BILKENT UNIVERSITY

CS201

Spring 2020

Homework 2

İrem Seven

21704269

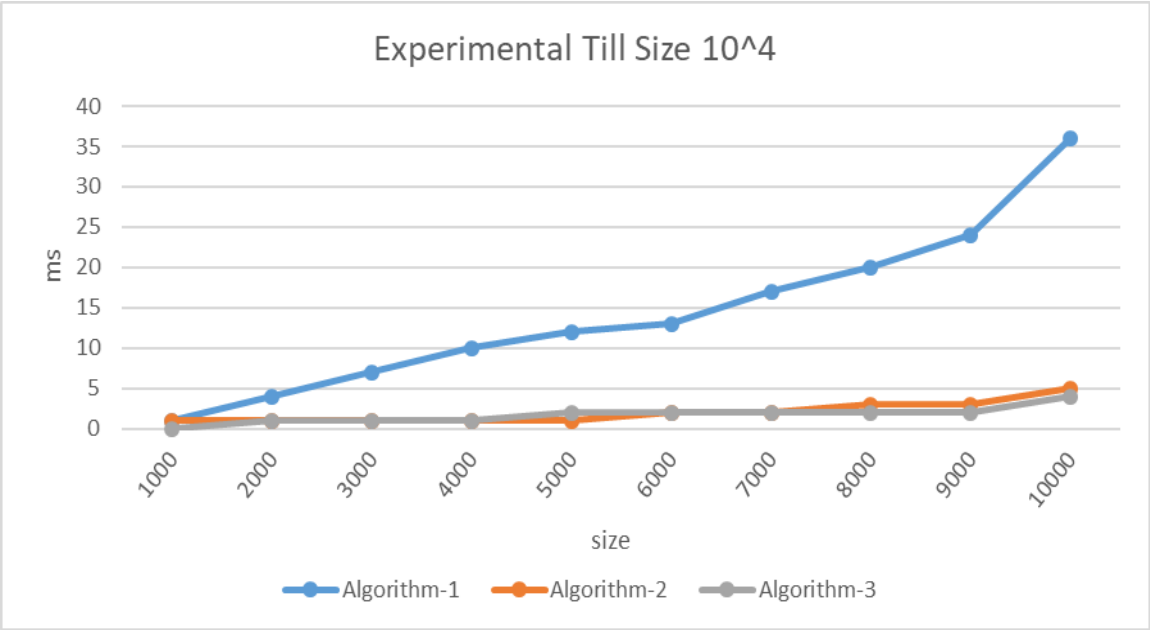
Finding K largest elements from an Array with Different Algorithms

Experimental Data Table

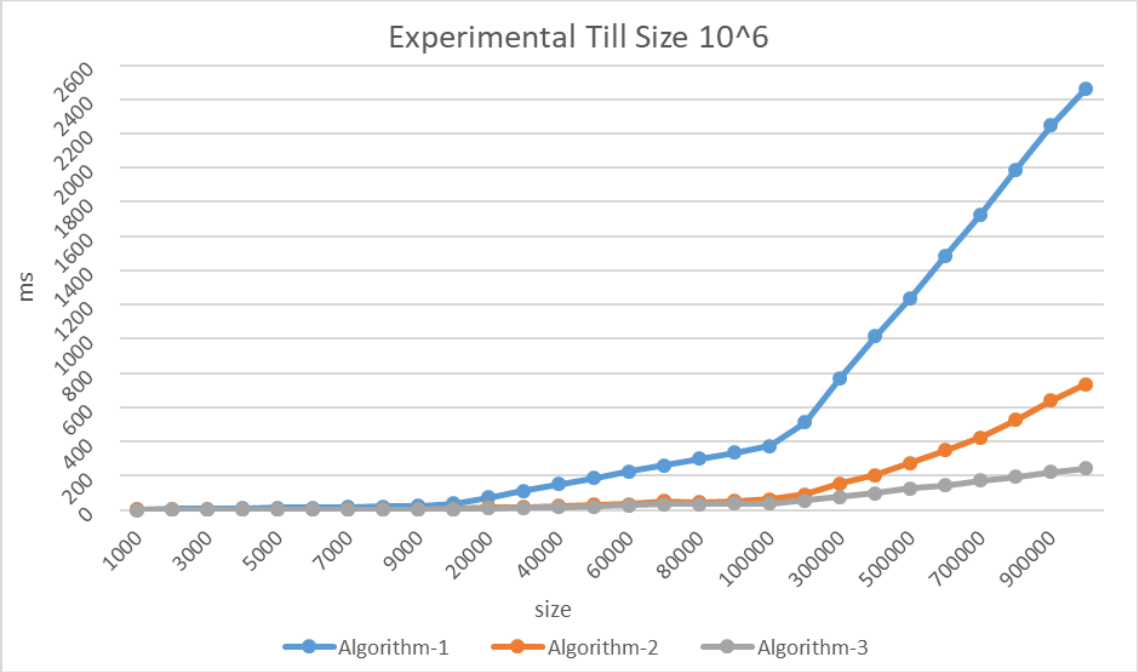
Array Size	Running Time (ms)		
n	Algorithm-1	Algorithm-2	Algorithm-3
1000	1	1	0
2000	4	1	1
3000	7	1	1
4000	10	1	1
5000	12	1	2
6000	13	2	2
7000	17	2	2
8000	20	3	2
9000	24	3	2
10000	36	5	4
20000	73	15	9
30000	111	16	11
40000	149	22	17
50000	185	29	19
60000	224	35	28
70000	261	51	35
80000	298	46	33
90000	336	52	39
100000	374	62	36
200000	511	89	55
300000	771	155	77
400000	1016	203	98
500000	1237	274	125
600000	1485	349	143
700000	1728	424	171
800000	1988	527	193
900000	2248	640	220
1000000	2465	734	242
10000000	24615	37106	2492
20000000	49310	140641	5011
30000000	76163	324863	8013
40000000	102615	571998	10608

