**Department of Computer Science and Engineering**

|  |  |
| --- | --- |
| **Course Code: CSE221** | **Credits: 1.5** |
| **Course Name: Algorithms** | **Semester: Fall’18** |

**Lab 02  
Sorting Algorithms**

1. **Topic Overview:**

Sorting is important in programming for the same reason it is important in everyday life. It is easier and faster to locate items in a sorted list than unsorted. In this lab students, will familiarize themselves with different sorting algorithms and their runtimes. They will write a java class (MyArray.java) that implements the following sorting algorithms: Insertion sort, Merge sort and Quicksort. Each of the sorting algorithms will take an array as an input and return the sorted array. The required algorithms are provided in the tasks below. Students will calculate and compare the time required to execute each algorithm. The MyArray class should not have any main method. The main method should be implemented in the SortTest class.

1. **Lesson Fit:**

There is prerequisite to this lab. Students must have a basic idea on the following concepts:

* 1. Constructors
  2. Recursion

1. **Learning Outcome:**

After this lecture, the students will be able to:

* 1. Implement different sorting algorithms
  2. Learn how different sorting algorithms perform in different scenarios

1. **Anticipated Challenges and Possible Solutions**
   1. Task 2: Array index may be out of bounds

**Solutions:**

* + 1. Debug to see if proper indexing has been done
    2. Recheck the loop condition
  1. Task 2-5: The original array has been modified

**Solutions:**

* + 1. Maintain a separate array for carrying out the sorting operation

1. **Acceptance and Evaluation**

Students will show their progress as they complete each problem. They will be marked according to their class performance. There may be students who might not be able to finish all the tasks, they will submit them later and give a viva to get their performance mark. The mark distribution for the lab will be as follows:

Code: 05

Viva: 05

1. **Activity Detail**
   1. **Hour: 1  
      Discussion:**Explain why sorting is needed and guide the students to finish the requirements needed in future tasks. **Problem Task:**
      1. Task 1-2 (Page 3-4)
   2. **Hour: 2**

**Discussion:**

Explain different sorting algorithms and check the progress of the students.

**Problem Task:**

* + 1. Task 3-4 (Page 4- 6)
  1. **Hour: 3**

**Discussion:**

Check task 3 and 4 while the students continue with the rest.

**Problem Task:**

* + 1. Task 5 ( Page 6)

