**D13NET**

* Sub 1: Ta sẽ duyệt tập con các cạnh rồi kiểm tra xem các định có thể kết nối với nhau như ban đầu hay không bằng thuật toán DSU. ( O2^nlogn).
* Sub 2: Ta sẽ quy hoạch động (t, pa[]) với ý nghĩa đang xét cạnh thứ t và đã đạt trạng thái pa. Hàm cal(t, pa[]) được tính như sau ( xét các cạnh từ 0 đến m – 1):
* Nếu t = m và trạng thái pa giống như trạng thái nếu nối đủ m cạnh thì return 1.
* Ta cập nhật mảng DSU theo cạnh có vị trí nhỏ hơn.
* Nếu sử dụng cạnh t thì sẽ cập nhật DSU pa’ rồi cnt += cal(t+1, pa’).
* Nếu không sử dụng cạnh t thì cnt += cal(t+1, pa);
* Lưu ý đễ giảm độ phức tạp so với cách 1 ta để ý nếu như cùng t và cùng pa thì ta sẽ cho ra cùng 1 kết quả nên ta sẽ dùng mảng f[t][pa] để lưu lại kết quả và giảm độ phức tạp của hàm cal(t, pa).

**D13NET**

* Thuật toán : Ta coi mỗi vị trí là 1 đỉnh và cạnh kết nối của nó là giữa các vị trí có giá trị giống nó theo input đề bài => chỉ cần kiểm tra xem có bao nhiêu thành phần liên thông bằng DSU.
* Tối ưu bài toán bằng cách sử dụng thuật toán RMQ cho mảng DSU. Ta có D[i][k] là root của i ở tầng k (k ở đây tức là từ i đến vị trí i+2^k-1 sẽ là các giá trị bằng nhau).
* Xử lí thông tin đầu vào : với mỗi p, q, l thì ta sẽ join các đoạn cùng hạng k của p, q lại với nhau bằng DSU ( hạng k là các bit 1 của l ).
* Truyền thông tin xuống các hạng dưới bằng cách join(i, r, k-1) và join(i, r+(1<<(k-1)), k-1) (với r là root của D[i][k]).
* Ta có kết quả sẽ là các thành phần liên thông khác nhau bằng cách đếm số giá trị khác nhau của D[i][0].