Universidad de los Andes
Facultad de Ingeniería
Reto 2- Curando y Explorando el MoMA: RESURRECTION
Estructuras de datos y algoritmos – sección 02

Estudiante 1:

Nombre: Sofia Torres Ramírez.
Código: 2020140872
Correo uniandes: s.torres21@uniandes.edu.co

Estudiante 2:

Nombre: Ana Margarita Flórez Ruiz.
Código: 201922242
Correo uniandes: a.florezr@uniandes.edu.co

Análisis de complejidad de cada requerimiento

REQ. 1: Listar cronológicamente los artistas (Grupal)

Este requerimiento tiene complejidad O(n) ya que este requerimiento cuenta con un For que recorre todos los años (llaves) del mapa con el que se cuenta para así poder agregar los artistas a una nueva lista en la que se puedan ordenar y obtener un orden cronológico de los artistas.

```
def getArtistByRange(catalog, initialDate,finalDate):
    """
    Retorna artistas en un rango de años
    """
    artistInRange =lt.newList("ARRAY_LIST")

    keys= mp.keySet(catalog["YearArtist"])

    for año in lt.iterator(keys):
      if int(año) >= int(initialDate) and int(año) <= int(finalDate):
        entry= mp.get (catalog["YearArtist"],año)
        valor= me.getValue(entry)
        artist= valor["Artist"]
        for artista in lt.iterator(artist):
          lt.addLast(artistInRange, artista)
    sortedlist=sortyear(artistInRange)

    return sortedlist
```

REQ. 2: Listar cronológicamente las adquisiciones (Grupal)

La complejidad de este algoritmo es O(n) debido a que para recorrer la lista que contiene las llaves de las fechas que están en el índice "yearartworks" es necesario crear un ciclo el cual recorre todos los datos, a pesar de que dentro de ese ciclo hay otro ciclo, este ciclo no se va a contar dentro del orden de complejidad debido a que este ciclo es mucho más pequeño debido a que no pasa por todos los datos.

```
#REQ. 2: listar cronológicamente las adquisiciones
def addartworkyear(catalog, date1,date2):
    date1=dt.date.fromisoformat(date1)
    date2=dt.date.fromisoformat(date2)
    artworksinrange=lt.newList("ARRAY_LIST")
    yearcatalog=catalog["yearartworks"]
    listtemp=mp.keySet(yearcatalog)
    sortlist=mg.sort(listtemp, compareadcquireddate)
    for date in lt.iterator(sortlist):
        datestr=date
        date=dt.date.fromisoformat(date)
        if date >= date1 and date <= date2:
            temp=mp.get(yearcatalog,datestr)
            temp=me.getValue(temp)
            for artwork in lt.iterator(temp):
                lt.addLast(artworksinrange,artwork)
    return artworksinrange
```

REQ. 3: Clasificar las obras de un artista por técnica (Individual)

Realizado por Sofía Torres.

La complejidad de este algoritmo es O(n) debido a que es necesario recorrer todas las obras de la lista de obras de un artista con un ciclo para poder responder correctamente al requerimiento.

```python
# Total de obras
def totalartworksartist (catalog, name):
    catalogname= catalog["Nameartist"]
    artistindex= mp.get(catalogname,name)
    artworksartist= me.getValue (artistindex)
    return artworksartist

#Total técnicas (medios) utilizados
def totalmediums(artworksartist):
    techniques=lt.newList("ARRAY_LIST", cmpfunction=cmpmediums)
    for artwork in lt.iterator(artworksartist):
      technique=artwork["Medium"]
      position=lt.isPresent(techniques,technique)
      if position>0:
          tec=lt.getElement(techniques, position)
          tec["value"]+=1
      else:
          tec={"Name":technique,"value":1}
          lt.addLast(techniques, tec)
    sortedlist=sorttecnicas(techniques)
    return sortedlist
```

REQ. 4: Clasificar las obras por la nacionalidad de sus creadores (Individual)

Realizado por Margarita Flórez.

