Introduction Task Report.

Adrian Jonsson Sjödin

Fall 2022

Introduction

The purpose of this task is to determine the time efficiency of three different operations over an array of elements, as well as familiarizing oneself with writing a report in LaTeX.

The three operations are:

- Random access: reading or writing a value at a random location in an array.
- Search: searching through an array looking for an item.
- Duplicates: finding all common values in two arrays.

Task 1

Set up a benchmark where the access method is called on a larger and larger array of size n and present your conclusions and observations on how much time is needed for a reading of a value at a random location in the array.

Method

To be able to measure the time it takes for one random access of an array of size n, I used the provided code from the task with some modifications. Mainly I adjusted it so that the number of times we searched for an element in the array is not the same as the size of the array. Instead we input the size of the array (n) and the nr of searches into the method call:

```
private static double access(int arraySize, int nrOfSearches) {
  int k = 1_000_000;
  int l = nrOfSearches;
 /* code here*/
}
```

This method will then return a time in nano seconds for the average time of one random array access.

Result

Array size	Nr. of searches	Time in ns
10	10 000	0.86
100	10 000	0.26
1000	10 000	0.26
10 000	10 000	0.31
50 000	50 000	0.50
100 000	10 000	0.72

Table 1: Output from the program being run once

Discussion

It seems like the time to access a random element in an array of size n increases with the size of the array, which I think is expected. The time increase does not seem to be linear but rather a slower time increase than that, which if true is good. However the trustworthiness of this result is questionable since every time I ran the program it gave me quite different results, where sometimes the time being lower and lower for larger n, and other times having one of the times in the middle being drastically larger than the rest.

I don't know if the reason behind this fluctuation of the time results are buggy code, or if it is something that Java does behind the scenes that could somehow change the results. It makes more sense that the reason lies in the code since one would think that whatever Java might do behind the scenes should be consistent and thus not influence the results. But even then I can't find where the problem would lie in the code which leads me to believe that the reasons has to be connected to the randomness of the accessing somehow.

Task 2

You will include some run-time measurements in your reports. You should the think about the number of significant figures that you use. Just because a benchmark took $1.2345678s$ does not mean that you should report it in this way. If you write this in your report you're implicitly saying - if I do this again the number will be the same. This could be true but I doubt that anything you do on a computer can be determined with an 8 figure accuracy. The next time you try it might very well take $1.2354678s$. What you report is maybe $1.235s$ or $1.2s$?

tables

Numbers are often best presented in a table. You will have to do some reading on how to format tables but the general structures is quite easy. This is for example a table with some run-time figures.

prgm	**runtime**	**ratio**
dummy	115	1.0
union	535	4.6
tailr	420	3.6

Table 2: Union and friends, list of 50000 elements, runtime in microseconds

As you see in the table above, the run time per se might not be interesting. The interesting thing is how it relates to something else. Look at the ratios above, it gives you the information that we are looking for. So when you include numbers, ask your self why you have these numbers in the report. What is the purpose, can you describe it in a better way?

no f*ing screen shots

I know that you are all very happy that things actually work and eagerly want to show what things look like on you screen but please, don't use *screen shots*. It looks ugly and it's impossible to mark or copy the things that you want to show. It also, most often, show a lot of irrelevant things so instead of using an image, copy the text and format it so it's easy to read.

graphs

Once you start to generate graphs make sure that they are readable and have sensible information on the axes.

There are many ways to generate graphs but you want to use a way that minimize manual work. My tool over the years has been *Gnuplot* and if you do not have a favorite tool you could give it a try.

Gnuplot is not a statistical program nor a program that is very good at manipulating numbers but it is very good at taking a sequence of numbers and generate a nice graph.

LaTeXthings

Some LaTeXerrors that I frequently see that could easily be avoided if you only know where they come from.

less than

If you in your LaTeX code write "5 < 7" it will look like 5 ¡ 7 and "9 > 7" will look like 9 ¿ 7. Using the characters < and > directly does not work ... so, how did I do it? I used the commands `\textless` and `\textgreater` to generate the symbols < and >.

You could also use {`\tt 5 < 7`} but then it will use the teletype font and look like this: `5 < 7`.

Still another way is to write it using so called `math mode`. This is a mode used for writing mathematical formulas in a nice way. You enclose your expression in `$` signs like this `$5 < 7$` and then it will look like this $5 < 7$.

If you have a larger mathematical expression you enclose it in double $ and the result is that it is written centered with some space around it like this:

$$5 < (3 * 8/3)$$

why strange font

If you want to write *foo* in teletype font you write like this `{\tt foo}`. If you forget the closing } then it will look like this: `foo. Now everything here after until the end of you report will look like this.`