CS367 Homework 2

Lecture 1, Spring 2018

Due by 11:59 pm on Friday, March 2, 2018 (not accepted late)

In this page: Questions | Handing in   Related pages: Assignments

Questions

Homework assignments must be done individually. Collaboration on homework assignments is not allowed.

Question 1:

Assume that a `LinkedList` class has been implemented as a doubly-linked chain of nodes **with** a (dummy) header node. The `LinkedList` class has the following fields:

```
private DblListnode<E> items;  // pointer to the header node
private int numItems;
```

`LinkedList` uses the class `DblListnode` that includes the public methods `getData`, `setData`, `getPrev`, `setPrev`, `getNext`, and `setNext`.

Below is an implementation of a new method, `reverse`, for the `LinkedList` class. This method reverses the order the objects in a specified sublist. Consider this list: `[1, 2, 3, 4, 5]`. The call `reverse(1,3)` reverses the sublist `[2, 3, 4]`, returning `[1, 4, 3, 2, 5]`. Note that the call `reverse(0,size()-1)` reverses the entire list.

The `reverse` method operates in a recursive manner. Given a list `L`, it first swaps the very leftmost and rightmost items. It then recursively reverses the remaining list (excluding the two items that have been swapped.) Given `[1, 2, 3, 4, 5]` it swaps 1 and 5 to obtain `[5, 2, 3, 4, 1]`. Then the sublist `[2, 3, 4]` is reversed in a recursive call. 4 and 2 are swapped to obtain `[4, 3, 2]`. The remaining sublist, `[3]`, is trivially reversed, and our final list, after all recursive calls return, is: `[5, 4, 3, 2, 1]`

```
 public void reverse(int pos1, int pos2){
  // If pos1 == pos2, reversing a single list element does nothing
  // If pos1 > pos2, reversing an expty sublist does nothing
  if (pos1 >= pos2)
     return;

  // We swap the 1st and last items in the sublist,
  //  then recursively reverse the remaining sublist
  // We stop when the remaining sublist has size 0 or 1

  // Swap list items at pos1 and pos2
  E temp = remove(pos2);
  add(pos2, get(pos1));
  remove(pos1);
  add(pos1, temp);

  // Now recursively reverse remainer of sublist (if any)
  // The remaining sublist is from pos1+1 to pos2-1
  reverse(pos1+1, pos2-1);
}
```

Recall that if a position is invalid, the `get` and `remove` methods throw an `IndexOutOfBoundsException`.

For this homework, **complete a second version of `reverse`** that functions that same as the code above. However, your version must **directly change the chain of nodes by unlinking the nodes to be swapped and re-linking them into the chain in the appropriate way**. To receive full credit, your `reverse` method must use only `DblListnode` methods (and cannot use any `LinkedList` methods).

Be sure that your code works for all cases such as when the list is empty, has just one item, has just two items, or has more than two items. Also consider when the positions of items to be swapped are next to each other, or one of the items to be swapped is at the front or the end of the chain.

Note: If you run `reverse` on very long lists, you may get a "StackOverflowError". This means you have run out of space to store pending recursive calls. To give your program more stack space call the Java run-time system as:

```
java -Xss4m YourClass
```

This runs your class file with 4 megabytes of stack space. To get even more space change the digit 4 to the number of megabytes you want.

Question 2:

Give the worst-case time complexity for your method above in terms of N, the list size. Identify what aspect of the method characterizes the problem size. Write a brief justification for the time complexity you give. Include in your justification any assumptions you make about the complexity of any methods that are called by your implementation.

Handing in

Please include your name at the top your file.

Put your answers to both questions into one file named `Homework2` with the appropriate file extension, e.g., `Homework2.pdf` (see File Format for acceptable file formats).

Electronically submit your work to the Homework 2 tab on Canvas.