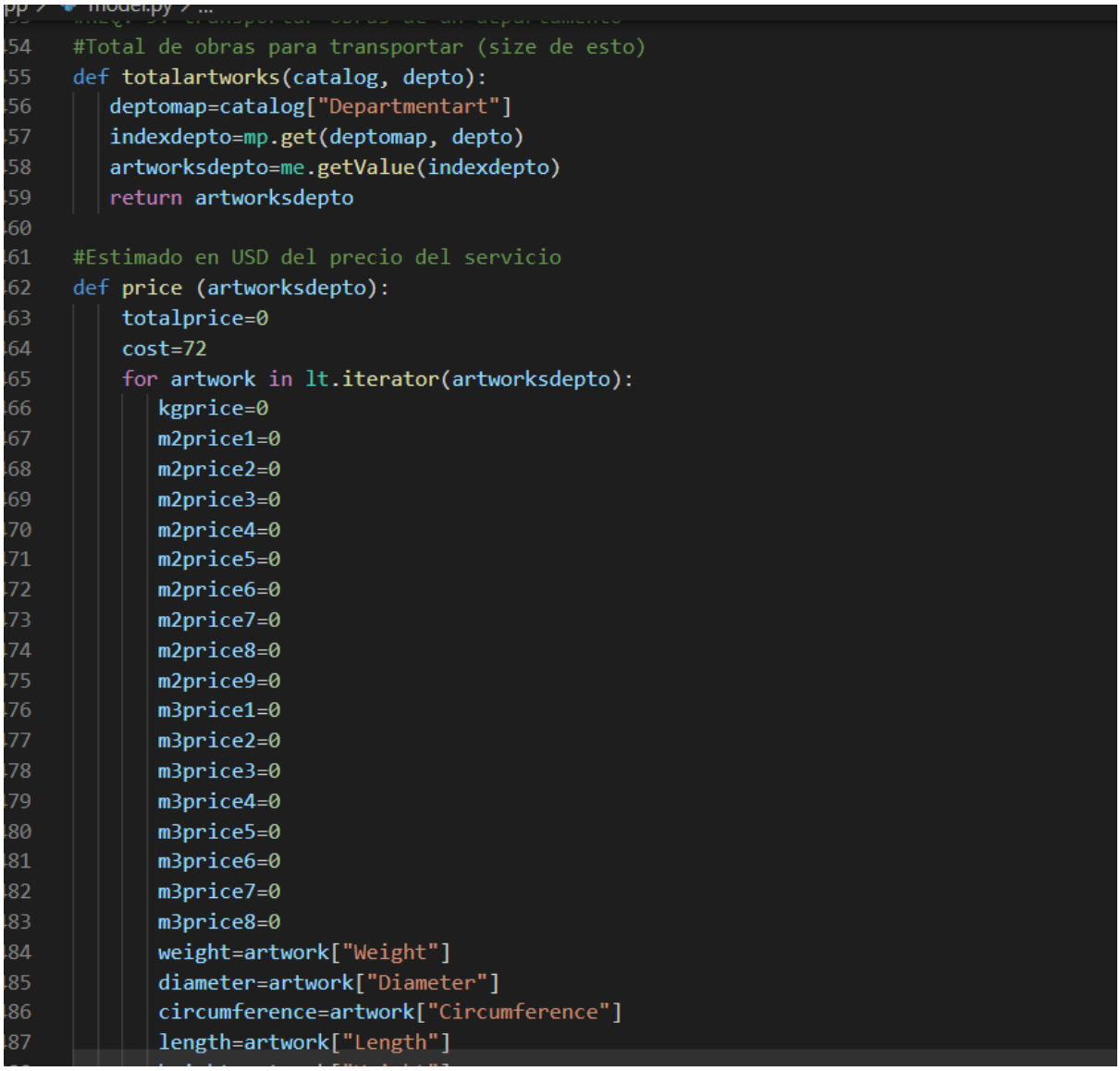
Este requerimiento cuenta con una complejidad de O(n) ya que solo cuenta con un recorrido For en el que se recorren las nacionalidades (llaves del map) para poder encontrar el total de obras de cada nacionalidad y así poder agregar este dato en una nueva lista y esta lista será ordenada con otra función para así poder obtener la respuesta de las nacionalidades ordenadas de mayor a menor.

```
#REQ. 4:clasificar las obras por la nacionalidad de sus creadores

def totalObras(catalog, nationality):
    nationalityIndex= mp.get(catalog ["Nationality"], nationality)
    artworksNationality= me.getValue(nationalityIndex)
    return artworksNationality

def clasifyByNationality(catalog):
    obras=lt.newList("ARRAY_LIST")
    keys= mp.keySet(catalog ["Nationality"])
    for nacionalidad in lt.iterator(keys):
        lista=totalObras(catalog, nacionalidad)
        lt.addLast(obras, f(nacionalidad,lista))
    sortedlist=sortObras(obras)
    return sortedlist

def sortObras(obras):
    sortedlist=mg.sort(obras, comparacionObras)
    return sortedlist

def f(nacionalidad,lista):
    return {"nacionalidad":nacionalidad,"obras":lista}

def comparacionObras(obra1, obra2):
    return lt.size(obra1["obras"])<= lt.size(obra2["obras"])
```

