CC4102 **Diseño y Análisis de Algoritmos** **2021A**

Jérémy Barbay, Javier Oliva

Ejercicio Formativo 1

(Fecha de entrega: [2021-03-23 Tue])

Sofía Valentina Bobadilla Ponce

1. Cual es el orden de complejidad del algoritmo de `Prim` en el peor caso por n y m fijado?

   En el peor de los casos el orden del algoritmo de Prim es cuadrático , $0(n^2)$, la razón de este orden se debe a que como ya es sabido en su comportamiento se tiene un ciclo o corrida que se asevera de no finalizar el proceso hasta que se hayan "capturado" todos los vértices , pero a la vez por cada nuevo vertice se va actualizando la información sobre los arcos vecinos al vertice actual de cada iteración , en el peor de los casos a parte de recorrer todos los vertices se recorren todos los vecinos por ello el orden para este caso corresponde a $O(n^2)$

2. Se puede mejorar el algoritmo de `Prim` con una lista ordenada de los arcos? Se podría mejorar mediante un heap, la idea es mantener la info de los vertices y sus arcos de forma ordenada lo que permitiría acceder de inmediato al arco correcto para añadir a la lista de vertices capturados, estoy podría conllevar a una mejora pero depende directamente de la cantidad de nodos con los que se trabaje. El orden de esta mejora sería $O(mlog(n))$ ya que recorrer el árbol en busca del arco es de orden $O(log(n))$ y lo cual se repite para todos los arcos aún no conectados , siendo m el total de arcos.

3. Se puede mejorar el algoritmo de `Prim` con una estructura de tipo "Union Find"? Para esta estructura contamos con dos ayudas el método de búsqueda y el de unión , ambos permiten hacer nuevas implementaciones para ciertos algoritmos , para el algoritmo de Kruskal se puede variar ordenando en una lista de prioridad los vertices vecinos con su arco correspondiente, la prioridad se hace en función del valor del arco , luego se toman los vertices en orden y se intenta realizar la conexión , si los vertices no se encuentran en el mismo arbol se conectan , si estan conectados por un arbol se omiten y si no estan conectados por un mismo arbol peor provienen de distinto se aplica el método de unión , y se continua con cada uno de ellos.
   Como se menciono previamente esto corresponde para el algortimo de Kruskal ,para el algoritmo de Prim veo en particular uns dificultad para implementarlo ya que de por sí se encuentra bastante optimizado y no veo forma de crear una implementación que **mejore al algoritmo**.