Linnéuniversitetet
Kalmar Växjö

Exercise 4 Report

Time Measurement

String Concatenation, String Builder Appending, Insertion and Merge Sort in one second

Supervicer: Jonas Lundberg
Author: Hailing Zhang (hz222bp)
Semester: Spring 2018
Course Code: 1DV507
Date: March 11, 2018

Table of Contents

1 Introduction

Firstly, this report aims to compare the number of concatenations and the length that could be added by different approaches in one second. The target strings consist of short strings (string with one character), and long strings (a row of string with 80 characters). Secondly, the speed of two different sort algorithms will be illustrated by the length of sorting integers and characters. The results are concluded from repeated experiments will be stated in the 8 tables. In the design, the independent variable is time, thus all method will run in one second (1000 milliseconds). In the last part, the reason of the advantages for the efficiency of StringBuilder will be analyzed.

2 String Concatenation

The number of both short and long string that can be added by the plus operator in a string in one second will be shown in this part.

2.1 Description

1) Initialize an empty string;
2) Begin to count time by calling the milliseconds method. Since for simplifying the calculation, the precise nanoseconds is too big to measure.
3) Create a loop with the time limitation. If the running-time is lesser than one second, the string keeps adding one character to the string by plus operator.
4) Record the length of string and the number of characters after loop.
5) Executing this method 5 times for gaining the average.
6) Change the short string to long string with 80 characters, then do the step 1 to 5 again.
7) Draw the tables and analyze the results.

2.2 Results

There are four algorithms, and I should run each of them five times. I consider to record the concatenations and length in an array then get the average directly from Eclipse. However, the results are not calculated by array since the memory space of my laptop is not enough (java.lang.OutOfMemoryError: Java heap space). In this case, I record the data then calculate the average by calculator.

Table 1: The plus operator add short string with one character

Experiment No.	Time(milliseconds)	Concatenations	Length

1	1000	75778	75778
2	1000	83046	83046
3	1000	85921	85921
4	1000	83242	83242
5	1000	80220	80220
Average	1000	81641	81641

Table 2: The plus operator add long string with 80 characters

Experiment No.	Time(milliseconds)	Concatenations	Length
1	1000	5724	457920
2	1000	6445	515600
3	1000	6343	507440
4	1000	6525	522000
5	1000	6382	510560
Average	1000	6283	408744

3 StringBuilder Appending

By appending method in the StringBuilder, the number of both short and long string that can be added in a string in one second will be shown in this part.

3.1 Description

1) Initialize a StringBuilder;

2) Begin to count time by calling the milliseconds method.

3) Create a loop with the time limitation. If the running-time is lesser than one second, the StringBuilder keeps appending one character.

4) Record the length of string and the number of characters after loop.

5) Executing this method 5 times for gaining the average.

6) Draw the tables and analyze the results.

3.2 Results

The time for convert them to string is included. I just keep the integer part of the average since within the limited time, the unfinished operator will not be stated.

Table 3: The StringBuilder append short string with one character

Experiment No.	Time(milliseconds)	Concatenations	Length
1	1000	157872691	157872691
2	1000	142046147	142046147
3	1000	158138626	158138626
4	1000	153016924	153016924
5	1000	155758589	155758589
Average	1000	153366595	153366595

Table 4: The StringBuilder append long string with 80 characters

Experiment No.	Time(milliseconds)	Concatenations	Length
1	1108	2174681	98816292
2	1067	1274864	120010217
3	1139	1600311	87179302
4	1171	2781677	92167351
5	1102	1002187	130363002
Average	1117	1266042	105707232

4 Insertion Sort

This part will discuss the number of integers and characters can be sorted by the insertion algorithm in one second.

4.1 Description

1)Write a method that returns an array of one thousand random integers. The range of the integers are from 1 to 10000.

2)Write a method to get an array consists 1000 characters.

3) Copy the two insertion sort algorithms for integers and string from assignment 3.

4) Sort the random array by insertion algorithm. The time before and after sorting are recorded. The time is measured in milliseconds.

5) For the accuracy, the loop will break when the running-time gets close around 1000, otherwise, the array size would be adjusted according to the running-time. Then the time will be reset. It means that the sort will be restart with the different length of array.

6) Repeat the step 4 to get the results of sorting string by insertion.

7)The average comes from executing the insertion sort five times in main method.

4.2 Results

Table 5: The insertion sort the integers

Experiment No.	Time(milliseconds)	Length
1	1001	46049
2	1000	46001
3	1000	46011
4	999	46011
5	999	46005
Average	1000	46015

Table 6: The insertion sort the string

Experiment No.	Time(milliseconds)	Length
1	1001	11300
2	1000	11502
3	999	11497
4	999	11500
5	999	11386
Average	1000	11437

5 Merge Sort

It is the VG task. This part will discuss the number of integers and characters can be sorted by the merge algorithm in one second. The method for getting an array with random integers or characters are finished before. Moreover, the sorting methods are finished in assignment 3.

5.1 Description

1) Copy the two insertion sort algorithms for integers and string from assignment 3.

2) Sort the random array by merge algorithm. Other part are same with insertion sort.

3) Static the results and calculate the average after executing five times.

5.2 Results

Table 7: The merge sort the integers

Experiment No.	Time(milliseconds)	Length
1	1001	3972007
2	1001	4360012
3	1000	3800999
4	999	4125013
5	999	4482045
Average	1000	4148015

Table 8: The merge sort the string

Experiment No.	Time(milliseconds)	Length
1	1001	581001
2	999	659910
3	1000	650203
4	1000	620803
5	1000	650113
Average	1000	632406

6 Analysis

This part analyzed the reason of the StringBuilder is faster than the plus operator.

The statement of the string is “private final”, what means that the string is unchangeable. In fact, the plus operator creates a new string object that includes the target string rather than changing it. Especially when using the loop, the plus operator would consume much more time and memory space to create the new objects. For example, the toString() method will create a temporary object every time.

For achieving the same aim, the steps of plus operator are following: 1. str1= “a”; 2. str2 = “b”; 3.str3=str1+str2; 4.str1=str3. The steps of StringBuilder are: 1.str1= “a”, str2= “b”, str1= str1+str2.

In this case, the advantage of writing StringBuilder on codes will be more obvious after executing the loop for more than then thousands.