1. **Home tasks**
   1. Task 6 (Page 6)
   2. Unfinished tasks

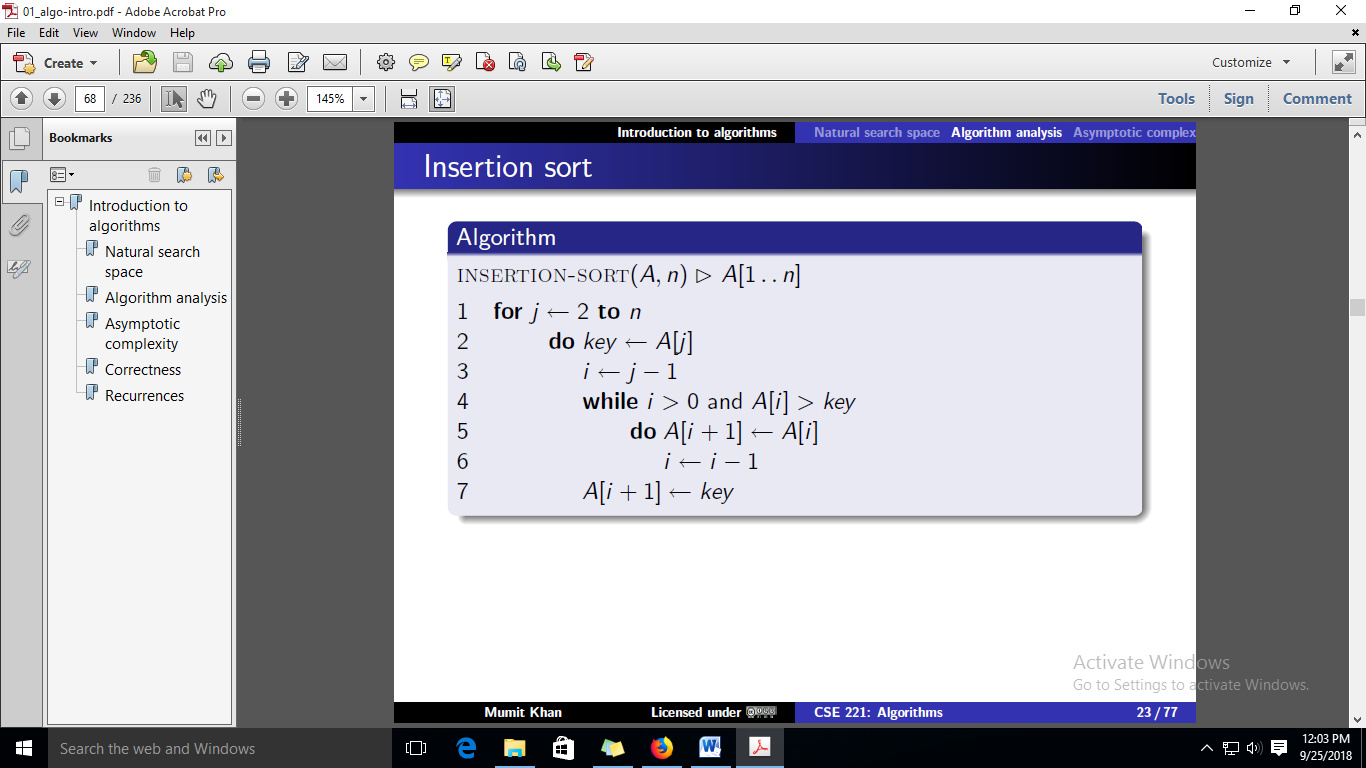
**Lab 2 Activity List**

**Task 1**

1. Create two classes “MyArray.java” and “SortTest.java”
2. Create the following methods in MyArray.java: insertionSort(), mergeSort(), quickSort() and toString(). The sorting algorithms will return the sorted array and toString will return a String. [**You may need to implement other methods depending on your requirement]**
3. MyArray.java will also have an integer array and a constructor that takes an array as parameter.
4. SortTest.java will have the main method and the following:
   * 1. Create an array of length 10 with the following elements: 3, 5 , 10, 23, 25, 8, 7, 9, 50, 47
     2. Create a MyArray Object.
     3. Sort the array using all the three sorting methods. (Do not change the original array)
     4. Print the result of each sorting using the toString method
     5. Print the time taken to sort in each of the three cases. **[Hint: System.currentTimeMillis(), 1 s = 1000 ms]**
     6. Finally use Java’s “sort” method from the “Array” class, print the sorted array and time taken to sort.

