Name, SID, Date .....

In Class Assignment 7: Bubble Sort

Benjamin Sanders, MS November 25, 2020

1 Introduction

You will need to work individually to complete this assignment. Write your name at the top of all pages for this assignment. Turn in all work to Blackboard on or before the deadline to receive credit.

You may use additional libraries and online resources, if you get them approved in writing, over email, from the instructor first. If you have received approval from the instructor, write the approved libraries and any references in the space below.

.....
.....
.....
.....

2 Assignment Description

2.1 Big Picture

This algorithm is the most popular first sorting algorithm for programmers to implement.

2.2 Algorithm Implementation

Implement the following algorithm in Java, using the Vector data structure for any 1-D array, 2-D array, or linear algebra purposes.

Input: A sequence of n numbers $\langle a_1, a_2, \ldots, a_n \rangle$.

Output: A permutation (reordering) $\langle a'_1, a'_2, \ldots, a'_n \rangle$ of the input sequence such that $a'_1 \leq a'_2 \leq \cdots \leq a'_n$.

BUBBLESORT(A)

```
1  for i = 1 to A.length − 1
2      for j = A.length downto i + 1
3          if A[j] < A[j − 1]
4              exchange A[j] with A[j − 1]
```

```
public class BubbleSort {
    public static int[] sortFunction(int[] a) {
        int n = a.length;
        for (int i = 0; i < n - 1; i++) {
            for (int j = n - 1; j > i; j--) {
                if (a[j] < a[j - 1]) {
                    // Swap elements if they are in the wrong order
                    int temp = a[j];
                    a[j] = a[j - 1];
                    a[j - 1] = temp;
                }
            }
        }
        return a;
    }
    public static void main(String[] args) {
        int[] data = {1, 3, 5, 2, 6};
        int[] sortedData = sortFunction(data);

        System.out.println("Sorted Array:");
        for (int i = 0; i < sortedData.length; i++) {
            System.out.println(sortedData[i]);
        }
    }
}
```

2.3 Time Complexity Analysis

Where n is the number of data points in A, analyze the time complexity of the given algorithm with respect to n. Write the result of your analysis in big-O notation, i.e. $O(n^2)$ in the space below.

[redacted] O(n^2)

.....

2.4 Space Complexity Analysis

Where n is the number of data points in A, analyze the space complexity of the given algorithm with respect to n. Write the result of your analysis in big-O notation, i.e. $O(n \cdot log(n))$ in the space below.

O(n)

.....

Name, SID, Date .....

2.5 Optimize the Algorithm for a Purpose

Choose an optimization, either in time or in space, for the given algorithm. Write your intended big-O notation, i.e. $O(1)$, $O(n)$, etc., in the space below, and write N/A in the other space.

- Time Complexity: O(n)
- Space Complexity:)(n)

What application would benefit from the purpose of the above optimization? Why? Write two sentences to answer these questions in the space below.

The optimized BubbleSort algorithm may be useful in scenarios where the input data (list of names, numbers) is already partially sorted or nearly sorted.
In such cases, the early exit conditions can lead to improved performance

2.6 New Algorithm Design and Implementation

In the space below, design an algorithm that achieves the same purpose of the given algorithm, but includes the optimization you have specified above. Use pseudocode written in a style similar to the given algorithm, and implement it in Java. You may use as many additional pages as necessary for this purpose.

```
public class BubbleSortOptimization {
    public static int[] sortFunction(int[] a) {
        int n = a.length;
        // Optimization 1: Use a boolean flag to check if any swaps were made
        boolean swapped;
        // Optimization 2: Reduce the upper bound of the inner loop to avoid unnecessary comparisons
        for (int i = 0; i < n - 1; i++) {
            swapped = false;
            for (int j = n - 1; j > i; j--) {
                if (a[j] < a[j - 1]) {
                    // Swap elements if they are in the wrong order
                    int temp = a[j];
                    a[j] = a[j - 1];
                    a[j - 1] = temp;
                    swapped = true;
                }
            }
            // Optimization 3: If no swaps were made, the array is already sorted
            if (!swapped) {
                break;
            }
        }
        return a;
    }
    public static void main(String[] args) {
        int[] data = {1, 3, 5, 2, 6};
        int[] sortedData = sortFunction(data);

        System.out.println("Sorted Array:");
        for (int i : sortedData) {
            System.out.println(i);
        }
    }
}
```

