



## Mice and Maze

Link: <http://www.spoj.com/problems/MICEMAZE/>

Solution:

C++	<a href="https://ideone.com/x1EO9v">https://ideone.com/x1EO9v</a>
Java	<a href="https://ideone.com/LkaDwh">https://ideone.com/LkaDwh</a>
Python	<a href="https://ideone.com/bN1T56">https://ideone.com/bN1T56</a>

**Tóm tắt đề:** Cho bạn N đỉnh 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 1 đỉnh là điểm mà các con chuột có thể thoát ra khỏi đồ thị trong thời gian quy định T.

Hỏi bạn hãy xác định số lượng con chuột có thể thoát ra khỏi đồ thị khi cho bạn

**Input:**

Dòng đầu tiên chứa số N ( $1 \leq N \leq 100$ ) 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 là số cạnh nối giữa các đỉnh trong đồ thị.

M dòng tiếp theo mỗi dòng chữ 3 số a, b và khoảng thời gian để đi hết đoạn đường đó.

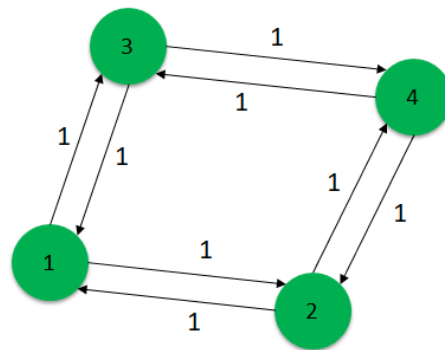
**Output**

Số con chuột có thể thoát ra khỏi đồ thị.

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:** Đồ thị có 4 đỉnh  $N = 4$ , cổng 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 dưới:



Các con chuột có thể thoát ra khỏi mê cung trên:

$1 \rightarrow 2$ : Có 1 con chuột.

$4 \rightarrow 2$ : 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 2 giải pháp được đưa ra.

**Giải pháp 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  $\leq T$  thì bạn sẽ tăng số lượng con chuột lên.

**Độ phức tạp:**  $O(V * E \log V)$  với  $E$  là số lượng cạnh (cung) trong đồ thị,  $V$  là số lượng đỉnh trong đồ thị.

**Giải pháp 2:** Giải pháp này sẽ giảm độ phức tạp xuống còn  $O(E \log V)$ , tức là bạn chỉ chạy Dijkstra 1 lần. Bằng cách 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 đỉnh 2, giờ bạn tìm đường đi từ đỉnh 2 đến các đỉnh khác, thì bạn cần phải quay ngược hướng các đồ thị lại để tìm đường đi với chi phí  $\leq T$ .

**Độ phức tạp:**  $O(E \log V)$  với  $E$  là số lượng cạnh (cung) trong đồ thị,  $V$  là số lượng đỉnh trong đồ thị.