**Rèn luyện kỹ năng duyệt trong đồ thị**

Để rèn luyện kỹ năng duyệt trong đồ thị , tôi xin giới thiệu cách giải một vài bài toán sau đây:

**Bài 1 ĐÈN TRANG TRÍ**

Rôn mua một bộ đèn trang trí gồm ***n*** đèn (1 ≤ ***n*** ≤ 1 000). Mỗi đèn có một công tắc để bật hay tắt riêng đèn đó. Mỗi giây Rôn có thể bật hoặc tắt một bóng đèn tùy chọn. Ban đầu tất cả các bóng đều ở trạng thái tắt. Một cấu hình của bộ đèn là trạng thái khi một số đèn nào đó được bật sáng, những đèn còn lại – tắt. Rôn đặc biệt thích một số cấu hình vì chúng có vẻ phù hợp với khung cảnh căn phòng của Rôn.Mỗi trạng thái của bộ đèn được biểu diễn bằng một xâu ***n*** ký tự từ tập {0, 1}. Ký tự thứ ***i*** xác định trạng thái đèn thứ ***i***, 0 tương ứng với trạng thái đèn tắt, 1 là trạng thái đèn được bật sáng. Ví dụ, với ***n*** = 3 và Rôn đặc biệt thích 3 cấu hình {1, 0, 1}, {0, 1, 0}, {1, 1, 1}. Để kiểm tra xem cấu hình nào là thích hợp nhất Rôn phải lần lượt bật tắt một số đèn. Trong trường hợp này Rôn cần 4 giây để xem xét hết mọi cấu hình.

***Yêu cầu***: Cho biết ***n*** và ***m***, trong đó ***m*** – số cấu hình khác nhau mà Rôn đặc biệt yêu thích (1 ≤ ***m*** ≤ 15). Hãy xác định thời gian tối thiểu cần thiết để kiểm tra hết tất cả các trạng thái mà Rôn quan tâm.

***Dữ liệu***: Vào từ file văn bản GARLAN.INP:

* Dòng đầu tiên chứa 2 số nguyên ***n*** và ***m***,
* Mỗi dòng trong ***m*** dòng tiếp theo chứa xâu ***n*** ký tự xác định một cấu hình Rôn yêu thích.

***Kết quả***: Đưa ra file văn bản GARLAN.OUT một số nguyên – thời gian tối thiểu kiểm tra các cấu hình.

***Ví dụ***:

|  |  |  |
| --- | --- | --- |
| GARLAN.INP |  | GARLAN.OUT |
| **3 3**  **101**  **010**  **111** |  | **4** |
|  |

**Lời giải : - Mỗi trạng thái coi như 1 đỉnh của đồ thị (trạng thái ban đầu là đỉnh số 0 )**

* **Trong số của mỗi cạnh là chi phí chuyển từ trạng thái nọ sang trạng thái kia**
* **Bìa toán trở thành tìm đường đi từ 0 qua lần lượt các đỉnh với tổng trọng số nhỏ nhất**
* **Gọi mỗi canh là (u,v,w), trong đó u,v là 2 đỉnh , w là trọng số .**

**Trong ví dụ trên ta có các cạnh (0,1,2) , (0,2,1) , (0,3,3) , (1,2,3) , (1,3,1) , (2,3,2)**

Ta được đường đi tốt nhất là 0-2-1-3 ( hoặc 0-2-3-1) với chi phí = 4.

**Bài 2 BẮC CẦU**

Chính phủ quốc đảo Oceani quyết định xây dựng ***m*** chiếc cầu nối ***n*** đảo của mình, tạo một mạng lưới giao thông đường bộ cho phép đi từ dảo bất kỳ tới đảo khác bằng đường bộ (trực tiếp hoặc qua một số đảo trung gian). Mỗi cây cầu sẽ nối 2 đảo khác nhau và cho phép đi lại hai chiều. Các đảo được đánh số từ 0 đến ***n***-1. Bị hạn chế bởi kinh phí và nguồn nhân lực, người ta quyết định sẽ xây dựng lần lượt từng chiếc cầu một và lên kế hoạch xác định cầu và trình tự xây. Mỗi cây cầu được xác định bởi cặp đảo ***u***, ***v*** mà nó nối. Trong quá trình thực hiện kế hoạch có thể đến một lúc nào đó từ một đảo đã có thể đi đến bất kỳ đảo khác bằng đường bộ.

