Efrain Vazquez Hernandez A01612936

**Double linked lists in Merge Sort**

Double linked lists are useful data structures that give the user the opportunity to easily adjust the nodes in the structure, this is because of its two pointers from both directions. Also, the structure leaves the users the capacity to get a value, while it also tracks the past values, however this advantage is also one of their worst weaknesses.

. However, the use of a merge sort doesn’t use any of the characteristics previously stablished, since the algorithm is almost always created for data structures like vectors or arrays, where to access a place it’s a constant time, to the double linked list is O(n/2). So, to make the use of double-linked lists useful you would have to recreate the same algorithm considering all the pros this data structure has.

In this case a better implementation of the algorithm would be that in the case of the merge, it would use the function of insert instead of creating a new data structure and insert there the new values.

However, for this reason, I would rather choose to create a new algorithm to sort the data. In this case, all the data would be inserted into queues stored in a stack, each queue could only store a value that is bigger than the past, so whenever this condition fails, it would create another queue. So, when you have stored all the data in the stack of queues. The next step would be to erase the linked list and create a new one. And for each stack, you would be going through the linked list inserting the values in the queue. This would be efficient by the fact that you would have the value of the next and actual node, and a pointer to the actual value. So, you could use the function insert, without having to traverse all the data structure again. Also, the fact that all the queues are sorted, simplify the process to insert the values through the double linked list.

This approach is similar to the quick sort. Since it divides all the data by just a random pointer.

Its complexity would be in the worst case, the list is in reverse, so we create a queue for each data. And the merge would be linear, going from the start to the end at each insertion. And int the best case it would be a single queue and just a sequence of insertions.