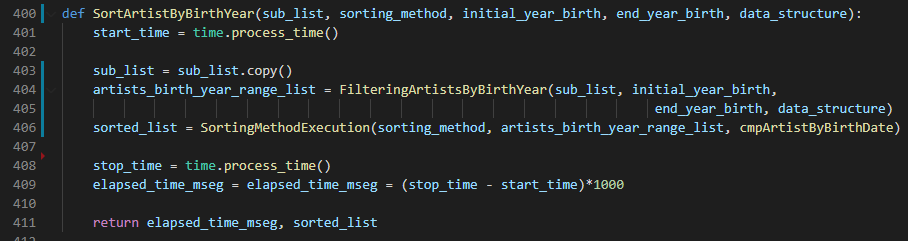
**Análisis de Complejidad**

**Nombre:** Nicolas Merchan Cuestas

**Código:** 202112109

**Correo:** n.merchan@uniandes.edu.co

**Requerimiento 1**

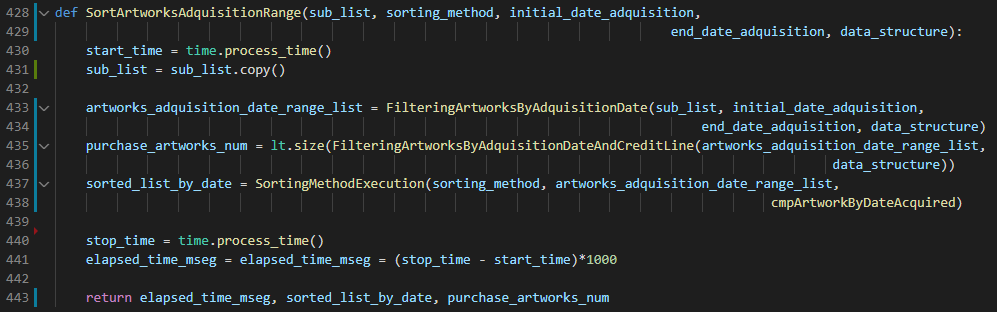


La complejidad asociada a **FilteringArtistsByBirthYear()** es **O(n)**, dado que separan los artistas nacidos en el rango de años indicado por medio de la comparación individual de cada elemento. De manera similar, la función **SortingMethodExecution()** realiza un ordenamiento en función del año de nacimiento de los artistas nacidos en el rango de años especificado. La complejidad de esta operación depende del tipo de algoritmo de ordenamiento utilizado. La complejidad es **O(n^2)**, **O(n^3/2)**, **O(n^2)** y **O(nlog(n))** para los algoritmos Insertion Sort, Shell Sort, Quick Sort y Merge Sort, respectivamente. Finalmente, la complejidad es modelada en su mayor parte por la complejidad en **SortingMethodExecution()**.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Requerimiento 1 - Lista Encadenada (mseg) | | | | | |
| N° Elementos | Porcentaje (%) | Algoritmo de Ordenamiento | | | |
| Insertion | Shell | Merge | Quick |
| 1522 | 10 | 390,63 | 262,50 | 220,34 | 320,63 |
| 3045 | 20 | 3187,50 | 2271,88 | 1314,90 | 3171,50 |
| 4567 | 30 | 11140,63 | 7180,75 | 5029,81 | 10320,63 |
| 6090 | 40 | 26109,38 | 21096,88 | 17468,86 | 23229,38 |
| 7612 | 50 | 46137,50 | 35043,75 | 21027,10 | 44134,40 |
| 9134 | 60 | 81365,63 | 67881,25 | 43071,40 | 77354,63 |
| 10656 | 70 | 165527,13 | 126891,35 | 54089,40 | 145827,13 |
| 12178 | 80 | 332492,26 | 250152,76 | 68089,40 | 322192,26 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Requerimiento 1 - Lista Ordenada (mseg) | | | | | |
| N° Elementos | Porcentaje (%) | Algoritmo de Ordenamiento | | | |
| Insertion | Shell | Merge | Quick |
| 1522 | 10 | 295,63 | 150,65 | 225,70 | 190,27 |
| 3045 | 20 | 2161,50 | 1786,62 | 1314,90 | 1890,52 |
| 4567 | 30 | 7320,18 | 6478,28 | 5029,81 | 6574,61 |
| 6090 | 40 | 20186,52 | 18925,39 | 17468,86 | 19645,25 |
| 7612 | 50 | 38925,02 | 34197,28 | 23027,10 | 35964,72 |
| 9134 | 60 | 76240,10 | 63651,25 | 48453,10 | 68836,73 |
| 10656 | 70 | 142065,87 | 116735,35 | 54089,30 | 134976,53 |
| 12178 | 80 | 290514,63 | 230254,76 | 67426,10 | 260545,83 |

