MÔN: TIN HỌC

CHUYÊN ĐỀ: LÝ THUYẾT ĐỒ THỊ VÀ CÁC BÀI TẬP ỨNG DỤNG

THÁNG 7/2025

MỤC LỤC

A. PHẦN LÝ THUYẾT

Bài 1. BÀI TOÁN "THÀNH PHỐ ĐÈN LỒNG"

Thành phố Đèn Lồng đang chuẩn bị cho lễ hội ánh sáng thường niên. Để tạo ra một màn trình diễn hoành tráng, ban tổ chức muốn treo các đèn lồng trên các cột trụ trong thành phố, sau đó nối các đèn lồng bằng các sợi dây ánh sáng.

Các đèn lồng được đánh số từ 1 đến n. Các sợi dây ánh sáng nối giữa các cặp đèn lồng, mỗi sợi dây có thể đi theo cả hai chiều.

Để tiết kiệm thời gian di chuyển, ban tổ chức muốn phân công một người phụ trách đi qua toàn bộ các sợi dây ánh sáng để kiểm tra trước khi lễ hội bắt đầu. Người phụ trách phải đi qua **mỗi sợi dây đúng 1 lần**, không được đi lại qua sợi dây nào đã đi qua trước đó.

Người phụ trách được cung cấp sơ đồ của mạng lưới đèn lồng dưới dạng **ma trận kề** (n dòng, mỗi dòng n số, với số 1 nếu có dây nối giữa hai đèn lồng, số 0 nếu không có dây nối).

Yêu cầu của bài toán:

1. Viết chương trình kiểm tra xem có tồn tại cách đi qua tất cả các sợi dây ánh sáng đúng 1 lần hay không (tức là có tồn tại **đường đi** hay không).

2. Nếu có, hãy in ra đường đi đó (danh sách các đèn lồng đi qua theo thứ tự).

3. Nếu không có, hãy in ra thông báo: **"Khong co duong di "**

Input:

- Dòng đầu tiên là số nguyên n (số lượng đèn lồng).
- Tiếp theo là ma trận kề kích thước n × n.

Output:

- Nếu có đường đi, in ra: **"Co duong di"** sau đó là danh sách các đèn lồng theo thứ tự đi qua.
- Nếu không có, in ra: **"Khong co duong di"**

Ví dụ:

input	output
4 0 1 1 0 1 0 1 1 1 1 0 1 0 1 1 0	Co duong di 1 2 3 4 2 1

3 0 1 0 1 0 1 0 1 0	Khong co duong di

Hướng dẫn: Bài toán yêu cầu **đi qua tất cả các sợi dây đúng 1 lần** → chính là bài toán **Đường đi Euler** trong đồ thị **vô hướng**.

- Mô hình hóa bài toán

+ Mỗi **đèn lồng** là 1 **đỉnh** của đồ thị.

+ Mỗi **sợi dây ánh sáng** là 1 **cạnh** của đồ thị.

+ Sơ đồ được đưa ra dưới dạng **ma trận kề**.

- Lý thuyết cần áp dụng: Một đồ thị vô hướng có **Đường đi Euler (Eulerian Path)** khi và chỉ khi:

+ Đồ thị **liên thông** (nếu bỏ các đỉnh bậc 0 thì phần còn lại liên thông).

+ Có **0 hoặc 2 đỉnh bậc lẻ**:

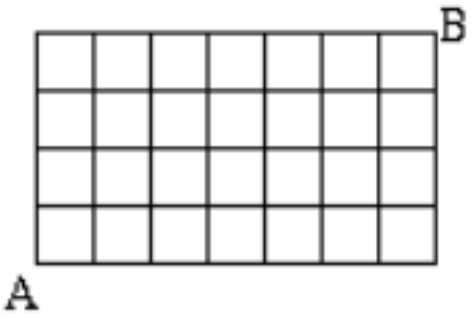
Nếu **0 đỉnh bậc lẻ** → có **Chu trình Euler** (đi khép kín).

Nếu **2 đỉnh bậc lẻ** → có **Đường đi Euler** (không khép kín).

Nếu có >2 đỉnh bậc lẻ → **không có** đường đi Euler.

Bài 2. LƯỚI Ô VUÔNG

Cho một lưới ô vuông kích thước m x n (m, n nhập từ bàn phím). Lập trình tính số các đường đi ngắn nhất từ A đến B và chỉ đi trên các cạnh của lưới. Kết quả thể hiện trên màn hình.



+ **Subtask 1: Kích thước nhỏ (m, n ≤ 10)**

- Với kích thước nhỏ như thế này, có thể sử dụng cách tính trực tiếp bằng công thức tổ hợp mà không lo ngại về hiệu suất.

- Đây là mức độ dễ dàng cho việc kiểm tra và xác nhận tính đúng đắn của thuật toán.

+ Subtask 2: Kích thước trung bình (10 < m, n ≤ 20)

- Đối với kích thước trung bình, vẫn có thể sử dụng công thức tổ hợp, nhưng cần chú ý hơn đến việc tối ưu hóa tính toán giai thừa để tránh tràn số.

- Có thể sử dụng phương pháp tính toán động (dynamic programming) để xây dựng một bảng lưu trữ số đường đi ngắn nhất từ điểm (0,0) đến từng điểm (i,j).

+ Subtask 3: Kích thước lớn (20 < m, n ≤ 50)

- Với kích thước lớn hơn, cần sử dụng dynamic programming để tối ưu hóa việc tính toán và giảm thiểu thời gian xử lý.

- Sử dụng một bảng 2D để lưu trữ số đường đi đến mỗi ô và tính toán dần dần từ góc trên bên trái đến góc dưới bên phải.

+ Subtask 4: Kích thước rất lớn (m, n > 50)

- Khi kích thước trở nên rất lớn, cần phải tối ưu hóa bộ nhớ và thời gian xử lý.

- Sử dụng công thức tổ hợp với kỹ thuật tối ưu hóa tính giai thừa để tránh tràn số và giảm thiểu bộ nhớ cần thiết.

*** Hướng dẫn**

1. Khởi tạo bảng:

- Tạo một bảng 2D 'dp' với kích thước (m+1) x (n+1) và khởi tạo tất cả các phần tử bằng 0.

2. Khởi tạo giá trị ban đầu:

- dp[0][0] = 1 vì chỉ có một cách để đứng yên tại điểm bắt đầu.

3. Tính toán số đường đi:

- Duyệt qua từng ô (i, j) trong bảng.

- Nếu không phải điểm bắt đầu, tính số đường đi đến ô (i, j) bằng cách cộng số đường đi từ ô phía trên (i-1, j) và ô bên trái (i, j-1).

4. Kết quả:

- Kết quả cuối cùng là giá trị tại dp[m][n], số đường đi ngắn nhất từ A đến B.

Bài 3. NHÀ THÁM HIỂM

Chi là một nhà thám hiểm nổi tiếng và đã được mời tham gia một cuộc thi khám phá các vùng đất mới. Cuộc thi này diễn ra tại một vương quốc xa xôi có tên là "Vương quốc Thành phố". Vương quốc này gồm **n** thành phố, được đánh số từ 1 đến n. Giữa một số thành phố có các con đường cổ xưa nối liền chúng, trong khi một số cặp thành phố khác không có đường đi trực tiếp.

