Jovanni Lozzi

Classwork Lab #1

COM210: Data Structures and Algorithms

Introduction

The main purpose for this lab is to revisit and practice programming in Java. The lab also acts as a good introduction to the topic of solving time complexity in our code. These activities can prove beneficial to study / practice basic programming and is also a good way to prepare for real world scenarios we may encounter.

Method / Implementation

I implemented my topics using common, beneficial coding practices. My coding style is consistent throughout each of the algorithms with a healthy amount of comments as well. For the first lab problem, I started off by using the javax.swing.JOptionPane import as a simple way for the user to input their data. First the code prompts the user for the name of the item they wish to input, then another prompt asks for the price of said item.

```
String item1 = JOptionPane.showInputDialog("Input the name of item 1");
String s1 = JOptionPane.showInputDialog("What is the price of item 1");
Double price1 = Double.parseDouble(s1);
```

This process will then repeat until 3 items are input and saved as variables. Next the three items are output along with their price. Using the price variables, the code then calculates the average price and outputs that as well.

```
System.out.println(item1+":"+price1);
System.out.println(item2+":"+price2);
System.out.println(item3+":"+price3);

double average = (price1 +price2 + price3)/3;

System.out.println("The average cost of the price is "+average);
```

The code for the second lab problem is the same as the first except for the output portion. For this problem the average price should only be output if one of the items is peas. The way I achieved this is by using a simple counter saved under the variable x. If any of the 3 items are equal to the string “peas” the value of x will increase. As long as x is above 0, it will then print the average output.

```
String peas = ("peas");

int x =0;

if (item1.toLowerCase().equals(peas))
    x ++;
if (item2.toLowerCase().equals(peas))
    x ++;
if (item3.toLowerCase().equals(peas))
    x ++;

if (x > 0)
    System.out.println("The average cost of the price is "+ average);
else
    System.out.println("no average output");
```

The final problem is more complex since using arrays is required. Similarly to the first two problems I make use of the JOptionPane dialog boxes once again but this time I first ask what the number of elements the user wishes to input is. Once the number of elements is saved I then initialize two arrays, One for the item prices and one for the item names.

```
String s1 = JOptionPane.showInputDialog("How many items are you going to input?");
int numItems = Integer.parseInt(s1);

Double ItemPrices[] = new Double[numItems];
String ItemNames[] = new String[numItems];
```

Next a for loop is implemented to input all the item names and prices. This loop will iterate until each element is filled out by the user.

```
for (int x = 0; x < numItems; x++) {// start for
    ItemNames[x] = JOptionPane.showInputDialog("What is the name of item number: " + (x + 1));
    String s2 = JOptionPane.showInputDialog("What is the price of item number: " + (x + 1));
    ItemPrices[x] = Double.parseDouble(s2);
}// end for
```

Now that both arrays have all of their elements, we need to print them in the reverse order that they were input. Using another for loop makes this a simple process, instead of starting at zero like the previous for loop, we start using the number of items variable and decrement by 1 for each loop that iterates.

```
for (int i = numItems; i > 0; i--) {// start for
    System.out.print(ItemNames[(i - 1)] + ": ");
    System.out.println(ItemPrices[(i - 1)]);
}// end for
```

Since problem 3 also only wants the average price to be output if one of the items are peas, we need to use another for loop to check each element in the array to see if they contain the “peas”. I use a counter under the variable j to increment when peas are in the array. I also use this for loop to add the prices together to find the total price and save it as a variable. Once the loop is over we can now use the total price variable to calculate the average price and finally use the j value to

check if the average price should be output or not.

```
String peas = ("peas");
int j = 0;

Double totalPrice = 0.0;

for (int x = 0; x < numItems; x++) {// Start for
    if (ItemNames[x].toLowerCase().equals(peas)) {//Start If
        j++;
    }//End If
    totalPrice = ItemPrices[x] + totalPrice;
}// End for

Double averagePrice = totalPrice / numItems;

if (j > 0) {
    System.out.println("The average cost of the price is " + averagePrice);
} else {
    System.out.println("no average output");
}
```

Analysis

Once finished with coding the three algorithms, I moved on to calculate the time complexity of each. Calculating the time complexity of the first two algorithms was easy since it used a set amount of items and contained no arrays. The time complexity for problem one and two are both O(1) since the code contains a fixed number of operations that are not dependent on the user's input and contains a constant number of operations. The third algorithm has a time complexity of O(n) because the code contains two for loops that iterate n times. Since the program requires the user's input of n for the amount of operations, the time taken to execute the code increases linearly with the size of the input.

Conclusions and future work

In conclusion, solving time complexity is not as daunting as it may first appear. It is a necessary skill that is worthwhile to absorb a complete grasp of in the field of computer science. Solving time complexity is an important way to measure the performance of an algorithm or data structure. Having this skill set allows us to determine which algorithms are better than others for different scenarios. In the future I will continue to use time complexity analysis with my code to increase my productivity and efficiency.