PRACTIAL WORK

THE MEASUREMENT OF RICK SANCHEZ

HENRY TAMEKLOE

2018500044

DEPARTMENT OF COMPUTER SCIENCE

FEDERAL UNIVERSITY OF MINAS GERAIS

Belo horizonte – MG – Brasil

henrany@ufmg.br

INTRODUCTION:

This project is to practice the concept of data structures such as linked list, queue and stack. It's about a scientist who wants to realize scientific experiments of measuring different type of reagent with cautiousness. He uses apparatus of different sizes to do his measurements. To do each measurement, he has to combine different apparatus to obtain the quantity needed. He also breaks some apparatus which needs to be removed and also find apparatus lost in the lab. The main aim of the problem is to write a program in which given apparatus of different sizes and also the measurement to be done in milliliters, return the minimum number of operations which needs to be done to get the amount required.

IMPLEMENTATION:

The implementation of the program was done in c++ using class. It has a header file, source file and the main function. The implementation was done using the linked list data structure. The declaration of the header file has a struct function for the data, count and also a pointer to the next of the list. The main class has the private attributes head and tail which are the beginning and end of the list, a count variable to count the total number of values inserted into the list and the container to get the containers for the measurements. The public attributes have declarations which can be understood by looking at the code.

The principal functions are the insertion of a node into the list, and removing node from the list and the operation to get the minimum operations to be done. The function for the insertion of the list is the insert_container(int container) and also insert_new_container(int container, int count) with parameters for the getting the container and the count. The insertion function insert_container(int container) start by declaring an auxiliary node "temp" which is later assigned to the new data to be inserted into the list. After that, we assign to point to null. With this, we check whether the head of the list is empty or not, if empty, assign both head and tail to the auxiliary variable and increment the count by 1. If the list already has an element, we insert the element at the end of the list. We assign the next of the tail to the auxiliary variable and then assign tail to point to null as the end of the list.

The other insertion function insert_new_container(int container, int count) has the same implementation of the already insert function insert_container(int container) with different parameters, in which each time we insert an element into the list, we assign the count to the inserted element. This is done to get the level of operation.

The next principal function is the removing of a node from the list. An auxiliary variable is created and assigned to the head of the list. First we verify if the element to be removed is at the beginning of the list. If it is, we set the prev to the next node and delete the node we want to remove and return. A new auxiliary variable is created and assigned to the next of the head and then we run through the list and try to find the element we want to remove, and also verify if the node we want to remove is at the end of the list, if it is at the end of the list, we assign the tail to the prev and assign it to the current node. After that we delete the node, this is done not to lose reference of the list. After that, we set the prev to the current and the delete the current node

The other important function was the operation function operations(int container) which is the getting the minimum operation needed. We first pass a parameter container which is the desired amount needed for the measurement, with that, we create a new node opera and set it to the head to get the already created list. We create other variables quant, cnt, sum and sub to get the quantity, the count, the summation and subtraction respectively. We then create a new list measure and insert it with 0 using the insert_new_container(int container , int count) function as the base of the operations. We create another node and assign it to the head of the newly created list. We then have a "for" loop to get the containers and the count in the new list. After that, we create another loop for the addition and subtraction of the containers and increment the counter. If the sub operation gives a negative value, we don't save it into the list. We then check whether the sum or sub manipulation is equal to the required amount needed, if not, we get the head of the new list and the original list to get each element and we return to the first for loop and assign them with different containers. If any of them is equal, we return the cnt + 1.

The other function is the print function to print the content of the list.

The destructor is used to free the space allocated as we run through the whole list and delete every node available.

INSTRUCTION FOR COMPILING:

A make file is used to compile the program. The program was done and compiled using c++ and the gcc compiler. To compile the program, enter "***g++ *.cpp –o tp1*** " or "***make tp1***"and execute it by entering ***"./tp1***" in the terminal.

COMPLEXITY ANALISES:

The time complexity is done by checking how long the various principal functions take. The principal functions to look at are the insertion, removing, operations, print, destructor and the main.

The insertion function insert_container() takes constant time O(1) to run since we only insert values and don't need to run through the whole list.

The removing function remove_container() have two time complexity. One is, it takes a constant time O(1) when the item to be removed is at the beginning of the list. And the other is when the item to be removed is at the middle or the end of the list. This way, it takes a linear time O(n) as we have run through the list to find the item we want to remove.

The other function is the operation function operations() which has time complexity of $O(n^2)$ as we first do a "for" loop to get the new list and after that another "for" loop to do the addition and the subtraction to get the minimum operation.

The print and destructor function all have a linear time complexity O(n) as we have to run through the list to print and also to free the memory.

The main function has a linear time complexity of O(n) as we have to get the user input since we don't know the size of it.

The total time complexity is the maximum time complexity of all the time complexities which are :

Max (O(1) , O(n) , (n^2), O(n), O(n), O(n)) = $O(n^2)$.

So the time complexity of the program is O (n^2)

CONCLUSION:

At the end of the day, with any quantity of apparatus of different sizes, Sir Rick would be able to know the minimum operations to be done to get his experiment done as fast as possible.

This work gave a lot of new ideas on the use of linked list to solve daily world problems, since we are not limited to a certain quantity of data that we can insert into our system.

The number one difficulty encountered in the resolution of the problem was getting the minimum operations to be done to get the desired amount. Apart from this, other implementations gave less work than getting the minimum operations.

BIBLIOGRAPHY:

Linked list in c++ https://www.geeksforgeeks.org/

Introduction to Algorithms Third Edition Thomas H. Cormen
Charles E. Leiserson The MIT Press Cambridge, Massachusetts London, England