Nhiệm vụ của Chi:

Chi được giao nhiệm vụ tìm con đường ngắn nhất để đi từ thành phố đầu tiên (thành phố 1) đến thành phố cuối cùng (thành phố n). Để làm được điều này, **Chi** cần xem xét một ma trận đối xứng **A** có kích thước **n x n**. Phần tử **aij** trong ma trận

này cho biết độ dài của con đường trực tiếp từ thành phố **i** đến thành phố **j**. Nếu **aij** là 0, điều đó có nghĩa là không có đường đi trực tiếp giữa hai thành phố đó.

Hãy viết chương trình để xác định độ dài của con đường ngắn nhất từ thành phố 1 đến thành phố n. Nếu không có cách nào để đi từ thành phố 1 đến thành phố n, chị cần thông báo rằng việc khám phá là không thể.

Yêu cầu:

1. Nhập vào số nguyên **n** ($2 \leq n \leq 1000$) - số lượng thành phố trong vương quốc.

2. Nhập vào ma trận đối xứng **A** với **n x n** phần tử, mỗi phần tử là một số nguyên không âm.

3. Tính và in ra độ dài của con đường ngắn nhất từ thành phố 1 đến thành phố n. Nếu không thể đi từ thành phố 1 đến thành phố n, hãy in ra "Khám phá không thể thực hiện".

Ví dụ:

input	output
5 0 10 0 0 5 10 0 1 0 0 0 1 0 2 0 0 0 2 0 3 5 0 0 3 0	8
4 0 2 0 0 2 0 2 0 0 2 0 2 0 0 2 0	6

Input:

- Số thành phố **n** là 5.
- Ma trận **A** miêu tả độ dài đường đi giữa các thành phố. Ví dụ, từ thành phố 1 đến thành phố 2 có độ dài là 10, từ thành phố 1 đến thành phố 5 có độ dài là 5, v.v.

Output:

- Độ dài của con đường ngắn nhất từ thành phố 1 đến thành phố 5 là 8. Cụ thể là: đi từ thành phố 1 đến thành phố 5 qua thành phố 3 (1 -> 5 = 5, 5 -> 3 = 3, tổng là 8).

+ Subtask 1: n = 2

Trường hợp đơn giản nhất, chỉ có 2 thành phố. Ta chỉ cần kiểm tra xem có đường đi trực tiếp giữa hai thành phố không.

+ Subtask 2: $3 \leq n \leq 5$

Với số lượng thành phố nhỏ, ta có thể kiểm tra tất cả các đường đi trực tiếp và sử dụng một thuật toán tìm đường đi ngắn nhất đơn giản như Dijkstra.

+ Subtask 3: $6 \leq n \leq 10$

Tăng số lượng thành phố, tiếp tục sử dụng thuật toán Dijkstra để tìm đường đi ngắn nhất nhưng cần tối ưu hóa một chút để đảm bảo hiệu suất.

+ Subtask 4: $11 \leq n \leq 100$

Với số lượng thành phố lớn hơn, cần chú ý đến tối ưu hóa việc sử dụng bộ nhớ và thời gian thực thi. Thuật toán Dijkstra với hàng đợi ưu tiên sẽ hữu dụng trong trường hợp này.

+ Subtask 5: $101 \leq n \leq 1000$

Đây là trường hợp tổng quát nhất với số lượng thành phố lớn. Cần đảm bảo thuật toán Dijkstra được triển khai hiệu quả và có thể xử lý dữ liệu lớn. Kiểm tra xem có cần tối ưu hóa thêm không, ví dụ bằng cách sử dụng thuật toán A* nếu có thể áp dụng.

*** Hướng dẫn:** Để giải bài toán tìm đường đi ngắn nhất giữa hai thành phố trong đồ thị vô hướng với ma trận trọng số, ta có thể sử dụng thuật toán Dijkstra. Thuật toán này sẽ giúp tìm đường đi ngắn nhất từ thành phố 1 đến thành phố n. Dưới đây là một cách triển khai thuật toán Dijkstra bằng ngôn ngữ C++:

```
```

Lưu ý rằng trong ma trận **graph**, giá trị **graph[i][j]** là 0 nếu không có đường đi trực tiếp giữa thành phố **i** và **j**. Đoạn mã này sẽ giúp ta tìm ra độ dài của đường đi ngắn nhất từ thành phố 1 đến thành phố n. Nếu không có đường đi nào tồn tại, chương trình sẽ thông báo điều đó.

Bài 4. THIẾT KẾ MẠNG LƯỚI CÁP QUANG

Một công ty viễn thông được giao nhiệm vụ lắp đặt mạng lưới cáp quang để kết nối n văn phòng trong một thành phố.

Yêu cầu kỹ thuật đưa ra như sau:

- Từ bất kỳ văn phòng nào, dữ liệu phải có thể truyền đến bất kỳ văn phòng nào khác, thông qua không quá một văn phòng trung gian (nghĩa là đi qua cùng lắm một lần chuyển tiếp).

- Chi phí lắp đặt tỷ lệ với số đường cáp quang cần dùng. Công ty muốn sử dụng ít đường cáp quang nhất có thể.

Nhiệm vụ của bạn là thiết kế các đường cáp quang cần lắp đặt và in ra danh sách các cặp văn phòng cần kết nối trực tiếp.

Input: Một số nguyên duy nhất n ($2 \leq n \leq 10^5$) — số lượng văn phòng.

Output

+ Dòng đầu tiên in ra số đường cáp quang cần lắp đặt.

+ Mỗi dòng tiếp theo gồm 2 số nguyên u v ($1 \le u, v \le n, u \ne v$), biểu thị rằng có một đường cáp quang nối trực tiếp văn phòng u với văn phòng v.

Ví dụ:

input	output
5	4 1 2 1 3 1 4 1 5

Giải thích

- Chọn văn phòng 1 làm trung tâm.
- Nối văn phòng 1 với tất cả các văn phòng còn lại.
- Từ bất kỳ văn phòng u đến v, có thể đi trực tiếp hoặc qua văn phòng 1, với nhiều nhất 1 lần chuyển tiếp.
- Cách này sử dụng đúng n - 1 = 4 đường cáp quang — là phương án tối ưu nhất.

Gợi ý

- Không cần thuật toán phức tạp.
- Chọn 1 văn phòng làm trung tâm, nối các văn phòng khác vào đó.
- Độ phức tạp thuật toán: **O(n)** (cực nhanh, phù hợp với $n \le 1e5$).

Bài 5. BÀI TOÁN TÌM SỐ BƯỚC ÍT NHẤT ĐỂ BIẾN SỐ A THÀNH SỐ B

Bạn được cho hai số nguyên dương **A** và **B**.

Bạn có thể thực hiện hai phép biến đổi sau:

1. Nhân số hiện tại với 2.
2. Cộng thêm 1.

Bạn cần tìm số lượng phép biến đổi ít nhất để biến **A** thành **B**.

