Reto 1

Documento de análisis

Est2: Juan Felipe Serrano Martinez – 201921654 – j.serrano@uniandes.edu.co

Est1: Santiago Sinisterra Arias – 202022177 – s.sinisterra@uniandes.edu.co

Análisis de complejidad de código:

Requerimiento 1:

```
104  def getagerange(catalog, date1, date2):
105      artists = lt.iterator(catalog['Artists'])
106      cronartists = lt.newList("ARRAY_LIST")
107      for men in artists:
108          if (int(men['BeginDate']) >= int(date1) and int(men['BeginDate']) <= int(date2)):
109              lt.addLast(cronartists, men)
110      sa.sort(cronartists, compareages)
111      return cronartists
```

En este pedazo de código es evidente como la complejidad es de N + N + N, = 3N, comenzando con un N en la linea 105 por el iterator de artists en la lista de artistas, después por el for de la linea 107 presente en la función y el ordenamiento que se da con Shell sort en la linea 110, dando una complejidad de O(N).

Requerimiento 2:

```
113  def cronartwork(catalog, date1, date2):
114      tamano = lt.size(catalog["Artworks"])
115      sublist = lt.subList((catalog['Artworks']), 1, tamano)
116      sublist = sublist.copy()
117      c = lt.size(sublist) - 1
118      while c > -1:
119          element = lt.getElement(sublist, c)
120          if element['DateAcquired'] < date1 or element['DateAcquired'] > date2:
121              lt.deleteElement(sublist, c)
122          sa.sort(sublist, cmpArtworkByDateAcquired)
123          c = c-1
124      return sublist
```

En este pedazo de código es evidente que hay una complejidad de N*(N/2)^2, pues se comienza con la creación de una sublista en la línea 144 del tamaño de la lista que da una complejidad N, posteriormente el while de la línea 118 da una complejidad de N/2 al ir poco a poco disminuyendo el valor de N a través del while, y esto se multiplica por el sort de la línea 122 que se encuentra

dentro del while, dando una complejidad de (N/2) por la misma razón que el while. Resultando en una complejidad considerablemente alta de: O(N^3)

Requerimiento 3:

```
126 def getartwoksandtech(catalog, artist):
127     prod = lt.iterator(catalog['Productions'])
128     for element in prod:
129         if element['Artist'] == artist:
130             bruh = element
131     sa.sort(bruh['Artworks'], comparetechniques)
132 
133 
134     popularity = lt.newList("ARRAY_LIST")
135     comparator = lt.iterator(bruh['Artworks'])
136     art1 = None
137     art2 = None
138     n = 0
139     tech_num = 0
140     artlist = lt.newList("ARRAY_LIST")
141     for artwork in comparator:
142 
143         art1 = artwork['Medium']
144         if art1 != art2:
145             tech_num = tech_num + 1
146             if art2 != None:
147                 lt.addLast(popularity, dic)
148             n = 1
149             artlist = lt.newList("ARRAY_LIST")
150             lt.addLast(artlist, artwork)
151             dic = {"Medium": artwork['Medium'], 'Number': n, 'Artworks': artlist}
152             art2 = art1
153         else:
154             n = n + 1
155             art2 = art1
156             lt.addLast(artlist, artwork)
157             dic = {"Medium": artwork['Medium'], 'Number': n, 'Artworks': artlist}
158     if art1 == None:
159         tech_num = 0
160     elif art1 == art2:
161         lt.addLast(popularity, dic)
162 
163     sa.sort(popularity, comparetechniques2)
164 
165     return bruh, popularity, tech_num
```

En este requerimiento se evidencia un desarrollo más extenso, del cual se obtiene una complejidad de N + N + N + N + N + N = 6N. Esta es una complejidad relativamente buena considerando la función que se le asigna, pues se suma la complejidad de la iteración de la línea 127, el for de la línea 128, la iteración de la línea 135, el for de la línea 141, y finalmente el sort de la línea 163. Lo cual da una complejidad de O(N).

