**OBSERVACIONES DEL LA PRACTICA**

Valentina Uribe 201817485

Gregorio Salazar 202022085

# **Preguntas de análisis**

1. ¿Qué instrucción se usa para cambiar el límite de recursión de Python?

sys.setrecursionlimit(2 \*\* 20)

en la vista del laboratorio está esta línea de código. Esto le da la instrucción a Python de poner el límite de recursión a 2^20=1048576.

1. ¿Por qué considera que se debe hacer este cambio?

Los algoritmos utilizados como dijkstra y kosaraju utilizan como base algoritmos recursivos como el DFS. Entonces se harán muchos llamados recursivos y si el límite es muy bajo marcará error. (recursion Depth exceded)

1. ¿Cuál es el valor inicial que tiene Python cómo límite de recursión?

1000.

1. ¿Qué relación creen que existe entre el número de vértices, arcos y el tiempo que toma la operación 4?

def minimumCostPaths(analyzer, initialStation):

    """

    Calcula los caminos de costo mínimo desde la estacion initialStation

    a todos los demas vertices del grafo

    """

    analyzer['paths'] = djk.Dijkstra(analyzer['connections'], initialStation)

    return analyzer

La operación 4 usa solamente el algoritmo de Dijkstra. Por lo visto en clase, su complejidad temporal es O( E logV), donde E es el número de arcos y V el número de vértices.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Archivo | Numero vertices | Num arcos | Time op 4 (ms) | Time op 6 (ms) |
| 50 | 74 | 73 | 20.31 | 0.07 |
| 150 | 146 | 146 | 27.41 | 0.41 |
| 300 | 295 | 382 | 84.21 | 0.47 |
| 1000 | 984 | 1633 | 230.22 | 0.52 |
| 2000 | 1954 | 3560 | 830.43 | 0.79 |
| 3000 | 2922 | 5773 | 1370.81 | 0.62 |
| 7000 | 6829 | 15334 | 5470.32 | 0.94 |
| 10000 | 9767 | 22758 | 13350.62 | 0.93 |
| 14000 | 13535 | 32270 | 22464.43 | 1.2 |

Vemos que la opción 4 se comporta como esperado (linearitmicamente). La opción 6 por lo visto, no hay una complejidad clara, pero es mejor que lineal (es cerca a O(1)).

1. ¿Qué características tiene el grafo definido?

analyzer['connections'] = gr.newGraph(datastructure='ADJ\_LIST',

                                              directed=True,

                                              size=14000,

                                              comparefunction=compareStopIds)

El grafo utilizado es un grafo dirigido.

1. ¿Cuál es el tamaño inicial del grafo?

El tamaño inicial del grafo es 14000. Como la lista de adyacencia es un mapa con los vértices, este tamaño inicial se usa como en los casos de los mapas cuando se pone un número de elementos esperado para evitar hacer muchos rehash.

1. ¿Cuál es la Estructura de datos utilizada?

La estructura de datos utilizada es “ADJ\_LIST” que representa una lista de adyacencia.

1. ¿Cuál es la función de comparación utilizada?

def compareStopIds(stop, keyvaluestop):

    """

    Compara dos estaciones

    """

    stopcode = keyvaluestop['key']

    if (stop == stopcode):

        return 0

    elif (stop > stopcode):

        return 1

    else:

        return -1

La función de comparación es compareStopIds, que compara si dos vértices son iguales. Esto es importante para la función getEdge.