**Requerimiento 2**

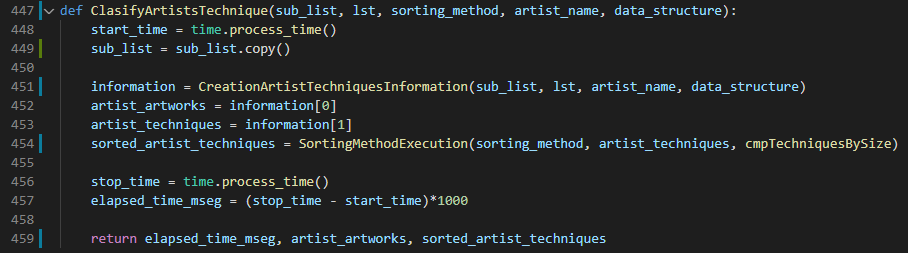


La complejidad asociada a **FilteringArtworksByAduisitionDate()** es **O(n)**, dado que separan las obras de arte adquiridas en el rango de fechas indicado por medio de la comparación individual de cada elemento. Igualmente, la complejidad de **FilteringArtworksByAduisitionDate()** es **O(n)**, porque separa las obras de arte adquiridas por compra revisando individualmente cada obra de arte. De manera similar, la función **SortingMethodExecution()** realiza un ordenamiento en función de la fecha de adquisición de las obras adquiridas en el rango de fechas especificado. La complejidad de esta operación depende del tipo de algoritmo de ordenamiento utilizado. La complejidad es **O(n)**, **O(nlog(n)),** **O(nlog(n))** y **O(nlog(n))** para los algoritmos Insertion Sort, Shell Sort, Quick Sort y Merge Sort, respectivamente. Ello se debe a que las obras de arte fueron agregadas a la lista de clasificación a medida que fueron adquiridas por el museo. De ese modo, existe un orden dentro de la lista a ordenar. Finalmente, la complejidad es modelada en su mayor parte por la complejidad en **SortingMethodExecution()**.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Requerimiento 2 - Lista Encadenada (mseg) | | | | | |
| N° Elementos | Porcentaje (%) | Algoritmo de Ordenamiento | | | |
| Insertion | Shell | Merge | Quick |
| 2763 | 2 | 325,54 | 374,28 | 231,82 | 231,82 |
| 5526 | 4 | 892,65 | 1255,86 | 1427,47 | 1638,43 |
| 8289 | 6 | 1323,46 | 1864,18 | 1929,23 | 2015,27 |
| 11052 | 8 | 1532,54 | 2396,45 | 2584,39 | 2648,12 |
| 13815 | 10 | 2178,17 | 3476,67 | 3568,40 | 3678,23 |
| 16578 | 12 | 2671,26 | 4381,15 | 4827,19 | 5025,80 |
| 19341 | 14 | 2915,83 | 6732,82 | 7001,20 | 7112,86 |
| 22104 | 16 | 3526,98 | 8453,92 | 9435,12 | 10172,12 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Requerimiento 2 - Lista Ordenada (mseg) | | | | | |
| N° Elementos | Porcentaje (%) | Algoritmo de Ordenamiento | | | |
| Insertion | Shell | Merge | Quick |
| 2763 | 2 | 306,76 | 360,13 | 221,30 | 200,34 |
| 5526 | 4 | 800,34 | 1003,78 | 1274,60 | 1439,40 |
| 8289 | 6 | 1234,20 | 1803,45 | 1893,50 | 2002,13 |
| 11052 | 8 | 1492,40 | 2017,30 | 2473,50 | 2648,40 |
| 13815 | 10 | 1902,40 | 3164,10 | 3110,30 | 3362,50 |
| 16578 | 12 | 2253,70 | 4182,00 | 4502,10 | 4836,10 |
| 19341 | 14 | 2437,50 | 5901,30 | 6403,40 | 6500,20 |
| 22104 | 16 | 3004,12 | 7925,40 | 8372,10 | 8263,40 |