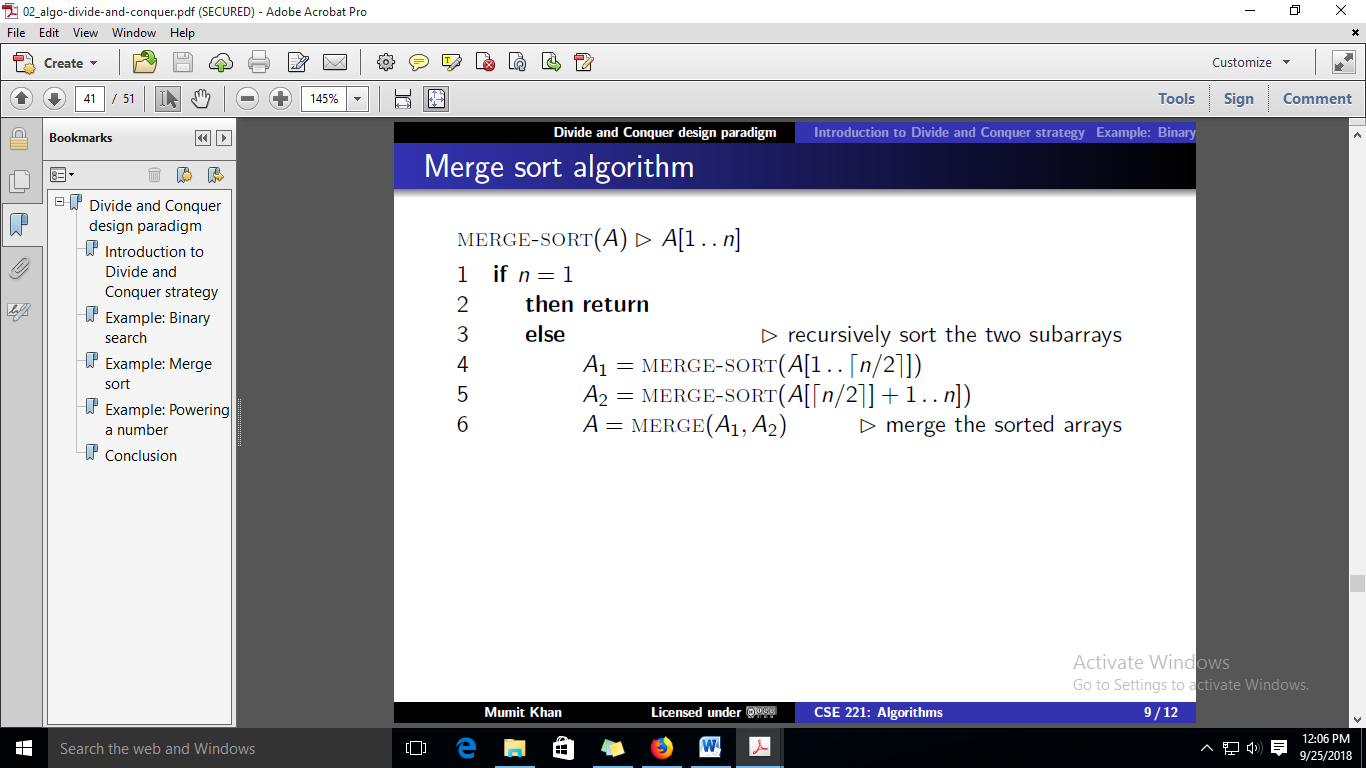
**Task 2**

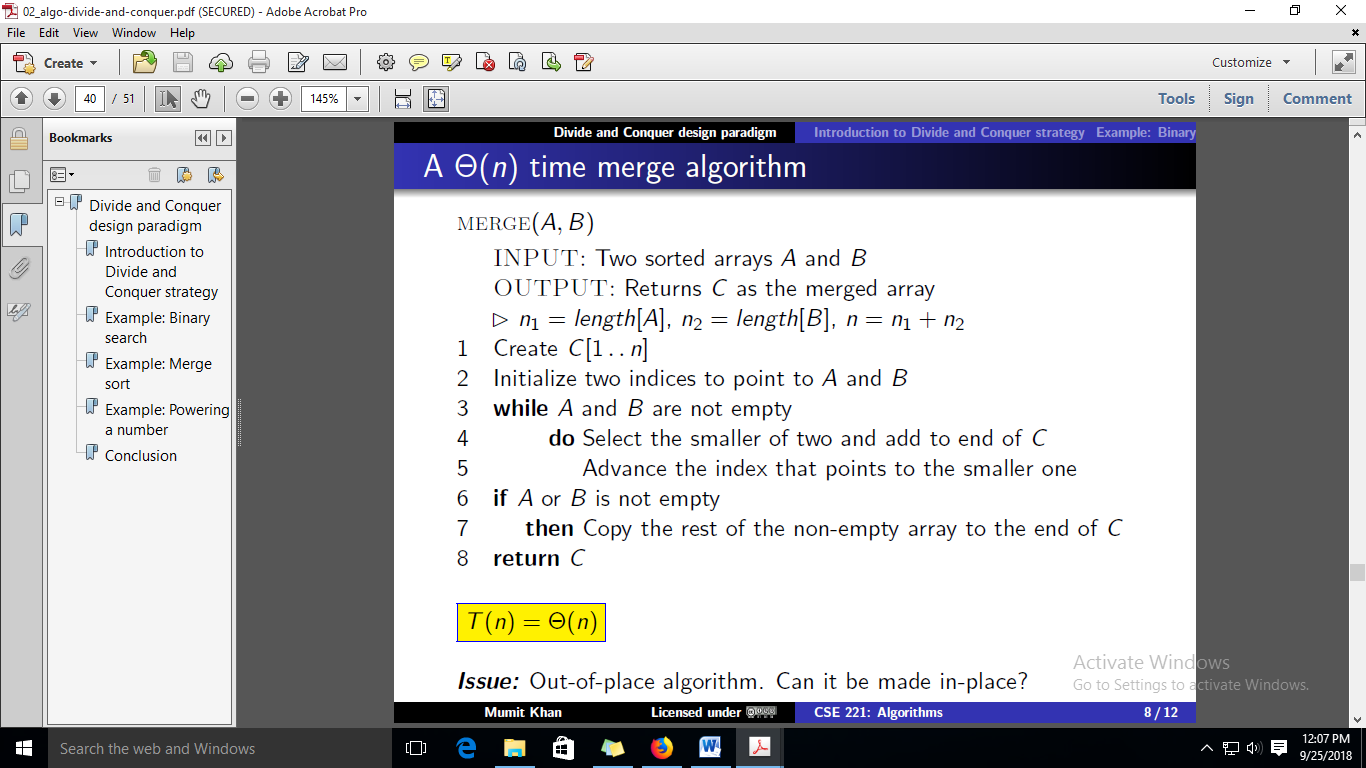
Study the algorithm below and implement insertionSort method in MyArray.java class. After sorting print both the unsorted array and sorted array and also the time it takes to complete sorting.