REQ. 5: transportar obras de un departamento (Grupal)

La complejidad de este algoritmo es O(n) debido a que se hace necesario realizar varias operaciones por cada obra para poder hallar el mayor precio de transporte para cada obra, por lo tanto, es necesario hacer uso de un ciclo para recorrer todas las obras de cada departamento.

Universidad de los Andes
Facultad de Ingeniería
Reto 2- Curando y Explorando el MoMA: RESURRECTION
Estructuras de datos y algoritmos – sección 02



```
#Total de obras para transportar (size de esto)
def totalartworks(catalog, depto):
    deptomap=catalog["Departmentart"]
    indexdepto=mp.get(deptomap, depto)
    artworksdepto=me.getValue(indexdepto)
    return artworksdepto

#Estimado en USD del precio del servicio
def price (artworksdepto):
    totalprice=0
    cost=72
    for artwork in lt.iterator(artworksdepto):
        kgprice=0
        m2price1=0
        m2price2=0
        m2price3=0
        m2price4=0
        m2price5=0
        m2price6=0
        m2price7=0
        m2price8=0
        m2price9=0
        m3price1=0
        m3price2=0
        m3price3=0
        m3price4=0
        m3price5=0
        m3price6=0
        m3price7=0
        m3price8=0
        weight=artwork["Weight"]
        diameter=artwork["Diameter"]
        circumference=artwork["Circumference"]
        length=artwork["Length"]
```

```
height=artwork["Height"]
width=artwork["Width"]
depth=artwork["Depth"]
if weight !="" :
  kgprice=float(weight)*cost
if height!= "" and width!= "":
  m2price1=(float(height)/100)*(float(width)/100)*cost
if height!= "" and length!= "":
  m2price2=(float(height)/100)*(float(length)/100)*cost
if height!= "" and depth!= "":
  m2price3=(float(height)/100)*(float(depth)/100)*cost
if length!= "" and width != "":
  m2price4= (float(length)/100)*(float(width)/100)*cost
if depth!="" and width!="":
  m2price5= (float(depth)/100)*(float(width)/100)*cost
if length!="" and depth!="":
  m2price6= (float(length)/100)*(float(depth)/100)*cost
if diameter !="":
  m2price8=3.1416*(((float(diameter)/2)/100)**2)*cost
if circumference!="":
  m2price9= 3.1416*(((float(circumference)/100)/(2*3.1416))**2)*cost

if length!="" and width != "" and depth != "":
  m3price1=(float(length)/100)*(float(width)/100)*(float(depth)/100)*cost
if height!="" and width!= "" and depth!= "":
  m3price2=(float(height)/100)*(float(width)/100)*(float(depth)/100)*cost
if height!="" and width!= "" and length!= "":
  m3price3=(float(height)/100)*(float(width)/100)*(float(length)/100)*cost
if length!="" and depth!= "" and height!= "":
  m3price4=(float(length)/100)*(float(depth)/100)*(float(height)/100)*cost
if height!="" and diameter!= "":
  m3price5=3.1416*((((float(diameter))/2)/100)**2)*(float(height)/100)*cost
if width!="" and diameter!="":
  m3price6=3.1416*((((float(diameter))/2)/100)**2)*(float(width)/100)*cost
```

```
        if length!="" and width != "" and depth != "":
            m3price1=(float(length)/100)*(float(width)/100)*(float(depth)/100)*cost
        if height!="" and width!= "" and depth!= "":
            m3price2=(float(height)/100)*(float(width)/100)*(float(depth)/100)*cost
        if height!="" and width!= "" and length!= "":
            m3price3=(float(height)/100)*(float(width)/100)*(float(length)/100)*cost
        if length!="" and depth!= "" and height!= "":
            m3price4=(float(length)/100)*(float(depth)/100)*(float(height)/100)*cost
        if height!="" and diameter!= "":
            m3price5=3.1416*((((float(diameter))/2)/100)**2)*(float(height)/100)*cost
        if width!="" and diameter!="":
            m3price6=3.1416*((((float(diameter))/2)/100)**2)*(float(width)/100)*cost
        if depth!="" and diameter!="":
            m3price7=3.1416*((((float(diameter))/2)/100)**2)*(float(depth)/100)*cost
        if length!="" and diameter!="":
            m3price8=3.1416*((((float(diameter))/2)/100)**2)*(float(length)/100)*cost
        lastprice= max(kgprice,m2price1,m2price2,m2price3,m2price4,m2price5,m2price6,m2pric
        if lastprice==0:
            lastprice=48
        artwork["Price"]=lastprice
        totalprice+=lastprice
    return (totalprice, artworksdepto)
```

