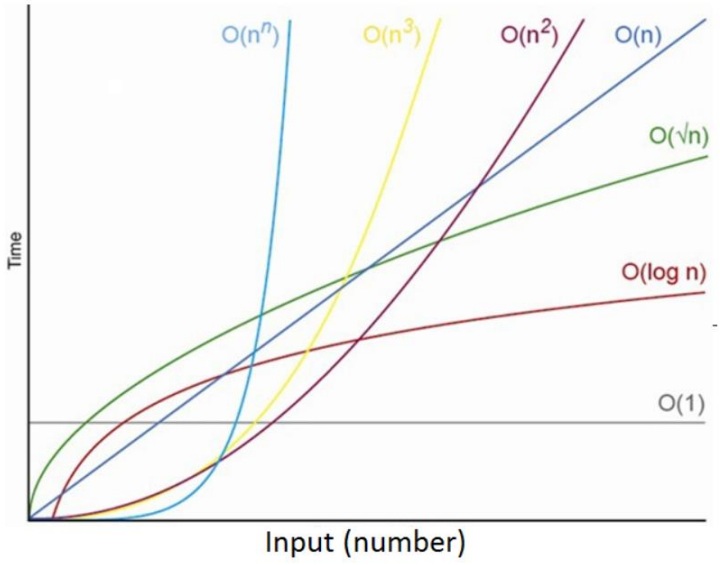
Ayşe Kelleci 21902532 CS201-3 HW2

5 December 2021

Time Complexity in Algorithms (Big-O notation)



Algorithms have two types complexity: memory and time complexity. Time complexity shows the elapsed time changes according to the size of parameters. This picture shows the theoretical values of time complexity.

Computer Properties

RAM: 16.00 GB

Processor: Intel(R) Core(TM) i7-10510U CPU @ 1.80GHz

Operating System: Windows 10

Algorithm Analysis

In algorithm 1, the complexity of subset method is n*m because the outer loop is m and the inner loop value is n. Therefore, n should multiply with m, then the result is equal to the O(n*m).

In second algorithm, firstly arr1 is sorted from the smallest to the biggest. Then each element of arr2 is searched by using binary search. Each binary search in arr1 has logn complexity and there are m binary search. Therefore, the complexity of algorithm2 is O(m*logn).

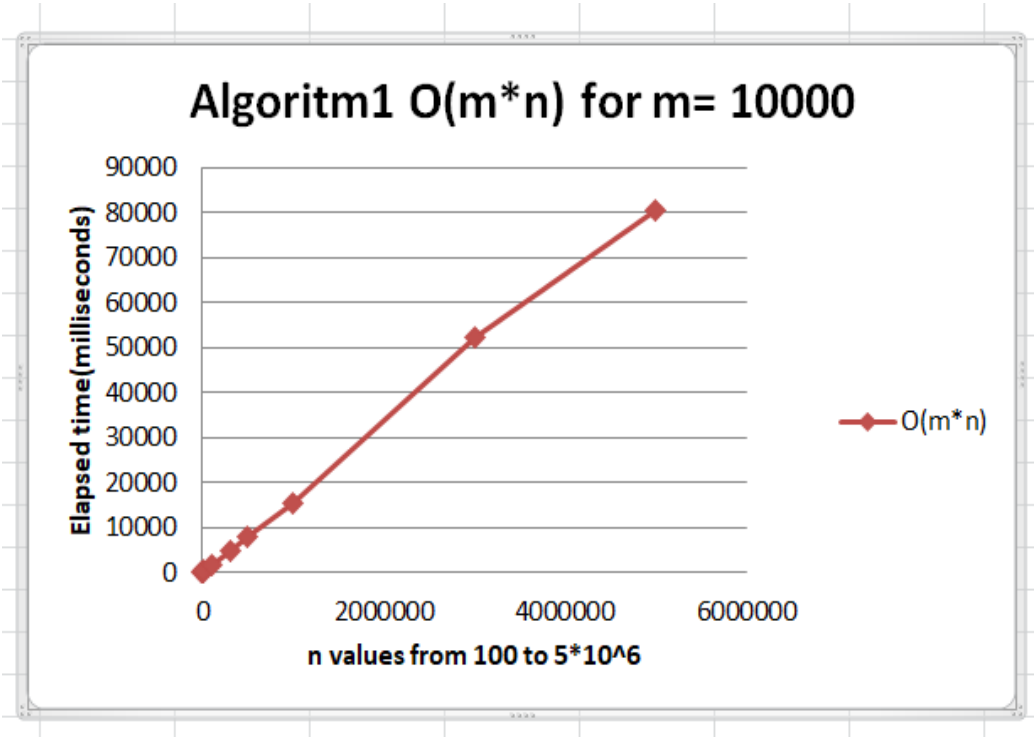
In third algorithm, the frequency table for arr1 is created. Then, for all arr2 elements are searched in frequency table by using its value as an index of frequency table array. Creating frequency table complexity is m because it is used for loops to find maximum value in array and place them. Then, search complexity is n. Consequently, the time complexity is O(m+ n).