Ví dụ, ở Oceani có 4 đảo và người ta quyết định xây dựng 5 cầu theo trình tự lần lượt là 0 – 1, 0 – 2, 1 – 2, 2 – 3, 3 – 0. Tuy vậy, không cần chờ đợi đến khi hoàn thành kế hoạch xây cầu, sau khi cầu thứ 4 được xây xong tất cả các đảo đã được nối liền bằng đường bộ.

***Yêu cầu***: Cho ***n***, ***m*** và các cây cầu dự kiến xây. Thông tin về các cây cầu đưa ra theo đúng trình tự xây dựng. Hãy xác định số cầu tối thiểu cần xây theo kế hoạch để từ một đảo đã có thể đi đến bất kỳ đảo khác bằng đường bộ.

***Dữ liệu***: Vào từ file văn bản BRIDGES.INP:

* Dòng đầu tiên chứa 2 số nguyên ***n*** và ***m*** (1 ≤ ***n*** ≤ 106, 1 ≤ ***m*** ≤ 5×106),
* Dòng thứ ***i*** trong ***m*** dòng tiếp theo chứa 2 số nguyên ***u*** và ***v*** xác định cây cầu thứ ***i*** cần xây.

***Kết quả***: Đưa ra file văn bản BRIDGES.OUT kết quả tìm được dưới dạng một số nguyên.

***Ví dụ***:

|  |  |  |
| --- | --- | --- |
| BRIDGES.INP |  | BRIDGES.OUT |
| **4 5**  **0 1**  **0 2**  **1 2**  **2 3**  **3 0** |  | **4** |
|  |

**Lời giải : - Tìm đáp số bằng tìm kiếm nhị phân ( Ds min = n-1 , ds max = m )**

* **Với mỗi ds dự đoán , ta chỉ việc kiểm tra tính liên thông với danh sách cạnh từ 1 đến ds .**

**Bài 3 Đường đi của Robot**

Một bảng hình chữ nhật có kích thước MxN (M,N nguyên dương và không lớn hơn 100) được chia thành các ô vuông đơn vị bằng các đường thẳng song song với các cạnh. Một số ô vuông nào đó có thể đặt các vật cản. Từ một ô vuông, Robot có thể đi đến một ô vuông kề cạnh với nó nếu ô vuông đó không có vật cản. Hỏi rằng nếu Robot bắt đầu xuất phát từ một ô vuông không có vật cản thuộc dòng K, cột L thì có thể đi đến được ô vuông không có vật cản thuộc dòng H, cột O hay không? Nếu có thì hãy chỉ ra đường đi qua ít ô vuông nhất.

Dữ liệu vào là tệp văn bản BAI3.INP có cấu trúc:

- Dòng đầu tiên ghi các chữ số M, N, K, L, H, O. Các số ghi cách nhau ít nhất một ký tự trống;

- M dòng tiếp theo, mỗi dòng ghi N số 1 hoặc 0 tuỳ thuộc vào ô vuông tương ứng trong bảng hình chữ nhật nêu trên có vật cản hay không (ghi số 1 nếu có vật cản); các số trên mỗi dòng ghi liên tiếp nhau.

Dữ liệu ra là tệp văn bản BAI3.OUT có cấu trúc:

Nếu Robot có thể đi được từ ô vuông thuộc dòng K, cột L đến ô vuông thuộc dòng H, cột O thì:

- Dòng đầu tiên ghi ‘Co duong di ‘;

- Các dòng tiếp theo, mỗi dòng ghi 2 số là chỉ số dòng và chỉ số cột của các ô vuông trong đường đi tìm được từ ô vuông thuộc dòng K, cột L đến ô vuông thuộc dòng H, cột O mà qua ít ô vuông nhất. Hai số trên mỗi dòng ghi cách nhau ít nhất một ký tự trống;

- Ngược lại, nếu Robot không thể đi được từ ô vuông thuộc dòng K, cột L đến ô vuông thuộc dòng H, cột O thì ghi ‘Khong co duong di’.