**Requerimiento 3**

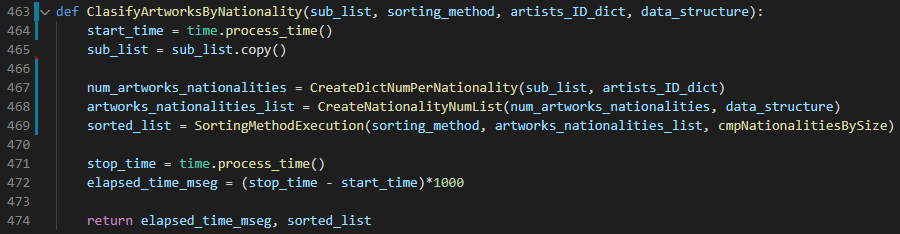


La función **CreationArtistsTechniquesInformation()** crea una TAD lista que contiene como elementos una lista con el nombre de la técnica y una TAD lista de todas la obras del autor que hacen uso de dicha técnica. La complejidad de esta función es **O(n)**, porque la función compara todas las obras respecto al autor y técnica utilizada en la mismas. De manera similar, la función **SortingMethodExecution()** realiza un ordenamiento en función de la cantidad de obras del artista en cuestión por técnica utilizada . La complejidad de esta operación depende del tipo de algoritmo de ordenamiento utilizado. La complejidad es **O(n^2)**, **O(n^3/2)**, **O(n^2)** y **O(nlog(n))** para los algoritmos Insertion Sort, Shell Sort, Quick Sort y Merge Sort, respectivamente. Finalmente, la complejidad es modelada en su mayor parte por la complejidad en **SortingMethodExecution()**.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Requerimiento 3 - Lista Encadenada (mseg) | | | | | |
| N° Elementos | Porcentaje (%) | Algoritmo de Ordenamiento | | | |
| Insertion | Shell | Merge | Quick |
| 2763 | 2 | 311,72 | 255,20 | 219,10 | 315,27 |
| 5526 | 4 | 2936,28 | 2651,87 | 1300,40 | 3162,90 |
| 8289 | 6 | 10284,82 | 8076,71 | 4526,30 | 13716,60 |
| 11052 | 8 | 23816,40 | 20183,16 | 18345,10 | 24371,18 |
| 13815 | 10 | 40155,10 | 31265,80 | 20345,90 | 46918,70 |
| 16578 | 12 | 75614,26 | 70162,30 | 42812,30 | 65812,30 |
| 19341 | 14 | 157291,10 | 120317,10 | 55273,10 | 139611,27 |
| 22104 | 16 | 304287,28 | 246193,17 | 62954,10 | 310261,19 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Requerimiento 3 - Lista Ordenada (mseg) | | | | | |
| N° Elementos | Porcentaje (%) | Algoritmo de Ordenamiento | | | |
| Insertion | Shell | Merge | Quick |
| 2763 | 2 | 300,57 | 240,34 | 208,20 | 310,20 |
| 5526 | 4 | 2538,12 | 2381,30 | 1284,30 | 3072,50 |
| 8289 | 6 | 7238,10 | 7523,50 | 4271,30 | 12840,00 |
| 11052 | 8 | 20491,30 | 18954,30 | 15723,10 | 21649,30 |
| 13815 | 10 | 35923,76 | 29485,20 | 17043,20 | 38465,20 |
| 16578 | 12 | 70273,40 | 64283,40 | 36823,50 | 50045,50 |
| 19341 | 14 | 149265,12 | 111845,00 | 45274,10 | 121845,40 |
| 22104 | 16 | 275294,15 | 239412,30 | 59253,50 | 260353,70 |