Table 1: Experimental Running Time Values

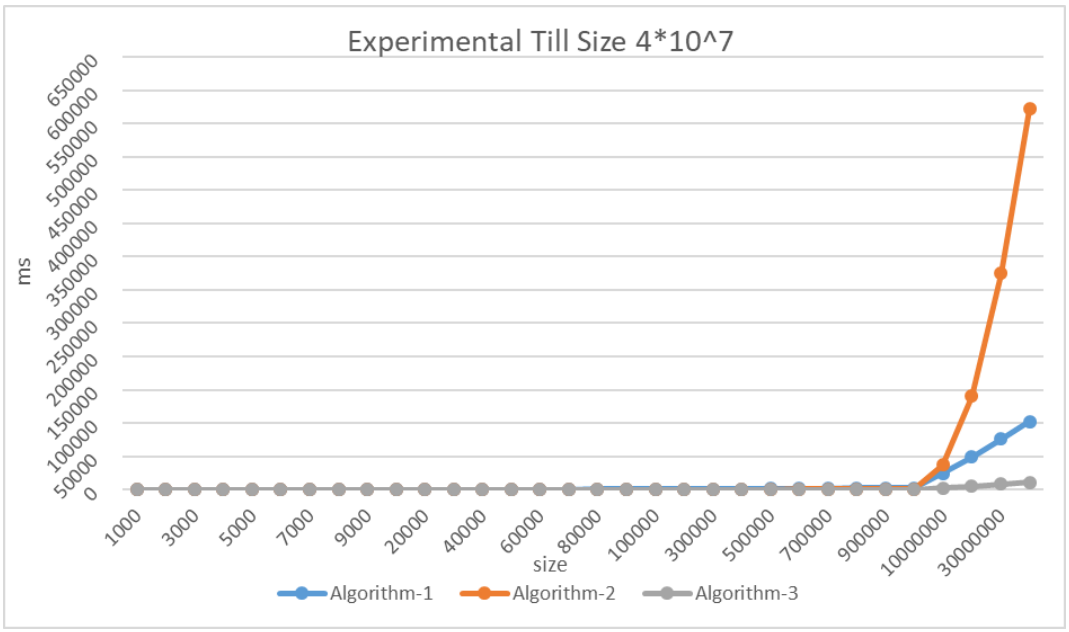
Experimental Data Plots



Plot 1: Experimental Comparison till n = 10^4



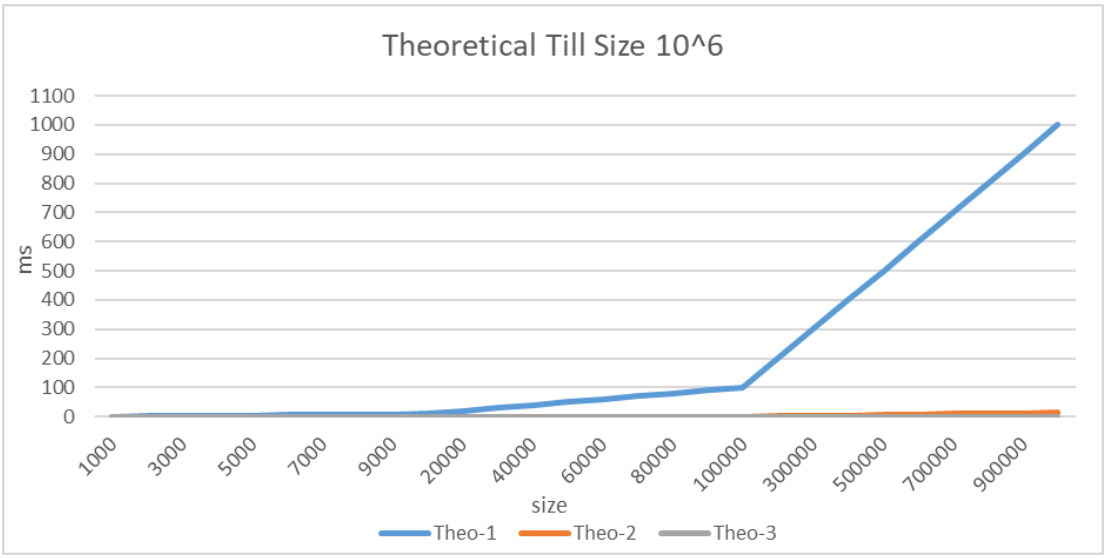
Plot 2: Experimental Comparison till n = 10^6