Ví dụ 1:

|  |  |
| --- | --- |
| robot.inp:  4 7 3 4 2 6  1000000  0010100  0000000  1101000 | robot.out:  Co duong di  3 4  3 5  3 6  2 6 |

Ví dụ 2:

|  |  |
| --- | --- |
| robot.inp:  4 7 2 2 1 3  1010000  0010100  0100000  1101000 | robot.out:  Khong co duong di |

**Phân tích:**

Yêu cầu của bài toán thực chất là tìm đường đi từ ô [K,L] đến ô [H,O] sao cho qua ít ô vuông nhất. Ta dễ thấy thuật toán để xử lý một cách hợp lý nhất là thuật toán Loang. Ta bắt dầu “loang” từ ô [K,L], nếu “loang” đến được ô [H,O] thì có đường đi, ngược lại không có đường đi.

Hàng đợi phục vụ “loang” được thể hiện bởi mảng 2 chiều Q:Array[1..2,Mmax\*Max] of Byte; hàng thứ 1 của Q để lưu thông tin chỉ số hàng, hàng thứ 2 lưu thông tin của chỉ số cột của các ô khi nạp vào Q.

Mảng A lưu thông tin tình trạng các ô - có vật cản hay không của bảng hình chữ nhật chứa các ô vuông.

Mảng P dùng để đánh dấu những ô đã “loang” đến; đồng thời để phục vụ cho việc truy xuất đường đi sau này nên khi ô [i,j] được “loang” đến thì P[i,j] được gán giá trị là r (r là giá trị tương ứng với hướng mà ô trước đó “loang” đến, hướng nào tương ứng với giá trị k bao nhiêu tuỳ theo quy định, ví dụ r = 1 - sang phải, 2 - đi xuống, 3 - sang trái, 4 - đi lên).

Sau khi thực hiện xong việc “loang”, nếu P[H,O] = 0 thì điều có có nghĩa là ô [H,O] chưa được “loang” đến (không có đường đi), nếu P[H,O] = r (r=1..4 - loang theo 4 hướng) thì dựa vào hướng “loang” đến mà ta tìm được ô trước đó, rồi ta lại dựa vào giá trị k của ô tìm được ta tìm được ô trước đó nữa ... quá trình trên kết thúc khi tìm được ô [K,L].

Sau khi “loang” xong thì giá trị các phần tử trong mảng Q không còn giá trị sử dụng nữa nên ta có thể dùng mảng Q phục vụ cho việc truy xuất kết quả.

**Bài 4 Gặp gỡ của hai Robot.**

Trên một lưới ô vuông MxN (M,N<100), người ta đặt Robot A ở góc trái trên, Robot B ở góc phải dưới. Mỗi ô của lưới ô có thể đặt một vật cản hoặc không (ô trái trên và ô phải dưới không có vật cản). Hai Robot bắt đầu di chuyển đồng thời với tốc độ như nhau và không Robot nào được dừng lại trong khi Robot kia di chuyển (trừ khi nó không thể đi được nữa). Tại mỗi bước, Robot chỉ có thể di chuyển theo 4 hướng - đi lên, đi xuống, sang trái, sang phải - vào các ô kề cạnh. Hai Robot gặp nhau nếu chúng cùng đứng trong một ô vuông. Bài toán đặt ra là tìm cách di chuyển ít nhất mà 2 Robot phải thực hiện để có thể gặp nhau.

Dữ liệu vào cho bởi tệp robot.inp:

- Dòng đầu ghi 2 số M, N cách nhau ít nhất một ký tự trống;

- M dòng tiếp theo, mỗi dòng ghi N số 0 hoặc 1 liên tiếp nhau mô tả trạng thái của các ô vuông: 1 - có vật cản, 0 - không có vật cản.