Requerimiento 4:

Requerimiento 5:

```
171 def getcostfordepa(catalog, departament):
172     indep = lt.newList('ARRAY_LIST')
173     inold = lt.newList('ARRAY_LIST')
174     dep = lt.iterator(catalog['Artworks'])
175     t_cost = 0.0
176     t_weight = 0.0
177     for depo in dep:
178         size = ''
179         cost = 0
180         if depo["Department"] == departament:
181 
182             if depo['Height (cm)'] != '' and depo['Width (cm)'] != '':
183                 size = float(depo['Height (cm)']) * float(depo['Width (cm)'])/10000
184 
185             if depo['Depth (cm)'] != '' and depo['Depth (cm)'] != '0':
186                 if depo['Diameter (cm)'] !='':
187                     size = float(depo['Depth (cm)']) * (float(depo['Diameter (cm)']))/1000000
188                 elif (depo['Height (cm)'] != '' and depo['Height (cm)'] != '0') and (depo['Width (cm)'] != '0' and depo['Width (cm)'] != ''):
189                     size = float(depo['Height (cm)']) * float(depo['Width (cm)']) * float(depo['Depth (cm)'])/1000000
190 
191             if depo['Circumference (cm)'] != '' and depo['Circumference (cm)'] != '0':
192                 size = ((float(depo['Circumference (cm)'])/2)**2)/3.14
193                 if depo['Diameter (cm)'] !='' and depo['Diameter (cm)'] !='0':
194                         size = float(depo['Circumference (cm)']) * float(depo['Diameter (cm)'])/10000
195                         if depo['Length (cm)'] != '' and depo['Length (cm)'] != '0':
196                             size = float(depo['Circumference (cm)']) * float(depo['Diameter (cm)']) * float(depo['Length (cm)'])/1000000
197             if size == '' or size == 0:
198                 cost = 48.00
199             elif (depo['Weight (kg)']) != '' and float(depo['Weight (kg)']) > size*72:
200                 size = float(depo['Weight (kg)'])
201                 print('bruh')
202 
203             if cost != 48.00:
204                 cost = size*72.00
205 
206             lt.addLast(indep, {'dep': depo, 'price': cost})
207             if depo['Date'] != '' and depo['Date'] != '0':
208                 lt.addLast(inold, {'dep': depo, 'Date': depo['Date']})
209             t_cost = t_cost + cost
210             if depo['Weight (kg)'] != '':
211                 t_weight = t_weight + float(depo['Weight (kg)'])
212         sa.sort(indep, comparecost)
213         sa.sort(inold, compareage)
214     return indep, inold, t_weight, t_cost
```

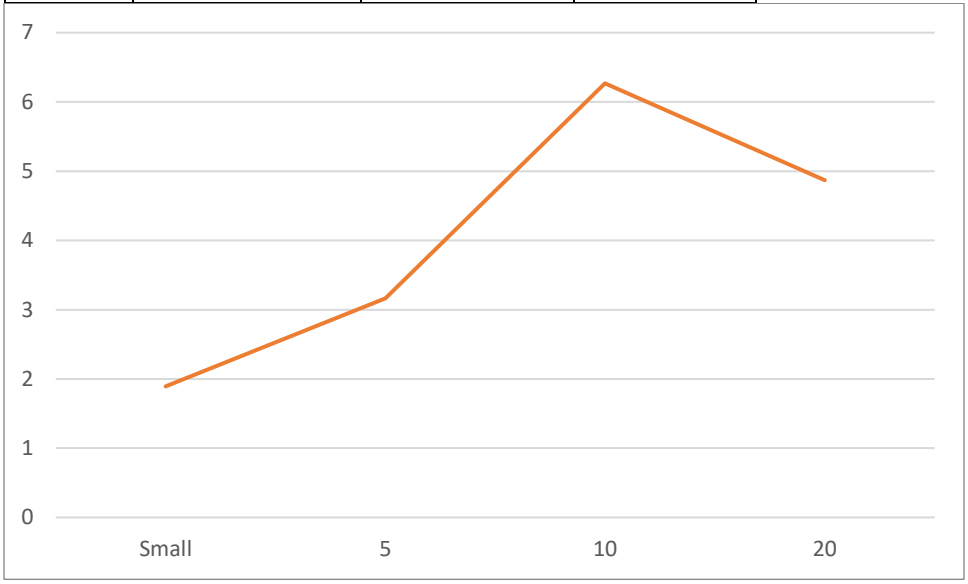
En este ultimo requerimiento se observa una complejidad de N + N + N + N = 4N. Esto se debe a que aunque es de gran tamaño la función, solo se encuentra una iteración en la línea 174 con complejidad N, un for de complejidad N en la línea 177, y par de sorts con complejidad N respectivamente en las líneas 212 y 213. Dando esto a una complejidad de O(N)

