Mice and Maze

Link submit: http://www.spoj.com/problems/MICEMAZE/

Solution:

C++	https://ideone.com/tWSWHa
Java	https://ideone.com/6QDk2g
Python	https://ideone.com/m2nHjW

Tóm tắt đề:

Cho bạn N đỉnh trong trong đồ thị. Tại mỗi đỉnh có một con chuột đang ở đỉnh đó. Có một số đường đi từ đỉnh này đến đỉnh kia, đường đi là một chiều. Cho bạn một đỉnh là điểm mà các con chuột có thể thoát ra khỏi đồ thị trong thời gian quy định T.

Bạn hãy xác định số lượng con chuột có thể thoát ra khỏi đồ thị.

Input:

Dòng đầu tiên chứa số N (1 ≤ N ≤ 100) là số đỉnh trên đồ thị, dòng tiếp theo số E là cổng thoát, tiếp theo là số T là thời gian mà con chuột có thể thoát khỏi đồ thị.

Dòng tiếp theo chứa số M cho biết số cạnh nối giữa các đỉnh trong đồ thị.

M dòng tiếp theo mỗi dòng chứa ba số a, b và khoảng thời gian để đi hết đoạn đường đó.

Output:

Số con chuột có thể thoát ra khỏi đồ thị.

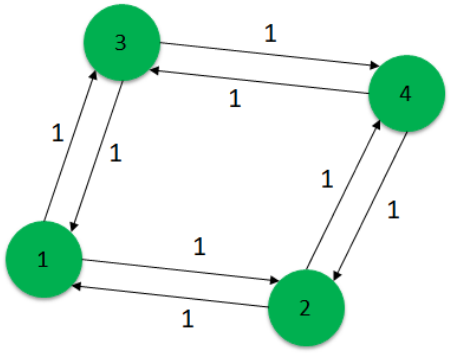
Ví dụ:

4 2 1 8 1 2 1 1 3 1 2 1 1 2 4 1 3 1 1 3 4 1 4 2 1 4 3 1	3

Giải thích ví dụ:

Đồ thị có 4 đỉnh N = 4, đỉnh số 2 là cổng thoát, thời gian đếm ngược để các chú chuột thoát khỏi là 1 giây.

Có 8 đường đi trong đồ thị được hiển thị như đồ thị bên.



Các con chuột có thể thoát ra khỏi mê cung:

1 → 2: Có 1 con chuột.

4 → 2: Có 1 con chuột.

Tại đỉnh 2: Có 1 con chuột.

Tổng có 3 con chuột có thể thoát khỏi đồ thị trên.

Hướng dẫn giải:

Có hai cách để giải quyết bài này.

Cách 1: do số lượng đỉnh N bé (100 đỉnh) nên bạn có thể chạy Dijkstra ở mỗi đỉnh rồi tìm đường đi từ các đỉnh đến đỉnh thoát ra. Nếu chi phí ≤ T thì bạn sẽ tăng số lượng con chuột lên.

Cách 2: cách này bạn chỉ chạy Dijsktra 1 lần, bạn sẽ chạy Dijkstra từ đỉnh thoát ra. Tuy nhiên trước khi chạy Dijkstra từ đỉnh này bạn phải quay ngược hướng toàn bộ đồ thị lại. Vì bạn cần tìm đường đi từ các đỉnh khác tới lối thoát, giờ bạn tìm đường đi từ đỉnh lối thoát đến các đỉnh khác, thì bạn cần phải quay ngược hướng các đồ thị lại để tìm đường đi. Sau đó xét các chi phí, chi phí nào nhỏ hơn bằng T thì tăng số lượng chuột có thể thoát lên.

Độ phức tạp: **O(V * ElogV)** cho cách 1 và **O(ElogV)** cho cách 2 với E là số lượng cạnh (cung), V là số lượng đỉnh trong đồ thị.