Dữ liệu ra ghi vào tệp robot.out:

- Nếu 2 Robot không thể gặp nhau thì ghi ký tự ‘#’.

- Ngược lại, ghi hai dòng, mỗi dòng là một dãy các ký tự viết liền nhau mô tả các bước đi của Robot: U - đi lên, D - đi xuống, L - sang trái, R - sang phải. Dòng đầu là các bước đi của Robot A, dòng sau là các bước đi của Robot B.

Ví dụ:

|  |  |  |  |
| --- | --- | --- | --- |
| robot.inp | robot.out | robot.inp | robot.out |
| 4 6  011000  000001  001001  010100 | DRRR  LULU | 3 4  0000  0000  0000 | # |

**Phân tích:**

Với dạng bài toán như vậy thì ta nghĩ ngay đến thuật toán Loang để tìm đường đi cho 2 Robot. Như vậy là phải “loang” từ 2 phía (loang của Robot A và loang của Robot B). Nhưng vì 2 Robot di chuyển đồng thời trong khi không cho phép ta cài đặt việc “loang” song song từ 2 phía nên ta phải thiết kế “loang” thế nào cho hợp lý.

Xin đề xuất một ý tưởng “loang” như sau: Cứ Robot A loang 1 lớp thì dừng lại để Robot B loang 1 lớp, quá trình đó được lặp đi lặp lại cho đến khi 2 Robot gặp nhau tại một ô hoặc 1 trong 2 Robot dừng “loang”. Một lớp “loang” ở đây là “loang” từ các phần tử hiện có trong hàng đợi (từ phần tử Queue[dau] đến phần tử Queue[cuoi]). Sau mỗi lớp “loang”, biến dau và biến cuoi lại được điều chỉnh để trở thành vị trí đầu và vị trí cuối của các phần tử mới trong Queue. Ta có thể mô tả cụ thể các lớp “loang” của 2 Robot với dữ liệu vào là tệp robot.inp thứ 2 ở trên:

Lớp 1 Lớp 2 Lớp 3

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Queue  Robot A | 1 | 2 | 1 | 3 | 1 | 2 | 1 | ..... |
| 1 | 1 | 2 | 1 | 1 | 2 | 3 |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Queue  Robot B | 3 | 3 | 2 | 3 | 2 | 3 | 1 | ...... |
| 4 | 3 | 4 | 2 | 3 | 4 | 4 |  |

Lớp 1 Lớp 2 Lớp 3

Q1,Q2 là 2 mảng dùng để biểu diễn cấu trúc hàng đợi để phục vụ việc “loang” của 2 Robot. Trong quá trình “loang” ta phải lưu giữ thông tin hàng, cột của ô khi “loang” đến, bởi vậy các phần tử của Q1, Q2 là các record có kiểu HC

HC = Record

h,c:Byte; *{h: lưu chỉ số hàng, c: lưu chỉ số cột}*

end;

Hai hàng đợi Q1, Q2 được khởi tạo như sau:

Procedure KT\_Queue;

Begin

dau1:=1;

cuoi1:=1;

Q1[cuoi1]:=1;

Q1[cuoi1]:=1; *{Robot A xuất phát từ ô [1,1]}*

dau2:=1;

cuoi2:=1;

Q2[cuoi2]:=M;

Q2[cuoi2]:=N; *{Robot B xuất phát từ ô [M,N]}*

End;

Ngay sau khi khởi tạo thì trong Q1 chứa ô [1,1], Q2 chứa ô [M,N]. Đó là các ô xuất phát để “loang” của 2 Robot.

Mỗi Robot từ một ô có thể “loang” theo bốn hướng: đi xuống, sang trái, đi lên, sang phải; nên để thuận tiện cho việc cài đặt ta sử dụng kỷ thuật “rào”: Mảng A[i,j] chứa thông tin các ô trong lưới ô vuông được khai báo A:Array[0..Mmax + 1,0..Nmax + 1] of Byte (chứ không phải như thông thường là [1..Mmax,1..Nmax]) và được khởi tạo FillChar(A,SizeOf(A),1) (như vậy là xung quanh lưới ô vuông được “rào” bới số 1); đồng thời sử dụng 2 mảng hằng Hi=(1,0,-1,0), Hj=(0,-1,0,1).

