Workshop 6 (Midterm Review)

CSC 172 (Data Structures and Algorithms)
Spring 2018
University of Rochester
March 04 - March 07

Introduction

You can't cover all these questions during the workshop. Find out the topics those interest you and limit your discussion on those only. But before you start, review the questions on the last quiz.

Problem 1 (Using Java ArrayList, LinkedList, Stack, and Queue)

Until now, we have covered four data structures besides Arrays. These are ArrayList, LinkedList, Stack, and Queue. By this time, you should be pretty confident using these data structures. Also, you should know methods each of this data structures support. In Lab 4, you have implemented URArrayList and URLinkedList and those methods are quite similar to ArrayList and LinkedList provided by Java Collection framework. For Stack and Queue, we have discussed in lecture, how can you implement these using Arrays (or ArrayLists or Vectors) or LinkedList (both Singly and Doubly) as the underlying data structures. Also, we have learned how to solve a problem iteratively and recursively. It's time to put all these together. For the following few coding problems, you need to provide for each, **both an iterative and a recursive solutions**.

Note: Throughout the course, we will assume 0-based indexing. That is, the index of the first element is always '0'; The following `Node` Class is given:

```
class Node<E> {       // Doubly linked list node
public E e;         // Value for this node
public Node<E> n;  // Pointer to next node in list

// Constructors
Node(E it, Node<E> next) { e = it; n = inn; }
Node(Node<E> inp, Node<E> next) { n = next; }
}
```

- Implement a method which takes three Stacks `Stack<Integer> allInput, evenOutput, OddOutput` object. The method pops each element from the stack `allInput`. If the element is even, it pushes the element in evenOutput stack, otherwise, in oddOutput stack. `allInput` stack should be empty after the operation.
- Implement another method which performs the same operation, but instead of three stacks, it takes three queues as parameters.
- Given an `ArrayList`, write a function that returns the minimum element from the arraylist.
- Given an `ArrayList`, write a function that returns the k^{th} minimum element from the arraylist. Note: 0^{th} minimum element is the smallest element in the ArrayList.
- Given a `(Java) LinkedList`, write a function that returns the minimum element from the arraylist.
- Given a `(Java) LinkedList`, write a function that returns the k^{th} minimum element from the arraylist
- Given a `LinkedList` of `Nodes`, write a function that returns the minimum element from the arraylist

- Given a `LinkedList` of `Nodes`, write a function that returns the minimum element from the linkedList
- Given a `LinkedList` of `Nodes`, write a function that returns sum of the elements

Problem 2 (Questions to Ponder)

- ArrayList implementation: what is the difference between `size` and `capacity`.
- Given a `LinkedList` of `Nodes`, how can you use it to implement a Stack?
- Given a `LinkedList` of `Nodes`, how can you use it to implement a Queue?
- Given a `Java LinkedList`, how can you use it to implement a Stack?
- Given a `Java LinkedList`, how can you use it to implement a Queue?
- Given a `ArrayList`, how can you use it to implement a Stack?
- Given a `ArrayList`, how can you use it to implement a Queue?

Problem 3 (Induction)

Review: Workshop #2.
Prove the following by induction. For any question, you may use/assume the result of previous questions without proving them again.

1. $$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$
2. $$\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$$

Problem 4 (Recurrence Tree)

Review: Workshop #4.
Use the recurrence tree method to solve the following recurrences.

1. $T(n) = 2T(n/2) + 1$
2. $T(n) = 4T(n/2) + n$

Problem 5 (Project 2 / Shunting Yard)

Convert the following infix expression into postfix using shunting yard algorithm and then evaluate the postfix expression using stack and list.
Infix Expresson: 1 + (2 - 3 + 4) * 5 + 6 (This is from Quiz 6)

Problem 6 (Generics)

Please refer Lab 2

Problem 7 (Efficiency)

Growth Rate

Order the following functions by growth rate:

$N, \sqrt{N}, N^{1.5}, N^2, N \log N, N \log \log N, N \log^2 N, N \log N^2, 2^N, 2^{N/2}, 37, N^2 \log N, N^3$.

Indicate which functions grow at the same rate.

Runtime

For each of the following six program fragments give an analysis of the running time (Big-Oh will do).

```
sum = 0;
for( i = 0; i < n; i++ )
sum++;
```

```
sum = 0;
       for( i = 0; i < n; i++ )
              for( j = 0; j < n; j++ )
                     sum++;
```

```
sum = 0;
for( i = 0; i < n; i++ )
       for( j = 0; j < n * n; j++ )
              sum++;
```

```
sum = 0;
for( i = 0; i < n; i++ )
       for( j = 0; j < i; j++ )
              sum++;
```

```
sum = 0;
for( i = 0; i < n; i++ )
       for( j = 0; j < i * i; j++ )
              for( k = 0; k < j; k++ )
                     sum++;
```

```
sum = 0;
for( i = 1; i < n; i++ )
       for( j = 1; j < i * i; j++ )
              if( j % i == 0 )
                     for( k = 0; k < j; k++ )
                            sum++;
```

Problem 8 (Recursion)

Write iterative and recursive methods solving the following problems. Practice writing 'real' Java code.

1. Sum of numbers 0 through n
2. Reverse a string (no need to modify the given string)
3. Return the binary representation (as a string) of a given number.

Problem 9 (Study Quizzes)

Review all the quiz questions and labs for the midterm.