3 What to Turn In

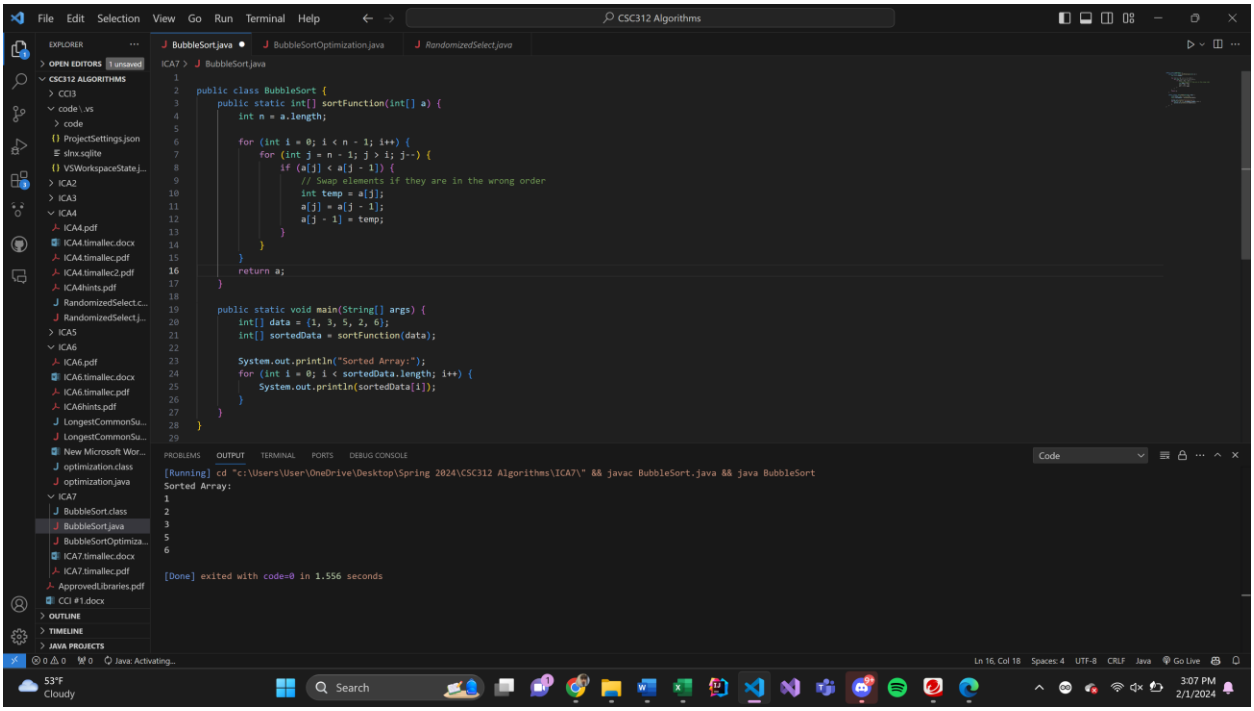
Turn in one PDF or Word document on Blackboard, containing the following items.

1. All pages scanned or photographed of the In Class Assignment completed document.
2. Any additional pages you used to complete the assignment.
3. All code created for the assignment, along with test cases.
4. One statement indicating which parts of your implementation(s) are working, and which parts are not.
5. Screenshots demonstrating the code working, if it is working.

BUBBLE SORT

```java
public class BubbleSort {
    public static int[] sortFunction(int[] a) {
        int n = a.length;

        for (int i = 0; i < n - 1; i++) {
            for (int j = n - 1; j > i; j--) {
                if (a[j] < a[j - 1]) {
                    // Swap elements if they are in the wrong order
                    int temp = a[j];
                    a[j] = a[j - 1];
                    a[j - 1] = temp;
                }
            }
        }
        return a;
    }

    public static void main(String[] args) {
        int[] data = {1, 3, 5, 2, 6};
        int[] sortedData = sortFunction(data);

        System.out.println("Sorted Array:");
        for (int i = 0; i < sortedData.length; i++) {
            System.out.println(sortedData[i]);
        }
    }
}
```

