American University of Armenia, CSE
CS121 Data Structures
Spring 2024

Homework Assignment 2

Due Date: Sunday, March 17 by 23:59 electronically on Moodle

Solve the programming tasks using Java, following good coding practices and required format.

1. **(BinarySearch | 15 points)** Create a class `BinarySearch` that will only contain one static method `binarySearch` that, given a `List` of generic elements that can be *compared* to each other (and are sorted in non-decreasing order) and a target elements, *efficiently* finds the element using the *binary search* algorithm.

   What makes this implementation efficient compared to the array-based implementation provided in the slides?

2. **(PositionalSorting | 30 points)** The task is to implement various sorting algorithms for a positional list of `Character`s. All of the methods should be implemented *in-place*.

   (a) Implement a method *selectionSort* that given a `Character` positional list PL, sorts it using *selection sort*. You are not allowed to swap the values (elements) at the positions. At each iteration, you should remove the position with smallest value and insert its value before the current position.

   (b) Implement a method *insertionSort* that given a `Character` positional list PL, sorts it using *insertion sort*. You are not allowed to swap the values (elements) at the positions. At each iteration, you should remove the current position (if it is not in its correct order at the moment) and insert its value into its correct position within the sorted portion.

   (c) Implement a method *bubbleSort* that given a `Character` positional list PL, sorts it using *bubble sort*. You are not allowed to swap the values (elements) at the positions.

   (d) Add a main method and test all the above methods on `LinkedPositionalList` objects.

3. **(ArrayList | 20 points)** Change the inner `ArrayIterator` class definition and implementation so that it becomes an efficient `ListIterator` (i.e., it should implement `ListIterator` instead of `Iterator`). Make sure to go over the documentation of `ListIterator` and make your methods throw appropriate exceptions when needed. All the methods of a `ListIterator` should be implemented (except for `forEachRemaining`).

4. **(LinkedList | 35 points)** Create a class `LinkedList` that efficiently implements the `List` interface using a doubly-linked structure. Use the concept of `header` and `trailer` for your implementation.

   Your implementation will be a hybrid between our implementations of the `LinkedPositionalList` and `ArrayList`. Make sure to copy all the necessary inner classes, instance variables and helper methods from these classes into this one. Add a method into the `Node` class for setting an element within a node.

   The class should only provide an efficient iterator over the elements.

   Include a helper method that returns a node at a given index i (assuming i is a valid index). Make sure to use the method when appropriate.

   Specify and justify the running times of all the main `List` methods in this implementation.

   In a `main` method, create two integer `List` objects: one of type `LinkedList` and one of type `ArrayList` (keep this order in all of the tests below).

In each list, insert all elements from 10000 till 0, each time inserting the element at index 0 (to get a sorted sequence from 0 till 10000)). Time the duration of this process for each list using `System.nanoTime()` and report the results. Run the program several times, and discuss the results that you got. Do they make sense?

Now, use the `binarySearch` method from the `BinarySearch` class to search for element 10001 in each list. Time the duration of this process for each list using `System.nanoTime()` and report the results. Run the program several times, and discuss the results that you got. Do they make sense?