A doubly linked list

Nariman Haidar

Fall term 2022

1 Introduction:

In this assignment we are going to write a doubly linked structures and find out their properties. Then we are going to write all methods than we need to add nodes and remove nodes and a bench mark to measure time that it takes to remove one node and compare it with a singly linked list.

2 Method:

A doubly linked list

In the singly linked list we learned that every node in this list has two fields data and a pointer next which points to the next node in the list, but doubly Linked List is a type of Linked List where each node apart from storing data has two links. One to points to the previous node and the other link points to the next node of the list. So the doubly linked list is much like a Single Linked List. Every link that points to previous node called Prev, and Every link that points to next node called next.

To implement the code in our assignment I started with write the first class and that is node and in this node that represents a node in the linked list. It has previous node, data, and the next node. Then I define another class that called Doubly Linked List to create a Doubly Linked List with one node head and size, then initialize values the head will be null and the size will be zero.

Then I created a method that add a node first. I started with creating node then check if the list is empty or not and if it empty so the new node n is pointed by head and it is the first node in the list so prev will point to null. If it not empty so the new node n next will point to the head and the head prev will point to the new node n and the size will be bigger. Then I write a method that show all data in this list. Then I write a method that delete a node from the beginning. I started check if the head is null and if that just return, but if it is not empty the head is pointed to head.next and head prev is pointed to null and size reduces. But in the assignment we

need to make a reference to the previous element will come in handy when we want to remove an element in the list without traversing the list from the beginning. And A doubly linked list is as the name suggests a linked list where each node also has a reference to the previous node in the list. The very last node in the list will as usual have a null next pointer but now we will also have special case for the first node in the list. So I started to write a method that find a node at a specific index and then return the data that in this index.

Then I started to write a method that delete that specific index that we get from the previous method.This method finished when the array that is used by remove method can find the the node that we need to remove it then it will delete that node in that position. Remove method is going to check if the head is null and if that just return. Then check if index is smaller than zero bigger than the size so just return. Then if given index does not meet these conditions then until we get to the given index we need to move forward in the list if the next pointer is null then update pointer, then if the previous pointer is null then both the pointer and the head elements are added and if the node is in the middle of the list, so update both pointers and update size of list.

Then I make some of bench mark that to keep a list of n elements and then perform k remove and add operations. The elements that removed selected by random. And I measure time just only to remove elements from the list.

Time complexity: The add function in the doubly linked list is different because if we wants to add at the first so we needs o(1) it is constant time, but if we need to add the new node at the end or in the mid so we need to go through all the needs or some of them in the list to put the new node and it takes o(n). And add in the mid is taking also o(n). The remove function is the same as the add function if we want remove the first node so it takes o(1) but if we wants to remove from the last or from the mid so it takes o(n).

code:In this code we can see how we can add a node to the beginning.

```
public void addFirst(int data) {
        DoublyNode n = new DoublyNode(data);
        if (head == null) {
            head = n;
        }
        else{
            n.next = head;
            head.prev = n;
            head = n;
```

```
        }
        size++;
    }
```

code:In this code we can see how we can remove a node from the beginning.

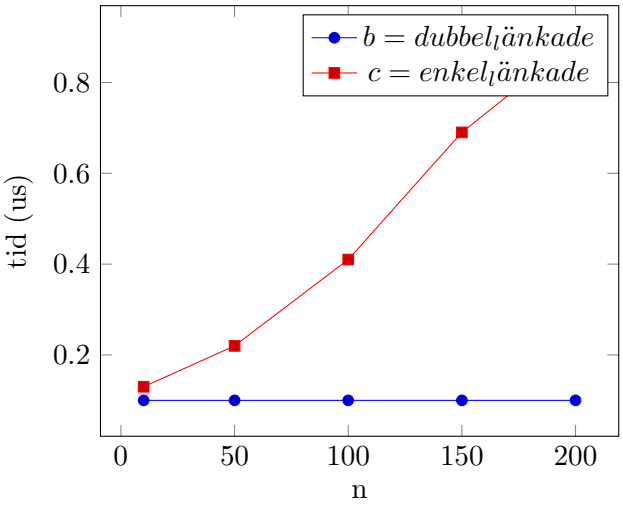
```
public void deletFirst() {
        if (head == null)
            return;

        head = head.next;
        head.prev = null;
        size--;
    }
```

table

n	Doubly Linked List	Singly Linked List
10	0.1	0.13
50	0.1	0.22
100	0.1	0.41
150	0.1	0.69
200	0.1	0.89

Table 1: This is the time out put of singly linked list and doubly linked list to delete method

Graph:Singly Linked List



Compare between doubly Linked List and singly linked list:

In singly linked list I used the same from the last assignment but I changed little with bench mark and with delete method to suit this assignment because in this assignment we want to remove a random node and then put it at first of list. In conclusion time complexity to doubly linked list takes constant time but in singly linked list takes o(n). Because in singly linked list it takes time to go through every element in the list but in doubly linked list it to remove an element in the list it doesn't have to traversing the list from the beginning.