**Name:**

**Java Programming**

**Lab Exercise 11/20/2024**

**Measures of Efficiency (Big O)**

1. What is the Big O value for a sequential search on an unordered list?
2. What requirement must we impose on a list before we can apply a binary search to it?
3. What is the time complexity order of a binary search on an ordered list?
4. Suppose an algorithm with a Big O value of O(n2) has a runtime of 20 sec for n = 5000. What will be the runtime for n = 1000?
5. for(j = 0;j < n + 5; j++)

{

…some code…

}

What is the Big O value?

1. for(j = 0;j < n + 5; j++)

for(k = 0; k < n; n+=8)

for(z = 0; z <= (n\*n); z++)

{ …some code… }

What is the Big O value?

1. for(j = 0;j < n - 5; j++)

{

for(k = 0; k < 7; k++)

{ …some code… }

}

What is the Big O value?

1. for(j = 2; j < n + 5; j\*=7)

{

…some code…

}

What is the Big O value?

1. There are two types of complexity analysis. What two things can be analyzed?
2. Which of these two types does Big O address?
3. Suppose a time complexity analysis yields 5000n2 + (1/1000) n3 + n – 2. What would be the Big O value?
4. Will a O(n) algorithm generally always win in a time-race over a O(n3) algorithm?
5. Which is generally the fastest, O(log n) or O(2n)?
6. An algorithm has a time complexity of the order 2n. How many times more slowly would this algorithm run when n= 200, as compared to n = 100?
7. for(j = 0 ;j < n - 5; j++)

{

for(k = 0; k < n; k++)

{ …some code… }

}

What is the Big O value?

1. for(j = 0 ;j < n; j++)

{

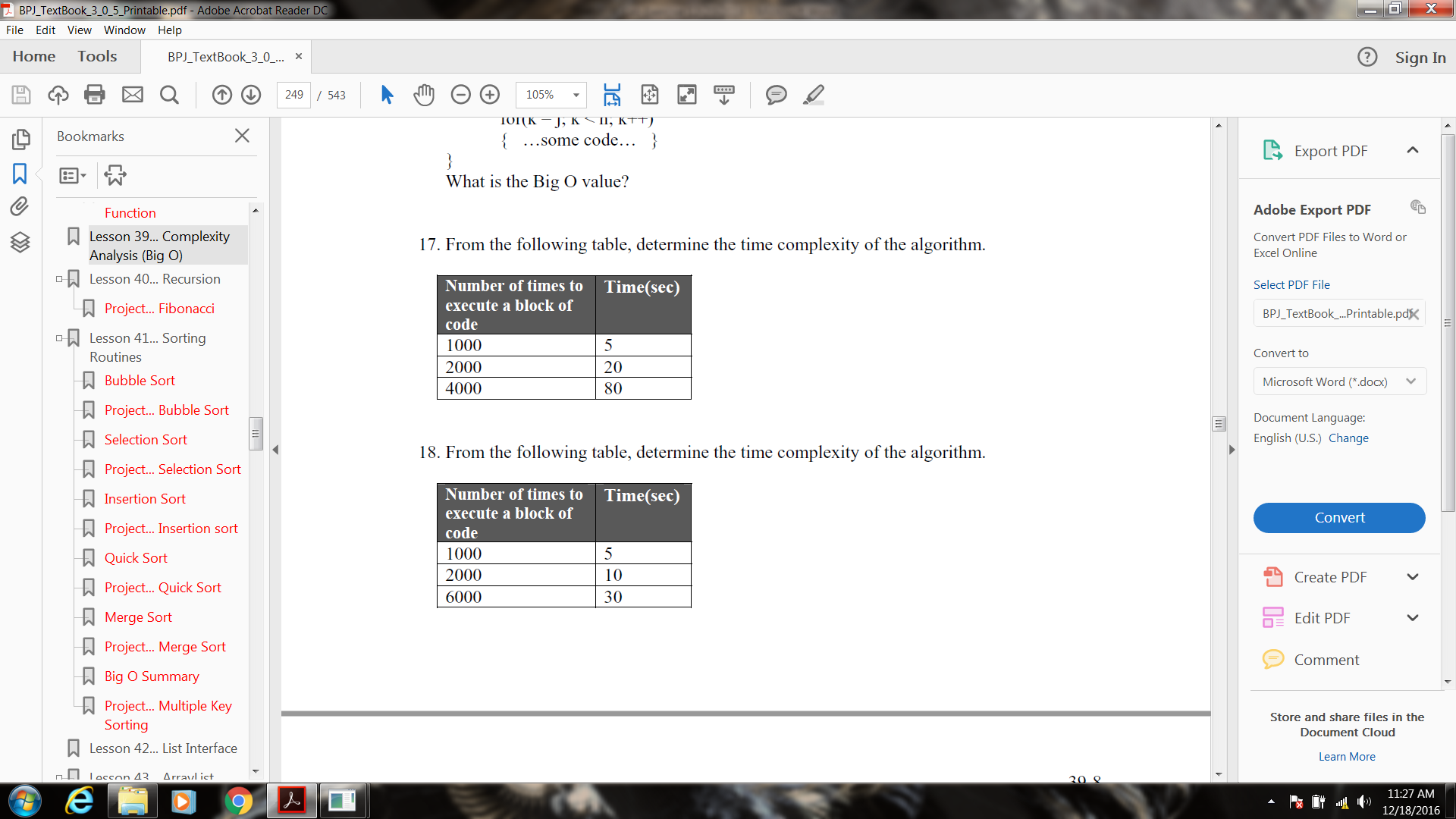
for(k = j; k < n; k++)

