Tafita Rakotozandry
CS150 Lab
09/11/2020

Lab 4 Report

INTRODUCTION

In this lab, a generic class called RandomStuffContainer was created. This class possesses the addToFront, addToBack , selectionSort ,bubbleSort and toString method. Another class named RandomIntegerContainer was also created which extends RandomStuffContainer. This subclass possesses an additional method called sum which returns the sum of all elements. An experiment controller class was created to measure the time performance of the sorting algorithm in RandomStuffContainer. The goal of the lab is to deepen understanding of generic classes , abstract classes and interfaces.

Unit Tests

The functions of each main methods are:

- addToBack(): allows to add elements to the back of the container
- addToFront(): allows to insert an element at the beginning of the container
- addSorted(): allows new insert new element on a sorted list
- selectionSort(): allow to sorts the list using selection sort method.
- bubbleSort(): allow to sorts the list using bubble sort technique

In addition to them, there were additional methods which were created to handle the data manipulation in general. For example, the reverseList() method was created because it was

interesting to know how the sorting algorithm would behave if the items are sorted in a descending order.

The 5 main methods of the classes were all tested in addition to the reverseList() method. The screenshot of the unit testing is as follows:

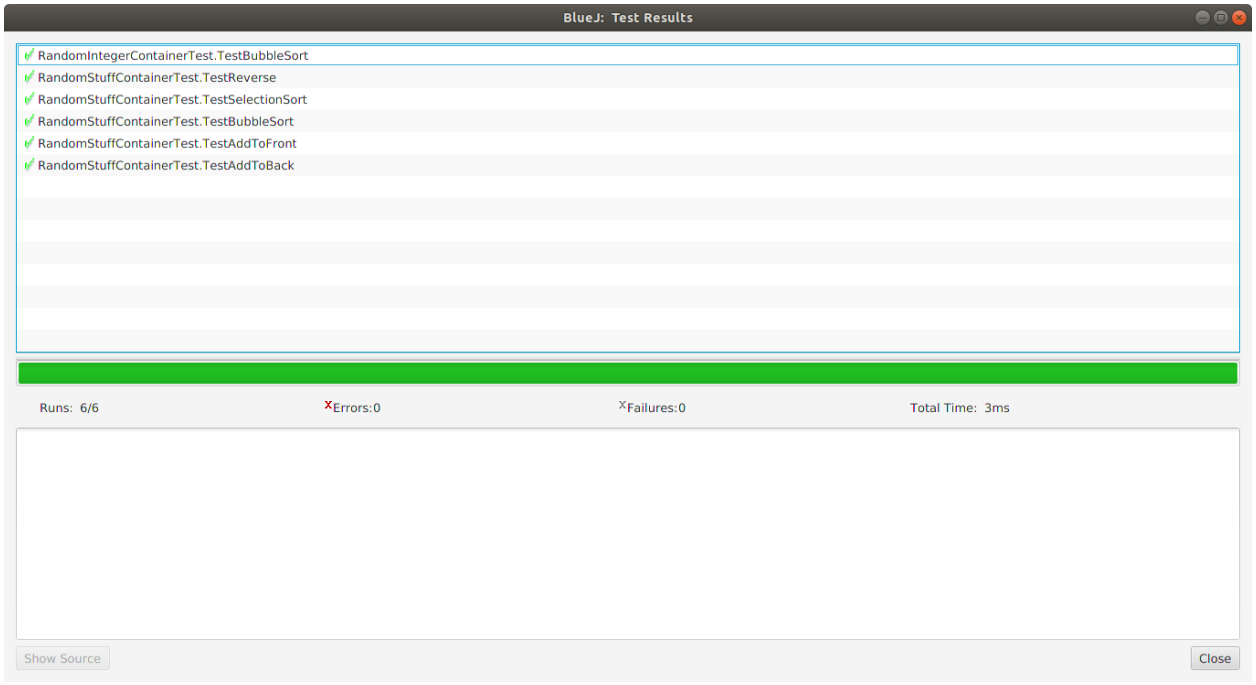

Figure 1: Screenshot Unit Testing Window

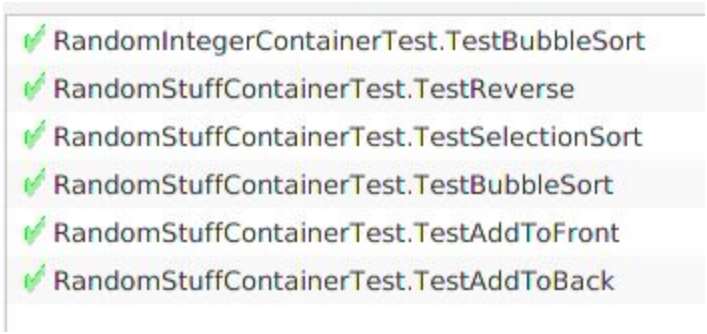

Figure 2: A zoom of the Unit Testing Window

Required Output

When comparing the two bubble sort and selection sort in the ExperimentController, three cases of situations were taken into consideration:

- Items on the list are set randomly (average case)
- Items on the list are sorted but in an ascending order (best case)
- Items on the list are sorted but in a descending order (worst case)

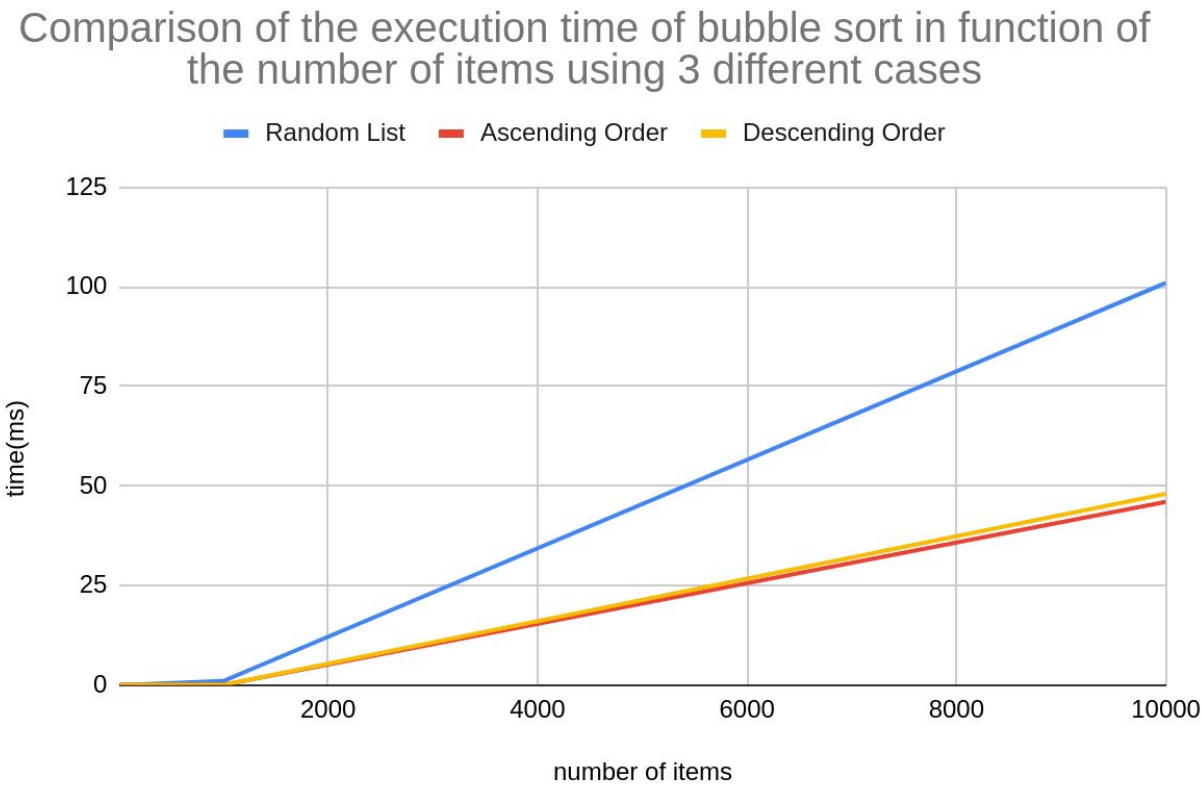

Figure 4: Comparison of the execution time of selection bubble in function of the different cases

Based on the graph above, we can interpret that sorting a random list takes much more time than sorting a sorted list. The graph also shows that an ascending order list and descending order list

takes almost the same time to the algorithm. In theory, the best case of a bubble sort is O(n) and the worst case takes O(n^2) . However, an ascending order list is not a worst case for a bubble sort because it compares element by element.

