# Tarea 7 - Programación y algoritmos 2

Erick Salvador Alvarez Valencia

CIMAT A.C.,
erick.alvarez@cimat.mx

**Resumen** En la presente tarea se describirá el código implementado para resolver el problema llamado *Chef and Graph Queries* usando una variante del algoritmo de MO la cual va a ser descrita así como los resultados obtenidos.

## 1. Descripción del problema

El problema consiste en dado un Grafo $G = (V, E)$ no dirigido y posiblemente con ciclos, se tiene que responder a una cantidad de queries en las cuales se dan dos valores $a$ y $b$ en donde hay que determinar cuál sería el número de componentes conexas en el grafo si se borraran todas las aristas excepto las contenidas en el conjunto $a \leq e_i \leq b$.

**Restricciones del problema:**

1. Número máximo de casos: 1000.
2. Máximo número de nodos, aristas y queries: 200000.

## 2. Algoritmo implementado

Este problema se puede resolver respondiendo a las queries de manera offline usando una variante del algoritmo de MO en la cual se hace uso de la estructura de datos **Union-Find** para llevar el conteo de las componentes conexas del grafo. Lo primero es ordenar las queries de la forma que establece el algoritmo de MO, dividiéndolas en bloques de tamaño $sqrtM$ por la parte izquierda, donde $M$ es el número de aristas del grafo, y en caso de que dos queries caigan en el mismo bloque, se desempata por la parte derecha. Esto anterior se hace para procesar queries en donde las aristas que están a lo más alejadas por una distancia de raíz cuadrada caigan en el mismo bloque.
Posteriormente se irán procesando las queries bloque por bloque y lo que se hace es obtener las aristas de la querie que estén en el siguiente bloque, todas estas se trabajan de manera normal con el Union-Find haciendo uniones. Posteriormente para las aristas que se encuentran en el mismo bloque se tiene que tener en cuenta que las siguientes queries pueden pedir que estas ya no se tomen en cuenta por lo que se necesitaría hacer un método *erase* en el UF, pero esta estructura de datos no lo permite por lo que hay que ir guardando las modificaciones que se van

haciendo en un vector y posteriormente revertirlas, pero antes de ello se guarda la respuesta en otro vector para poder imprimirla al final. La complejidad final de este algoritmo es $O(Qsqrt(M))$ porque el Union-Find posee una complejidad amortizada de $O(1)$.

**Algorithm 1** Algoritmo de MO para árboles.

```
1: procedure MOsTree(Queries, Edges, DSU, BlockSize, NoVertex)
2:     idx ← 0
3:     answers ← array[Queries.size()]
4:     sort(Queries, QuerieComparator)
5:     for k = 0 → BlockSize do
6:         DSU.restart()
7:         res ← NoVertex
8:         ll ← BlockSize * (k + 1)
9:         while idx < Queries.size() and Queries[idx].bl = k do
10:            if Queries[idx].bl = Queries[idx].br then        ▷ Todas las aristas de la query están en el mismo bloque
11:                answers[Queries[idx].id] ← NoVertex - DSU.UnionMul(Queries[idx].left, Queries[idx].right, Edges)  ▷ El método anterior une todas las aristas en el rango indicado y regresa el número de uniones hechas
12:            else                                            ▷ Queries en bloques diferentes
13:                while ll ≤ Queries[idx].right do
14:                    if DSU.UnionSingle(Edges[ll].u, Edges[ll].v) then
15:                        res ← res - 1
16:                    ll ← ll + 1
17:                answers[Queries[idx].id] ← res - DSU.UnionMul(Queries[idx].left, BlockSize * (k + 1) - 1, Edges)
18:            idx ← idx + 1
19:    return answers
```

En el algoritmo anterior se usan dos variantes del método *Union* del UF, el *UnionSingle* realiza la unión de manera normal y el *UnionMul* une todas las aristas en un rango pero revirtiendo el proceso una vez que se obtuvo el número de componentes generadas, y para ello este último no hace compresión de caminos al hacer el método *Find*.

## 3. Resultados

Se envió el código a la plataforma de CodeChef y se obtuvo el resultado de *Aceptado*.

| ID | Date/Time | User | Result | Time | Mem | Lang | Solution |
|---|---|---|---|---|---|---|---|
| 21639008 | 3 hours ago | ericksav22 | ✔ | 7.16 | 21.8M | C++14 | View |

(a) Figura 1. Resultado obtenido al enviar el código a Codechef.