ID1021: Measuring time variance in different array operations

Alexander Lundqvist

30-08-2022

Introduction

This report we try to measure the execution time for different operations on array elements and what happens when we increase the size of the array.

We also discuss how these different operations differ and how they can be described mathematically.

Method

To implement this project I started a Netbeans project with one single file with private methods for the different tasks. The main method was responsible for the actual testing of the methods.

Task 1

In this task we are supposed to measure the time it takes to measure the time it takes to access a random element in an array and how the time is affected when the array grows in size **n**.

To start of we create an array with random numbers ranging from 0 up to but not including n. This will serve as an array with indexes that will later be used to access another array. We continue by creating an array populated with some dummy value, in this case 1s.

To test the array access time we utilize the built in Java function nanoTime which lets us accurately measure the execution time.

```
int sum = 0;
long t0 = System.nanoTime();
for (int i = 0; i < k; i++) {
    for (int j = 0; j < n; j++) {
            sum += dummyArray[index[j]];
        }
```

```
}
long t_access = (System.nanoTime() - t0);
```

The sum variable is declared as to have the function actually do something and not let the compiler skip any step in the code.

Task 2

TBD

Task 3

TBD

Result

Here the result of the different executions are presented as tables. The result is discussed in the last section.

Task 1

ns	n
0.295	250
0.342	500
0.323	750
0.292	1000
0.295	1250
0.291	1500
0.288	1750
0.290	2000
0.285	2250
0.289	2500

Table 1: Task 1: Random array access with n being the array size and ns the time in nanoseconds

Task 2

Task 3

Discussion

In thís section we discuss the different result that we got from the measurements.

ns	n
57.3	250
102.4	500
148.9	750
252.4	1000
329.5	1250
387.5	1500
442.8	1750
526.3	2000
591.4	2250
651.0	2500

Table 2: Task 2: Array search with n being the array size and ns the time in nanoseconds

ns	n
900.0	10
3200.0	20
5400.0	30
13700.0	40
30100.0	50
330000.0	60
42800.0	70
69600.0	80
69800.0	90
113100.0	100

Table 3: Task 3: Finding duplicates in array with n being the array size and ns the time in nanoseconds

Task 1

When plotting table 1 as a graph we can see that the time to access an element decreases when the size of the array is increased. We can see that

Task 2

Task 3