**D13IJ**

* Đây là bài toán tìm đường đi ngắn nhất cho moi cặp đỉnh nên ta chuẩn bị trước mảng c[i][j] bằng thuật toán Foyld.

-Khi bổ sung thêm một đỉnh và một số cạnh vào đồ thị thì ta sẽ cập nhật mảng c[][] :

**+**Cập nhật đường đi ngắn nhất từ mọi đỉnh đến đỉnh mới.

+Cập nhật đường đi ngắn nhất từ đỉnh mới đến mọi đỉnh.

+Cập nhật lại đường đi ngắn nhất giữa mọi cặp đỉnh

**Chú ý :** Thứ tự cập nhật ở đấy thì bước 1 và 2 có thể đổi chỗ cho nhau nhưng bước 3 phải làm sau

**D13SERVICE**

* Tính c[i][j] với Floyd
* Với mỗi I thì ta sẽ lấy cạnh nhỏ nhất mà nối tới i

Ans += min(c[i][j]) với j là đỉnh kề I

**D13SNOW**

-Sub 1: k = n thì bài toán trở về cây khung ngắn nhất, ta dùng thuật toán kruskal để giải quyết.

* Sub 2: k = 2 đường đi ngắn nhất giữa 2 đỉnh bất
* Sub 3: k = 3 thì ta sẽ tìm một đỉnh trung gian giữa 3 đỉnh để kết nối các đỉnh lại với nhau, kết quả sẽ là min(c[node1][i] + c[node2][i] + c[node3][i]).
* Sub 4: k <= 10 nên ta sẽ sử dụng bitmask ở đây.
* f[i][t] ( t là bitmask ) với ý nghĩa đang đứng ở i và liên thông tập t ( t là tập con của k đỉnh cần kết nối )
* Ta sẽ sử dụng thuật toán dijkstra để cập nhật kết quả cho f[i][t].
* TH 1: f[i][t] cập nhật tiếp cho các đỉnh kề i:
* f[i][t] cập nhật cho f[j][t] nếu j ko thuộc tập k đỉnh.
* f[i][t] cập nhật cho f[j][t|(1<<2^j)] nếu j thuộc tập k đỉnh.
* TH 2: f[i][t] cập nhật cho các tập hợp lớn hơn nhưng vẫn đứng ở i: f[i][t’] = f[i][t’^t] + f[i][t] ( với t là tập con của t’ ).
* Sub 5: Để ý yêu cầu bài toán thì ta chỉ cần kết nối n – k đỉnh còn lại với nhau tức là hầu hết các đỉnh trong đồ thị kết nối với nhau bằng một cây khung. Mặt khác với k <= 10 thì ta có thể dùng bitmask để duyệt các khả năng các đỉnh còn lại có nằm trong cây khung đó hay không. Bài toán nhỏ ta sẽ sử dụng thuật toán Kruskal để giải quyết như Sub1.