Khi đó việc “loang” theo lớp của Robot A được thực hiện như sau:

Procedure LoangA;

Var

k:Byte;

Begin

j:=Cuoi1;

For i:=dau1 to cuoi1 do

For k:=1 to 4 do

Begin

h:= Q1[i].h + Hi[k]; *{k=1 - đi xuống, 2 - sang trái, 3 - đi lên, 4 - sang phải}*

c:= Q1[i].c + Hj[k];

If A[h,c] = 0 then *{ô [h,c] không có vật cản và chưa “loang” đến}*

Begin

Inc(j);

Q1[j].h:= h;

Q1[j].c:= c; *{Nạp ô [h,c] vào hàng đợi Q1}*

A[h,c]:=k;*{Đánh dấu ô bằng cách gán giá trị tương ứng với hướng loang}*

B[h,c]:=True; *{Dấu hiệu cho Robot B nhận biết đã gặp Robot A}*

End;

End;

dau1:=cuoi1 + 1;

cuoi1:=j; *{Điều chỉnh lại biến dau1, cuoi1 cho các phần tử mới trong Q1}*

If dau1 > cuoi1 then ST:=True; *{ST=True là Q1 rỗng, kết thúc “loang”}*

End;

Việc “loang” theo lớp của Robot B cũng tương tự như Robot A nhưng chỉ khác ở chổ khi “loang” đến một ô [h,c] nào đó thì phải xét dấu hiệu B[h,c] xem thử đã gặp Robot A chưa:

........

If B[h,c] then *{Nếu tại ô [h,c] Robot B gặp Robot A thì}*

Begin

lk:=k; {*Lưu lại giá trị tương ứng với hướng “loang” để lấy kết quả}*

hm:=h; *{Lưu lại chỉ số hàng của ô mà 2 Robot gặp nhau để lấy kết quả}*

cm:=c; *{Lưu lại chỉ số cột của ô mà 2 Robot gặp nhau để lấy kết quả}*

TT:=True; *{Dấu hiệu dừng “loang” của 2 Robot vì đã gặp nhau}*

Exit;

End;

.........

Sở dĩ ta phải lưu lại giá trị tương ứng với hướng “loang” (lk:=k) là vì tại ô gặp nhau [h,c] Robot A đã “loang” đến trước nên đã gán giá trị của A[h,c] bằng giá trị tương ứng với hướng “loang” đến nên khi Robot B “loang” đến ô [h,c] buộc ta phải lưu lại giá trị tương ứng với hướng “loang” vào biến lk để sau này truy xuất đường đi của Robot B.

Quá trình “loang” theo từng lớp của 2 Robot được thực hiện như sau:

Procedure Loang\_lop;

Begin

TT:=False;

ST:=False;

While (ST=False) and (TT=False) do

Begin

FillChar(B,SizeOf(B),False); *{Đánh dấu theo từng lớp loang}*

Loang1;

Loang2;

End;

End;

Lệnh đánh dấu theo từng lớp “loang” tại vị trí như ở trên: FillChar(B,SizeOf(B),False) là rất quan trọng vì Robot B gặp Robot A tại ô [h,c] chỉ khi B[h,c] = True tại thời điểm lớp “loang” của Robot A cùng lớp “loang” với Robot B. Còn nếu B[h,c] = True của lớp “loang” trước nào đó của Robot A thì không thể kết luận 2 Robot gặp nhau vì khi đó 2 Robot sẽ di chuyển khập khểnh chứ không đồng thời.

Việc lấy kết quả dựa vào giá trị của biến TT: TT=True - Hai Robot gặp nhau, TT=False - Hai Robot không gặp nhau.

Trong trường hợp gặp nhau thì dựa vào việc đã lưu thông tin ô gặp nhau vào 2 biến hm ,cm (hm - chỉ số hàng, cm - chỉ số cột) ta sẽ truy xuất đường đi của 2 Robot.