Traffic Network

Link submit: http://www.spoj.com/problems/TRAFFICN/en/

Solution:

C++	https://ideone.com/EBCS61
Java	https://ideone.com/NtBt4I
Python	https://ideone.com/qpPSPx

Tóm tắt đề:

Đề bài cho bạn n điểm đánh số từ 1 đến n, và m đường một chiều nối từ điểm này sang điểm kia. Sau đó, cho bạn k đường hai chiều. Yêu cầu bạn dùng một đường hai chiều trong danh sách các đường đã cho sao cho đường đi từ đỉnh s đến đỉnh t là đường đi ngắn nhất.

Input:

Dòng đầu tiên chứa số nguyên dương không lớn hơn 20 là số lượng test case.

Mỗi case gồm những thông tin sau:

- Dòng đầu gồm 5 số nguyên dương n ($n \le 10000$), m ($m \le 100000$), k ($k < 300$), s ($1 \le s \le n$), t ($1 \le t \le n$).
- m dòng tiếp theo, mỗi dòng chứa 3 số nguyên dương di, ci, li với li ($0 < li \le 1000$) là độ dài của đường một chiều thứ i từ nút di đến nút ci.
- k dòng tiếp theo, mỗi dòng chứa 3 số nguyên dương uj, vj, qj với qj ($0 < qj \le 1000$) là độ dài của đường hai chiều thứ j từ nút uj đến nút vj.

Output:

Với mỗi case, in ra độ dài nhỏ nhất có thể của đường đi ngắn nhất sau khi chọn xây một đường hai chiều từ danh sách đề xuất trên một dòng. Nếu không có đường đi từ s đến t, in -1.

Ví dụ:

1 4 5 3 1 4 1 2 13 2 3 19 3 1 25 3 4 17 4 1 18	35

1 3 23 2 3 5 2 4 25	

Giải thích ví dụ:

Trong ví dụ có 1 bộ test. Bộ này có 4 điểm, 5 đường một chiều, 3 đường hai chiều, tìm đường ngắn nhất đi từ 1 đến 4 khi chọn một trong 3 đường hai chiều để xây dựng.

Chọn đường hai chiều 2 đến 3 với độ dài 5, thì ta tìm được đường ngắn nhất là 1 → 2 → 3 → 4 với độ dài là 13 + 5 + 17 = 35.

Hướng dẫn giải:

Đầu tiên bạn sẽ gắn toàn bộ đường đi một chiều vào đồ thị sau đó lần lượt gắn từng đường đi hai chiều vào, nếu đường nào cho ra đường đi ngắn nhất thì bạn chọn đó là kết quả cuối cùng.

Lưu ý: để làm được việc này thì mỗi lần gắn đường đi hai chiều vào bạn phải chạy lại thuật toán Dijkstra. Việc này sẽ làm bài bạn bị quá thời gian (TLE). Bạn phải cải tiến thuật toán Dijkstra và cải tiến luôn việc không thể gọi Dijkstra mỗi lần gắn đường đi hai chiều mới vào bằng cách sau:

- Tính Dijkstra từ s → t và tính chiều ngược lại t → s (gọi Dijkstra hai lần).
- Sau khi chạy Dijkstra ta có:
  - distS[u]: tìm đường đi từ đỉnh s đến đỉnh u (trong đồ thị bên dưới là 1 → 5).
  - distT[v]: tìm đường đi từ đỉnh v đến đỉnh t (trong đồ thị là 3 → 4).
- Gắn từng đường đi hai chiều vào, lúc này dùng công thức bên dưới, công thức này giống như việc gọi Dijkstra lại mỗi lần gắn đường đi hai chiều vào.
- Sau khi gắn d vào, nếu tổng này nhỏ hơn đường đi hiện tại đang có thì nó chính là đường đi ngắn nhất.

```
int a = distS[u] + d + distT[v];
int b = distS[v] + d + distT[u];
result = min(result, min(a, b));
```

Độ phức tạp: **O(T * ElogV)** với E là số lượng cạnh của đồ thị, V là số lượng đỉnh của đồ thị và T là số lượng bộ test trong từng dữ liệu đầu vào.

Note: time limit của bài này quá chặt, nên java và python bị TLE, sắp tới Big-O sẽ tạo test và up lên trang algote để các bạn submit bằng hai ngôn ngữ này, khi nào có sẽ thông báo sau.