Universidad de los Andes
Facultad de Ingeniería
Reto 2- Curando y Explorando el MoMA: RESURRECTION
Estructuras de datos y algoritmos – sección 02

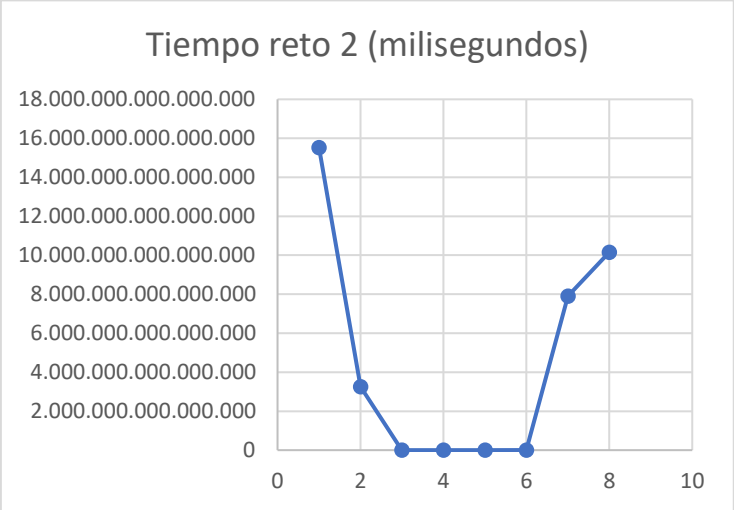
Pruebas de tiempos de ejecución y memoria utilizadas para cada uno de los requerimientos.

Las pruebas de tiempos de ejecución se realizaron es un equipo con las siguientes características:

Características	Máquina
Procesadores	Intel(R) Core (TM) i7-10510U CPU @ 1.80GHz 2.30 GHz
Memoria RAM (GB)	16,0 GB
Sistema Operativo	Sistema operativo de 64 bits, procesador basado en x64

REQ. 0: Carga de datos.

