là số nguyên không vượt quá 500).

Kết quả: Ghi ra một dòng của file TRIP.OUT dòng thông báo no solution . Nếu không tồn tại hành trình tham quan, hoặc chỉ số của các nút giao thông trên hành trình tham quan ngắn nhất theo thứ tự hành trình đi qua chúng (tức là các số từ x_1 đến x_k theo định nghĩa hành trình tham quan đã nêu ở trên).

Ví dụ 1:

TRIP.IN

```
5 7
1 4 1
1 3 3000
3 1 10
1 2 16
2 3 100
2 5 15
5 3 20
```

TRIP.OUT (một trong các lời giải đúng)

1 3 5 2

Ví dụ 2:

TRIP.IN

1 3
1 2 10
1 3 20
1 4 30

TRIP.OUT (duy nhất một lời giải đúng)

No solution