**Práctica 2: Conjuntos Disjuntos y Algoritmo de Kruskal. El Problema del Viajante**

Pablo Sánchez Redondo. David Kaack Sánchez

Pareja 03

**II. Algoritmo de Kruskal**

Cuestiones:

Dado un grafo con n nodos y m aristas Kruskal realiza dos tareas, primero crea una cola de prioridad, que tiene coste O(m log m), dado que coge cada arista y la inserta según su longitud, comparando con cada una de las aristas previamente.

Una vez creada la cola de prioridad, hacemos dos finds y posiblemente una union con cada una de las aristas O(m). Y sobre cada una de ellas hace una operación de union-find, que tiene coste O(log n). El coste de esto queda como O(m log n).

En el peor de los casos, un grafo completo, m = n2. Con lo que el tiempo de creación de la cola de prioridad es:

O(n2 log n2) = O(2n2 log n) = O(n2 log n) = O(m log n)

Ambas partes tienen la misma complejidad temporal. Normalizando los tiempos observamos que esto es así.A graph with numbers and a line

Description automatically generated

**III. El problema del viajante de comercio**

1. 1.

Codicioso: O(n²\*log(n)): Pasamos todas las ciudades: O(n) y en cada ciudad medimos las distancias hacía las otras ciudades. Como las opciones disminuyen con cada ciudad pasada el coste es O(n\*log(n)). Añadidos los dos costes: O(n)+O(n\*log(n)) = O(n²\*log(n))

Exhaustivo: O(n!): Crea todas las permutaciones (o circuitos) de las ciudades que son n! en total. Después compara cada circuito que también es O(n!). Dadas las reglas de la notación Big-O nos quedamos con O(n!) + O(n!) = O(n!)

Repeated greedy: Esta función aplica el algoritmo codicioso con coste O(n²\*log(n)) a todos los n nodos que sería O(n²\*log(n)) \* O(n). Luego compara todos los circuitos en los n nodos: O(n). En total es, teniendo en cuenta las reglas de Big-O, O(n3\*log(n)) + O(n) = O(n3\*log(n)).

La solución greedy no es óptima para un grafo:

A black and white diagram with circles and lines

Description automatically generated matriz de distancias:A number and numbers on a white background

Description automatically generated

Solución greedy desde el nodo 0:

Nodos: 0---1---3---2 | Distancias: 0 + 1 + 4 + 6 = 11

Una solución óptima es:

Nodos: 3---1---0---2 | Distancias: 0 + 4 + 1 + 3 = 8