Ràng buộc: $1 \le A \le B \le 10^6$

Ý tưởng giải bằng BFS:

- Mỗi số nguyên có thể coi là một **đỉnh** trong đồ thị.
- Từ một số **x**, ta có thể sinh ra các số $x \times 2$ và $x+1$, tức là có 2 **cạnh** đi từ mỗi đỉnh.
- Bài toán trở thành bài toán **tìm đường đi ngắn nhất từ đỉnh A đến đỉnh B** trong đồ thị này.
- BFS rất phù hợp để tìm đường đi ngắn nhất trong đồ thị **vô hướng hoặc có trọng số bằng nhau** như ở đây (mỗi phép biến đổi coi như có trọng số 1).

- Ta sẽ sử dụng một hàng đợi (queue) để duyệt theo mức và một mảng/Set để đánh dấu các số đã duyệt.

Ví dụ minh họa:

input	output
A = 2 B = 9	3 (2 → 4 → 8 → 9 (3 bước))
A = 1 B = 10	4 (1 → 2 → 4 → 8 → 9 → 10 (4 bước))

*** Tóm tắt kiến thức lý thuyết học được từ bài toán này:**

+ BFS giúp ta tìm đường đi ngắn nhất trong **đồ thị phi trọng số** hoặc các bài toán có thể mô hình hóa thành đồ thị.

+ Kỹ thuật "biến trạng thái → trạng thái mới" có thể dùng BFS rất hiệu quả.

+ Bài toán không cần xây đồ thị tường minh mà sinh đỉnh khi duyệt → tiết kiệm bộ nhớ.

Bài 6. KHÁM PHÁ MÊ CUNG CỔ ĐẠI

Một nhóm nhà khảo cổ học phát hiện ra một **mê cung cổ đại** dưới lòng đất. Mê cung gồm nhiều **phòng thông với nhau qua các hành lang**. Mỗi phòng được đánh số từ 1 đến N.

Các nhà khảo cổ muốn kiểm tra xem từ phòng số S, họ có thể **đi đến được tất cả các phòng khác** trong mê cung hay không (vì một số hành lang có thể đã bị chặn hoặc sụp đổ).

Bạn được giao nhiệm vụ viết một chương trình sử dụng thuật toán **DFS** để giúp nhóm khảo cổ trả lời câu hỏi đó.

Input:

+ Một số nguyên N — số phòng trong mê cung ($1 \le N \le 10^5$).

+ Một số nguyên M — số hành lang ($0 \le M \le 10^5$).

+ M dòng tiếp theo, mỗi dòng chứa hai số nguyên u và v ($1 \le u, v \le N$), cho biết có hành lang nối giữa phòng u và phòng v (hành lang hai chiều).

+ Một số nguyên S — phòng bắt đầu.

Output:

+ In ra **YES** nếu từ phòng S có thể đi đến tất cả các phòng khác.

+ Ngược lại, in ra **NO**.

Hướng dẫn:

+ Sử dụng **DFS** để duyệt qua các phòng có thể tiếp cận từ S.

+ Nếu sau khi duyệt DFS, số phòng đã thăm == N → **YES**.

+ Ngược lại → **NO**.

Ví dụ minh họa:

Input	**Output**
5 4 1 2 1 3 3 4 3 5 1	**YES**
4 2 1 2 3 4 1	**NO**

*** Tóm tắt kiến thức lý thuyết học được từ bài toán này:**

DFS là thuật toán duyệt đồ thị theo chiều sâu; duyệt đồ thị từ một đỉnh gốc để khám phá tất cả các đỉnh có thể đến được.

Phân tích bài toán dưới dạng đồ thị.

Một đồ thị vô hướng được gọi là liên thông nếu từ một đỉnh bất kỳ có thể đi đến mọi đỉnh còn lại.

DFS có độ phức tạp **O(N + M)** với N là số đỉnh, M là số cạnh — rất hiệu quả với đồ thị lớn.

B. PHẦN BÀI TẬP ÁP DỤNG

Bài 1. CHI PHÍ PHÁT SINH

Đất nước Alpha có n thành phố được đánh số $1,2,3,\ldots,n$. Giữa các thành phố này có m tuyến đường hai chiều đảm bảo kết nối giữa n thành phố. Mỗi tuyến đường thứ i $(1 \le i \le n)$ được mô tả bởi cặp số (u_i, v_i) kết nối trực tiếp hai thành phố u_i và v_i $(u_i \ne v_i)$ và được gán hai số nguyên dương c_i, d_i, trong đó c_i là độ đẹp và d_i là chi phí phát sinh khi đi qua tuyến đường.

Giả sử s và t là hai thành phố của đất nước Anpha. Một công ty ABC đang cần vận chuyển một chuyến hàng có trọng lượng W từ thành phố s tới thành phố t. Ta gọi một đường đi từ s đến t là một dãy $z_0, z_1, z_2, ..., z_{k-1}, z_k$, trong đó $z_0 = s,\ z_k = t,\ (z_{i-1},\ z_i)$ là tuyến đường với $i = 1,2,...,k$. Chi phí cần vận chuyển một chuyến hàng có trọng lượng W từ thành phố s đến thành phố t theo đường đi nói trên là:

$$d(z_0, z_1) + d(z_1, z_2) + ... + d(z_{k-1}, z_k) + \frac{W}{C_{\min}}$$

Trong đó $d(z_{i-1},\ z_i)$ là chi phí phát sinh của tuyến đường $(z_{i-1},\ z_i)$, C_{min} là độ đẹp nhỏ nhất trong các tuyến đường trên đường đi mà xe hàng đi qua.

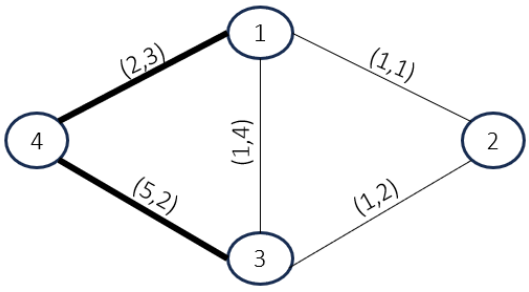
Yêu cầu: Cho trước hai thành phố s và t. Hãy tìm đường đi để vận chuyển một chuyến hàng có trọng lượng W với chi phí nhỏ nhất.

Dữ liệu vào: Từ tệp văn bản **CHIPHI.INP** có cấu trúc như sau:

- Dòng đầu tiên chứa hai số nguyên $n,\ m$ $(2 \le n \le 150,\ 1 \le m \le 5000)$, mỗi số cách nhau một khoảng trắng;
- Dòng thứ hai chứa ba số nguyên dương $s,\ t,\ W$ $(1 \le s,t \le n;\ 1 \le W \le 10^4)$, mỗi số cách nhau một khoảng trắng;
- Dòng thứ i trong số m dòng tiếp theo chứa bốn số nguyên dương u_i, v_i, c_i, d_i mô tả thông tin về con đường thứ i $(1 \le u_i,\ v_i \le n;\ 1 \le c_i,\ d_i \le 10000,\ i = 1, 2, \ldots, m)$, mỗi số cách nhau một khoảng trắng.

Kết quả: Ghi ra tệp văn bản **CHIPHI.OUT** gồm một số duy nhất là chi phí nhỏ nhất để vận chuyển chuyến hàng (kết quả làm tròn đến hai chữ số sau dấu thập phân).

