**ASSIGNMENT 3 – Quick Sort**

|  |
| --- |
| Topics |
| * Partition * QuickSort algorithm |
| Readings |
| * CLRS, Chapter 6, Chapter 7.1, 7.2 |
| Instructions | |
| 1. Select a **partner** and inform instructor who you will work with  2. Do the problems and answer the questions listed in the next section   * Keep in mind Guidelines on plagiarism.   3. Follow instructions for submitting your work.  PROBLEMS AND QUESTIONS | |
| Problems and Questions |
| Part A Tracing (10 pts) |

Apply quicksort to sort the list “Las Vegas”, “Boston”, “Orlando”, “San Francisco”, “Montpelier”, “Dallas”, “Austin”, “Columbus”, “Hartford”, “Miami” in alphabetical order. Show the steps: how the partition is done on each step

We have chosen the initials of every word to make it easier for us to run the algorithm.

* A close up of text on a whiteboard

  Description automatically generatedPivot Point: The right most letter after each step

|  |
| --- |
| Part B Quick Sort implementation ( 60 pts) |
| Implement QuickSort algorithm as a Java program with generics. For partition, select a pivot by median-of-three rule. Apply insertion sort for an array with length not exceeding 4.For a demo, do the same as you did in Assignment 2 (extract words from the text file) |

|  |
| --- |
| Bonus problem 1 (5 pts) |

Run quicksort on arrays of random integers (range 1 to 50) of increasing sizes: . Record running times and describe your observations.

According to my observations, although the size of the array grows exponentially, the time it takes to sort the array does not increase as rapidly.

**We can see that when we get to run 10^6, the program overflows. Apparently, quicksort the limit of quicksort lies between 10^5 and 10^6. The overflow is to expect due to its instability. Also, we noticed that quicksort and insertion sort are very similar when it comes to time efficiency. If we apply quicksort and insertion sort to the same list, we wouldn’t see much of a difference (if any).**

|  |
| --- |
| Bonus problem 2 (5 pts) |

Implement the alternative version of QuickSort as described in the slide 14 of Lec3a: stop quicksort when subarray is not larger than 4 elements, and then apply insertion sort to the whole array.

2. **Summary questions:**

a. What concepts did you have trouble with? What still confuses you?

**We did not have big problems learning the concepts, but the pivot point was a bit confusing. In the real world, choosing a pivot point for our application might be challenging because there are so many options. We could choose the last number to be the pivot point, the average, the middle number, the first element and so on. There are so many options.**

b. Suggestions for improving this assignment in the future?

**In the future, finding the most optimal pivot point for a problem should be one of the bonus exercises.**

|  |
| --- |
| Submitting your work |

1. Make sure that your name(s) are in all your files.
2. If you have more than one file for your solution, make a .zip file for your project
3. In Blackboard, attach your solution file to the submission for this assignment.

GUIDELINES ON

|  |
| --- |
| Guidelines on Plagiarism in Computer Science |

Outlined in the Syllabus