Double linked list

Leo Halfar

September 2023

Introduction

This report will discuss the implementation of a double linked list as well as functions that can be used to manipulate a linked or double linked list.

Double linked list

The double linked list contained a sequence of cells that contained a value as well as references to the previous and following cell. A cell first was created and set to the first cell in the sequence in order to always have access to the beginning of the list. To create a cell the function cell was called which was called with an integer that became the cells value as well as a cell that became the previous reference.

Add

The add function was called with an integer that was to become the next cells value. Firstly a cell current was set to the cell first. An if statement then checked if the list was empty. If that was the case a new cell was created and the cell first was set to the new cell. In the case the list wasn't empty a while loop iterated through the sequence using current.next. If the cell after current was null a new cell was created, the value was set to the integer add was called with. The previous reference was subsequently set to the current cell and the current cell's next reference was set to the new cell.

Remove

The remove method was to remove the cell with the value the function was called with. Initially a cell current was set to the cell first. Using if statements the function then checked if the list was empty or if it only contained one cell. If the sequence was empty the function simply returned since there was nothing to remove. In the case the list only contained one cell and this cell had the value that was to be removed this cell was set to a null cell. If neither of these statements where true a while loop iterated through the list until the cell after current had the value that was to be removed. The current cell's next reference was set to current.next.next and the cell after the to be removed cell's previous reference was set to the current cell.

Length and find

Both the length and find function where unchanged from the linked list implementations and will therefore not be discussed in this report.

Unlink

The unlink method was called with a cell that was to be unlinked(removed) from the list lets call this cell un.

Double linked list

In the double linked list the function contained an if, if else and else statement. The if statements checked if un was the first in the sequence. If that was the case the function simply changed the cell first to the next cell in the sequance thereby unlinking un. The else if statement checked if un was the last cell in the sequance. If that was true the method set the next reference of the cell before un to null unlinking un from the sequance. If neither of these statements was true the function set the next reference of the cell before un to the cell after un and the previus reference of the cell after un to the cell before un. The implementation of this can be seen below:

```
void unlink(Cell un) {
        if (un.prev == null) {
            first = un.next;
            un = null;

        } else if (un.next == null) {

            un.prev.next = un.next;

        } else {
            un.prev.next = un.next;
            un.next.prev = un.prev;
        }
    }
```

linked list

The linked list implementation of the unlink method was to remove a cell in the sequence. First of a cell current was set to the first cell. The function then checked if the list was empty or if it only contained one cell. If it was empty the function simply returned since there was nothing to unlink. If the list contained a single cell and this cell was the one to be removed the first cell was simply set to a null cell. In the case that neither of these if statement where true the current cell was set to the current.next cell until current.next was equal to the cell that was to be removed. When this occurred the cell current.next was set to the current.next.next cell.

Insert

The insert method was to insert a cell in the beginning of a list, lets call the cell dos.

Double linked list

The double linked list implementation simply set next reference of dos to the first cell in the sequence and the first cell's previous reference to dos. The cell first was subsequently set to dos since dos was now the first cell in the sequence. the implementation of this can be seen below:

```
void insert(Cell dos) {
        dos.next = first;
        first.prev = dos;
        first = dos;

    }
```

Linked list

The linked list implementation of insert set the next reference of dos to the cell first and then set the cell first to dos.

Benchmarks

The benchmarks for measuring the time it took to unlink and insert a cell in a sequence was set up by creating linked and double linked lists of different sizes. At the same time all the cells in the sequence was put into an array. An unsorted array of 100 integers was also created. The benchmark then iterated through the array of integers and used the current integer to get a cell from the array of cells. This cell was then unlinked and subsequently inserted inserted in the sequence. If the sequence was for example 1000 cells long the integers in the array where random numbers between 0 and

999. The time it took to unlink and insert a cell was measured using the system.nanotime function.

Result

list size	**Linked list**	**Double linked list**
1000	6000	150
2000	12000	160
4000	20000	160
6000	35000	160

Table 1: Avarage time consumption when unlinking and inserting 100 cells in a sequence

As one can see in the table above the linked list has a linear time consumption, when the size of the list doubles it takes roughly double the time to locate the cell, unlink it and then insert it. This is to be expected because we have to go through each cell in the sequence until we find the cell we want to unlink. When analyzing the results for the double linked list we can see it has the time complexity O(1). This is also to be expected since the size of the list doesn't affect the time it takes to locate the cell we want to unlink since we already have all the information we need.

Discussion

When analyzing the already mentioned results for the linked list we can see that the results aren't precisely linear as they should be. Could this be improved upon by changing our benchmark? Perhaps, if we did the measurements for example a thousand times over we could have gotten a better result. The other possibility is that the array of integers affected our result. Since the array was completely randomized and the time it takes to find a cell in a linked list is entirely dependent upon what integer we get from the array this is very likely to be the case. However as of this very moment this question can not be answered.

conclusion

In this report, we explored the implementation of a double linked list data structure along with various functions for manipulating both linked and double linked lists. In conclusion a doubly linked list is significantly faster than a linked list especially when it comes to removing a certain cell from the sequence. The downside of the double linked list is that it takes up more memory.