**Task 3**

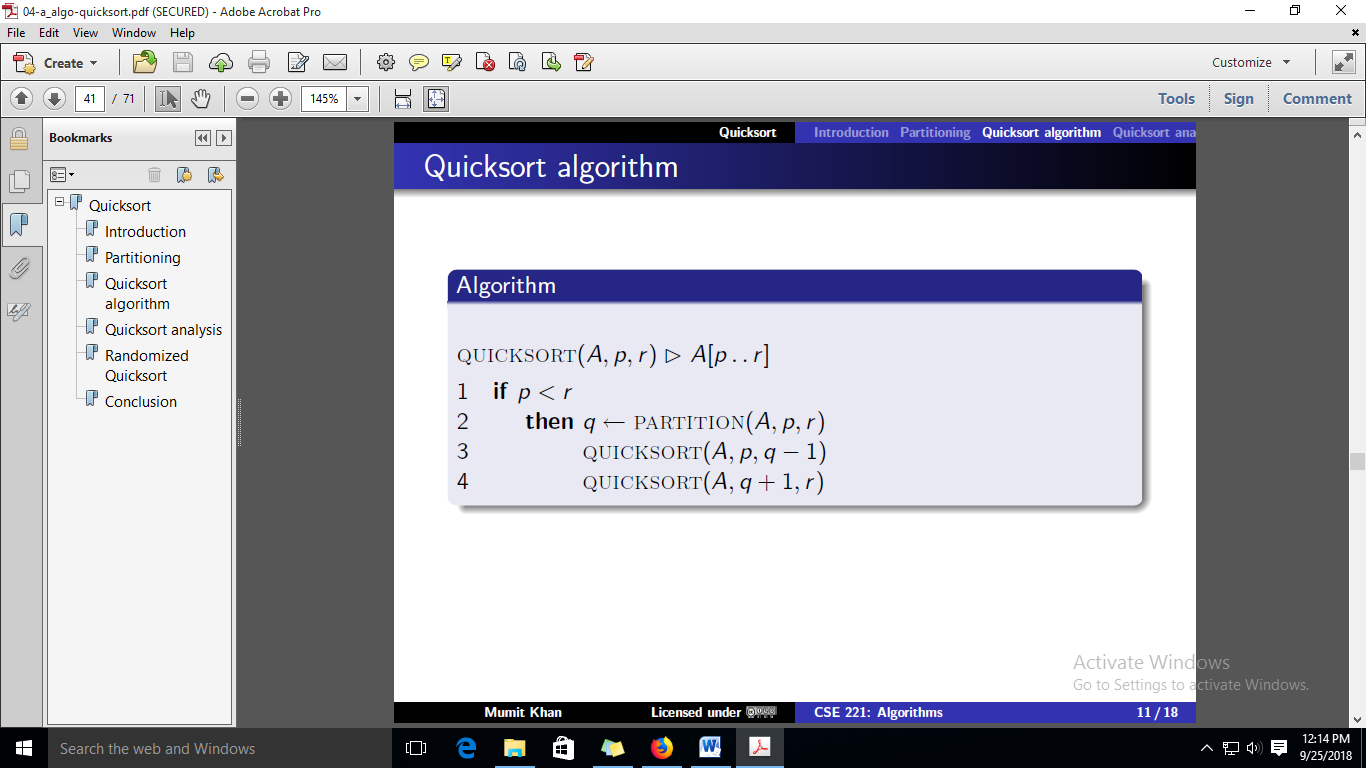
Study the algorithms below and implement mergeSort method in MyArray.java class. Additionally, you will also need to implement the “merge” method. After sorting print both the unsorted array and sorted array and also the time it takes to complete sorting.

