|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ADT # | File | Time #1 | Time #2 | Time #3 | Average Time |
| 1 | File 1 | 0.129 | 0.119 | 0.126 | 0.124666667 |
|  | File 2 | 57.073 | 57.79 | 57.704 | 57.52233333 |
|  | File 3 | 0.1 | 0.094 | 0.099 | 0.097666667 |
|  | File 4 | 30.918 | 32.287 | 31.024 | 31.40966667 |
| 2 | File 1 | 0.096 | 0.087 | 0.086 | 0.089666667 |
|  | File 2 | 219.872 | 220.141 | 218.73 | 219.581 |
|  | File 3 | 0.111 | 0.1 | 0.101 | 0.104 |
|  | File 4 | 111.266 | 157.741 | 155.406 | 141.471 |
| 3 | File 1 | 0.083 | 0.081 | 0.082 | 0.082 |
|  | File 2 | 0.077 | 0.076 | 0.076 | 0.076333333 |
|  | File 3 | 0.077 | 0.076 | 0.076 | 0.076333333 |
|  | File 4 | 0.078 | 0.076 | 0.077 | 0.077 |
| 4 | File 1 | 0.115 | 0.113 | 0.113 | 0.113666667 |
|  | File 2 | 0.094 | 0.091 | 0.092 | 0.092333333 |
|  | File 3 | 0.092 | 0.091 | 0.091 | 0.091333333 |
|  | File 4 | 0.092 | 0.091 | 0.09 | 0.091 |
| 5 | File 1 | 0.085 | 0.083 | 0.084 | 0.084 |
|  | File 2 | 0.082 | 0.081 | 0.081 | 0.081333333 |
|  | File 3 | 0.08 | 0.085 | 0.085 | 0.083333333 |
|  | File 4 | 0.082 | 0.081 | 0.086 | 0.083 |
| 6 | File 1 | 0.287 | 0.29 | 0.282 | 0.286333333 |
|  | File 2 | 0.211 | 0.211 | 0.215 | 0.212333333 |
|  | File 3 | 0.243 | 0.253 | 0.265 | 0.253666667 |
|  | File 4 | 0.436 | 0.453 | 0.464 | 0.451 |

ADT Write-Up #1

File 1:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ADT | Individual Insertion | Individual Deletion | Series of Insertions | Series of Deletions | Entire File |
| 1 | O(1) | N/A | O(N) | N/A | O(N) |
| 2 | O(1) | N/A | O(N) | N/A | O(N) |
| 3 | O(1) | N/A | O(N) | N/A | O(N) |
| 4 | O(1) | N/A | O(N) | N/A | O(N) |
| 5 | O(1) | N/A | O(N) | N/A | O(N) |
| 6 | O(logN) | N/A | O(NlogN) | N/A | O(NlogN) |

File 2:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ADT | Individual Insertion | Individual Deletion | Series of Insertions | Series of Deletions | Entire File |
| 1 | O(1) | O(N) | O(N) | O(N2) | O(N2) |
| 2 | O(1) | O(N) | O(N) | O(N2) | O(N2) |
| 3 | O(1) | O(1) | O(N) | O(N) | O(N) |
| 4 | O(1) | O(1) | O(N) | O(N) | O(N) |
| 5 | O(1) | O(1) | O(N) | O(N) | O(N) |
| 6 | O(logN) | O(logN) | O(NlogN) | O(NlogN) | O(NlogN) |

File 3:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ADT | Individual Insertion | Individual Deletion | Series of Insertions | Series of Deletions | Entire File |
| 1 | O(1) | O(1) | O(N) | O(N) | O(N) |
| 2 | O(1) | O(1) | O(N) | O(N) | O(N) |
| 3 | O(1) | O(1) | O(N) | O(N) | O(N) |
| 4 | O(1) | O(1) | O(N) | O(N) | O(N) |
| 5 | O(1) | O(1) | O(N) | O(N) | O(N) |
| 6 | O(logN) | O(logN) | O(NlogN) | O(NlogN) | O(NlogN) |