SCREENSHOT OF OUTPUT:

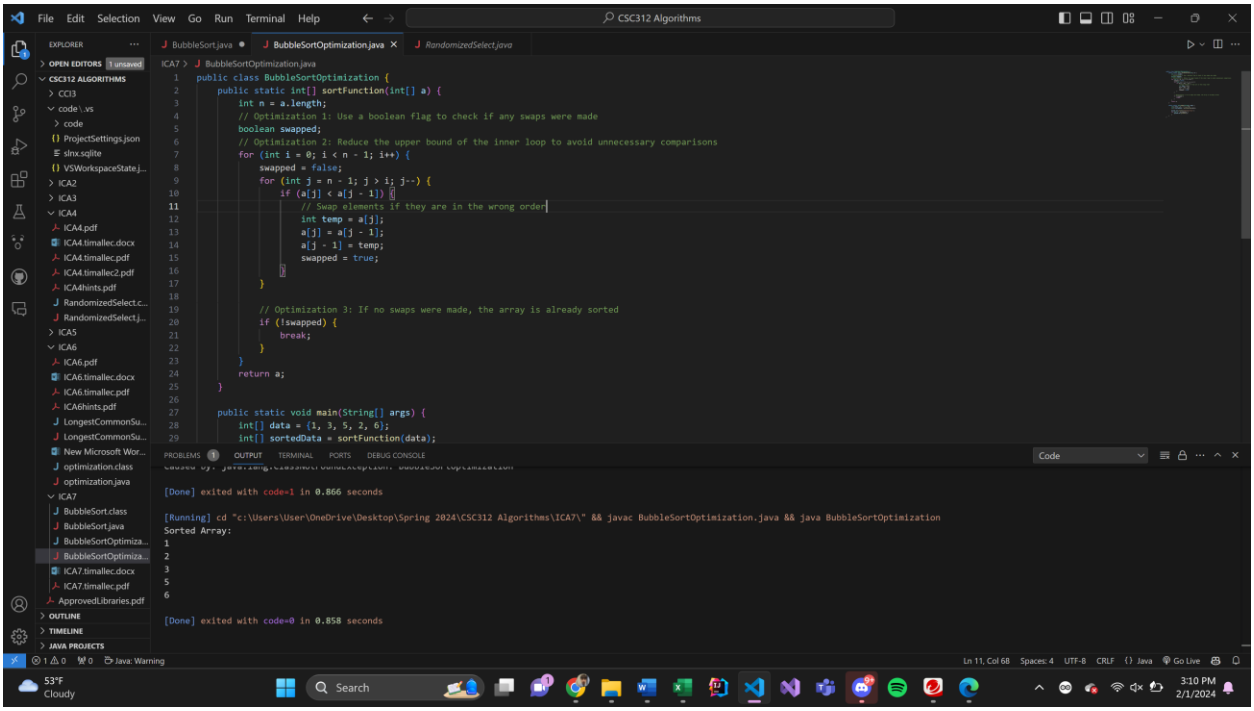


BUBBLE SORT OPTIMIZATION

```
public class BubbleSortOptimization {
    public static int[] sortFunction(int[] a) {
        int n = a.length;
        // Optimization 1: Use a boolean flag to check if any swaps were made
        boolean swapped;
        // Optimization 2: Reduce the upper bound of the inner loop to avoid
unnecessary comparisons
        for (int i = 0; i < n - 1; i++) {
            swapped = false;
            for (int j = n - 1; j > i; j--) {
                if (a[j] < a[j - 1]) {
                    // Swap elements if they are in the wrong order
                    int temp = a[j];
                    a[j] = a[j - 1];
                    a[j - 1] = temp;
                    swapped = true;
                }
            }

            // Optimization 3: If no swaps were made, the array is already sorted
            if (!swapped) {
                break;
            }
        }
```

```
        return a;
    }

    public static void main(String[] args) {
        int[] data = {1, 3, 5, 2, 6};
        int[] sortedData = sortFunction(data);

        System.out.println("Sorted Array:");
        for (int i : sortedData) {
            System.out.println(i);
        }
    }
}
```

SCREENSHOT BUBBLE SORT OPTIMIZATION OUPUT:



ALL CODE WORKS AS INTENDED