Tarea 7 - Programación Eficiente 14 de Noviembre del 2018
Miguel Angel Ruiz Ortiz

Graph Queries: MO's algorithm

Sea G un grafo no dirigido con N vértices y M aristas, $1 \leq N, M \leq 200000$. Cada arista tiene un único índice del 1 al M. Tenemos que contestar las siguientes Q queries, $1 \leq Q \leq 200000$: ¿cuántas componentes conectadas del grafo quedan al quitar todas las aristas en G excepto aquellas con índice $X \in [L_i, R_i]$, $1 \leq L_i, R_i \leq M$ y $i = 1, \ldots, Q$. Daremos respuesta a las queries en un orden conveniente, de manera que podamos contestar las Q queries en $O((M+Q)\sqrt{M})$.

Supongamos que $Query[1, \ldots, Q]$ es un arreglo tal que $Query[i]$ es la pareja (L_i, R_i), y también supongamos dicho arreglo está ordenado bajo la relación de orden $<_Q$, donde

$$Query[i] <_Q Query[j] \quad \Leftrightarrow \quad \begin{cases} \frac{L_i}{\sqrt{M}} < \frac{L_j}{\sqrt{M}}, & \text{si } \left\lfloor \frac{L_i}{\sqrt{M}} \right\rfloor \neq \left\lfloor \frac{L_j}{\sqrt{M}} \right\rfloor, \\ R_i < R_j, & \text{si } \left\lfloor \frac{L_i}{\sqrt{M}} \right\rfloor = \left\lfloor \frac{L_j}{\sqrt{M}} \right\rfloor. \end{cases}$$

Es decir, si $K = \lfloor \sqrt{M} \rfloor$, estamos ordenando las queries de manera que las queries con L_i en $[nK, (n+1)K-1]$, $n = 0, 1, \ldots, K$, estén juntas en, digamos, un bloque de tamaño a lo más K, y en cada bloque ordenamos con respecto a las R_i. Podemos usar la función $sort()$ para que ordene $Query[\,]$ con $<_Q$.

Contestaremos las queries por bloques. La idea es utilizar la información de una query que contestemos en determinado momento para contestar la siguiente. Se explicará cómo contestar las queries con L_i en $[0, K-1]$, y para contestar los siguientes bloques se realiza de manera análoga.

El problema nos pide contestar cuántas componentes conectadas hay si G sólo tiene las aristas con índice en cierto rango $[L, R]$, y esto lo podemos relacionar con la estructura *Union-Find Disjoint Sets* (UFDS) porque al colocar la arista (u, v) debemos unir las componentes conectadas a la que pertenecen u y v. Para nosotros, "agregar la arista (u, v) al UFDS" significará llamar al método $unionSet(u, v)$ de nuestra estructura, el cual unirá las componentes conectadas a la que pertenecen u y v.

Para las queries (L, R) tales que $R < K$, simplemente agregamos las aristas con índice $[L, R]$ al UFDS y regresamos el número de conjuntos en nuestra estructura. Para las demás queries en el bloque, supondremos que tenemos dos métodos en nuestro UFDS llamados $fix()$ y $rewind()$. Llamar $fix()$ en determinado momento hará que la próxima vez que llamemos a $rewind()$ volveremos a tener la información en el UFDS como lo teníamos en el momento en el que llamamos a $fix()$.

Consideremos j el primer índice en el bloque tal que $R_j \geq K$. Las aristas con índice en $[K, R_j]$ las agregamos al UFDS y llamamos a $fix()$. Para contestar $Query[j] = (L_j, R_j)$ nos falta agregar las aristas con índice en $[L_j, K-1]$ y podremos dar respuesta. Para contestar la siguiente query, $Query[j+1] = (L_{j+1}, R_{j+1})$, llamamos a $rewind()$ y agregamos al UFDS las aristas en $[R_j + 1, R_{j+1}]$. En este momento hacemos $fix()$ y agregamos las aristas en $[L_{j+1}, K-1]$. Con esto ya podremos dar respuesta la query (L_{j+1}, R_{j+1}). Luego hacemos $rewind()$ y así sucesivamente vamos contestando las demás queries faltantes, en cada bloque se trata de manera análoga.

Los métodos $fix()$ y $rewind()$ funcionan de la siguiente manera: en el UFSD tenemos un arreglo $p[1,\ldots,N]$ y un arreglo $past[1,\ldots,N]$ tal que $p[i]$ es el índice del nodo representante del conjunto al que pertenece el nodo i y $past[i]=0$ si después de llamar a $fix()$ no se ha modificado $p[i]$, y $past[i]=k$ si después de llamar a $fix()$ se tenía $p[i]=k$ y luego se modificó $p[i]$. Además tendremos una pila $modified$ en la que iremos agregando los nodos i tales que $p[i]$ fue modificado después de hacer $fix()$.

Nota: Por cada vez que llamamos $fix()$ sólo podemos llamar una sola vez $rewind()$.