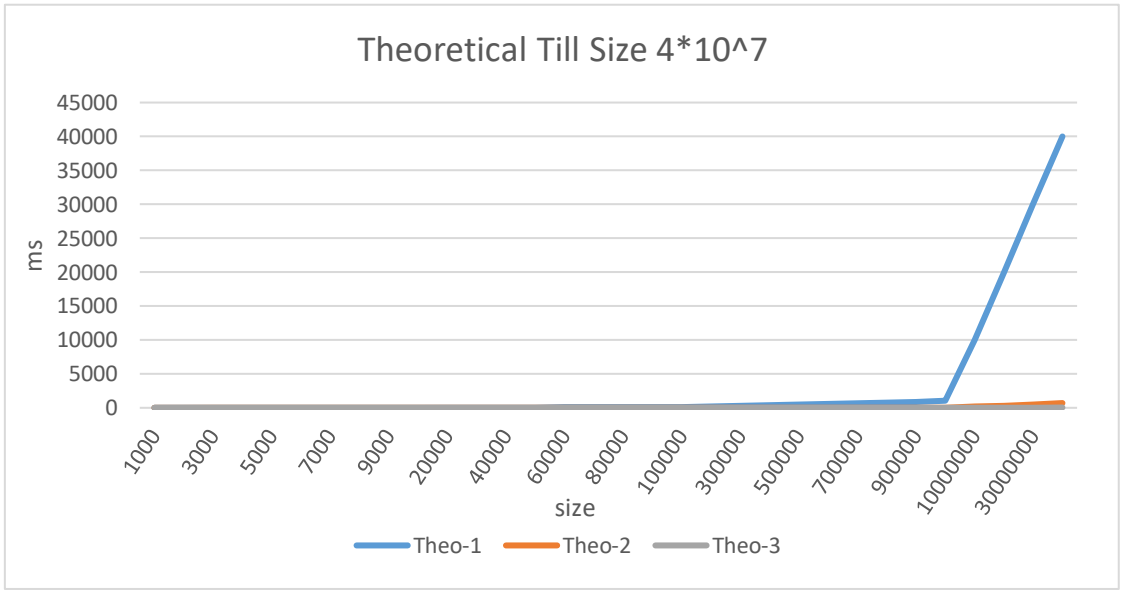
Plot 3: Experimental Comparison till n = 4*10^7

Theoretical Data Plots

Plot 4: Theoretical Comparison till n = 10^4



Plot 5: Theoretical Comparison till n = 10^6



Plot 6: Theoretical Comparison till n = 4*10^7

Computer Specifications:

Intel Core i7-7700HQ CPU @2.80GHz, 2808 MHz

x64-based PC

16 GB RAM

Operating System: Windows

IDE: Visual Studio

Language: C++

Discussion:

In this experiment, three different algorithms are used to find k largest numbers in an array of integers. The time complexities of each algorithm are measured by fixing k as 1000. Arrays are generated randomly for each array size(n) and the same randomly generated array is used for each algorithm to measure time complexities efficiently. Theoretically, algorithm 1 has the time complexity of O(k*n), the reason is that it iterates n times for each k value starting from 0 to k. Considering algorithm 2, it has a time complexity of O(nlog), since it uses a quicksort algorithm to first sort the array. Algorithm 3 has a linear time complexity of O(n) with the help of the select method. Thus, it is expected to observe a lower growth rate in algorithm 3 than the others. On the other hand, normally algorithm 1 expected to have the highest growth rate if the k value is large enough. In the experiment, the time complexity of algorithm 1 has measured higher compared to the others when the input size is between 1000 and 10^6, with k 1000. In plots 1 and 2, it can be seen that algorithm 1 has a significantly higher growth rate than the others. Also, the growth rate of algorithm 2 is higher than of algorithm 3. However, in plot 3 it can be seen that when the experiment is conducted with very large input sizes algorithm 2 has shown significantly higher running time than algorithm 1. The reason might be that for that n value log(n) becomes higher than the k. Since k value is fixed through the experiment, at some point algorithm 2 passes its time complexity. Thus, it can be said that if the k value is small relative to the array size choosing algorithm 2 will provide a more efficient solution. On the other hand, algorithm 3 gives the most efficient solution since its time complexity is always the smallest no matter what happens, so it can be considered as the fastest solution. However, it requires much more code segment than the others. Plot 4,5 and 6 show the theoretically expected growth rates, and it can be said that similar growth rates achieved in the experiment. However, comparing plots 3 and 6 , when the array size is very large, they differ significantly for the first two algorithms. When calculating O(nlog(n)) base of the log has taken as 2 for theoretical values. But in reality log(n) might be passed the k value in running rime, it might be caused by computer specifications or the randomness of array elements.