There may some differences between theoretical and experimental executing time because of many reasons (worst case- best case situation or compiler environment). However, the difference between these 3 algorithms can be seen clearly by looking the data table and plots.

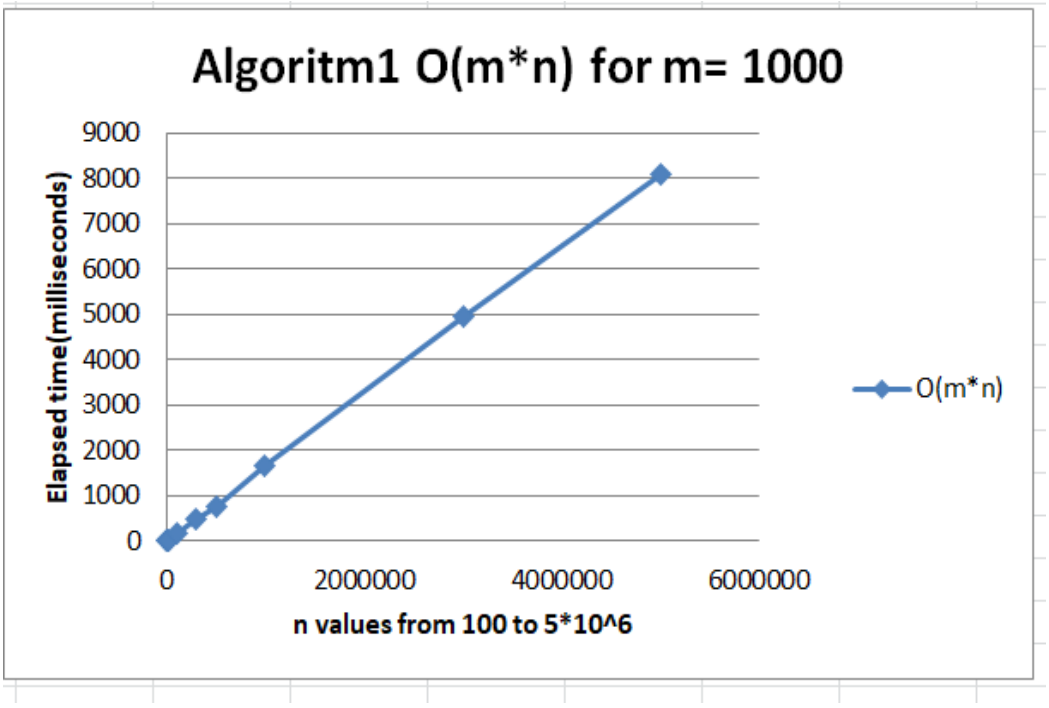
DATA TABLE FOR 3 ALGORİTHMS

n	Algorithm 1		Algorithm 2		Algorithm 3	
	$m = 10^3$	$m=10^4$	$m = 10^3$	$m= 10^4$	$m = 10^3$	$m=10^4$
10^2	0,0003	0,0005	0,0464	0,4905	0,0029	0,02
10^3	0,047	0,351	0.21	0.6579	0.008	0.32
10^4	16.09	162	0.162	1.2092	0.064	0.07
$2*10^4$	32.79	371	0.164	1.2824	0.113	0.16
$3*10^4$	48.11	487	0.162	1.4469	0.142	0.28
10^5	159.36	1633	0.167	1.6116	0.570	0.56
$2*10^5$	322.2	3211	0,169	1.6706	1.027	0.94
$3*10^5$	476.9	4833	0,172	1.60	1.79	1.77
$4*10^5$	653.8	6482	0,175	1.86	2.22	2.22
$5*10^5$	751.9	8092	0.169	2.00	2.49	2.55
10^6	1660	15385	0.20	1.98	6.41	4.93
$3*10^6$	4969	52281	0.19	2.08	15.10	16.87
$5*10^6$	8070	80608	0.21	2.04	27.45	26.08

