***Explicación del problema Rutas (CNC 2016 2B)***

Este problema trata de hallar sucesivamente el par de nodos más lejanos en subárboles de un árbol dado. Expongamos primeramente algunas definiciones y teoremas de teoría de grafos que usaremos.

***Def. Diámetro en un grafo***: Costo del camino entre el par de nodos más lejanos en el grafo.

***Def. Grafo conexo***: Entre cada par de vértices existe al menos un camino simple.

***Def. Árbol***: Grafo conexo y acíclico (no tiene ciclos).

Además **en un árbol para cada par de vértices existe un único camino**, de lo contrario, si para dos vértices existieran dos caminos distintos entonces podríamos encontrar un ciclo con algún subconjunto de las aristas de esos dos caminos, lo que contradice la definición de árbol. Otra característica muy importante de **los árboles es que tienen N – 1 aristas donde N es la cantidad de vértices en el grafo**, esto es consecuencia de la unicidad de los caminos entre cada par de vértices: si escogemos un nodo cualquiera y lo llamamos nodo raíz entonces hay N – 1 caminos desde la raíz hacia los restantes N – 1 nodos y podemos establecer una biyección entre el conjunto de esos caminos y las aristas asociando a cada camino la arista final del mismo, no habrán dos caminos distintos con la misma arista final porque contradice la unicidad de los caminos y todos las aristas son el final del camino de la raíz a al nodo más lejano de los dos que definen la arista.

A continuación las soluciones propuestas:

**Solución 1**: Cada vez que tengamos que “quitar” una arista y se obtengan dos subárboles a los cuales tenemos que hallarle el diámetro hacemos lo siguiente para cada uno de ellos:

sol = 0

Para cada nodo v en el subárbol:

Para cada nodo u en el subárbol:

Si sol < (x = camino(v, u)):

sol = x

La rutina camino() recibe dos nodos y retorna el costo del camino entre los dos nodos. Esta rutina puede programarse con un DFS o BFS y el costo de la misma es O(N), ya que el costo del DFS/BFS es O(N + E) donde N es la cantidad de vértices y E la cantidad de aristas (para convencerse de ese costo notemos que tanto del DFS como el BFS examinan cada nodo a lo sumo una vez y cada arista a lo sumo dos veces). Por lo que el peor caso de esta solución es O(N4). Esta solución da en tiempo para N <= 100 y obtiene el 10% de los puntos. Una mejora bastante rápida es evitar el segundo ciclo:

sol = 0

Para cada nodo v en el subárbol:

Si sol < (x = explora(v)):

sol = x

Nuevamente la rutina explora(v) es un DFS/BFS que retorna el costo del camino al nodo más lejano del nodo v. Esta pequeña mejora reduce la complejidad en el peor caso a O(N3) y obtiene el 15% de los puntos.

**Solución 2:** Primero veamos el pseudocódigo y luego demostremos su correctitud, sea A el árbol al cual queremos hallarle el par de nodos más lejanos y la rutina lejano(v) que retorna el nodo más lejano a v (igualemente se programa con un DFS/BFS):

x = nodo cualquiera en el árbol A

nod1 = lejano(x)

nod2 = lejano(nod1)

diametro = Costo del camino del nod1 al nod2 //esto ya puede haber quedado //calculado en una variable global en la segunda llamada a lejano()

Este algorimo tiene costo O(N) puesto que solo hace dos llamadas a un DFS/BFS.

La complejidad temporal y espacial esta solución es O(FIXME).

**Habilidades requeridas**: FIXME

**Actividades propuestas**: FIXME

PSN 2016 - Frank Arteaga Salgado, farteaga@ipvce.lt.rimed.cu