Documento de análisis | Reto 2

Integrantes:

Moisés Agudelo, 202113485, m.agudeloo@uniandes.edu.co
Sergio Franco, 202116614, s.francop@uniandes.edu.co

Requerimiento 1

Análisis de complejidad

```
años = mp.keySet(catalogo['years'])
```
O(n)

```
for c in lt.iterator(años):
   if int(c) >= int(año1) and int(c) <= int(año2):
       valor = mp.get(catalogo['years'],c)
       for i in lt.iterator(me.getValue(valor)['books']):
          lt.addLast(nueva,i)
```
O(n^2)

```
orden = sortnacidos(nueva)
```
O(nlongn)

O(n + n^2+ nlogn)
O(n^2)

Pruebas de rendimiento

Entradas	1800 - 1900	1900 - 1920	1945 - 1980	Promedio
Maquina 1	29.28	40.23	28.63	32.71
Maquina 2	93.75	93.75	171.88	119.79

Comparación de los promedios de tiempo de ejecución

Reto	Reto 1	Reto 2
Maquina 1	20.83	32.71
Maquina 2	36.46	119.79

Comparación de complejidades

Para el primer requerimiento, en el reto 1, dio una complejidad de O(n). Luego, para el reto 2 salió una complejidad de O(n^2). Para este caso, la complejidad del segundo reto salió mucho más compleja que la del primer reto. Esto se puede dar por la organización de los datos, ya que al tener una estructura de mapas buscar un rango en las llaves y sacar los valores se vuelven más complejo. Puesto que, para la función de búsqueda del rango, para el primer reto y su orden eran mucho más sencillos de manejar. Sin embargo, el hecho de poder tener una organización de los datos por medio de llaves tiende a ser mejor. Realmente, la complejidad se aumenta al querer ordenar los datos, ya que se usan los ordenamientos del Tad Lista.

Requerimiento 2

Análisis de complejidad

```
    llaves = (mp.keySet(catalog['AdquisicionFecha']))
```

O(n)

```
          for c in lt.iterator(llaves):
        if c >= fecha1 and c <= fecha2:
            valor = mp.get(catalog['AdquisicionFecha'],c)
            lista = me.getValue(valor)['books']
            for j in lt.iterator(lista):
                lt.addLast(nueva,j)
```

O(n^2)

```
       orden = sortobras(nueva)
```

O(nlogn)

O(n + n^2+ nlogn)
O(n^2)

Pruebas de rendimiento

Entradas	1900/01/01	1929/11/19	1950/12/01	Promedio
Maquina 1	5078.69	4698.36	6125.69	5300.91
Maquina 2	20859.98	1178.5	10500	10845.89

Comparación de los promedios de tiempo de ejecución

Reto	Reto 1	Reto 2
Maquina 1	2656.28	5300.91
Maquina 2	5567.71	10845.69

Comparación de complejidades

Para el segundo requerimiento, en el reto 1, dio una complejidad de O(n$\log n$). Luego, para el reto 2, dio una complejidad de O(n^2). En este caso, la complejidad del reto 2, para este requerimiento, es mucho mayor. Puesto que, es recorrer casi los datos nuevamente. Sin embargo, el tiempo tiende a ser menor en el reto 2 en comparación al reto 1. La complejidad en este caso se aumenta, ya que al lograr ordenar las obras se tiende a realizar algo más. Esto de más, es el recorrido para poder ordenar los valores internos en las fechas usadas. Pasa algo parecido a lo mencionado en el párrafo anterior, que, al tener una organización de mapas, para buscar un rango, es mucho mayor que en las listas.

Requerimiento 3 | Sergio Franco

Análisis de complejida

```
 valores = mp.get(catalog['Nombres_Artistas'],Artista)
```
O(n)

```
valores_especificos = mp.get(catalog['Codigos_Artistas'],me.getValue(valores))
```
O(n)

```
for i in lt.iterator(tecnicas_final):
        posauthor = lt.isPresent(cantidad_de_tecnicas_veces, i)
        if posauthor > 0:
            artista = lt.getElement(cantidad_de_tecnicas_veces, posauthor)
            artista['Cantidad'] += 1
else:
            artista = newTecnica(i)
            artista['Cantidad'] = 1
            lt.addLast(cantidad_de_tecnicas_veces, artista)
```
O(n)

```
    for p in lt.iterator(cantidad_de_tecnicas_veces):
        if int(p['Cantidad']) > k:
            k = p['Cantidad']
            maximo = p['Tecnica']
```
O(n)

```
orden = merge.sort(catalog, comparecanitdad)
```
O(nlogn)

```
    for obra in lt.iterator(me.getValue(valores_especificos)['obras']):
        if obra['Medium'] == tecnicas[0]:
            lt.addLast(obras,obra)
```
O(n)

O(n + n + n + n + nlogn + n) => <u>O(nlogn)</u>

Pruebas de rendimiento

Entradas	Libero badil	Chip Lord	Vladimir Blur	Promedio
Maquina 1	0	0	0	0
Maquina 2	0	0	15.63	5.21

Comparación de los promedios de tiempo de ejecución

Reto	Reto 1	Reto 2
Maquina 1	20.83	0
Maquina 2	36.46	5.21

Comparación de complejidades

Para el tercer requerimiento, en el reto1, dio una complejidad de O(nlogn). Para el reto 2, dio una complejidad de O(nlogn). En este caso, se denota que la búsqueda de los autores, tiende a ser similar. Puesto que, los recorridos tienden a ser iguales para lograr extraer su información. Para poder lograr extraer su información, es muy similar, ya que no toca entrar dos veces a la organización si ambos están relacionados de la misma manera. Realmente, el uso de Tad lista y mapas en este requerimiento es muy similar, y su implementación tiende a ser muy similar.

