Reto No. 3 Soundtrack Your Timeline

Documento de análisis

Estudiante 1: Ehimar Andres Vargas Malaver - - e.vargasm@uniandes.edu.co - - 202014902

Análisis de complejidad

- Requerimiento 1

  La complejidad computacional de este requerimiento es O(n^2) ya que hay dos "For" dentro de los algoritmos planteados como solución, estos "For" son necesarios ya que se debe recorrer los elementos que retorna el TAD árbol binario RBT.

  ```
  for event in lt.iterator(lst):
      totalevets += lt.size(event['tracks'])
      listartist = mp.keySet(event['artists'])
      for event in lt.iterator(listartist):
          countuniqueartist(maptartist, event)
  ```

- Requerimiento 3.

  Este requerimiento presenta una complejidad de O(n^2) debido a que se realizan dos "For" para recorrer el árbol de Instrumentalnees y otro "For" para recorrer el árbol de tempo; sin embargo

  ```
  for event in lt.iterator(lstI):
      for event in lt.iterator(event['tracks']):
          entry = mp.get(mapsongs, event['track_id'])
          if ((entry is None) and (mini<=event['instrumentalness']<=maxi) and (mint<=event['tempo']<=maxt)):
              mp.put(mapsongs, event['track_id'], event)

  for event in lt.iterator(lstT):
      for event in lt.iterator(event['tracks']):
          entry = mp.get(mapsongs, event['track_id'])
          if ((entry is None) and (mini<=event['instrumentalness']<=maxi) and (mint<=event['tempo']<=maxt)):
              mp.put(mapsongs, event['track_id'], event)
  return lt.size(mp.valueSet(mapsongs)), mp.valueSet(mapsongs)
  ```

- Requerimiento 4.

  Este requerimiento presenta dos complejidades dependiendo de lo que el usuario elija cuando ingresa la información, si el usuario decide no ingresar un nuevo genera la complejidad del algoritmo será O(n^2) debido a que se utilizan dos "For" para obtener los resultados que se piden. Si el usuario desea agregar un nuevo genero a la búsqueda, la complejidad del algoritmo es O(n^3) dado a que se utiliza un "For" adicional para buscar el nuevo género.