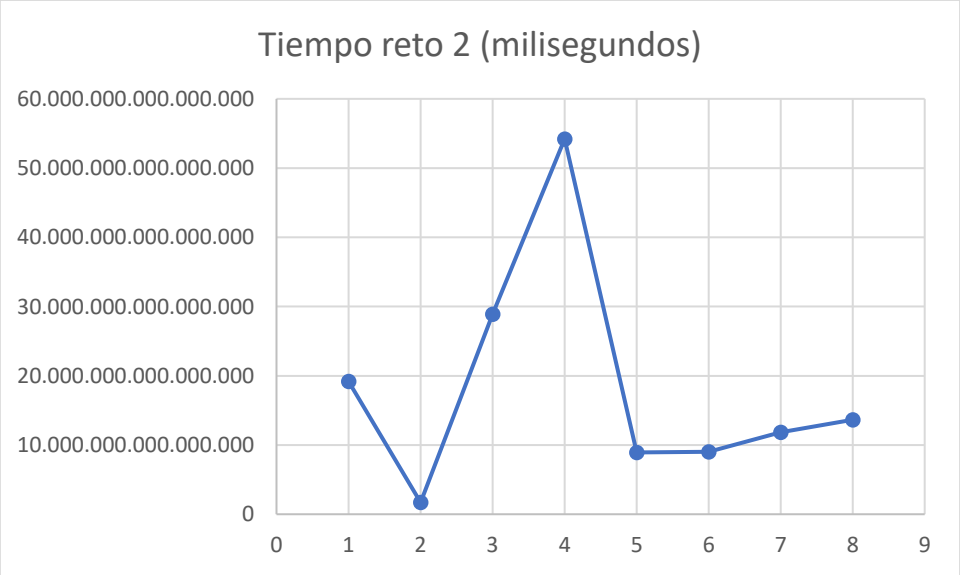
Tamaño del archivo	Tiempo reto 2 (milisegundos)	Memoria utilizada (GB) Reto 2
Small	15527.329000000002	6
5%	326403.7949999999	7.5
10%	826602.505	7.7
20%	1844027.476	8.12
30%	3002268.14	8.13
50%	6281687.612	8.76
80%	7904528.078000001	9.14
Large	10156295.242999999	9.9



Universidad de los Andes
Facultad de Ingeniería
Reto 2- Curando y Explorando el MoMA: RESURRECTION
Estructuras de datos y algoritmos – sección 02

REQ. 1: Listar cronológicamente los artistas (Grupal)

Tamaño del archivo	Tiempo reto 2 (milisegundos)	Memoria utilizada (GB) Reto 2
Small	1917.7840000000047	2.9
5%	17082.42400000003	3.3
10%	28900.681000000077	3.4
20%	54210.668000000056	3.6
30%	89167.47299999997	4.1
50%	90437.87900000098	4.9
80%	118613.69699999955	5.2
Large	136312.48300000152	5.4



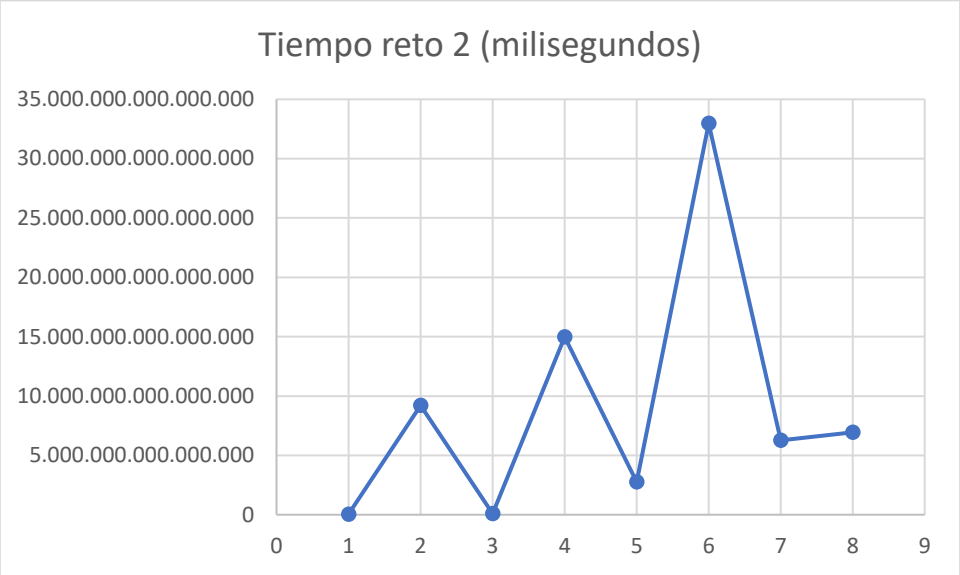
REQ. 2: Listar cronológicamente las adquisiciones (Grupal)

Tamaño del archivo	Tiempo reto 2 (milisegundos)	Memoria utilizada Reto 2
Small	948.0150000000265	2.1
5%	2896.3939999999866	2.6
10%	4207.733000000076	3
20%	5857.808000000091	3.3
30%	6500.283000000309	3.66
50%	7691.131999999925	3.8
80%	7912.1600000007675	4.2
Large	8851.556999998138	4.4

REQ. 3: Clasificar las obras de un artista por técnica (Individual)

Realizado por Sofía Torres.

