**OBSERVACIONES DEL LA PRACTICA**

Tomas Angel Gallon Cod 2020

Daniel Osorio Cárdenas Cod 202022996

# **Preguntas de análisis**

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

La instrucción utilizada es sys.setrecursionlimit(2\*\*20).

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

El límite de recursión debe modificarse porque al trabajar con grafos los algoritmos que se utilizan para trabajar sobre estos son algunos recursivos, y es muy probable que estos algoritmos recursivos de los grafos superen el límite de recursión preestablecido por Python, que es de tan solo 1000. Por lo tanto, si un algoritmo como por ejemplo el DFS se llama a si mismo más de 1000 veces el límite de Python pararía el programa, y no necesariamente porque esté mal el algoritmo sino porque el límite de recursión es ese.

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

El valor inicial que tiene Python como límite de recursión es de 1000.

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

La relacion que exite entre el numero de vertices, arcos y el tiempo que toma la operación 4, es que entre mas vertices alla, mas arcos puede tener un mismo vertice, aumentando asi el tiempo que toma en cargar toda la informacion para la “estacion base” que se ingresa como parametro en la opcion.

1. **¿Qué características tiene el grafo definido?**

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

                                      directed=*True*,

                                      size=14000,

                                      comparefunction=compareStopIds)

1. **¿Cuál es el tamaño inicial del grafo?**

El tamaño inicial del grafo es de 14000

1. **¿Cuál es la Estructura de datos utilizada?**

La estructura de datos utilizada es un grafo implementado como una lista de adyacencias.

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