Alice in Amsterdam, I mean Wonderland

Link submit: http://www.spoj.com/problems/UCV2013B

Solution:

	Lưu trữ đồ thị bằng ma trận kề	Lưu trữ đồ thị bằng danh sách cạnh
C++	https://ideone.com/vZJPgW	https://ideone.com/bUivXd
Java	https://ideone.com/jrQ0XL (TLE)	https://ideone.com/mJbtBp (dùng array)
Python	https://ideone.com/toVmoG (TLE)	https://ideone.com/1cKfKa (TLE)

Tóm tắt đề:

Alice đến xứ sở thần tiên và gặp nhiều điều kỳ lạ xảy ra. Khoảng cách giữa các tượng đài trong thành phố đôi khi là số âm, nhưng khoảng cách bằng 0 giữa hai tượng đài khác nhau lại có nghĩa là không có đường đi trực tiếp giữa chúng. Ngoài ra, mỗi tượng đài có thể có khoảng cách bằng 0 hoặc âm tới chính nó, còn trong trường hợp dương chúng ta coi như khoảng cách đó bằng 0.

Trong trường hợp đường đi là một chu kỳ với khoảng cách âm, sẽ luôn có một đường ngắn hơn để đến cùng một tượng đài. Vậy nên theo Alice, con đường ngắn nhất sẽ ngắn hơn nếu đi con đường đó lặp đi lặp lại vô tận.

Cho một danh sách tượng đài của thành phố và khoảng cách của chúng. Tìm đường đi ngắn nhất giữa một số cặp tượng đài.

Input:

Có nhiều test case, mỗi test case có cấu trúc như sau:

Dòng đầu là số nguyên N ($1 \le N \le 100$) – số tượng đài trong thành phố, các tượng đài được đánh số từ 0 đến N - 1.

N dòng tiếp theo, mỗi dòng chứa một chuỗi K và N số nguyên Kj ($0 \le j < N, -2^{30} \le Kj \le 2^{30}$) – lần lượt là tên của tượng đài i (tối đa 20 ký tự chữ và số), Kj là khoảng cách giữa tượng đài i và j.

Dòng tiếp theo chứa một số nguyên Q ($1 \le Q \le N^2$) – số truy vấn.

Q dòng tiếp theo, mỗi dòng chứa hai số nguyên U, V ($0 \le U, V < N$) là chỉ số của hai tượng đài cần xác định khoảng cách.

Kết thúc input là một dòng chứa số 0 (không xử lý).

Output:

Với mỗi test case, dòng đầu in "Case #tc:" với tc là thứ tự của test case đang xử lý bắt đầu từ 1.

Q dòng tiếp theo sẽ miêu tả kết quả các truy vấn. Nếu đường đi tối ưu từ U với V có thể nhỏ vô hạn thì in ra "NEGATIVE CYCLE". Ngược lại, dòng bắt đầu với "tên_tượng_đài_bắt_đầu-tên_tượng_đài_kết_thúc" rồi đến kết quả. Nếu không có đường đi từ U đến V thì in "NOT REACHABLE", ngược lại in ra một số nguyên là khoảng cách ngắn nhất giữa hai tượng đài.

Ví dụ:

2 Nieuwkerk -1 1 Oudekerk 1 0 4 0 0 0 1 1 1 1 0 3 Nieuwkerk 0 -5 0 Oudekerk 10 0 0 Pierteck -100 -100 0 9 0 0 0 1 0 2 1 0 1 1 1 2 2 0 2 1 2 2 0	Case #1: NEGATIVE CYCLE NEGATIVE CYCLE NEGATIVE CYCLE NEGATIVE CYCLE Case #2: Nieuwkerk-Nieuwkerk 0 Nieuwkerk-Oudekerk -5 Nieuwkerk-Pierteck NOT REACHABLE Oudekerk-Nieuwkerk 10 Oudekerk-Oudekerk 0 Oudekerk-Pierteck NOT REACHABLE Pierteck-Nieuwkerk -100 Pierteck-Oudekerk -105 Pierteck-Pierteck 0

Giải thích ví dụ:

Ví dụ trên có hai bộ test.

Bộ 1: gồm 2 tượng đài, 4 truy vấn.

- Truy vấn đầu, từ 0 đến 0 có thể lặp vô hạn với chi phí luôn giảm (mỗi lần -1).
- Các cặp đỉnh khác cũng có thể đi tới 0 và lặp vô hạn rồi tới điểm kết thúc (0 → 0 → 0 → ... → 1; 1 → 0 → 0 → 0 → ... → 1; 1 → 0 → 0 → 0 → ... → 0).
- Vậy nên kết quả của cả 4 truy vấn là NEGATIVE CYCLE.

Bộ 2: gồm 3 tượng đài, 9 truy vấn.

- Truy vấn đầu, từ 0 đến 0 với chi phí là 0 (0 chỉ có thể đến chính nó – chi phí 0, qua 1 – chi phí -5, và từ 1 có thể qua 0 – chi phí 10, vậy nên nếu đi 0 → 1 → 0 →... chi phí càng tăng, nên ta chỉ đi 0 → 0).
- Truy vấn hai, từ 0 đến 1 với chi phí là -5 (tương tự truy vấn 1, nên ta chỉ đi 0 → 1).
- Truy vấn ba, từ 0 đến 2 không có đường đi nên kết quả là NOT REACHABLE.
- Các truy vấn còn lại tương tự.

Hướng dẫn giải:

Sử dụng ma trận kề để lưu trọng số các cạnh. Hoặc có thể lưu thông tin đồ thị ở dạng danh sách cạnh, dùng cách này chỉ cần lưu những cạnh có trọng số khác 0 trừ các cặp (i, j) với i == j.

Tiền xử lý: Nếu a[i][i] < 0 thì cho a[i][i] = -INF có nghĩa là khoảng cách ngắn nhất từ i đến chính nó là nhỏ vô hạn, nếu a[i][i] > 0 thì cho a[i][i] = 0 theo đề bài.

Sử dụng thuật toán Ford-Bellman để tìm đường đi ngắn nhất và kiểm tra chu trình âm với N điểm bắt đầu. Đồng thời gọi neg[u][v] là biến đánh dấu nếu có thể gặp chu trình âm trên đường đi từ u đến v.

Giả sử đang tìm đường đi ngắn nhất từ đỉnh s đến tất cả đỉnh khác. Nếu ở bước lặp thứ N mà đỉnh v nào còn cần cập nhật thì đỉnh đó nằm trên chu trình âm, ta đánh dấu neg[s][v] = true. Chú ý ban đầu để dist[s][s] = a[s][s].

Độ phức tạp: **O(T * N^4)** và với T là số lượng bộ test, N là số tượng đài trong thành phố khi sử dụng ma trận kề. Cách lưu danh sách cạnh là **O(T * N^2 * M)** với M là số cạnh.

Note: Python chậm hơn so với C++ và Java, nên bị TLE, sắp tới Big-O sẽ tạo test và up lên trang algote để các bạn submit bằng Python, khi nào có sẽ thông báo sau.