CSC 323-01 Algorithm Design and Analysis, Fall 2019
Instructor: Dr. Natarajan Meghanathan

Quiz 2: Evaluation of the Bubble Sort Algorithm with and without the Optimization Approach

Due by: Oct. 3rd, 11.59 PM

In the pseudo code analysis of the Bubble Sort algorithm, we discussed an optimization approach to minimize the number of comparisons. The optimization approach is that if no swapping happens to be done in an iteration, then the array is already sorted and we break from the algorithm without going through the rest of the iterations. In this quiz, you are going to examine whether such an approach works for really large arbitrary arrays.

You are given a startup code with the main function already setup to generate the array and measure the actual sorting time (in real-time) as well as the average number of comparisons. Note that the return type of the Bubble Sort function in the code is *int* corresponding the number of comparisons incurred.

Your task is to implement the Bubble Sort function with and without optimization and measure the actual sorting time (in milliseconds) and the average number of comparisons for array size values ranging from 1000 to 10000, in increments of 1000. Run your experiments for two different values (50 and 5000) for the maximum value for an element. The number of trials is 50 for each combination of array size and maximum value.

Submission:

(1-60 pts) Submit two .cpp files: one for the Bubble Sort function implemented without any optimization and another for the Bubble Sort function implemented with optimization.

(2-40 pts) Submit a PDF file featuring the following:

(i) A table (see below for the layout) displaying the actual run time values (in milliseconds) for the Bubble Sort algorithm with and without optimization for array size values of 1000 to 10000, in increments of 1000, for maximum values of an element as 50 and 5000.

	Bubble Sort (No Optimization)		Bubble Sort (with Optimization)	
	Max. Val = 50	Max. Val = 5000	Max. Val = 50	Max. Val = 5000
1000				
2000				
3000				
...				
...				
10000				

(ii) A table (with a similar layout as the one above) displaying the averaging number of comparisons for the Bubble Sort algorithm with and without optimization

(iii) Plot the results of (i) in Excel. You should generate two plots (one plot for Max. Value = 50 and another plot for Max. Value = 5000) comparing the actual run time (in milliseconds) for Bubble Sort with and without optimization.

(iv) Plot the results of (ii) in Excel. You should generate twp plots (one plot for Max. Value = 50 and another plot for Max. Value = 5000) comparing the average number of comparisons for Bubble Sort with and without optimization.

(v) Interpret/Explain the results observed in (i)-(iv), including the following aspects:

- -- Did the use of the optimization approach appreciably reduce the actual run time and/or the average number of comparisons?
- -- We saw the overall theoretical time complexity of Bubble Sort is $\Theta(n^2)$ for an array of size 'n'. Do the results/plots for the actual run time (refer to (iii)) illustrate this?
- -- Are the plots in (iii) and (iv) of the same trend or different?