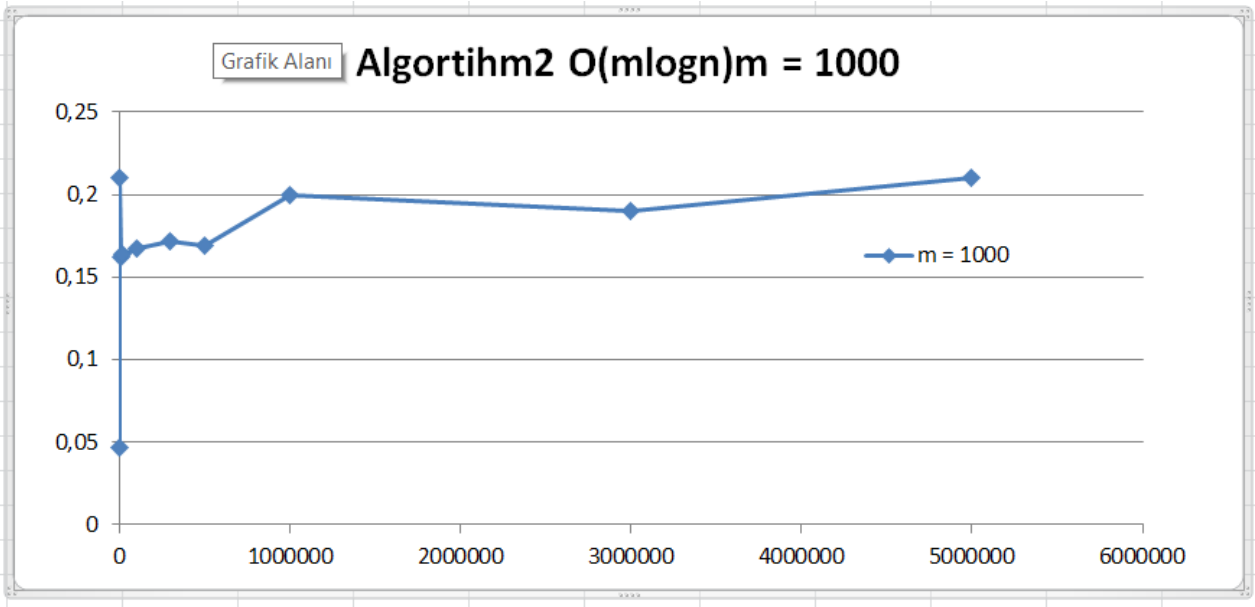
1. The plot1 show the complexity of algorithm1 O(m*n) when $m=10^4$



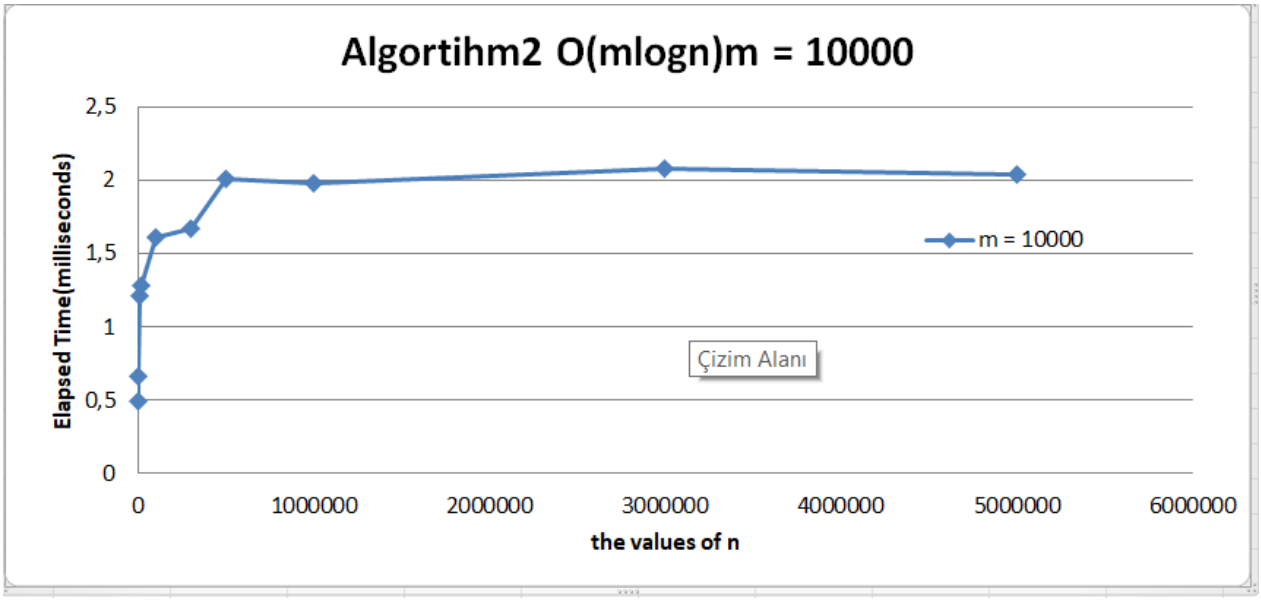
2. The plot2 show the complexity of algorithm1 O(m*n)when $m=10^3$



