Documento de análisis reto 2

Daniel Arango Cruz – 202110646

Oscar Iván García – 201630048

Para realizar el análisis de este segundo reto una vez más empezaremos analizando el nivel de complejidad de cada requerimiento en orden a la complejidad de sus funciones. Adicionalmente, para poder hacer una comparativa efectiva de tiempos de ejecución tomaremos y compararemos los tiempos que toma cada requerimiento con un tamaño de datos distintos, siendo en primer lugar los archivos small, luego el de tipo “50pct” y finalmente los archivos large.

**Carga de datos:**

|  |  |
| --- | --- |
| **Funciones** | **Complejidad** |
| initCatalog() | **O(k)** |
| loadData() |  |

**Tiempo ejecución small: 0.10 seg**

**Tiempo ejecución 50pct: 5.14 seg**

**Tiempo ejecución large: 9.375 seg**

**Requerimiento 1:**

|  |  |
| --- | --- |
| **Funciones** | **Complejidad** |
| getartistsByrange() |  |
| sortArtistByDate() |  |
| printSortResults1() |  |

**Tiempo ejecución small: 0.015 seg**

**Tiempo ejecución 50pct: 0.27 seg**

**Tiempo ejecución large: 0.34 seg**

**Requerimiento 1 (Reto 1 )**

|  |  |
| --- | --- |
| **Funciones** | **Complejidad** |
| **SortArtistByDate()** |  |
| **Consultar\_rango\_edad()** |  |
| **printsortResults()** |  |
| **Algoritmo comparación** |  |

En este requerimiento nos dimos cuenta de que el tipo de estructura de datos que resuelve con menor complejidad, teniendo en cuenta la implementación hecha en ambos retos, es el Tad lista pues mientras el consultar el rango de edad en Tad lista tarda en Tad tabla de hash tarda **,**  sin embargo, el volumen de datos en mapas a la hora de filtrarlos gracias al cómo están ordenados, reducen consideradamente.

**Requerimiento 2:**

|  |  |
| --- | --- |
| **Funciones** | **Complejidad** |
| Artorksinrange() |  |
| sortDate() |  |
| printSortResults2() |  |

**Tiempo ejecución small: 0.093 seg**

**Tiempo ejecución 50pct: 9.15 seg**

**Tiempo ejecución large: 18.90 seg**

**Requerimiento 2 (Reto 1)**

|  |  |
| --- | --- |
| **Funciones** | **Complejidad** |
| **sortDate()** |  |
| **Consultar\_adq\_rango()** |  |
| **printSortResults()** |  |
| **Algoritmo comparación** |  |

En este requerimiento al igual que en el primero el Tad lista tiene una aparente menor complejidad en la función de consultar adquisiciones dentro del rango, sin embargo, es necesario recalcar que aunque los mapas tienen un nivel de complejidad mayor, el volumen de datos que procesan es mucho menor gracias a la facilidad de filtrar por llaves de esta estructura de datos.

**Requerimiento 3 (Oscar Iván García):**

|  |  |
| --- | --- |
| **Funciones** | **Complejidad** |
| searchConstituentIDByName() |  |
| sortByMediumsQuantity() |  |
| printSortResults3() | **y 6** |
| listaRepeticionesMediums() |  |

**Tiempo ejecución small: 0.015 seg**

**Tiempo ejecución 50pct: 0.046 seg**

**Tiempo ejecución large: 0.062 seg**

**Requerimiento 4 (Daniel Arango Cruz):**

|  |  |
| --- | --- |
| **Funciones** | **Complejidad** |
| NumArtByNat() |  |
| sortArtVsNatBynum() |  |
| printSortResults5() | **y 6** |

**Tiempo ejecución small: 0.015 seg**

**Tiempo ejecución 50pct: 0.015 seg**

**Tiempo ejecución large: 0.046 seg**

**Requerimiento 4 – Daniel Arango Cruz (Reto 1)**

|  |  |
| --- | --- |
| **Funciones** | **Complejidad** |
| **ListaNacionalidades()** |  |
| **Artworksbynat()** |  |
| **Artsbynat()** |  |
| **LoadNat()** |  |

Tanto en el requerimiento 3 como en el 4 evidenciamos como el Tad tabla de hash facilito significativamente la solución del problema, pues redujo el numero de funciones necesarias para llegar a la solución gracias a su manera de almacenar datos mediante una llave, pues únicamente necesitamos acceder a una llave en especifica y jugar con su información. Esto lo logramos ver en la comparación de complejidades de los algoritmos.

**Requerimiento 5:**

|  |  |
| --- | --- |
| **Funciones** | **Complejidad** |
| Calcular\_Costo\_dep() |  |
| sortDepabydate() |  |
| printSortResults5() |  |
| SortDepbyprice() |  |

**Tiempo ejecución small: 0.031 seg**

**Tiempo ejecución 50pct: 3.67 seg**

**Tiempo ejecución large: 7.95 seg**

**Requerimiento 5**

|  |  |
| --- | --- |
| **Funciones** | **Complejidad** |
| **Búsqueda\_dep()** |  |
| **Calculo\_costo()** |  |
| **Sortbydate()** |  |
| **Sortbyprice()** |  |
| **printResults5()** |  |
| **Algoritmo comparación** |  |

Al igual que con los requerimientos 3 y 4, la cantidad de funciones que se necesitaron con la implementación de mapas fueron significativamente menores a las anteriormente utilizadas en el reto 1 para este requerimiento gracias a que únicamente se buscó mediante el departamento que llegaba como parámetro dentro de las llaves y de esa forma hacia que el volumen de datos y recorridos fueran mucho menores.