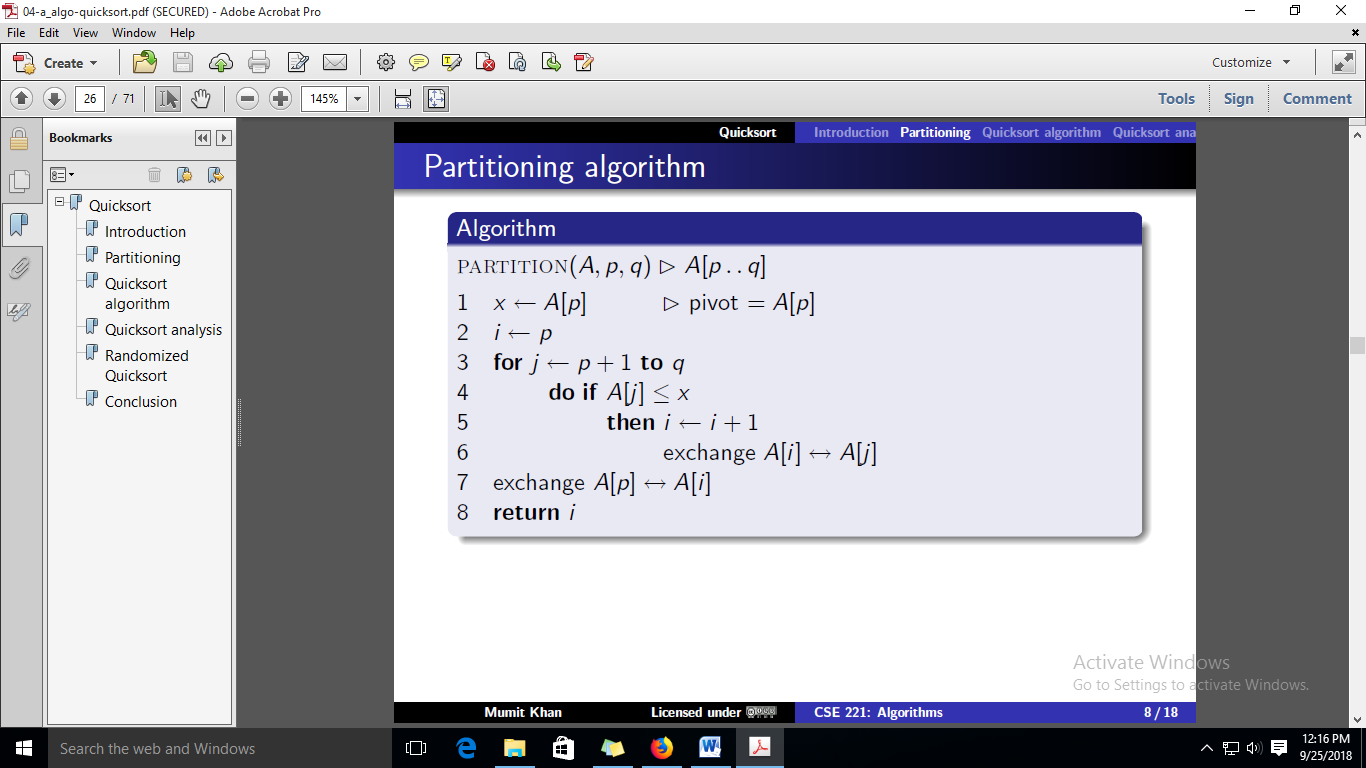




**Task 4**

1. Study the algorithm below and implement quickSort method in MyArray.java class. Additionally you will also need to implement the “partition” method. After sorting print both the unsorted array and sorted array and also the time it takes to complete sorting.





1. Implement an algorithm “findK” that uses the “partition” algorithm to find the kth (smallest) element from an array without sorting. E.g. for the array in our example, the 5th element will be “9”

**Task 5**

Implement the randomized quicksort algorithm. (i.e. use random number to pick the next pivot)

**Task 6**

Once you learn the heapsort algorithm in the theory lecture, implement it in the MyArray class. You will need to implement two methods buildMaxHeap and heapsort.