Ví dụ:

CHIPHI.INP	CHIPHI.OUT	HÌNH MINH HỌA
4 5 1 3 8 1 2 1 1 1 3 1 4 1 4 2 3 3 4 5 2 3 2 1 2	9.00	

Ràng buộc:

+ ***Subtask1.*** *Có 40% số test ứng với 40% số điểm của bài thỏa mãn* $c_i = d_i = 1\ (\forall i = 1,2,3,...,m)$;

+ ***Subtask2.*** *Có 30% số test ứng với 30% số điểm của bài thỏa mãn* $c_i = 1\ (\forall i = 1,2,3,...,m)$;

+ ***Subtask3.*** *Có 20% số test ứng với 20% số điểm của bài thỏa mãn* $m = n - 1$;

+ ***Subtask4.*** *Có 10% số test ứng với 10% số điểm của bài không có ràng buộc bổ sung*.

Hướng dẫn thuật toán

+ **Subtask 1:** Nhận thấy chi phí và độ đẹp các con đường như nhau nên chỉ cần sử dụng BFS tìm đường đi từ s đến t, cộng thêm w ta tìm được chi phí phát sinh.

+ **Subtask 2:** Do độ đẹp các con đường đều bằng 1 nên ta sử dụng đường đi ngắn nhất tìm được đi từ s đến t, cộng với w ta có chi phí phát sinh.

+ **Subtask 3:** Đồ thị dạng cây nên chỉ có duy nhất một đường đi. Ta có thể sử dụng BFS/DFS tìm đường đi

+ **Subtask 4:** Duyệt tất cả các giá trị $cmin$. Với mỗi $cmin$, tìm đường đi ngắn nhất trên tập các cạnh có độ đẹp $\geq cmin$. Ta có được chi phí phát sinh với $cmin$ đó. Cập nhật chi phí phát sinh tối ưu.

BÀI 2. DU LỊCH VIỆT NAM

Việt Nam được biết đến với thương hiệu là một đất nước hòa bình và là một trong những quốc gia có nhiều địa điểm du lịch nổi tiếng. Theo thống kê, hiện nay trên toàn bộ lãnh thổ Việt Nam có n địa điểm du lịch được đánh số từ 1 đến n. Giữa n điểm du lịch là một mạng lưới gồm m tuyến đường hai chiều, trong đó tuyến đường thứ i kết nối hai điểm du lịch x_i, y_i và có khoảng cách là d_i, đảm bảo không có hai điểm du lịch nào được nối bởi quá một tuyến đường.

Điểm du lịch 1 và điểm du lịch n là hai điểm du lịch trung tâm và hệ thống tuyến đường đảm bảo luôn có ít nhất một cách đi từ điểm du lịch 1 đến điểm du lịch n. Tuy nhiên, cả hai điểm du lịch này đều có dấu hiệu quá tải về số lượng khách du

lịch. Vì vậy, chính phủ quyết định chọn thêm một điểm du lịch nữa để nâng cấp hạ tầng thành một điểm du lịch trung tâm thứ ba. Điểm du lịch này sẽ tạm ngưng mọi hoạt động, cũng như mọi luồng lưu thông vào và ra để nâng cấp nhưng phải đảm bảo đường đi ngắn nhất từ điểm du lịch 1 đến điểm du lịch n không bị thay đổi.

Vị trí và đường nối giữa n điểm du lịch được mô tả như một đồ thị gồm n đỉnh và m cạnh. Hãy giúp chính phủ đếm số lượng điểm du lịch có thể chọn làm điểm du lịch trung tâm thứ ba sao cho điểm du lịch được chọn thỏa mãn các điều kiện trên.

Dữ liệu vào: Từ tệp văn bản **VNTRIP.INP** có cấu trúc như sau:

•Dòng đầu tiên ghi 2 số nguyên dương n và m là số điểm du lịch và số tuyến đường $(2 < n \le 30000, 1 \le m \le 10^5)$ mỗi số cách nhau một khoảng trắng;

•Dòng thứ i trong số m dòng tiếp theo ghi 3 số nguyên dương x_i, y_i và d_i (mỗi số cách nhau một khoảng trắng) với ý nghĩa tuyến đường thứ i có độ dài d_i $\left(1 \le d_i \le 10^3\right)$ và nối giữa 2 điểm du lịch x_i, y_i $\left(1 \le x_i, y_i \le n\right)$.

Kết quả ra: Ghi ra tệp văn bản **VNTRIP.OUT** gồm:

•Dòng đầu tiên ghi số tự nhiên S là số lượng các điểm du lịch có thể chọn làm điểm du lịch trung tâm thứ ba.

•S dòng tiếp theo, mỗi dòng ghi một số nguyên dương là số thứ tự của điểm du lịch được chọn (các điểm du lịch in ra theo thứ tự tăng dần).

Ví dụ:

VNTOUR.INP	VNTOUR.OUT
5 6 1 3 1 1 4 2 2 3 10 2 5 20 3 5 2 4 5 3	2 2 4

Ràng buộc:

+ Có 20% số test ứng với 20% số điểm của bài khi các tuyến đường chỉ nối đỉnh i với đỉnh i+1;

+ Có 80% số test ứng với 80% số điểm của bài không có thêm ràng buộc gì.

Hướng dẫn thuật toán

Subtask 1:

+ Bước 1: Sử dụng thuật toán Dijisktra tạo mảng D[i][j] với ý nghĩa là đường đi ngắn nhất từ i→j.

+ Bước 2: Tạo k hoán vị của k điểm du lịch cần thăm, với mỗi hoán vị tính chi phí nhỏ nhất của mỗi hành trình.

Subtask 2::

+ Sử dụng QHĐ bit: `Tinh(i,t)` (đang ở địa điểm i và t là dãy bit thể hiện trạng thái đã thăm hoặc chưa thăm của k địa điểm) trả về chi phí nhỏ nhất từ địa điểm i về trạng thái đích.

Bài 3. GRAPH

Trên bản đồ thành phố HP có n địa điểm chiến lược (đánh số từ 1 đến n) và m con đường hai chiều (đánh số từ 1 đến m), con đường i $(1 \le i \le m)$ nối giữa hai địa điểm x_i và y_i $(1 \le x_i, y_i \le n)$.

Có q truy vấn, truy vấn thứ i $(1 \le i \le q)$ cho hai số l và r, bạn hãy cho biết mọi cặp địa điểm chiến lược có thể di chuyển được với nhau không nếu chỉ dùng các con đường $l, l+1, \dots, r$.

Dữ liệu: Vào từ file **GRAPH.INP**:

+ Dòng đầu tiên gồm hai số nguyên dương n, m $(2 \le n \le 100; 1 \le m \le 100.000)$;

+ m dòng sau, dòng thứ i $(1 \le i \le m)$ gồm hai số nguyên dương x_i, y_i $(1 \le x_i, y_i \le n)$ mô tả một con đường nối giữa hai địa điểm x_i và y_i;

+ Dòng tiếp theo chứa số nguyên dương q $(1 \le q \le 100.000)$;

+ q dòng sau, dòng thứ i $(1 \le i \le q)$ gồm hai số nguyên l và r mô tả truy vấn i $(1 \le l \le r \le m)$;

Kết quả: Ghi ra file **GRAPH.OUT** q dòng, dòng thứ i in ra "**Yes**" nếu mọi cặp địa điểm chiến lược có thể đến được nhau và "**No**" nếu ngược lại.