{ …some code… }

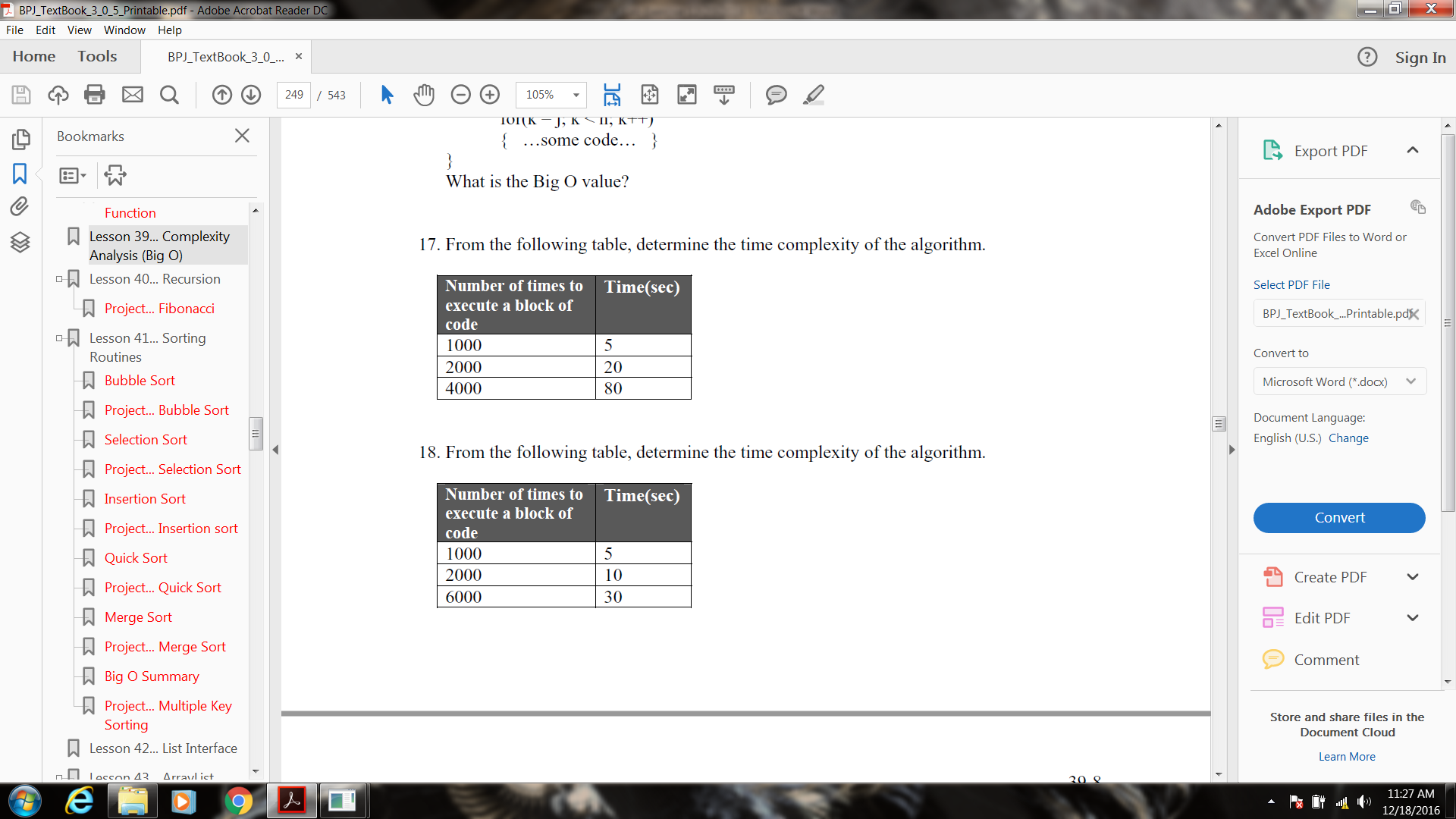
}

What is the Big O value?

17. From the following table, determine the time complexity of the algorithm.



18. From the following table, determine the time complexity of the algorithm.



**A class for testing sorting routines:**

You can test the various methods of sorting with the following code:

public class Tester

{

public static void main(String args[])

{

int theArray[] = {4,2,5,1,3,18,0,9,6};

sort(theArray);

for(int j = 0; j < theArray.length; j++)

{

System.out.print(theArray[j] + " ");

}

System.out.println(" ");

}

public static void sort(int a[ ])

{

**…specific code for a particular sorting method…**

}

}

Here is some code for a couple of types of sorting:

public static void sort(int a[]) //Bubble Sort

{

boolean loopSomeMore;

do

{

loopSomeMore = false;

for(int j = 0; j < a.length -1; j++)

{

if(a[j] > a[j+1])

{

//swap a[j] and a[j+1]

int temp = a[j];

a[j] = a[j+1];

a[j+1] = temp;

loopSomeMore = true;

}

}

}while(loopSomeMore);

}

public static void sort(int a[ ]) //selection sort

{

int min, minIndex;

for(int i = 0;i < a.length; ++i)

{

min = a[i];

minIndex = i;

for (int j = i + 1; j < a.length; ++j) // Find minimum

{

if (a[j] < min)

{

min = a[j];

minIndex = j;

}

}

a[minIndex] = a[i]; // swap

a[i] = min;

}

}

**Create a project that implements both of these sorts. Use an array size of 10000 integers. Fill the array with random integers using the nextInt() method. Print out a sample run that includes execution time, attach to this sheet and turn in.**

**Other Problems**

1. **Find the word**

Starting with an input string of words, find the second-to-last word of the string. For example, an input of “I love Codecademy” should return “love.”

To make your program more challenging, allow for a second numerical input, *n*, that results in returning the *n*th word of a string. So, for the string “I can program in Java” and *n* = 3, the output should be the third word, “program.”

1. **Anagrams**

Two words are anagrams if they contain the same letters but in a different order. Here are a few examples of anagram pairs:

* “listen” and “silent”
* “binary” and “brainy”
* “Paris” and “pairs”

For a given input of two strings, return a Boolean TRUE if the two strings are anagrams.

As an added challenge, for a given array of strings, return separate lists that group anagrams together. For example, the input {“tar,” “rat,” “art,” “meats,” “steam”}, the output should look something like {[“tar,” “rat,” “art”], [“meats,” “steam”]}.