**Requerimiento 4**

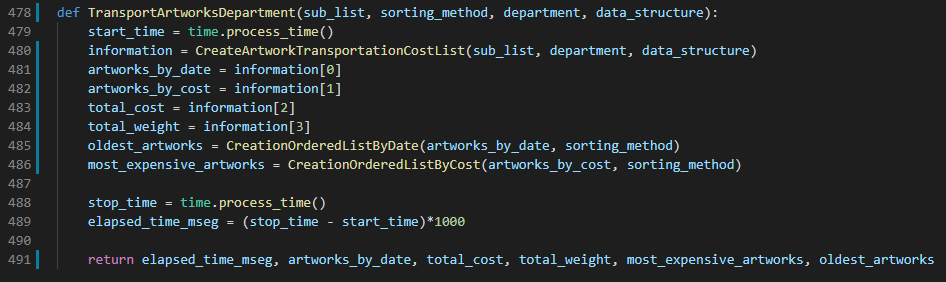


La función **CreateDictNumPerNationality()** cuenta la cantidad de obras de arte de cada nacionalidad y guarda dicha información en un diccionario donde la nacionalidad es la llave y el número de obras de arte es el valor de dicha llave. La función **CreateNationalityNumList()** convierte el diccionario generado en **CreateDictNumPerNationality()** en un TAD lista donde los elementos es son lista que contienen las nacionalidades y sus respectivos números de obras. La complejidad del proceso anteriormente mencionado es **O(n)**, dado que para completarlo es necesario recorrer la lista exactamente una vez comparando todas las obras de arte. De manera similar, la función **SortingMethodExecution()** realiza un ordenamiento en función de la cantidad de obras del artista en cuestión por nacionalidad . La complejidad de esta operación depende del tipo de algoritmo de ordenamiento utilizado. La complejidad es **O(n^2)**, **O(n^3/2)**, **O(n^2)** y **O(nlog(n))** para los algoritmos Insertion Sort, Shell Sort, Quick Sort y Merge Sort, respectivamente. Finalmente, la complejidad es modelada en su mayor parte por la complejidad en **SortingMethodExecution()**.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Requerimiento 4 - Lista Encadenada (mseg) | | | | | |
| N° Elementos | Porcentaje (%) | Algoritmo de Ordenamiento | | | |
| Insertion | Shell | Merge | Quick |
| 2763 | 2 | 370,17 | 270,34 | 205,40 | 319,26 |
| 5526 | 4 | 3015,20 | 2371,20 | 1523,12 | 3281,20 |
| 8289 | 6 | 14293,12 | 7829,10 | 4536,34 | 9374,30 |
| 11052 | 8 | 27392,50 | 23745,17 | 17238,40 | 24934,10 |
| 13815 | 10 | 40128,20 | 37217,10 | 22853,10 | 43016,20 |
| 16578 | 12 | 78395,10 | 65283,10 | 39274,10 | 77826,30 |
| 19341 | 14 | 180552,10 | 119274,20 | 51273,00 | 146829,70 |
| 22104 | 16 | 356874,12 | 260162,65 | 66283,10 | 325845,20 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Requerimiento 4 - Lista Ordenada (mseg) | | | | | |
| N° Elementos | Porcentaje (%) | Algoritmo de Ordenamiento | | | |
| Insertion | Shell | Merge | Quick |
| 2763 | 2 | 363,50 | 240,56 | 150,34 | 278,45 |
| 5526 | 4 | 3275,40 | 2103,40 | 1429,40 | 3365,23 |
| 8289 | 6 | 12734,50 | 7734,10 | 3912,30 | 7465,23 |
| 11052 | 8 | 20031,20 | 23790,10 | 14726,40 | 15945,30 |
| 13815 | 10 | 34924,10 | 30264,10 | 20388,60 | 38674,40 |
| 16578 | 12 | 59263,50 | 58264,10 | 30476,20 | 60374,50 |
| 19341 | 14 | 140346,34 | 101364,10 | 44836,10 | 125476,30 |
| 22104 | 16 | 330153,40 | 230131,50 | 51768,30 | 278675,30 |

