Alican Özeloğlu - 66277019558

EEM480 ALGORITHMS AND COMPLEXITY

HOMEWORK 2

TASK

The purpose of the homework is to create a LinkedList with double sides nodes. LinkedList will have the methods specified in the LinkedList interface (HW2Interface). These methods are as follows, Insert, Delete, ReverseLink, SquashL, OplashL, Output, ROutput, toString, LinkedListException.

HOW IT'S DONE

The Project was created in the Apache NetBeans IDE 11.1 environment. A new java class in the name of "DLNode" was created. This class have an element variable which is type of integer and two left and right variable which is type of DLNode. The element variable holds the element of the array and the left and right variables hold the addresses of the previous and next nodes in the LinkedList. A new java interface in the name of "HW2Interface" was created and the methods written in there. Another java class in the name of "LinkedList" was created and this class implements HW2Interface. This class contains a constructor to set the value of n to zero, the variable that holds the number of elements in the list.

Insert method takes 2 parameters. One is the element to be added to the list, and the other is where to add the element. First, the number of element of the array was checked. There are three different situation can be listed as follow, the element to be added is the first element, the element will be added between two nodes, the element will be added end of the list. In the first situation, element's right and left address variables will be null. In the second, a pointer called reference leads to the desired position. The left variable of the new node created from the DLNode class is set to hold the address of the reference and the right variable set to hold the address of the right variable of the reference. The previous elements in the list are set to hold the address of the new node and the operation is completed. Third situation is not very different. This time the right variable of the new node is set to hold null and other necessary connections are made.

Delete method takes parameters that indicate which element in which position to delete. Similar to the insert method, the position to be deleted is accessed by reference and this time links of the node to be deleted is broken and necessary connections are made.

ReverseLink method does not take any parameter. Reverse the LinkedList when it called. It starts from the first node to the last node and saves the elements in a static array at each stage. This array will be saved to the nodes in the reverse order.

SquashL method does not take parameters. Lists the elements in the list and the number of times that element exist. The element was retrieved and the number of repetitions was reached by checking whether it was the same as the next node element. This method could be done using the insert and delete methods. A new array was created by retrieving element information from the old array. Although this method cost longer lines of code, it was preferred because it was simpler as an algorithm.

OplashL method does not take any parameter. This method is the reverse of the SquashL method. In this method, two elements were taken from the linked lists each time. One had the value of the element and the other had the information about how many time it would repeat. Delete and Insert methods were used in this method. After getting the number of repetitions of the element, the node holding this information was deleted and Insert method was applied according to this information.

Two different methods were used to print the list on the screen. One is the Output method, which allows us to write the list in the original order, and the other is the ROutput method, which prints the list in reverse order. In the **Output method**, the first node of the list was received and the addresses of the variables named "right" of the nodes were followed and the elements were printed on the screen. In the **ROutput method**, the last node of the list was received from the variable named “last” and the addresses of the variables named "right" of the nodes were followed and the elements were printed on the screen.

The problem may be caused by the incorrect position information given in Insert and Delete methods. In such cases, we created our own exception to prevent the code from crashing. Name of the Exception is **LinkedListException** and it has a method name “toString” for returning the error message. A class named LinkedlistException was created and the Exception class was extended and the exception class was created.

A method called “**toString**” has been created in LinkedList so that the elements of the list can be written on the screen when System.out.println(mylist) command is executed. In the toString method, array elements were converted to String.

COMPLEXIES (Theta)

- Insert (1)
- Delete (1)
- ReverseLink (n)
- SquashL (n)
- OplashL (n)
- Output (n)
- ROutput (n)
- toString (1)
- Exception (1)