Arrays and performance

Kevin Wang

Fall term August 2022

1 Introduction

In this task i will explain something called performance measurement and the difference between the formulas d O(1), O(n) and O(n2). Where the formulas is used to calculate the time complexity. I will discuss the time complexity in respect to three different methods in searching a element in a array. Those are Random access, Search, Duplicates.

2 Random Access

In java there is a built in method called System.nanoTime that prints out the execution time in nanoseconds. With this i will try to analyze the accuracy of clock. To make conclusion from the testing of the speed of the program we actually need to do a lot of times for instance hundred times to actually say something about the result.

From the first test of code we can see that the nanoseconds are very close to each other and the reason is probably because of the program loops in a empty array. it basically doesn't do anything.

the first code:

n	Prgm1
1	$26\mu s$
2	$17\mu s$
3	$14\mu s$
4	$15\mu s$
5	$16\mu s$
6	$14\mu s$
7	$14\mu s$
8	$12\mu s$
9	$12\mu s$
10	$10\mu s$

From the second test of code we are testing lopping array with sum that adds every element in the array. It took 759 nanoseconds to access the first element which is significantly more delayed than other. the second test

n	Prgm2
1	$76\mu s$
2	$22\mu s$
3	$12\mu s$
4	$14\mu s$
5	$14\mu s$
6	$18\mu s$
7	$18\mu s$
8	$17\mu s$
9	$27\mu s$
10	$18\mu s$

The third test of code we are testing

n	Prgm3
1	$17\mu s$
2	$11\mu s$
3	$8\mu s$
4	$9\mu s$
5	$10\mu s$
6	$16\mu s$
7	$13\mu s$
8	$24\mu s$
9	$18\mu s$
10	$11\mu s$

3 Random Access - more operation

In this section we have to compare the time accuracy of the code where we use java.util.Random method. and the result i got is;

n	Prgm4
1	$18\mu s$
2	$11\mu s$
3	$8\mu s$
4	$9\mu s$
5	$10\mu s$
6	$16\mu s$
7	$14\mu s$
8	$24\mu s$
9	$18\mu s$
10	$15\mu s$

The first time i run the program i got very long numbers of nanoseconds and its probably because the compiler is compiling to machine code and its

takes more time for it. Here the program is where i used the completion code and the run time after. Program 2 is where i run the program without the completion code.

n	Prgm1	prgm2
1	0.006ms	0.01ms
10	0.01ms	0.2ms
100	0.07ms	2ms
1000	0.6ms	1ms

from table 1 we can observe that the time efficiency is stable and program 1

Filling the array with dummy = 1

n	Prgm1
$\eta H1$	$10.3\mu s$
10	$0.02\mu s$
100	$0.7\mu s$
1000	$0.002\mu s$
10000	$0.0005\mu s$

4 search for an item

There is a algorithm to find a random element in a array that could be a given element we are looking for. And now the purpose is to observe how the execution time various as the length of the array grows.

n	Prgm1	prgm2
1	2.3ms	$520\mu s$
10	9.5ms	4.2mss
100	8.2ms	48.8ms
1000	0.6ms	420ms

we can see from the table how the time various as we increase "n" which is the variable for the array. It takes obviously longest time to search for the key when the array is larger.

i will now determine how i can do so that this program can give me a predictable result as i change the value of k and m.

k	m	n	Time
10	10	1	$26\mu s$
10	10	10	$31\mu s$
10	10	100	$37\mu s$
10	10	1000	$59\mu s$

I decided to set m, the number of search operation in each round, as 10. And set k, the number of rounds, as 10. It actually doesn't matter if we set k higher or lower. Because i think the number of rounds is not that essential in this task as we are testing of how fast the program can find keys in a array, while it seems more interesting to investigate how the time would change if we increase the number of search operation in each round. The time that we can see from the table actually don't differs so much. It seems like my hypothesis is correct. The result is increasing higher and higher and there is a large jump in time from n=100 to n=1000 might be [$O(n^2)$].

5 Duplicate

The meaning in this section is that there will be two arrays with the same size and one of them contains a key. and we will iterate over one array and see. For this assignment i did a program very similar to the one in task 2, where there is a array and a array with keys. what i did was that i declared k and m to the same variable and with the same size n. The result that i got is that when n is increasing the graph, where i stimulated in excel is increasing pretty stable over time, as this formula O(n).

n	Time
1	$0.8\mu s$
10	$0.9\mu s$
100	$0.7\mu s$
1000	$0.4\mu s$
10000	$0.5\mu s$