Time Measurements

Ghayth Sabeaallil

Abstract- This is a simple report which contains description and analysis of the both exercises 6 and 7 in assignment 4 1DV507 – Programming and Data Structures. The main idea of this report is to find how many letters can compute during 1 second, depending on adding only one character or long string (about 80 characters) using two different ways:
1) Normal plus operator. (str = str + "...")
2) Using the StringBuilder class. (append ("..."))
Our experiments show that about 87516 letters can be computed in one second using plus operator short string (only one letter). And about 854288 letters using plus operator long string (80 letters). And for StringBuilder the experiments show that about 15377562 (15.3 million) letters can be computed in one second using plus operator short string. And about 313613480 (313.6 million) letters for long strings. For sorting integers using insertionSort I got 3587 elements and for sorting random strings I got 2553.

I. Exercises

A. Repeated string concatenations can be done in two ways: 1) Using the plus operator you can construct a long string by constantly increasing the length as: str = str + "...", 2) Using the StringBuilder class and repeated use of method append ("..."). Your task is to find the fastest approach by measuring how many concatenations, and the length of the final string, each of them can compute *in 1 second* when:
   1. Adding short strings containing only one character
   2. Adding long strings representing a row with 80 character

B. In Assignment 3 you implemented 4 different sorting algorithms: Insertion Sort (for both strings and integers) and Merge Sort (VG Exercise, for both strings and integers). How many strings and integers can be sorted in 1 second using these four algorithms?
   1. For integers: Sort arrays with random generated integers. The range used by the random generator should be larger than the array size in order to reduce the number of duplicate elements.
   2. For strings: Sort arrays in alphabetical order using arrays of random generated strings where each string contains 10 randomly generated characters.

II. Experimental Setup

All experiments were done on a Lenovo Thinkpad S430 with an Intel® Core™ i7-3520M CPU @ 2.90GHz × 4 with 8GB of memory. The OS is Ubuntu 19.10. I use Java 13.0.2, Java (TM) SE Runtime Environment (build 13.0.2+8). In the beginning a heap space was 2GB but later on I allowed the Java Virtual Machine to use a heap space of 4GB (4096MB). Only fireFox was running during the process. I use System.nanoTime() to calculate the time (x^9).

The java code is a simple one, I start the time using *(long start = System.nanoTime()).* Then I use while loop, within the while loop I start to add the character. Then I create a variable *end* which contains the time during the while loop is running. The while loop stops when and just when *(end - start >= 1000000000)* which means after around 1 second . As an example, the code below shows my explanation:

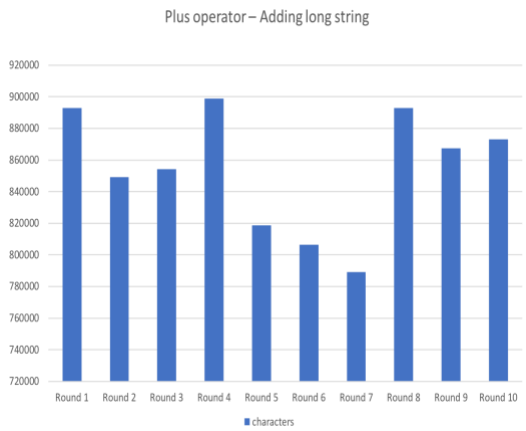
```
String oneChar = "o";
```

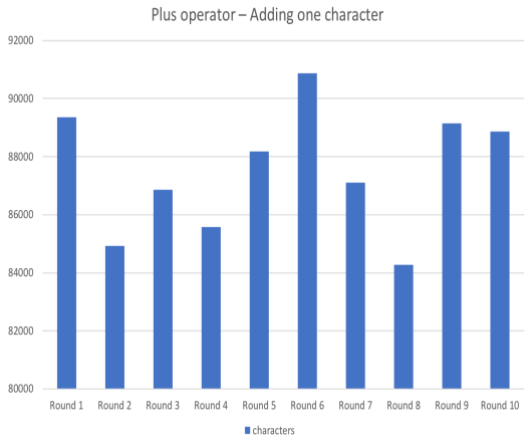
```
StringBuilder one = new StringBuilder();
long start = System.nanoTime();
        long end;
        while (true) {
        one.append(oneChar);
        end = System.nanoTime();
        if ((end - start) >= 1000000000)
        break;
        }
        System.out.println("Total (add one char by StringBuilder): " + one.length());
```

I use the same idea of the code above during all our experience. The only differences are the operation (plus or StringBuilder) and the length of the string. The variables *start* and *end* are to handle the time, *start* is before our process starts and end is to calculate the time during the process, in the end the difference between *start* and *end* should be bigger or equal to 1000000000 (nanoTime).