Ràng buộc:

+ 20% số điểm thoả mãn: $m, q \le 100$;

+ 20% số điểm tiếp theo thoả mãn: $l = 1, \forall i \in [1; q]$;

+ 20% số điểm tiếp theo thoả mãn: $l \le 50, \forall i \in [1; q]$;

+ 40% số điểm còn lại không có ràng buộc gì thêm.

Ví dụ:

GRAPH.INP	GRAPH.OUT	Giải thích
4 6 **1 2** **2 3** **3 4**	**Yes** **No**	- Ở truy vấn đầu tiên, các con đường có thể sử dụng là (1, 2), (2, 3), (3, 4). Tất cả 4 địa điểm chiến lược đều có thể đến được với nhau;

4 1 **1 3** **2 3** **2** **1 3** **3 5**		- Ở truy vấn thứ hai, các con đường có thể sử dụng là (3, 4), (4, 1), (1, 3). Các địa điểm 1, 3, 4 có thể đến được với nhau, tuy nhiên địa điểm 2 lại bị cô lập.

Hướng dẫn thuật toán

+ Subtask 1 : mỗi truy vấn dựng đồ thị với các cạnh từ l đến r, sau đó DFS để kiểm tra xem có thể thăm tất cả địa điểm không

Độ phức tạp: $(\boldsymbol{n} + \boldsymbol{m}) * \boldsymbol{q}$

+ Subtask 2:

- ✓ sort lại các truy vấn theo r tăng dần;
- ✓ khi xét đến truy vấn $(1, r)$, thêm các cạnh từ 1 đến r bằng DSU
- ✓ nếu có n - 1 cạnh có thể được thêm vào DSU thì khi đó đồ thị sẽ liên thông

Độ phức tạp: $\boldsymbol{q} * \boldsymbol{log2}(\boldsymbol{q}) + \boldsymbol{n} * \boldsymbol{log2}(\boldsymbol{n})$

+ Subtask 3 : gọi fi là chỉ số nhỏ nhất mà nếu dùng các cạnh từ $i \rightarrow fi$ thì đồ thị sẽ liên thông

Nhận xét : $fi \leq fi{+}1\ \forall 1 \leq i < n$

- ✓ Ta tính mọi fi với mọi i từ 1 đến 50:
- ✓ tương tự như sub2 thêm dần các cạnh đến khi đồ thị liên thông, fi sẽ là chỉ số của cạnh thứ n - 1 được thêm vào DSU.
- ✓ khi đến truy vấn (l, r) , ta kiểm tra $r \geq fi$, nếu có thì ra Yes, nếu không thì in ra No.

Độ phức tạp: $\mathbf{50} * \boldsymbol{m} * \boldsymbol{log2}(\boldsymbol{n})$

+ Subtask4 : ở Subtask 3, mỗi khi cần tính f_l, ta duyệt các cạnh từ trái qua phải và thêm vào DSU, giả sử các cạnh đó là $p_{l_1}, p_{l_2}, \ldots, p_{l_{n-1}}$

- ✓ nhận xét $p_{l_i} \leq p_{l_{i+1}}\ \forall 1 \leq i < n$
- ✓ vì vậy, mỗi khi tính f_l ,ta chỉ cần duyệt từ bên phải của các cạnh đã thêm vào cây khung khi tính f_{l-1}

Độ phức tạp: $\boldsymbol{n} * \boldsymbol{m} * \boldsymbol{log2}(\boldsymbol{n})$

Bài 4. ĐIỆN TOÁN ĐÁM MÂY

Hiện nay, điện toán đám mây đang trở nên phổ biến và phát triển vượt bậc, trở thành xu thế công nghệ hàng đầu mà các tổ chức, doanh nghiệp hướng tới. Một trong những nhà cung cấp tiên phong trong dịch vụ điện toán đám mây là Amazon

Web Service (AWS). AWS đang dẫn đầu về doanh thu và sự phổ biến với hơn 50% doanh thu về dịch vụ này trên.

Để có thể tạo ra những service phục vụ hàng triệu người dùng trên toàn cầu, AWS xây dựng được một hạ tầng trên toàn thế giới. Theo thống kê, hầu hết các nước phát triển trên toàn cầu đều có trung tâm dữ liệu do AWS xây dựng, giúp cho người sử dụng có thể truy cập dễ dàng hơn. Hơn nữa, trên mỗi quốc gia không chỉ có một mà còn có nhiều trung tâm dữ liệu khác để phòng trường hợp một trung tâm dữ liệu bị hỏng không thể sử dụng được.

Hiện tại, AWS có N trung tâm dữ liệu, mỗi trung tâm đặt ở một địa điểm khác nhau. Đặc biệt, có một vài trung tâm dữ liệu được đặt rất gần nhau tạo thành một cụm máy. Cụ thể, trong N trung tâm dữ liệu này có G cụm máy, mỗi cụm gồm một số trung tâm dữ liệu, các trung tâm dữ liệu trong cùng một cụm kết nối trực tiếp với nhau và có cùng độ trễ là cost. Hơn nữa, một trung tâm dữ liệu có thể thuộc nhiều cụm máy.

Để các trung tâm dữ liệu có thể kết nối đến nhau, AWS thêm M đường truyền, mỗi đường nối trực tiếp 2 trung tâm dữ liệu u và v có độ trễ là w.

Trong N trung tâm dữ liệu, có đúng S trung tâm được gọi là server chứa các dịch vụ cho người dùng. Các trung tâm còn lại muốn được cung cấp các dịch vụ đó phải kết nối với một trong số S server trên. Trung tâm a có thể kết nối với trung tâm b nếu tồn tại đường đi từ a đến b trên mạng lưới đó, với độ trễ là tổng độ trễ các đường truyền phải đi qua.

AWS cần trả lời Q câu hỏi có dạng sau: trung tâm dữ liệu X muốn được cung cấp các dịch vụ thì độ trễ nhỏ nhất khi kết nối vào một trong các server là bao nhiêu?

Yêu cầu: Em hãy viết chương trình trả lời Q câu hỏi trên.

Dữ liệu vào cho trong tệp AWS.INP

- Dòng đầu tiên gồm 4 số nguyên N, S, G, M ($1 \leq N, M \leq 10^5$; $S, G \leq N$) lần lượt là số trung tâm dữ liệu, số server, số cụm máy và số đường dây mạng trên mạng lưới đó.

- Dòng thứ hai gồm S số là các trung tâm máy tính server trong số N trung tâm.

- G dòng tiếp theo, mỗi dòng có cấu trúc như sau:

+ Số đầu tiên là num ($1 \leq num \leq N$), biểu thị cho số lượng các trung tâm máy tính trong cụm máy đó.

+ num số tiếp theo là chỉ số của các trung tâm dữ liệu trong cụm đó. Không có 2 trung tâm dữ liệu nào trùng nhau.

+ Cuối cùng là $cost$ ($1 \leq cost \leq 10^9$), độ trễ để kết nối giữa 2 trung tâm bất kỳ trong cụm máy đó.

