The Benefactor

Link submit: https://www.spoj.com/problems/BENEFACT/

Solution:

C++	https://ideone.com/HMAQqk
Java	https://ideone.com/JX6lSF
Python	https://ideone.com/8VWtsE

Tóm tắt đề:

Cho bạn danh sách các con đường trong thị trấn. Biết rằng giữa hai địa điểm bất kì chỉ có duy nhất một con đường nối. Nhiệm vụ của bạn là tìm con đường dài nhất trong thị trấn đó.

Lưu ý: con đường dài nhất không phải con đường đi qua hết các địa điểm mà là con đường có độ dài lớn nhất nối giữa hai địa điểm bất kì.

Input:

Dòng đầu tiên là một số nguyên t – số lượng test case.

Mỗi test case có định dạng như sau:

- Dòng đầu tiên gồm một số nguyên n – số địa điểm trong thị trấn ($2 \le n \le 50.000$).
- Mỗi con đường được xác định bởi cặp địa điểm a, b nối bởi con đường ấy ($1 \le a, b \le n$) và độ dài l của con đường ($0 \le l \le 20.000$).

Output:

Với mỗi test case, in ra trên một dòng độ dài của con đường dài nhất trong thị trấn.

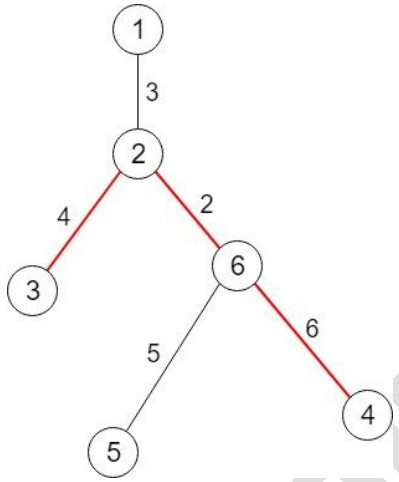
Ví dụ:

1 6 1 2 3 2 3 4 2 6 2 6 4 6 6 5 5	12

Giải thích ví dụ:

Ví dụ trên có một bộ test.

Con đường dài nhất là con đường nối giữa địa điểm 4 và 6 và có độ dài 4 + 2 + 6 = 12.



Hướng dẫn giải:

Xem danh sách được cho là một đồ thị với các đỉnh là các địa điểm trong thị trấn và con đường nối giữa hai địa điểm là một cạnh của đồ thị. Ta có:

- Giữa hai đỉnh bất kì luôn chỉ có một cạnh nối → đồ thị dạng cây (E = V – 1).
- Độ dài lớn nhất giữa hai đỉnh bất kì chính là đường kính của cây.

Như vậy bài toán của ta trở thành bài toán tìm đường kính của cây.

Ta lại có nhận xét đường kính của cây chính là khoảng cách lớn nhất giữa hai node lá. Như vậy nhiệm vụ của mình là phải tìm hai node lá này.

Giả sử ta thực hiện DFS từ một đỉnh bất kì. Điều chắn chắn rằng node có khoảng cách xa nhất từ điểm mà ta bắt đầu lan chính là một node lá cần tìm. Khi này chỉ cần sử dụng tiếp DFS để đi từ node lá mới tìm được là ta sẽ có được node lá thứ hai, đồng thời tính được khoảng cách giữa hai node lá đó, tức đường kính của cây. Thuật toán này có tên là “Double DFS” được mở rộng từ thuật toán DFS cơ bản.

* Lưu ý: Để lưu trọng số của mỗi cạnh, ta có thể sử dụng pair – kiểu dữ liệu lưu một cặp giá trị.

Tóm lại, bài này ta có thể thực hiện như sau:

- Bước 1: Đọc dữ liệu. Biết số cạnh = số đỉnh – 1 (tính chất của cây).
- Bước 2: Khởi tạo biến lưu khoảng cách xa nhất từ node mà ta đang xét là max_dist = 0. Khai báo biến f lưu node lá.

- Bước 3: Chạy DFS từ một đỉnh bất kì. Nếu tìm được đỉnh v có khoảng cách đến điểm bắt đầu lớn hơn max_dist:
    - Cập nhật max_dist = dist[v]
    - Cập nhật node lá f = v
- Bước 4: Chạy DFS từ đỉnh f. Nếu tìm được đỉnh v có khoảng cách đến điểm bắt đầu lớn hơn max_dist thì cập nhật max_dist = dist[v].
- Bước 5: In ra kết quả trong max_dist.

Độ phức tạp: **O(T * (V + E))** với V là số lượng đỉnh và E là số lượng cạnh của đồ thị và T là số lượng bộ test cho mỗi dataset.