CSDS 233 Spring Session 3

SI Leader: Jakob Danninger

2/2/2023

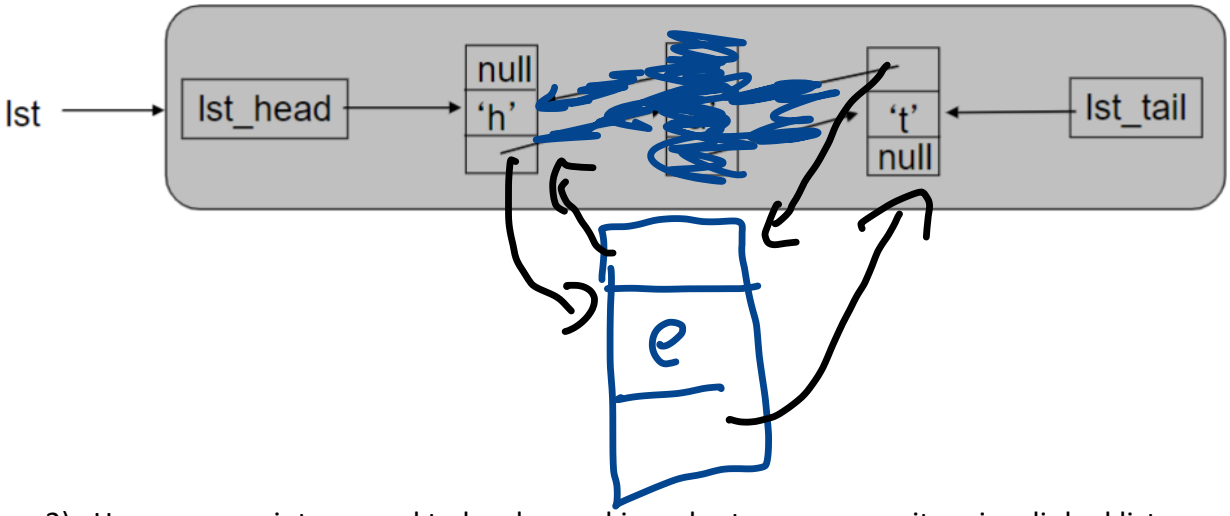
Disclosure: This is a supplement to class, not a replacement. This should not be your only study activity for exams, it should aid you in studying. I do not have the actual exam so questions here will differ from those on the exam.

Session Objectives:

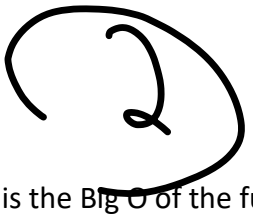
1) Explain what a linked list is (including how to add remove and search
2) Understand how computer memory works and how that related to array and linked list
3) Decide when to use a list vs array

Practice Problems:

1) Remove the node with the letter "a" then add a new one. Show each step and which pointers you changed



2) How many pointers need to be changed in order to remove an item in a linked list



3) What is the Big O of the function below:

```
public StringNode getNode(int i) {
    if (i < 0 || i >= theSize) throw new Exception("Index out of bounds");
    StringNode ptr;
    if (i < theSize/2) {
        ptr = lst_head;
        for (j = 0; j != i; j++) ptr = ptr.next;
    } else {
        ptr = lst_tail;
        for (j = theSize-1; j != i; j--) ptr = ptr.prev;
    }
    return ptr;
}
```

$\frac{n}{2}$

$\frac{n}{2}$

$O(n)$

$O(\frac{n}{2}) = O(\frac{1}{2}n)$

4) How does having a head and tail pointer effectively reduce run times in half

Worst case you can traverse in reverse thus worst case is $\frac{1}{2}n$ not n. Both are O(N) but this is a smaller O(N)... a slight optimization

5) How does doubly linked list reduce performance compared to a linked list

extra memory usage

6) What are the pros and const of doubly linked lists

Can Do Backwards

extra memory used.. more pointers to manipulate

Coding Problem:

The code can be found here under Session 3: feel free to copy and paste the code into your preferred development environment or you can clone the repository (if you already cloned it you can just pull changes)

https://github.com/jdanninger/CSDS233-Supplemental-Instruction

The basics of a doubly linked lists is written, you need to code a

Insert method

- Takes a given value and index and adds that value into that index (a lot of pointer manipulation)

Into Array

- Returns the current list in an array

The goal is to be comfortable modifying and traversing doubly linked lists