Doubly Linked List

## ReflexAct 2.3

Mariana Esquivel Hernandez A01641244

A doubly linked list is a variation of the singly linked list. It differs from the singly linked list in that where each node contains an extra pointer to the previous node along with the next pointer. A linked list is a linear data structure that does not store the elements at contiguous memory locations. Rather, they are stored at random locations connected through pointers.

This presence of an extra pointer facilitates insert, delete operations on the doubly linked list but at the same time requires extra memory to store these extra pointers.

The doubly linked list has various uses in real-time scenarios like browser cache, MRUs, etc. We can also represent other data structures like trees, hash tables, etc. using a doubly linked list.

A doubly linked list can be applied in various real-life scenarios and applications, for example:

* A Deck of cards in a game is a classic example of a doubly linked list. Given that each card in a deck has the previous card and next card arranged sequentially, this deck of cards can be easily represented using a doubly linked list.
* Browser history/cache – The browser cache has a collection of URLs and can be navigated using the forward and back buttons is another good example that can be represented as a doubly linked list.
* Most recently used (MRU) also can be represented as a doubly linked list.
* Other data structures like Stacks, hash table, the binary tree can also be constructed or programmed using a doubly linked list.

# Advantages

* The doubly linked list can be traversed in forward as well as backward directions, unlike singly linked list which can be traversed in the forward direction only.
* Insertion operation can be done easily in a doubly linked list when compared to the singly linked list.
* It allows traversing in both forward and backward directions because of the next and previous pointers, unlike the singly linked list, which allows traversing in only one direction.
* Deletion of nodes is easier compared to a singly linked list. This is because to delete a node; we need access to the node to be deleted and the node before it. So, in a singly linked list, for deletion, we require two pointers, while in a doubly linked list, there is no need to maintain an extra pointer as each node itself carries the information of the previous node.

# Disadvantages

* As the doubly linked list contains one more extra pointer, previous, the memory space taken up by the doubly linked list is larger when compared to the singly linked list.
* Since two pointers are present, previous and next, all the operations performed on the doubly linked list have to take care of these pointers and maintain them thereby resulting in a performance bottleneck.
* It consumes extra memory space compared to a singly linked list due to the extra previous pointer it must maintain for each node.
* It is impossible to access the list’s elements randomly because they are stored at random locations, and we can only access them sequentially.
* All the operations require more time due to the overhead of handling extra pointers. For instance, if you must insert a new node, you also need to modify the previous pointers and the next pointers, similarly in the case of deletion.

# Approach in the activity and Complexity of the functions into the activity

The program uses merge Sort and Sequential Search.

We decided to use this method to sort the IP once we obtain them in a specific format and implement a Sequential Search to find certain IPs when a user ask for a range.

We get the IP from the line of the text by checking the quantity of spaces before it and after it, so we can send only the IP to another function to convert it to an integer.

We checked the max quantity per numbers between the dots of the IP, for example, the default IP is xxx.xx.xxx.xx:xxxx, so if it founds a format that doesn’t fit the default format we added 0 to the left once we hit the dot.  
And after that we take away the “.” And “:” so we can work with the IP as an integer, sort it and found the nearest IP the user range ask for.

**Merge Sort**  
The Merge Sort algorithm is a sorting algorithm that is based on the Divide and Conquer paradigm. In this algorithm, the array is initially divided into two equal halves and then they are combined in a sorted manner.

**Sequential Search**

Method for finding an element within a list. It sequentially checks each element of the list until a match is found or the whole list has been searched.

As first step we started opening the txt file and reading line by line (this has a complexity of O (n)) so then we can storage each line into a vector and take the information of the line with the substring method.

Then we converted the data to integers to sort them with a function, once as integers we could apply a merge Sort algorithm to order the IPs in ascending order.

Once having the information in order, we could apply search algorithms, we chose to use sequential Search with a complexity O (n).  
We used two Sequential Search, one for the start IP and other for the end IP asked from the user to obtain the range into the ordered information.

And for last we return that information into a new txt file to storage the data asked with a complexity of O (n); the Total Complexity of the program is O (n log n).