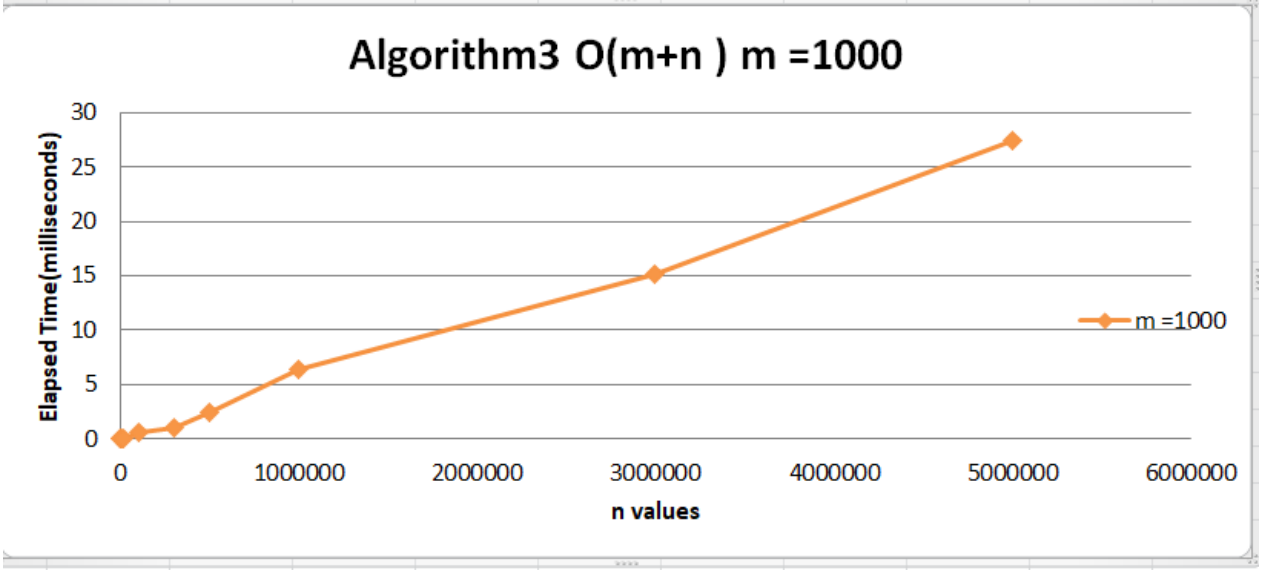
3. The plot3 shows the algorithm2 which has O(mlogn) complexity when m = 10^3