Tamaño del archivo	Tiempo reto 2 (milisegundos)	Memoria utilizada (GB) Reto 2
Small	47.852000000006	1
5%	92.04600000003893	1.3
10%	103.287000000023	2
20%	149.91500000007818	2.6
30%	276.8049999997402	2.7
50%	329.61199999931523	3.3
80%	627.2579999995287	3.6
Large	695.3090000001794	3.8



REQ. 4: Clasificar las obras por la nacionalidad de sus creadores (Individual)

Realizado por Margarita Flórez.

Tamaño del archivo	Tiempo reto 2 (milisegundos)	Memoria utilizada (GB) Reto 2
Small	44.07599999999512	1.3
5%	32.20100000000059	1.7
10%	51.258000000018455	1.89
20%	1781787.229	2
30%	81.47300000018731	2.8
50%	109.90100000071834	3.11
80%	137.517000000571	3.47
Large	30.031000001145003	4

REQ. 5: transportar obras de un departamento (Grupal)

Tamaño del archivo	Tiempo reto 2 (milisegundos)	Memoria utilizada Reto 2
Small	1393.620999999996	4
5%	177315.01299999992	4.58
10%	656709.3799999998	4.99
20%	2578695.964	5.3
30%	5950472.221	5.9
50%	12197681.069999998	6.32
80%	30678770.609	7
Large	52567386.254	7.97

Universidad de los Andes
Facultad de Ingeniería
Reto 2- Curando y Explorando el MoMA: RESURRECTION
Estructuras de datos y algoritmos – sección 02

<u>Comparar la complejidad de los requerimientos implementados en el Reto No 1 con los implementados en este reto</u>

Req 1: Comparando la complejidad obtenida para este requerimiento en el Reto 1(O(n)) y en este reto (O(n)), se puede ver que se tuvo la misma complejidad teóricamente hablando en notación Big O, pero hay que tener en cuenta que si se usara otra notación como puede ser sigma se podría apreciar mejor la diferencia y en tiempos también se puede ver cómo cambia la función de una estructura de datos a otra.

Req 2: Comparando la complejidad obtenida para este requerimiento en el Reto 1 (O(n)) y en este reto (O(n)) se puede apreciar que se obtuvo teóricamente la misma complejidad, sin embargo, también es importante tener en cuenta que la complejidad de este requerimiento en el reto 2 la dio un ciclo en una solo función, mientras que en el reto 1 se utilizaron más ciclos en más funciones.

Req 3: Comparando la complejidad obtenida para este requerimiento en el Reto 1 (O(n)) y en este reto (O(n)), al igual que en el requerimiento pasado se puede apreciar que se obtuvo teóricamente la misma complejidad, sin embargo, también se puede apreciar que la complejidad de este requerimiento en el reto 2 la dieron dos ciclos en dos funciones, mientras que en el reto 1 se utilizaron más ciclos en más funciones.

Req 4: Comparando la complejidad obtenida en el requerimiento 4 en este reto O(n) y en el reto O(nˆ2) se puede notar una mejora extraordinaria ya que se pudo hacer con un recorrido menos y además con menos funciones la consulta, haciendo el trabajo de mejor manera, más eficiente y optima.

Req 5: Comparando la complejidad obtenida para este requerimiento en el Reto 1 (O(n)) y en este reto (O(n)), al igual que en los requerimientos pasados se puede apreciar que se obtuvo teóricamente la misma complejidad, sin embargo en el reto 1, para la creación de la lista base de las obras de un departamento se utilizó un ciclo extra, mientras que en este caso (al igual que en los casos anteriores de este reto) este proceso se facilita ya que al usar maps, para

buscar estas listas, en lugar de utilizar un ciclo el cual en la mayoría de ocasiones tiene una complejidad O(n), se utiliza en map con una complejidad de O(1).