File 4:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ADT | Individual Insertion | Individual Deletion | Series of Insertions | Series of Deletions | Entire File |
| 1 | O(1) | O(N) | O(N) | O(N2) | O(N2) |
| 2 | O(1) | O(N) | O(N) | O(N2) | O(N2) |
| 3 | O(1) | O(1) | O(N) | O(N) | O(N) |
| 4 | O(1) | O(1) | O(N) | O(N) | O(N) |
| 5 | O(1) | O(1) | O(N) | O(N) | O(N) |
| 6 | O(logN) | O(logN) | O(NlogN) | O(NlogN) | O(NlogN) |

The first ADT to be examined is the LinkedList. For this data type, the time complexity varies depending up the order of the numbers that are inserted or deleted. For example, we see a significant difference between the operations of File2 and File3 because the order of deletions is reversed although the order of insertions remains the same. In File2 the number that is being deleted is found at the end of the list, while for File3 the number being deleted is located at the beginning of the list. This causes the Big-Oh to be O(N) for File2 rather than O(1) since the whole list has to be traversed in order to find the value to be deleted. With this being said, the Big-Oh for File3 is O(1) since the number being deleted is always found at the beginning of the list. For entire series of deletions, we repeat these operations N times which causes the complexities to be O(N2) for File2 and O(N) for File3. For File, the order of insertions and deletions is completely unique, however for insertions we know that the element is inserted at the front causing the Big-Oh to be O(1). This is different for deletions where we assume that the element being deleted will be found at the end of the list causing the Big-Oh to be O(N) since the whole list must be traversed. For an entire series of deletions, a single deletion occurs N times causing the Big-Oh to be O(N2) still assuming the worst case scenario.

For the second ADT, a CursorList, we obtain the same Big-Oh values due to the fact that this data type embodies the same operations as a LinkedList. These operations include an insertion always occurring at the beginning of the list, and a deletion always occurs after traversing the list and finding the value to be deleted. However, despite the Big-Oh values being the same for both ADTs, there is a dramatic difference in time for CursorList causing it to be much slower. This is attributed to the constant associated with the Big-Oh value that comes from having to access the code for the vector class since CursorList utilizes a vector.

For ADTs three, four and five (StackAr, StackLi, and QueueAr) the order of the values in the file had no effect on the Big-Oh values between each individual file. Regardless of the structure of the file, the insertions and deletions occur at the same location for StackAr and StackLi. Both of these operations occur at the top of the Stack because of their push() and pop() operations. This situation is also true for QueueAr where insertions always occur at at the beginning of the Queue using the function enqueue() and deletions always occur at the end of the Queue using the function dequeue().

For the final, sixth ADT, SkipList we see once again that there is very little variation between its Big-Oh values. This is because this ADT is able to search for values more efficiently than the other ADTs. The manner in which a SkipList is able to search more efficiently is derived from its structure, which utilizes leveled nodes to traverse large areas of data quickly. By being able to effectively search through large gaps of data quickly and efficiently, there is little to no change between the running times of each file.

In order to compare these ADTs among each other, they will be divided into two groups. The first group will contain StackAr, StackLi, and QueueAr since they all share the same Big-Oh values. There is not much that distinguishes these data types other than the fact that QueueAr and StackAr have to be initialized with a capacity while StackLi is able to expand with insertion operations. The functions within these data types all do the same thing with the exception of dequeue() deleting from the back of QueueAr rather that deleting at the top of the stack for both StackLi and StackAr using pop().

The next group contains the ADTs: SkipList, CursorList and LinkedList. The reason that these ADTs are grouped together to examin is because of the significant time differences in the running time of each file. Through looking at the running times of each file we can see that SkipList performed much quicker for File2 and File4 than CursorList and LinkedList. This quicker running time is attributed to the structure of the SkipList, which we discussed earlier. Its leveled node structure allows it to traverse larges gaps of data quickly when searching for a specific value whereas both LinkedList and CursorList utilize a linear search, typically resulting in the entire list having to be traversed. However, when looking at File1 and File3, CursorList and LinkedList perform much faster because of the order in which the values are inserted and deleted. For a SkipList, the values are inserted in any location within the list and a random number generator determines the height of the node. For CursorList and LinkedList, the values are always inserted at the beginning of the list causing them to be quicker for File1, and for File3 the order of the values for deletions causes the deletions to also occur at the front of the list resulting in no search being necessary.