Tổng số trung tâm dữ liệu trong G cụm không quá 10^5.

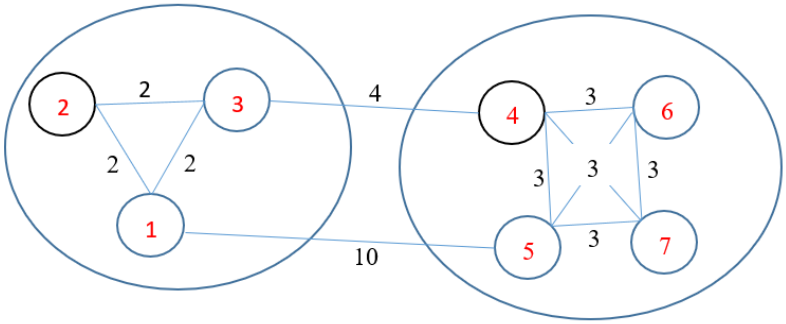
- M dòng tiếp theo, mỗi dòng gồm 3 số u, v, w là một đường dây mạng 2 chiều nối giữa trung tâm u và v, với độ trễ w ($1 \leq w \leq 10^9$).

- Dòng tiếp theo chứa 1 số nguyên Q ($1 \leq Q \leq 10^5$), số lượng truy vấn.

- Q dòng cuối, mỗi dòng gồm 1 số là trung tâm dữ liệu X ($1 \leq X \leq 10^5$)

Kết quả đưa ra tệp AWS.OUT Q dòng, mỗi dòng một số nguyên là độ trễ nhỏ nhất cho Q câu hỏi tương ứng trong tệp dữ liệu vào.

Ví dụ:

AWS.INP	AWS.OUT	Giải thích ví dụ
7 1 2 2 1 3 1 2 3 2 4 4 5 6 7 3 3 4 4 1 5 10 1 5	9	 - Tổng độ trễ nhỏ nhất từ 5 đến 1 là 9

Ràng buộc:

+ Có 10% số test ứng với 10% số điểm có $S = 1, G = 0$.

+ 20% số test ứng với 20% số điểm có $S = 1$, tổng số máy trong G cụm không quá 10^5.

+ 10% số test ứng với 10% số điểm có $G = 0$.

+ 20% số test ứng với 20% số điểm có tổng số máy trong G cụm không quá 10^3.

+ 40% số test ứng với 40% số điểm không có ràng buộc gì thêm.

Hướng dẫn thuật toán

Bài toán có 2 nhiệm vụ cần giải quyết là: Xây đồ thị và Tìm đường đi ngắn nhất đến 1 trong các đỉnh đích.

1. Xây đồ thị

- Với mỗi đường dây mạng (u, v) với độ trễ w, xây dựng 1 cạnh (u, v) với trọng số w trên đồ thị.

- Với cụm trung tâm $c_1, c_2, c_3, \ldots$ với độ trễ là cost, ta xây các cạnh vô hướng $(c_1, c_2), (c_1, c_3), \ldots, (c_2, c_3), \ldots$ với cùng trọng số là cost.

=> Số cạnh có thể lên đến N^2.

- Cải tiến: Với mỗi cụm trung tâm đó, tạo 1 đỉnh giả có nhiệm vụ liên kết cả cụm trung tâm đó. VD cho trường hợp cụm trên:

+ Tạo đỉnh giả g_1.

+ Tạo các cạnh 1 chiều $(c_1, g_1), (c_2, g_1), (c_3, g_1)$, ... với cùng trọng số cost.

+ Tạo các cạnh 1 chiều $(g_1, c_1), (g_1, c_2), (g_1, c_3)$, ... với cùng trọng số 0.

=> Số cạnh giảm xuống cỡ M+G.

2. Tìm đường đi ngắn nhất

- Nếu số server chỉ có duy nhất 1 đỉnh s, chỉ cần tìm đường đi ngắn nhất từ đỉnh s đến các đỉnh còn lại. Sử dụng thuật toán Dijkstra để giải quyết.

- Nếu các server là s_1, s_2, s_3, ..., thay vì chỉ đẩy đỉnh s vào hàng đợi ưu tiên ban đầu, ta đẩy tất cả các đỉnh đó vào hàng đợi, sau đó sử dụng Dijkstra để tìm đường đi ngắn nhất.

- Độ phức tạp $O((N + G + M)\log(N + G)$.

+ Subtask 1:

- Tìm đường đi ngắn nhất từ đỉnh 1 tới các đỉnh
- Xử lý từng yêu cầu.

+ Subtask 2:

- Cải tiến xây dựng đồ thị ở trên và tìm đường đi ngắn nhất từ đỉnh 1 tới các đỉnh.
- Xử lý từng yêu cầu.

+ Subtask 3:

- Khởi tạo hàng đợi ưu tiên là các đỉnh server, dùng thuật toán Dijkstra
- Xử lý từng truy vấn.

+ Subtask 4:

- Xây dưng đồ thị đầy đủ cho các cụm G.
- Khởi tạo hàng đợi ưu tiên là các đỉnh server, dùng thuật toán Dijkstra
- Xử lý từng truy vấn.

+ Subtask 5:

- Xây dựng đồ thị như cải tiến trên
- Khởi tạo hàng đợi ưu tiên là các đỉnh server, dùng thuật toán Dijkstra
- Xử lý từng truy vấn.

Bài 5: GẶP GỠ

Đất nước Z có n thành phố, các thành phố được đánh số từ 1 đến n. Có đúng $n - 1$ con đường hai chiều nối giữa các thành phố thỏa mãn điều kiện: có thể đi từ thành phố bất kì đến tất cả các thành phố còn lại theo đường trực tiếp hoặc gián tiếp qua các thành phố khác. Đất nước Z thường có các sự kiện văn hóa lớn, mỗi lần sự kiện sẽ được tổ chức tại một thành phố, điều này ảnh hưởng tới chi phí di chuyển trên các con đường. Cụ thể, nếu thành phố u là thành phố tổ chức sự kiện văn hóa,

khi đó các con đường hướng tới thành phố u sẽ có chi phí là a còn các con đường đi xa thành phố u sẽ có chi phí là b. Con đường từ thành phố i tới thành phố j được gọi là hướng tới u nếu đường đi ngắn nhất từ i tới u dài hơn đường đi ngắn nhất từ j tới u, ngược lại thì con đường từ i tới j được gọi là đi xa thành phố u. Khi sự kiện văn hóa diễn ra, một người di chuyển qua s con đường sẽ bị mất chi phí bằng tổng của từng lần di chuyển, lần di chuyển thứ k $(1 \leq k \leq s)$ sẽ mất chi phí $k \times cost_k$, trong đó $cost_k$ bằng a hoặc b tùy thuộc lần di chuyển thứ k đi qua con đường hướng tới thành phố tổ chức sự kiện hay đi xa thành phố tổ chức sự kiện.

Một câu hỏi thường gặp ở đất nước Z là: nếu sự kiện văn hóa diễn ra tại thành phố u, có hai người ở thành phố i và thành phố j thì chi phí nhỏ nhất để hai người gặp nhau tại một thành phố nào đó là bao nhiêu.