Plus operator

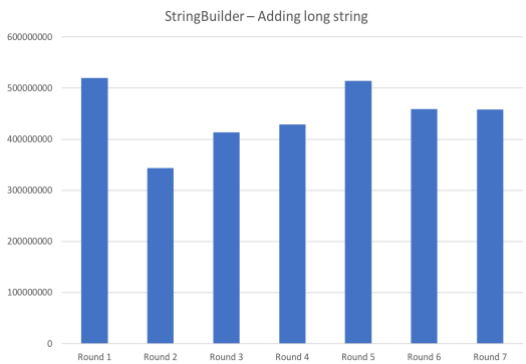
I run the code 10 times when I study the plus operator case. Later on I divide the result by 10 to get the Average. However, the memory was only 2GB in this case. The code runs both when I add only one letter and 80 letters at the same time. And I get those result:

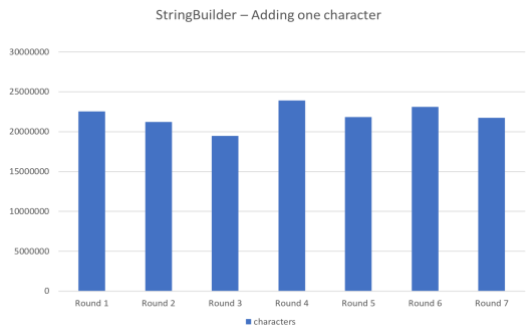




StringBuilder

In this case, I run the code only 7 times, and I was forced to change the size of the memory to 4096Mb (4GB), and the reason is Error heap space. However, adding one character and long string did not run at the same time. And I get those result:

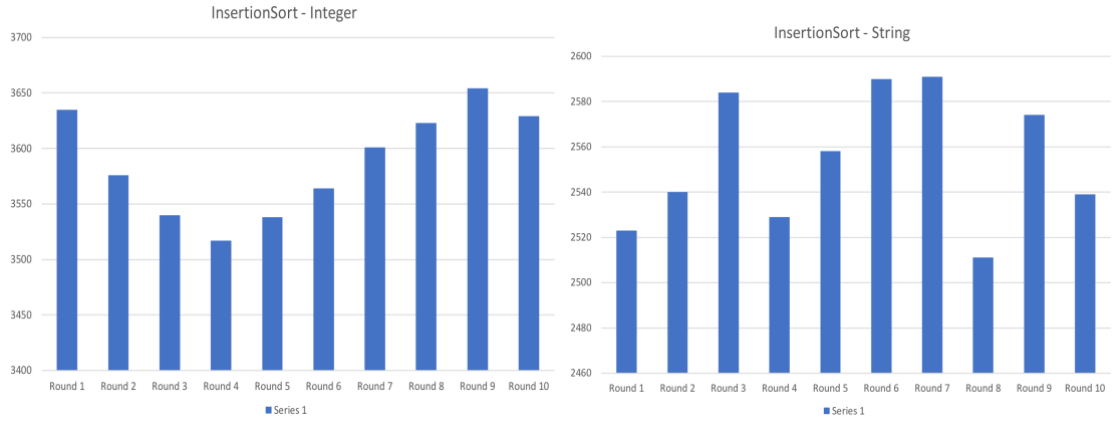




InsertionSort Integer

In this case I generate a random integer and add it to an arrayList, and during the adding I sort it using the insertionSort algorithm. However, all things did in a while loop that is doing all processes in one second. The code below shows how to add and sort all integers in just one second.

```
ArrayList<Integer> arrList = new ArrayList<Integer>();
        Random rnd = new Random();
        int num;
        long start = System.nanoTime();
        long end;
while (true) {
        num = rnd.nextInt((arrList.size() + 1) * 2);
        arrList.add(num);
        int element;
        int elementBefore;
        int temp;
        for (int i = 1; i < arrList.size(); i++) {
        element = arrList.get(i);
        elementBefore = i - 1;
        while (elementBefore >= 0 && element < arrList.get(elementBefore)) {
                temp = arrList.get(elementBefore); //num
                arrList.remove(elementBefore);
                arrList.add(elementBefore, arrList.get(elementBefore));
                arrList.remove(elementBefore + 1);
                arrList.add(elementBefore + 1, temp);
                elementBefore--;
        }
        }
        end = System.nanoTime();
        if ((end - start) >= 1000000000)
        break;
        }
```



Our experience did not get an exactly answer about how many integer or string sorted in 1 second, or how many characters can we add in one second. And of course it depends on a lot of

things such as (Process, OS, memory and program). So, I did not give an exact answer all my answer is averages.