Prueba de tiempos de ejecución:

Requerimiento 1:

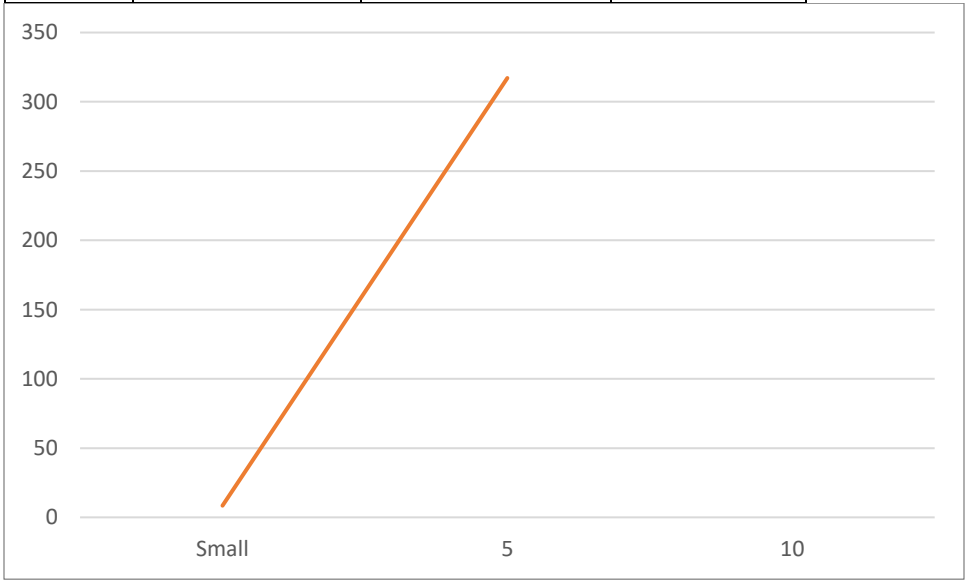
Req1	Tamaño_artistas	Tamaño_obras	Tiempo (ms)
Small	1948	768	1.893
5	4996	7572	3.165
10	6656	15008	6.268

20	8724	29489	4.869



Requerimiento 2:

