Fundamentos de Algoritmos

Universidad Torcuato Di Tella

TP 2

4) **Se analiza complejidad algoritmica del programa localizar.py (ej2):**

```
if __name__ == '__main__':
    try:
        FILENAME = FOLDER + sys.argv[1]                                    # O(1)
        LATITUD = float(sys.argv[2])                                       # O(1)
        LONGITUD = float(sys.argv[3])                                      # O(1)
    except Exception as e:
        print('Los parametros <FILENAME> <LATITUD> <LONGITUD> no fueron ingresados correctamente. \n' +
str(e))                                                                    # O(1)
    dependencias = abrir_archivo(FILENAME)                                 #  O(2*N)
    print(localizar_dj(dependencias, LATITUD, LONGITUD))                   #  O(N)
```

El analisis de complejidad de las funciones abrir_archivo() y localizar_dj() se encuentra en el codigo fuente del trabajo.

La funcion abrir_archivo() realiza limpieza de header previa a la creacion de la lista final con objetos DependenciaJudicial() haciendo dos iteraciones sobre las dependencias. Estrictamente el orden es O(len(dependencias)+1 + len(dependencias)) ya que en la segunda iteración se remueve una línea (header). *Siendo len(departamentos) = M* y *len(dependencias judiciales) = N*

Contemplando el orden mas alto de complejidad, el orden es: **O(2*N)**

Se analiza complejidad algoritmica del programa departamentos_judiciales.py (ej3):

```
if __name__ == '__main__':
    try:
        SRC_FILENAME = FOLDER + sys.argv[1]                                        # O(1)
        DST_FILENAME = FOLDER + sys.argv[2]                                        # O(1)
    except Exception as e:
        print('Los parametros <SRC FILENAME> <DST LATITUD> no fueron ingresados correctamente. \n' + str(e))
                                                                                   # O(1)
    dependencias = abrir_archivo(SRC_FILENAME)                                     # O(2*N)
    dependencias_formateadas = formatear(dependencias)                 # O(len(departamentos)*len(dependencias))
    # Ordenamos departamentos (llaves del diccionario que devuelve funcion formatear)
    departamentos_ordenados = sorted(dependencias_formateadas.keys())              # O(M*(log(M))
    # Ordenamos las dependencias de todos los departamentos
    for departamento, dependencias in dependencias_formateadas.items():            # O(M*(N*log(N)))
        # Suponemos como peor caso deben ordenarse M deparamentos
        # bubble_sort(dependencias_formateadas[departamento])                      # O(N^2)
        dependencias_formateadas[departamento] = sorted(dependencias_formateadas[departamento])
                                                                                   # O(N*log(N))
    # Guardamos los datos en el formato y orden requerido
    guardar_dependencias(departamentos_ordenados, dependencias_formateadas, DST_FILENAME)# O(N*M)
```

El analisis de complejidad de las funciones utilizadas se encuentra en el codigo fuente.

Contemplando el orden mas alto de complejidad, el orden es: **O(M*(N*log(N)))**