Figure 4: Comparison of the execution time of selection sort in function of the different cases

The graph above shows the execution time of selection sort in function of the three cases that we considered. They overlap which makes sense because no matter how the elements are listed, the selection sort will go through the elements one by one. The big O of a selection sort is O(n^2) for each case.

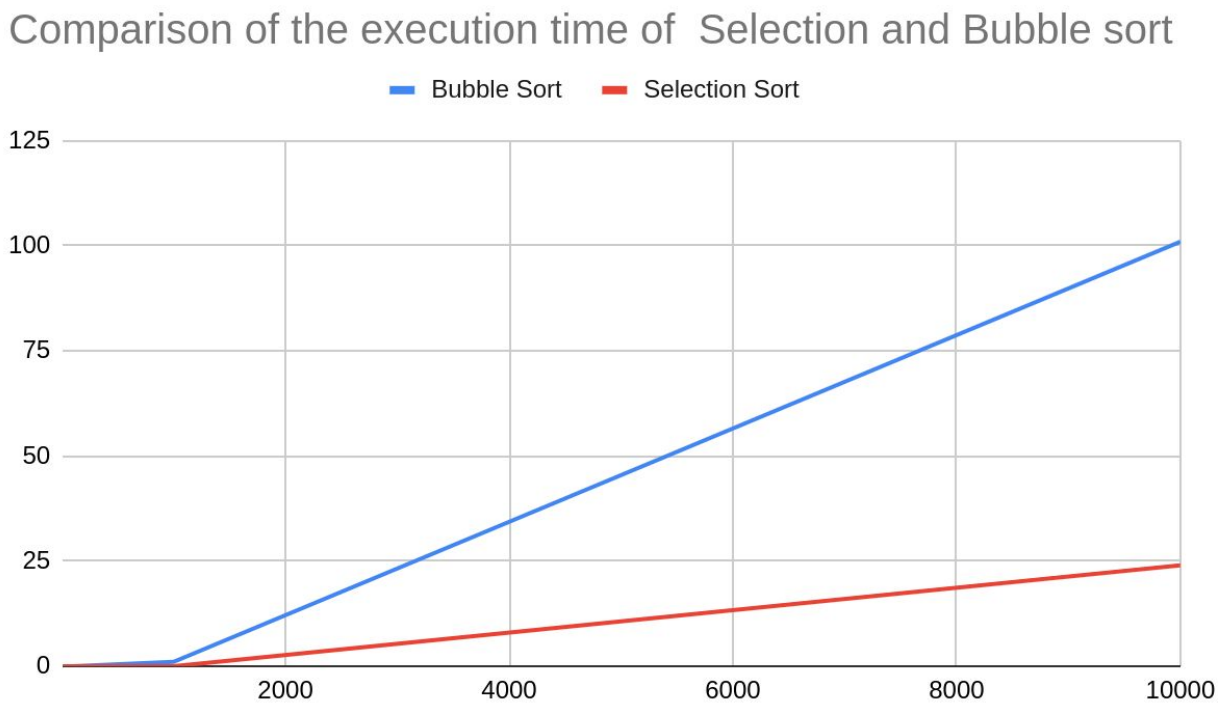

Figure 5:Selection Sort in comparison to BubbleSort

The graph above shows the execution time comparison between selection and bubble sort in case of a random list container. Bubble sort seems to take much longer than selection sort.

Trouble Report

N/A

References

Admin. (2020, July 28). Selection sort java ---- Flower Brackets ---- code here. Retrieved September 05, 2020, from https://www.flowerbrackets.com/selection-sort-java/

Oracle (2019). ArrayList Documentation. Oracle.Com. https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html

Michael Kölling (2015). U0nit Testing in BlueJ https://www.bluej.org/tutorial/testing-tutorial.pdf

Weiss, M. A. (1998). Data structures and problem solving using Java. ACM SIGACT News, 29(2), 42-49.