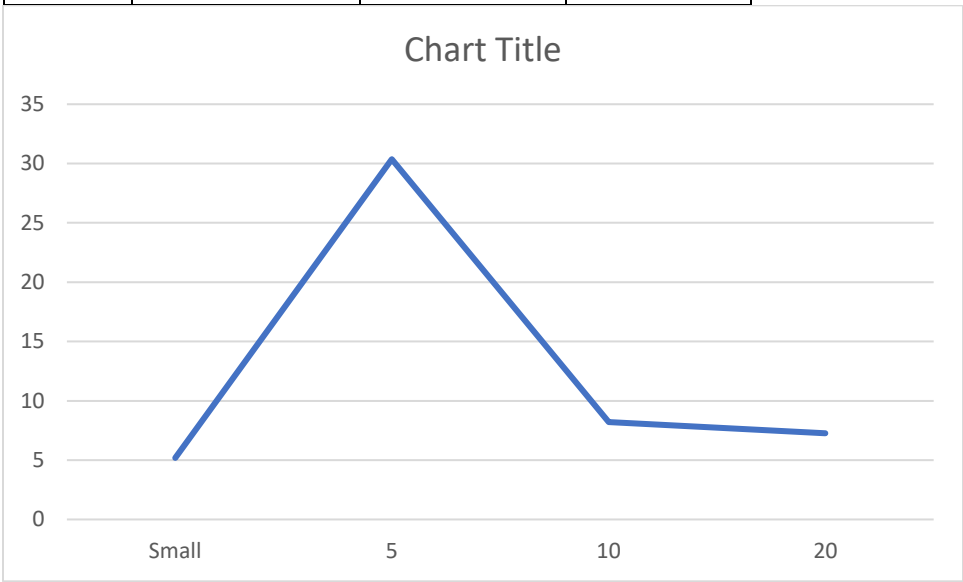
Req2	Tamaño_artistas	Tamaño_obras	Tiempo (ms)
Small	1948	768	8.444
5	4996	7572	317.143
10	6656	15008	



Requerimiento 3:

Req3	Tamaño_artistas	Tamaño_obras	Tiempo (ms)
Small	1948	768	5.189
5	4996	7572	30.36585951
10	6656	15008	8.215

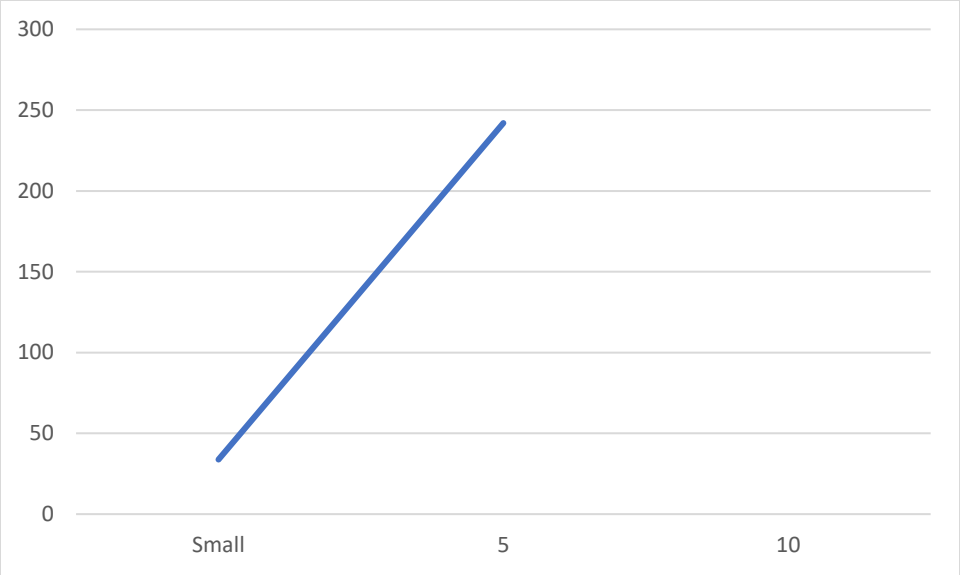
20			7.257
30			
50			



Requerimiento 4:

Requerimiento 5:

Req5	Tamaño_artistas	Tamaño_obras	Tiempo (ms)
Small	1948	768	33.785
5	4996	7572	242
10	6656	15008	
20			
30			
50			



Respecto a la implementación:

El primer y segundo requerimiento se trabajaron en grupo. El tercer requerimiento lo hizo el estudiante 1 y el cuarto requerimiento lo hizo el estudiante 2.