Requerimiento 4 | Moisés Agudelo

Análisis de complejidad

```
    for obra in lt.iterator(libros):
        codigo_obra = obra["ObjectID"]
        lt_codigos = lt.newList('ARRAY_LIST')

        obras = obra['ConstituentID'].replace("[","")
        obras = obras.replace("]","")
        obras = obras.split(",")
        for c in obras:
            codigo = c.strip()
            nacionalidad = me.getValue(mp.get(artistas, codigo))
            if nacionalidad == "" or nacionalidad == "Nationality unknown":
                nacionalidad = "Unknown"

            estado = mp.contains(catalogo["Orden_naciones"], nacionalidad)

            if estado:
                llave_valor = mp.get(catalogo["Orden_naciones"], nacionalidad)
                suma = (me.getValue(llave_valor) + 1)
                me.setValue(llave_valor, suma)

                llave_valor2 = mp.get(catalogo["Codigos_orden"], nacionalidad)
                lista = me.getValue(llave_valor2)
                lt.addLast(lista, codigo_obra)
                me.setValue(llave_valor2, lista)

            else:
                lt.addLast(lt_codigos, codigo_obra)
                mp.put(catalogo["Orden_naciones"], nacionalidad, 1)
                mp.put(catalogo["Codigos_orden"], nacionalidad, lt_codigos)
                lt.addLast(lt_naciones, nacionalidad)
```

O(n^2)

```
for pais in lt.iterator(lt_naciones):
    llave_valor1 = mp.get(catalogo["Orden_naciones"], pais)
    cantidad = me.getValue(llave_valor1)
    mp.put(catalogo["Naciones_cantidad"], cantidad, pais)
```

O(n)

```
llaves = mp.keySet(catalogo["Naciones_cantidad"])
```

O(n)

```
orden = merge.sort(lista, comppaises)
```

O(nlogn)

O(n^2 + n + n + nlogn)
O(n^2 + 2n + nlogn)
O(n^2 + n + nlogn)
O(n^2)

Pruebas de rendimiento

Entradas	Obras totales	Obras totales	Obras totales	Promedio
Maquina 1	3531.5	3578.13	3578.13	3562.58
Maquina 2	4125	4093.75	4078.13	4098.96

Comparación de los promedios de tiempo de ejecución

Reto	Reto 1	Reto 2
Maquina 1	1601986.83	3562.58
Maquina 2	1759276.88	4098.96

Comparación de complejidades

Para el cuarto requerimiento, en el reto1, dio una complejidad de O(n^3). Luego, para el reto 2, dio una complejidad de O(n^2). En este caso, se ve un cambio muy significativo ya que no se hace un triple for, buscando 1 solo resultado, ahora en el reto 2 es mucho más dinámico y rápido. Los recorridos tienden a ser más cortos y precisos gracias a la combinación de Tad lista y Mapas, que en el reto 1 se hizo con diccionarios violando el uso único de Tad listas en ese entonces, logrando un cambio muy significativo.

equerimiento 5

Análisis de complejidad

```
total_departamento = mp.get(catalog['Departamento'],departamento)
```
O(n)

```
obras_artista = me.getValue(total_departamento)['books']
```
O(n)

```
for obra in lt.iterator(catalog):

     peso = obra['Weight (kg)']
     altura = obra['Height (cm)']
     ancho = obra['Width (cm)']
     profundidad = obra['Depth (cm)']
     longitud = obra['Length (cm)']
     diametro = obra['Diameter (cm)']
```
O(n)

```
orden = merge.sort(catalog, comparacostos)
```
O(nlogn)

```
 for p in lt.iterator(catalog):
     suma += p['costo']
```
O(n)

```
    for p in lt.iterator(catalog):
        suma += float(p['peso'])
```
O(n)

```
    for obra in lt.iterator(ordenadas):
        if obra['fecha'] != '':
            lt.addLast(con_fecha, obra)
```
O(n)

O(n + n + n + n + n + n + nlogn)
O(6n + nlogn)
O(n + nlogn)
O(nlogn)

Pruebas de rendimiento

Entradas	Drawings	Photography	Painting	Promedio
Maquina 1	8069.85	3150.69	328.16	3849.56
Maquina 2	11343.75	4125	437.5	5302,08

Comparación de los promedios de tiempo de ejecución

Reto	Reto 1	Reto 2
Maquina 1	3968.75	3849.56
Maquina 2	8718.75	5302.08

Comparación de complejidades

Para el quinto requerimiento, en el reto 1, dio una complejidad de O(nlogn). Luego, para el reto 2 salió una complejidad de O(nlogn). En este caso las complejidades son iguales, como en el requerimiento 3. Para este caso, se facilitó por un lado la organización de las obras. Pero, el cálculo de su precio es muy similar. También, la complejidad tiende

a ser igual, ya que se usan las Tad lista como valores principales. Realmente, cualquiera de las dos estructuras es muy similares. En las complejidades resulta, interesante que la búsqueda de elementos se asimilar. Puesto que, los mapas tienden a ser mejores en ese caso. Sin embargo, en este caso su implementación fue muy similar, en el sentido, de su complejidad.