**Requerimiento 5**



La función **CreateArtworkTransportationCostList()** calcula el valor de transporte de cada obra de arte del departamento ingresado por el usuario. La complejidad de esta función es **O(n)**, dado que la misma recorre todas las obras de arte y verifica si pertenecen al departamento indicado y calcula el costo de transporte simultáneamente. Posteriormente, las funciones **CreationOrderedListByDate()**  y **CreationOrderedListByCost**  ordenan la lista generada en **CreateArtworkTransportationCostList()** en base a la fecha de creación y costo de transporte, respectivamente. La complejidad de esta operación depende del tipo de algoritmo de ordenamiento utilizado. La complejidad es **O(n^2)**, **O(n^3/2)**, **O(n^2)** y **O(nlog(n))** para los algoritmos Insertion Sort, Shell Sort, Quick Sort y Merge Sort, respectivamente. Finalmente, la complejidad es modelada en su mayor parte por la complejidad en **SortingMethodExecution()**.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Requerimiento 5 - Lista Encadenada (mseg) | | | | | |
| N° Elementos | Porcentaje (%) | Algoritmo de Ordenamiento | | | |
| Insertion | Shell | Merge | Quick |
| 2763 | 2 | 393,34 | 252,34 | 215,26 | 333,29 |
| 5526 | 4 | 3219,30 | 2371,40 | 1425,12 | 3478,00 |
| 8289 | 6 | 13823,70 | 6812,50 | 4372,12 | 8900,23 |
| 11052 | 8 | 28364,90 | 20012,10 | 16297,30 | 25865,20 |
| 13815 | 10 | 44923,40 | 36283,20 | 24923,21 | 49273,40 |
| 16578 | 12 | 81926,90 | 68263,10 | 40023,70 | 79375,70 |
| 19341 | 14 | 155823,10 | 130562,20 | 52945,30 | 159728,31 |
| 22104 | 16 | 351922,60 | 245925,30 | 65934,45 | 345823,20 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Requerimiento 5 - Lista Ordenada (mseg) | | | | | |
| N° Elementos | Porcentaje (%) | Algoritmo de Ordenamiento | | | |
| Insertion | Shell | Merge | Quick |
| 2763 | 2 | 381,45 | 230,10 | 193,40 | 302,80 |
| 5526 | 4 | 2965,40 | 2034,50 | 1264,60 | 3276,80 |
| 8289 | 6 | 10214,60 | 5523,50 | 37458,00 | 7946,80 |
| 11052 | 8 | 23475,20 | 17394,50 | 13796,50 | 23512,40 |
| 13815 | 10 | 37225,70 | 25734,50 | 20375,60 | 40364,50 |
| 16578 | 12 | 60253,50 | 35182,40 | 29761,80 | 65835,60 |
| 19341 | 14 | 146143,20 | 103475,30 | 42745,70 | 132475,60 |
| 22104 | 16 | 325364,10 | 231486,90 | 57036,55 | 304845,60 |