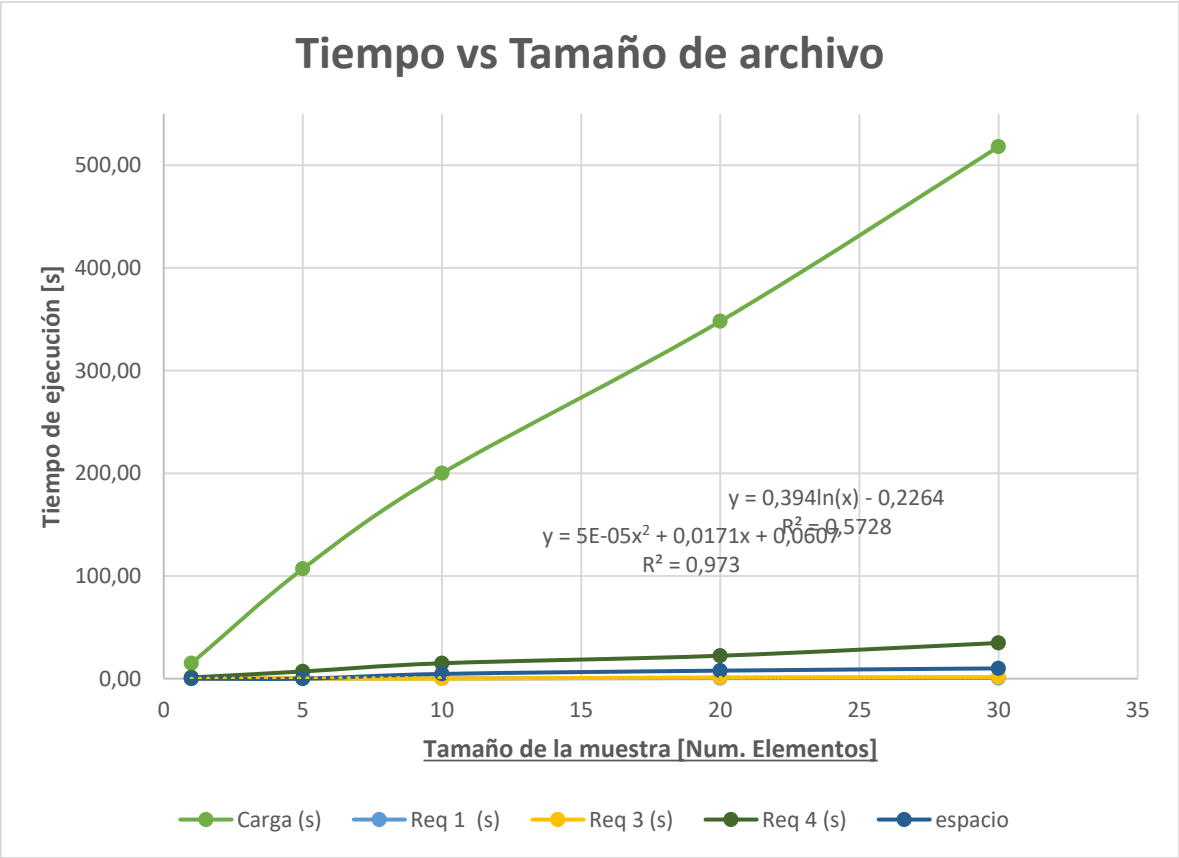
```python
def findreproductions(maptracks,genre, RBTT,listg, newg, minT, maxT):
    Totalrp = 0
    for element in listg:
        a = int((genre[element][0]))
        b = int((genre[element][1]))
        lstT = om.values(RBTT, a, b) #obtener los valores en cada genero recibido
        for event in lt.iterator(lstT): #diccionario con la informacion {list, hasht}
            Totalrp += lt.size(event['tracks'])
            for event in lt.iterator(event['tracks']):
                for element in listg:
                    a = int((genre[element][0]))
                    b = int((genre[element][1]))
                    if (float(a)<=float(event['tempo'])<=float(b)):
                        countreproductions(maptracks, element, event['artist_id'], a, b)

    if (newg != 'none'):
        lstT = om.values(RBTT, minT, maxT)
        for event in lt.iterator(lstT):
            Totalrp += lt.size(event['tracks'])
            for event in lt.iterator(event['tracks']):
                for element in listg:
                    a = int((genre[element][0]))
                    b = int((genre[element][1]))
                    if (float(a)<=float(event['tempo'])<=float(b)): # element es el genero
                        countreproductions(maptracks, element, event['artist_id'], a, b)
    return Totalrp
```

Hay dos For que no se cuentan ya que estos representan valores muy pequeños que dependen en gran medida del usuario.

Tiempos de carga y Tiempos de respuesta.

Tamaño de la muestra	**Carga (s)**	**Req 1 (s)**	**Req 3 (s)**	**Req 4 (s)**	**Espacio (GB)**
1	15,00	0,10	0,10	1,20	1,2
5	107,00	0,14	0,15	7,00	3.6
10	200,00	0,19	0.31	15,00	4,60
20	348,00	0,47	1,20	22,31	7,80
30	518,00	0,60	1,48	34,76	9,98



Los resultados de la gráfica podrían llevar a la conclusión de que las complejidades anteriormente descritas están incorrectas o que no correspondes a las líneas de tendencia observadas en la gráfica; sin embargo, esto tiene una explicación y radica en que los datos se almacenan dentro de árboles binarios, especialmente RBT los cuales permiten descartar grandes cantidades de información y reducir el tamañoo del n en gran medida. Cabe resaltar que las complejidades computacionales están dadas para el peor caso, el cual seria cuando se pidiera toda la información que esta dentro del árbol binario.

Cabe resaltar que los tiempos de ejecución son muy bajos respecto al volumen de datos que se están manejando en las prueba, ya que superan los 3 millones de datos.