# Lab Assignment 22 (Algorithm Analysis)

1. Write a program named **Lab22A** that will create an ArrayList and a TreeSet of 1000 items and will compare the time it takes to fill them, search for a number, and remove a number.   
   (Remember that you can find reviews of dealing with ArrayLists in the Java user manual as well.)
   * Create a void method that will:
     1. Accept an ArrayList of integers and a TreeSet of integers as parameters (Remember to use <Integer> with both)
     2. Fill the ArrayList with 1000 random numbers that have values between 0 and 2000. This list will not be sorted.
     3. Record how much time (in nanoseconds) it took to fill the whole ArrayList and print that value with a descriptive label. (Remember that the sample code from the Ch 22 lecture showed how to determine the time for an operation.)

However, use **System.nanoTime()** instead of **System.currentTimeMillis()** when recording start and end time. The time will be in nano seconds.

* + 1. Fill the TreeSet with 1000 random numbers that have values between 0 and 2000. This list will not be sorted.  
       NOTE – the ArrayList & the TreeSet will have different values
    2. Record how much time (in nanoseconds) it took to fill the whole TreeSet and print that value with a descriptive label.
  + Create a void method that will:
    1. Accept an ArrayList of integers and a TreeSet of integers as parameters
    2. Ask the user for a number between 1 and 2000 to search for.
    3. Use the .contains method to see if the number is in the ArrayList and print a statement to say if the number was there.
    4. Record the time it took to execute the if statement and print that time with a label. (Meaning you need to set your starttime before the if statement and endtime after it.)
    5. Use the .contains method to see if the number is in the TreeSet and print a statement to say if the number was there.
    6. Record the time it took to execute the if statement and print that time with a label.

(NOTE – Try entering 2500 one time when you run and test your program because it definitely won’t be there, so it will be a worst-case scenario.)

* + In the main method,
    1. Declare an ArrayList of Integers and a TreeSet of Integers
    2. Call the first method to fill the lists sending the ArrayList and TreeSet as parameters.
    3. Print the ArrayList & TreeSet
    4. Call the second method to search for valuessending the ArrayList and TreeSet as parameters.
  + NOTE: When you print the times, always include commas. (Use printf or another formatting method to do this.)

1. Write a program named **Lab22B**  that will do the following.
   * Create a String method that receives a String parameter.
     1. It should return the longest substring inside that has identical characters.  
        Example: if the user entered **adgannoooqrbbrow**, then method would return “**ooo**” that is the largest set of identical characters.
     2. Print each of the possible substrings of 2 or more identical characters as you find them, but only return the longest one. (Don’t forget labels for all print statements.)
   * In the main method ask the user to enter a string and call your method sending the string as a parameter. Print the result.
   * Analyze the code in your method and print a statement to say what order of complexity it would be and why.
2. Write a program named **Lab22C** that will process a matrix (2-dimensional array) of random numbers.
   * Write a void method that receives a 2-dimensional array (4 rows, 4 columns) of integers. It should:
     1. Copy all the values from the 2D array into a one-dimensional integer array that is big enough to fit all the values.
     2. Sort the 1D array using the selection sort algorithm given below. Do not use an automated function; write the sort routine yourself.
     3. Copy the values from your 1-dimensional array back into the 2D array, so that they will be sorted in the 2D array as well. (The first 4 values go in the first row, next 4 in the second row, etc.)
   * In your main method:
     1. Create a 2-dimensional array of integers with 4 rows and 4 columns and fill it with randomly generated numbers between 0 and 50
     2. Print the array as a matrix (with one row per line)
     3. Call your method, sending the array as a parameter
     4. Print a blank line and then print the array again.
     5. Analyze your sort routine and print a statement saying what its order of complexity is and why.

Pseudocode for **sortArray**:

declare 3 variables: smallest, index & temp

set len = the array length

for loop with i starting at 0 and condition: i < len-1

smallest = arr[i]

index = i

for loop with k starting at i+1 and condition: k < len

if arr[k] < smallest

smallest = arr[k]

index = k

end if

end for loop

swap arr[i] and arr[index] values (using temp as temporary holder)

end for loop

\*\*NOTE – watch where the for loops end and make sure you don’t put the swap statements inside the inner for loop.