ANALISIS DE COMPLEJIDAD
RETO 3

Juliana Sofia Ahumada Arcos - 201921471 - j.ahumadaa@uniandes.edu.co

Daniela Parra Martínez - 202013036 - d.parram2@uniandes.edu.co

REQUERIMIENTO 1

```python
# REQUERIMIENTO 1 (CONTAR LOS AVISTAMIENTOS EN UNA CIUDAD)
def AvistamientosCiudad(ciudad, catalog):
    datos = lt.newList('ARRAY_LIST')   #O(1)
    cuantos = 0                #O(1)
    valores = om.keySet(catalog['datetime']) #O(n)
    for i in lt.iterator(valores):     #O(n)
        fecha = om.get(catalog['datetime'], i) #O(nlogn)
        mapcity = me.getValue(fecha)['City'] #O(1)
        city = om.get(mapcity, ciudad)#O(nlogn)
        if city is not None:  #O(1)
            avist = me.getValue(city)['UFOS'] #O(1)
            avistamientos = lt.size(avist) #O(1)
            cuantos += avistamientos    #O(1)
            data = avist['first']['info']  #O(1)
            lt.addLast(datos, data)  #O(1)

    primeros_3 = lt.subList(datos, 1, 3)  #O(1)
    ultimos_3 = lt.subList(datos, lt.size(datos) - 2, 3)  #O(1)

    return cuantos, primeros_3['elements'], ultimos_3['elements']
```

Complejidad:

$n+n^*(n\log n)$

Justificación:

```python
valores = om.keySet(catalog['datetime']) #O(n)
for i in lt.iterator(valores):     #O(n)
    fecha = om.get(catalog['datetime'], i) #O(nlogn)
    mapcity = me.getValue(fecha)['City'] #O(1)
    city = om.get(mapcity, ciudad)#O(nlogn)
```

REQUERIMIENTO 2 - DANIELA PARRA MARTÍNEZ

```python
# REQUERIMIENTO 2 (CONTAR LOS AVISTAMIENTOS POR DURACIÓN)
def avistamientosRangosec(S_min, S_max, catalog):
    maxima_d = om.maxKey(catalog['Duration']) #O(1)
    datos = lt.newList('ARRAY_LIST') #O(1)
    rango = om.values(catalog['Duration'], float(S_min), float(S_max)) #O(n)
    for i in lt.iterator(rango):#O(n)
        valores = om.valueSet(i['City']) #O(n)
        for j in lt.iterator(valores): #O(n)
            for data in lt.iterator(j['UFOS']):#O(n)
                lt.addLast(datos, data)#O(1)

    cuantos = lt.size(datos) #O(1)

    primeros_3 = lt.subList(datos, 1, 3) #O(1)
    ultimos_3 = lt.subList(datos, lt.size(datos) - 2, 3) #O(1)

    return cuantos, primeros_3['elements'], ultimos_3['elements'], maxima_d
```

Haga clic para agregar texto

Complejidad:

O(n**3)

Justificación:

```python
for i in lt.iterator(rango):#O(n)
    valores = om.valueSet(i['City']) #O(n)
    for j in lt.iterator(valores): #O(n)
        for data in lt.iterator(j['UFOS']):#O(n)
            lt.addLast(datos, data)#O(1)
```

REQUERIMIENTO 3 - JULIANA SOFIA AHUMADA ARCOS

```python
# REQUERIMIENTO 3 (CONTAR LOS AVISTAMIENTOS POR HORA/MINUTOS DEL DÍA)
def AvistamientosPorHora(H_I, H_FN, catalog):
    maxima_h = om.maxKey(catalog['time']) #O(1)
    H_I = datetime.datetime.strptime(H_I, '%H:%M:%S') #O(1)
    H_FN = datetime.datetime.strptime(H_FN, '%H:%M:%S') #O(1)
    datos = lt.newList('ARRAY_LIST') #O(1)
    rango = om.values(catalog['time'], H_I.time(), H_FN.time()) #O(n)
    for i in lt.iterator(rango): #O(n)
        valores = om.valueSet(i['Date']) #O(n)
        for j in lt.iterator(valores): #O(n)
            for data in lt.iterator(j['UFOS']): #O(n)
                lt.addLast(datos, data) #O(1)

    cuantos = lt.size(datos) #O(1)

    primeros_3 = lt.subList(datos, 1, 3) #O(1)
    ultimos_3 = lt.subList(datos, lt.size(datos) - 2, 3) #O(1)

    return cuantos, primeros_3['elements'], ultimos_3['elements'], maxima_h
```

Complejidad:

O(n**3)

Justificación:

```python
for i in lt.iterator(rango): #O(n)
    valores = om.valueSet(i['Date']) #O(n)
    for j in lt.iterator(valores): #O(n)
        for data in lt.iterator(j['UFOS']): #O(n)
            lt.addLast(datos, data) #O(1)
```