Cho thông tin về các con đường của đất nước Z và q câu hỏi, mỗi câu hỏi được mô tả bằng 5 số u, i, j, a, b bạn hãy trả lời chi phí nhỏ nhất để hai người gặp nhau.

Dữ liệu: vào từ file văn bản **MEETING.INP**

+ Dòng đầu tiên chứa 2 số nguyên dương n và q $(n, q \leq 10^5)$.

+ $n-1$ dòng sau, mỗi dòng chứa 2 số nguyên x, y mô tả con đường nối giữa 2 thành phố x, y;

+ q dòng sau, mỗi dòng chứa 5 số nguyên dương u, i, j, a, b $(a, b \leq 10^6)$.

Kết quả: ghi ra file văn bản **MEETING.OUT**

+ Gồm q dòng, mỗi dòng là trả lời của mỗi câu hỏi trong dữ liệu vào.

Ví dụ :

MEETING.INP	MEETING.OUT
8 3 1 2 5 6 5 3 4 3 8 2 3 1 7 5 3 3 2 5 2 5 8 7 8 12 1 4 7 10 2	6 80 20

Ràng buộc:

- 20% số test tương ứng với 20% số điểm có $n, q \leq 100$;
- 20% số test tương ứng với 20% số điểm có $n, q \leq 1000$;

- 20% số test tương ứng với 20% số điểm có $n \leq 2000, q \leq 10^5$;
- 20% số test tương ứng với 20% số điểm có $n, q \leq 10^5$ và $b \geq n \times a$;
- 20% số test tương ứng với 20% số điểm không có ràng buộc bổ sung.

Hướng dẫn thuật toán

+ Subtask 1: Ta sẽ dùng thuật toán bfs để tìm những đường đi từ i đến j và tính chi phí rồi chọn đường đi có chi phí ngắn nhất từ i đến j.

Độ phức tạp : $O(q \times n^2)$.

+ Subtask 2: Nhận thấy rằng đường đi từ i đến j sẽ có dạng i -> x -> j nên ta sẽ duyệt x và tìm chi phí bé nhất để đi từ i đến x và từ j đến x. Để làm được điều đó thì ta sẽ dùng thuật toán bfs với mỗi đỉnh là một trạng thái (x, k) với x là đỉnh đang xét và k là số đường đi đã đi được. Ta sẽ duyệt các đỉnh y kề với x và cập nhật như sau : nếu dist(x, u) > dist(y, u) thì d[v] = d[u] + k * a, ngược lại thì d[v] = d[u] + k * b.

Ta sẽ duyệt x và cập nhật đáp án là giá trị nhỏ nhất của d1[x] + d2[x] (với d1[x] là chi phí nhỏ nhất khi đi từ i đến x, d2[x] là chi phí nhỏ nhất khi đi từ j đến x).

Độ phức tạp : $O(q \times n)$.

Subtask 5: DFS cây lấy 1 làm gốc.

Có 2 bước cần giải quyết cho mỗi truy vấn.

-Tìm LCA của 2 đỉnh i,j trên cây lấy u làm gốc. Chính là đỉnh thấp nhất trong 3 đỉnh LCA(i,j), LCA(i,u) và LCA(j,u) trên cây gốc 1. Gọi đỉnh này là z.

-Tính vị trí tốt nhất để làm điểm hẹn.

Dễ dàng chứng minh được điểm hẹn sẽ nghiêng về nhánh phía dài hơn trong 2 nhánh (z,i) và (z,j). Viết phương trình bậc 2 một ẩn tính kết quả với ẩn là số lần đi xuống (dùng cost b) và tìm điểm cực tiểu.

Xét ẩn là 2 số nguyên không âm gần với điểm cực tiểu nhất, tính và chọn đáp án tốt hơn.

Độ phức tạp : $O(q \times \log_2 n)$.

Bài 6. MẠNG CÁC TRƯỜNG HỌC (network of Schools)

Một số trường học được nối với nhau bởi một mạng máy tính. Có một sự thoả thuận giữa các trường học này: mỗi trường học có một danh sách các trường học (gọi là danh sách các trường "nhận") và mỗi trường khi nhận được một phần mềm từ một trường khác trong mạng hoặc từ bên ngoài, cần phải chuyển phần mềm nhận được cho các trường trong danh sách các trường nhận của nó. Cần chú ý rằng nếu B thuộc danh sách các trường nhận của trường học A thì A không nhất thiết phải xuất hiện trong danh sách các trường nhận của trường học B.

Người ta muốn gửi một phần mềm đến tất cả các trường học trong mạng. Bạn cần viết chương trình tính số ít nhất các trường học cần gửi bản sao của phần mềm

này để cho phần mềm đó có thể chuyển tới tất cả các trường học trong mạng theo thoả thuận trên (Câu A). Ta muốn chắc chắn rằng khi bản sao phần mềm được gửi đến một trường học bất kỳ, phần mềm này sẽ được chuyển tới tất cả các trường học trong mạng. Để đạt mục đích này, ta có thể mở rộng danh sách các trường nhận bằng cách thêm vào các trường mới. Tính số ít nhất các mở rộng cần thực hiện sao cho khi ta gửi một phần mềm mới đến một trường bất kỳ trong mạng, phần mềm này sẽ được chuyển đến tất cả các trường khác (Câu B). Ta hiểu một mở rộng là việc thêm một trường mới vào trong danh sách các trường nhận của một trường học nào đó.

Dữ liệu vào

- Dòng đầu tiên chứa số nguyên N - số trường học trong mạng. $2 \le N \le 10^4$.

- Tiếp theo là N dòng, mỗi dòng thứ i+1 mô tả danh sách các trường nhận của trường i.

- Mỗi dòng là dãy các số nguyên dương (số hiệu trường nhận), kết thúc bằng số 0.

- Một dòng chỉ chứa 0 nghĩa là trường i không gửi cho trường nào cả.

Dữ liệu ra

Ghi ra 2 dòng:

- Dòng thứ nhất ghi số nguyên dương — lời giải cho **Câu A**.

- Dòng thứ hai ghi lời giải cho **Câu B**.

Giới hạn

- Tổng số cạnh trong đồ thị (tổng số trường nhận) không quá 5×10^5.

network_of_schools.inp	network_of_schools.out
5 2 4 3 0 4 5 0 0 0 1 0	1 2

+ Subtask 1: 10 điểm $N \le 10$ giới hạn tổng số cạnh ≤ 100

+ Subtask 2: 15 điểm $N \le 30$ giới hạn tổng số cạnh ≤ 300

+ Subtask 3: 20 điểm $N \le 100$ giới hạn tổng số cạnh ≤ 1.000

+ Subtask 4: 25 điểm $N \le 1000$ giới hạn tổng số cạnh ≤ 10.000

+ Subtask 5: 30 điểm $N \le 10^4$ giới hạn tổng số cạnh ≤ 500.000

Hướng dẫn: Thuật toán sẽ là:

+ Kosaraju 2 lần DFS → tìm SCC

+ Xây **DAG các SCC** → tính số SCC có bậc vào bằng 0 (câu A), số mở rộng cần thêm (câu B).