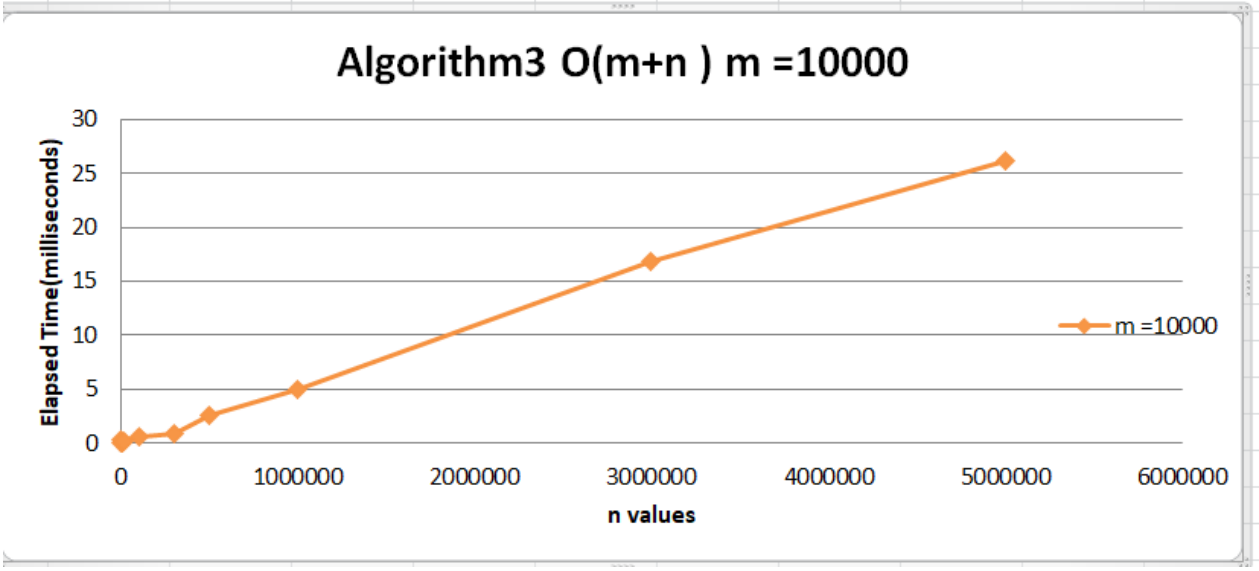
4. The plot4 shows the algorithm2 which has O((logn)*m) complexity when m = 10^4

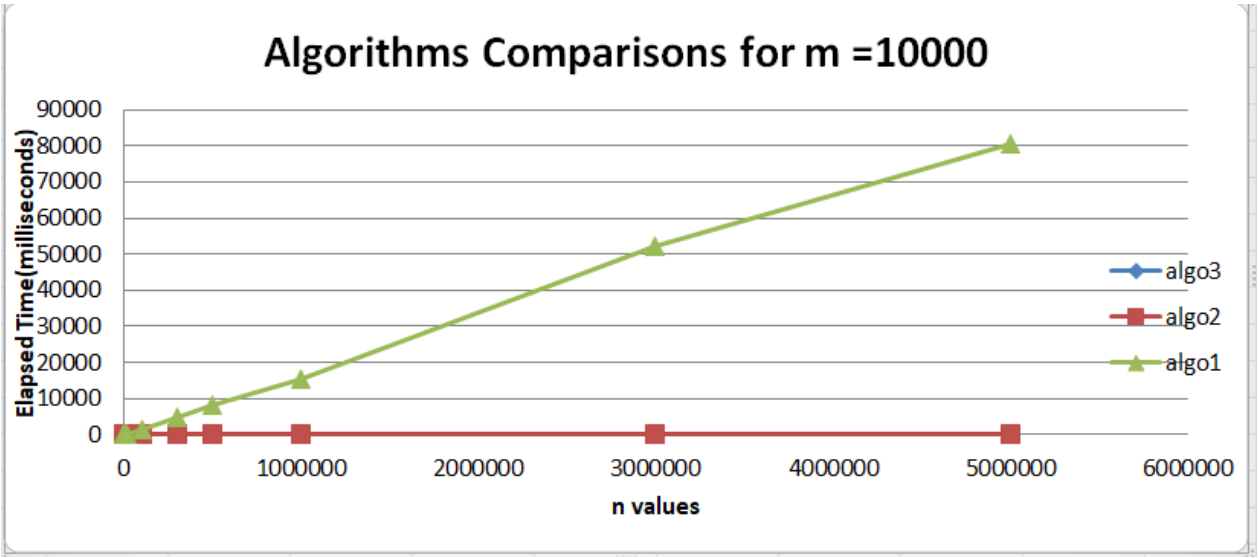


5. The plot5 shows the experimental complexity of algorithm3 O(m+n) when m = 10^3.



6. The plot6 shows the experimental complexity of algorithm3 O(m+n) when m = 10^4.





By using my experiment result, the time of complexity of algoritm3 and algorith2 is negligible compare to algorithm1. There are some differences between theoretical time complexities. However, it seems they work correctly.