REQUERIMIENTO 4

```python
# REQUERIMIENTO 4 (CONTAR LOS AVISTAMIENTOS EN UN RANGO DE FECHAS)
def AvistamientosRangoFechas(F_I, F_FN, catalog):
    F_I = datetime.datetime.strptime(F_I, '%Y-%m-%d') #O(1)
    F_FN = datetime.datetime.strptime(F_FN, '%Y-%m-%d') #O(1)
    datos = lt.newList('ARRAY_LIST') #O(1)
    cuantos = 0 #O(1)
    rango = om.values(catalog['datetime'], F_I.date(), F_FN.date()) #O(n)
    for avistamiento in lt.iterator(rango): #O(n)
        avist = avistamiento['UFOS'] #O(1)
        data = avist['first']['info'] #O(1)
        lt.addLast(datos, data) #O(1)
        cuantos += lt.size(avistamiento['UFOS']) #O(1)

    primeros_3 = lt.subList(datos, 1, 3) #O(1)
    ultimos_3 = lt.subList(datos, lt.size(datos) - 2, 3) #O(1)

    return cuantos, primeros_3['elements'], ultimos_3['elements']
```

Complejidad:

O(n+n)

Justificación:

```python
rango = om.values(catalog['datetime'], F_I.date(), F_FN.date()) #O(n)
for avistamiento in lt.iterator(rango): #O(n)
    avist = avistamiento['UFOS'] #O(1)
    data = avist['first']['info'] #O(1)
    lt.addLast(datos, data) #O(1)
    cuantos += lt.size(avistamiento['UFOS']) #O(1)
```

REQUERIMIENTO 5

```python
# REQUERIMIENTO 5 (CONTAR LOS AVISTAMIENTOS DE UNA ZONA GEOGRÁFICA)
def AvistamientosZona(L_I, L_FN, LT_I, LT_FN, catalog):
    datos = lt.newList('ARRAY_LIST') #O(1)
    rango = om.values(catalog['Latitud'], float(LT_I), float(LT_FN)) #O(n)
    for i in lt.iterator(rango): #O(n)
        Long = mp.keySet(i['Long']) #O(1)
        for j in lt.iterator(Long): #O(n)
            if j <= float(L_I) and j >= float(L_FN): #O(1)
                avista = mp.get(i['Long'], j) #O(1)
                avi = me.getValue(avista) #O(1)
                for data in lt.iterator(avi['UFOS']): #O(n)
                    lt.addLast(datos, data) #O(1)

    cuantos = lt.size(datos) #O(1)

    primeros_5 = lt.subList(datos, 1, 5) #O(1)
    ultimos_5 = lt.subList(datos, lt.size(datos) - 4, 5) #O(1)

    return cuantos, primeros_5['elements'], ultimos_5['elements'], primeros_5, ultimos_5
```

Complejidad:

$O(n+n^{**}3)$

Justificación:

```python
rango = om.values(catalog['Latitud'], float(LT_I), float(LT_FN)) #O(n)
for i in lt.iterator(rango): #O(n)
    Long = mp.keySet(i['Long']) #O(1)
    for j in lt.iterator(Long): #O(n)
        if j <= float(L_I) and j >= float(L_FN): #O(1)
            avista = mp.get(i['Long'], j) #O(1)
            avi = me.getValue(avista) #O(1)
            for data in lt.iterator(avi['UFOS']): #O(n)
                lt.addLast(datos, data) #O(1)
```

REQUERIMIENTO 6

```python
# REQUERIMIENTO 6 (VISUALIZAR LOS AVISTAMIENTOS DE UNA ZONA GEOGRÁFICA)
def AvistamientosGeo(L_I, L_FN, LT_I, LT_FN, catalog):
    avistamientos = AvistamientosZona(L_I, L_FN, LT_I, LT_FN, catalog) #O(1)
    primeros_5 = avistamientos[3] #O(1)
    ultimos_5 = avistamientos[4] #O(1)
    mapa = folium.Map()
    tooltip = '¡Click para ver las coordenadas!' #O(1)
    for i in lt.iterator(primeros_5): #O(n)
        coordenadas = ('latitud:', i['latitude']), ('longitud:', i['longitude']) #O(1)
        folium.Marker([i['latitude'], i['longitude']], tooltip=tooltip,
                      popup=coordenadas).add_to(mapa)
    for j in lt.iterator(ultimos_5): #O(n)
        coordenadas = ('latitud:', j['latitude']), ('longitud:', j['longitude']) #O(1)
        folium.Marker([j['latitude'], j['longitude']], tooltip=tooltip,
                      popup=coordenadas).add_to(mapa)
    mapa.save('Avistamientos en la zona.html')
```

Complejidad:

O(n)

Justificación:

La complejidad es o(n) por que estamos haciendo dos for en los cuales estamos recorriendo la lista de los primeros 5 y últimos 5 avistamientos. En realidad, seria o(2n) pero es como multiplicar el infinito **2 es por esto que la complejidad es o(n). Por otro lado, la librería Folium normalmente están muy optimizadas y tienden a ser constantes.

```python
for i in lt.iterator(primeros_5): #O(n)
    coordenadas = ('latitud:', i['latitude']), ('longitud:', i['longitude']) #O(1)
    folium.Marker([i['latitude'], i['longitude']], tooltip=tooltip,
                  popup=coordenadas).add_to(mapa)
for j in lt.iterator(ultimos_5): #O(n)
    coordenadas = ('latitud:', j['latitude']), ('longitud:', j['longitude']) #O(1)
    folium.Marker([j['latitude'], j['longitude']], tooltip=tooltip,
                  popup=coordenadas).add_to(mapa)
```