Bài 7: MẠNG LƯỚI BẢO MẬT (network.cpp)

Tập đoàn X quản lý một mạng lưới server gồm N server (đánh số từ 1 đến N). Một số server được kết nối với nhau thông qua các đường truyền bảo mật, cho phép truyền dữ liệu 2 chiều.

Một Mạng kín bảo mật cấp 4 là một nhóm 4 server phân biệt kết nối với nhau thành một vòng khép kín với các đường truyền như sau:

- Server A kết nối với Server B
- Server B kết nối với Server C
- Server C kết nối với Server D
- Server D kết nối với Server A

Hai vòng kín được xem là giống nhau nếu chỉ khác thứ tự quay (ví dụ (A,B,C,D) và (C,D,A,B) được tính là 1 vòng).

Quản trị viên muốn biết có bao nhiêu Mạng kín bảo mật cấp 4 đang tồn tại trong mạng lưới.

Yêu cầu

Cho N và M, danh sách các đường truyền giữa các server.
Hãy tính số lượng Mạng kín bảo mật cấp 4 khác nhau.

Input:

- Dòng đầu tiên chứa 2 số nguyên N, M ($1 \le N \le 1000$, $0 \le M \le 10000$) — số server và số đường truyền.
- M dòng tiếp theo, mỗi dòng chứa 2 số nguyên u, v ($1 \le u, v \le N$), mô tả một đường truyền giữa 2 server.

Output: Một số nguyên duy nhất — số lượng nhóm mạng kín cấp 4.

Subtask 1: $N \le 50$
Subtask 2: $N \le 200$
Subtask 3: $N \le 500$
Subtask 4: $N \le 1000$

Ví dụ:

Network.inp	Network.out
4 4 1 2 2 3 3 4 4 1	1

Thuật toán tối ưu ($O(N^2 * deg^2)$)

a) Biểu diễn đồ thị:

- Dùng ma trận kề adj[u][v] → kiểm tra cạnh tồn tại O(1).
- Dùng danh sách kề G[u] → duyệt các đỉnh kề u.

b) Duyệt cặp cạnh (u,v):

- Với mỗi cặp cạnh (u,v), tức là có cạnh u—v:

+ Duyệt các đỉnh x kề u (khác v).

+ Duyệt các đỉnh y kề v (khác u).

+ Kiểm tra xem có cạnh x—y hay không:

- Nếu có → (u,x,y,v) tạo thành 1 vòng kín cấp 4.

c) Vì sao cách này nhanh?

- Duyệt $O(N^2)$ cặp cạnh (u,v).

- Mỗi cặp (u,v), duyệt deg(u) * deg(v) cặp (x,y).

- Kiểm tra cạnh x—y O(1).

→ Tổng độ phức tạp: $O(N^2 * deg^2)$ → đủ tốt với N=1000.

d) Lưu ý quan trọng: Đếm vòng

Mỗi vòng kín (A,B,C,D) sẽ bị đếm 8 lần:

+ 4 điểm có 4! = 24 hoán vị → vì vòng là không có hướng, và có thể bắt đầu từ bất cứ đỉnh nào → chia 8 là đúng.

→ Kết quả cuối cùng: count / 8.

Bài 8. LẬP BẢN ĐỒ RỪNG QUỐC GIA (lanhdia.cpp)

Khu rừng quốc gia Pha Luông đang là nơi sinh sống của hai loài động vật quý hiếm: Hổ Đông Dương và Gấu ngựa. Để bảo tồn các loài này và phục vụ cho công tác nghiên cứu sinh học, các nhà khoa học tiến hành lập bản đồ lãnh thổ sinh sống của hai loài.

Khu rừng được chia thành một lưới ô vuông kích thước N x M. Mỗi ô trong lưới có thể nằm trong:

+ Biên lãnh thổ của Hổ.

+ Biên lãnh thổ của Gấu.

+ Biên lãnh thổ của cả hai loài.

+ Hoặc là ô đất trống không thuộc lãnh thổ nào.

Quy ước mã hóa ô:

+ Nếu ô (i,j) có giá trị là 1, thì ô đó nằm trên biên lãnh thổ của Hổ.

+ Nếu ô (i,j) có giá trị là 2, thì ô đó nằm trên biên lãnh thổ của Gấu.

+ Nếu ô (i,j) có giá trị là 3, thì ô đó nằm trên biên lãnh thổ của cả Hổ và Gấu.

+ Nếu ô (i,j) có giá trị là 0, thì ô đó là ô đất trống.

Đảm bảo rằng biên lãnh thổ của mỗi loài tạo thành một vòng khép kín.

Nhiệm vụ

Các nhà khoa học cần xác định có bao nhiêu ô trong khu rừng nằm hoàn toàn trong vùng lãnh thổ của cả hai loài động vật, tức là thuộc cả lãnh thổ của Hổ và của Gấu.

Ghi chú: Một ô nằm trong cả hai lãnh thổ nếu nó được bao quanh bởi biên của cả hai loài.

Dữ liệu vào

- Dòng đầu tiên chứa hai số nguyên N,M (1≤N,M≤1000) - số hàng và số cột của lưới bản đồ.

- N dòng tiếp theo, mỗi dòng gồm M số nguyên A[i][j] (0 ≤ A[i][j] ≤3), mô tả trạng thái từng ô.

Dữ liệu ra

- Một số nguyên không âm - số ô nằm trong cả lãnh thổ của Hổ và Gấu.

Lanhdia.inp	Lanhdia.out
5 5 0 0 0 0 0 0 3 1 2 0 0 1 0 2 0 0 1 2 2 0 0 0 0 0 0	1

*** Giới hạn**

+ Subtask 1 (25% số điểm): N=1.

+ Subtask 2 (25% số điểm): N,M ≤ 100 và biên của hai lãnh thổ trùng nhau.

+ Subtask 3 (25% số điểm): N,M ≤ 100.

+ Subtask 4 (25% số điểm): Không có ràng buộc thêm.

*** Ý tưởng thuật toán chuẩn**

- Ta cần xác định số ô nằm bên trong cả hai lãnh thổ → chính là các vùng liên thông nằm bên trong biên của Hổ *và* biên của Gấu.

- Mỗi lãnh thổ có biên khép kín → vùng bên trong có thể được tìm bằng:

+ Chạy flood-fill (DFS hoặc BFS) từ biên ngoài bản đồ để đánh dấu những ô thuộc bên ngoài lãnh thổ.

+ Những ô nào không bị đánh dấu, sẽ nằm bên trong lãnh thổ.

+ Làm bước này riêng biệt cho lãnh thổ Hổ và Gấu → ta có 2 mảng đánh dấu insideTiger[][] và insideBear[][].

+ Đếm các ô mà insideTiger[i][j] == true và insideBear[i][j] == true.

*** Độ phức tạp:** O(N×M)

Độ phức tạp cụ thể theo ràng buộc đề bài:

+ N,M ≤ 1000 → tối đa 1 triệu ô → chạy rất tốt với O(NM).

*** Thuật toán chuẩn học sinh HSG nên đạt**

+ Thực hiện 2 lần BFS hoặc DFS trên bảng 2 chiều.

+ Không cần tối ưu đặc biệt hơn O(NM).

+ Có thể ưu tiên BFS để tránh lỗi stack overflow khi dùng DFS đệ quy.