2021–2022 FALL SEMESTER

CS201 – HW1

Algorithm Analysis

NAME: ABDULLAH RIAZ

ID : 22001296

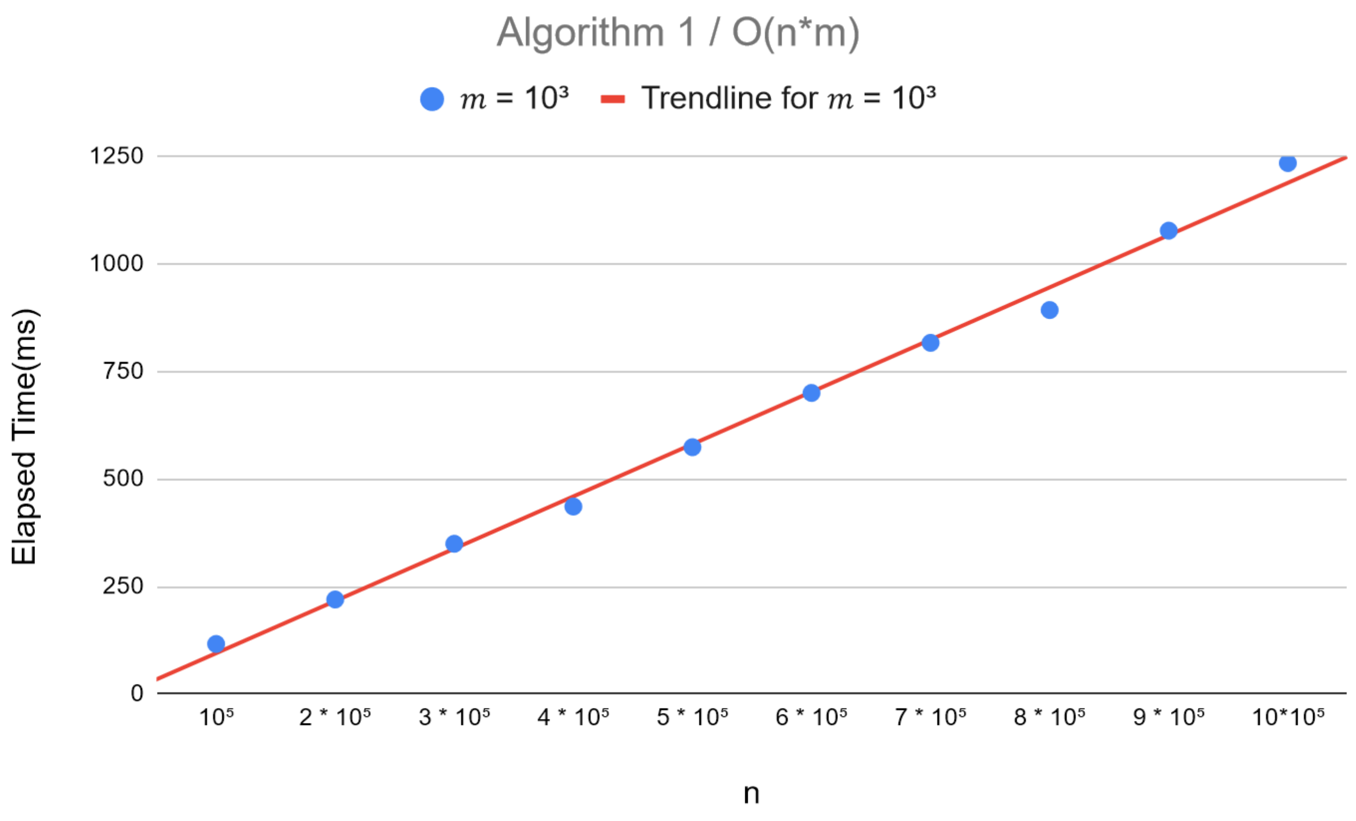
COURSE:CS201 – FUNDAMENTAL STRUCTURES OF COMPUTER SCIENCE I

SECTION: 01

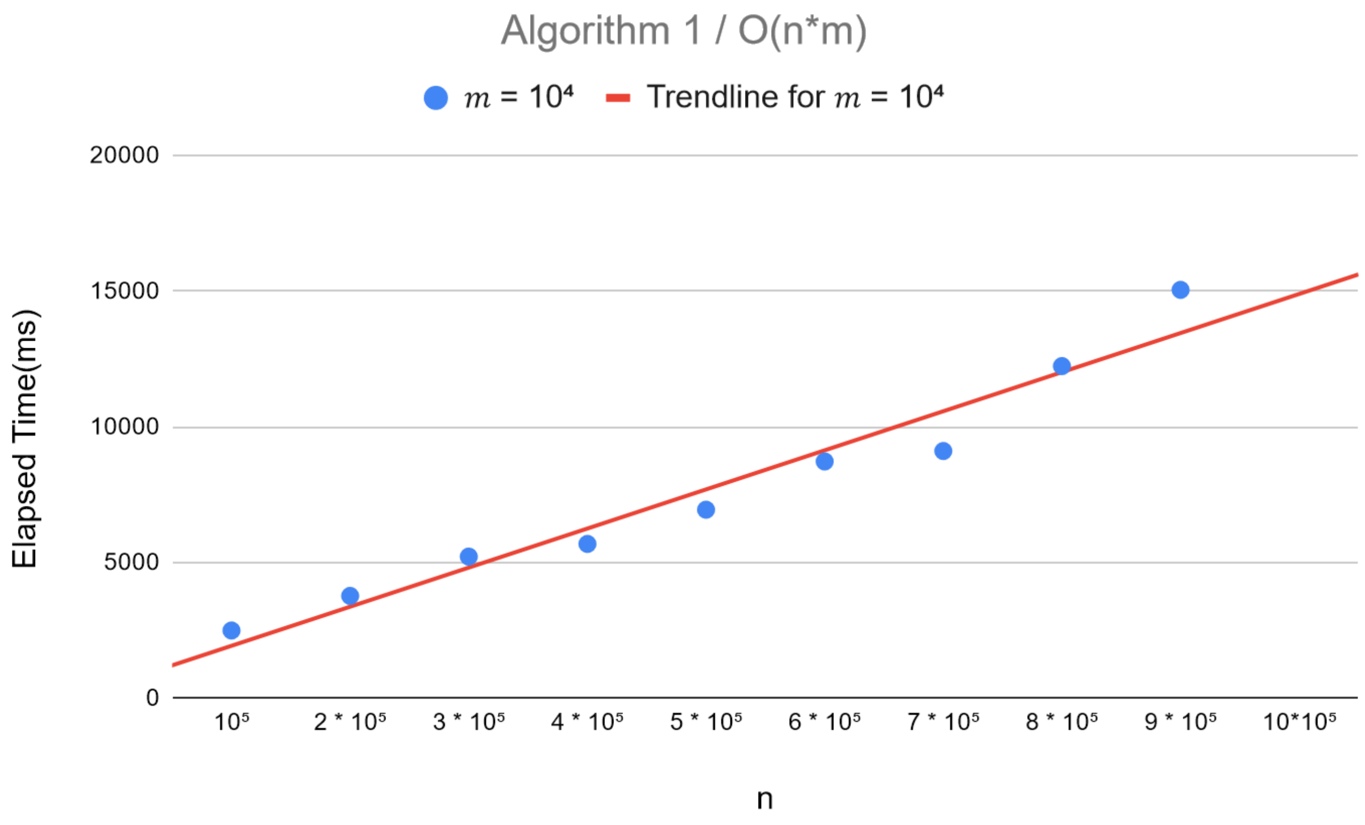
DATE: 05/12/2021

Algorithm Analysis (milliseconds)						
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
10^5	116.4	1135.2	0.013	0.131	0.225	0.253
2 * 10^5	219.8	2505	0.014	0.131	0.459	0.486
3 * 10^5	349.4	3780.3	0.015	0.135	0.691	0.781
4 * 10^5	436.0	5228.5	0.016	0.137	0.998	1.035
5 * 10^5	574	5696.4	0.015	0.136	1.23	1.221
6 * 10^5	700.2	6957	0.016	0.136	1.508	1.45
7 * 10^5	816.8	8735	0.016	0.138	1.648	1.598
8 * 10^5	893	9120.8	0.017	0.142	1.761	1.781
9 * 10^5	1077.6	12251.6	0.017	0.143	1.85	1.925
10*10^5	1234.8	15060.8	0.018	0.145	2.076	2.38

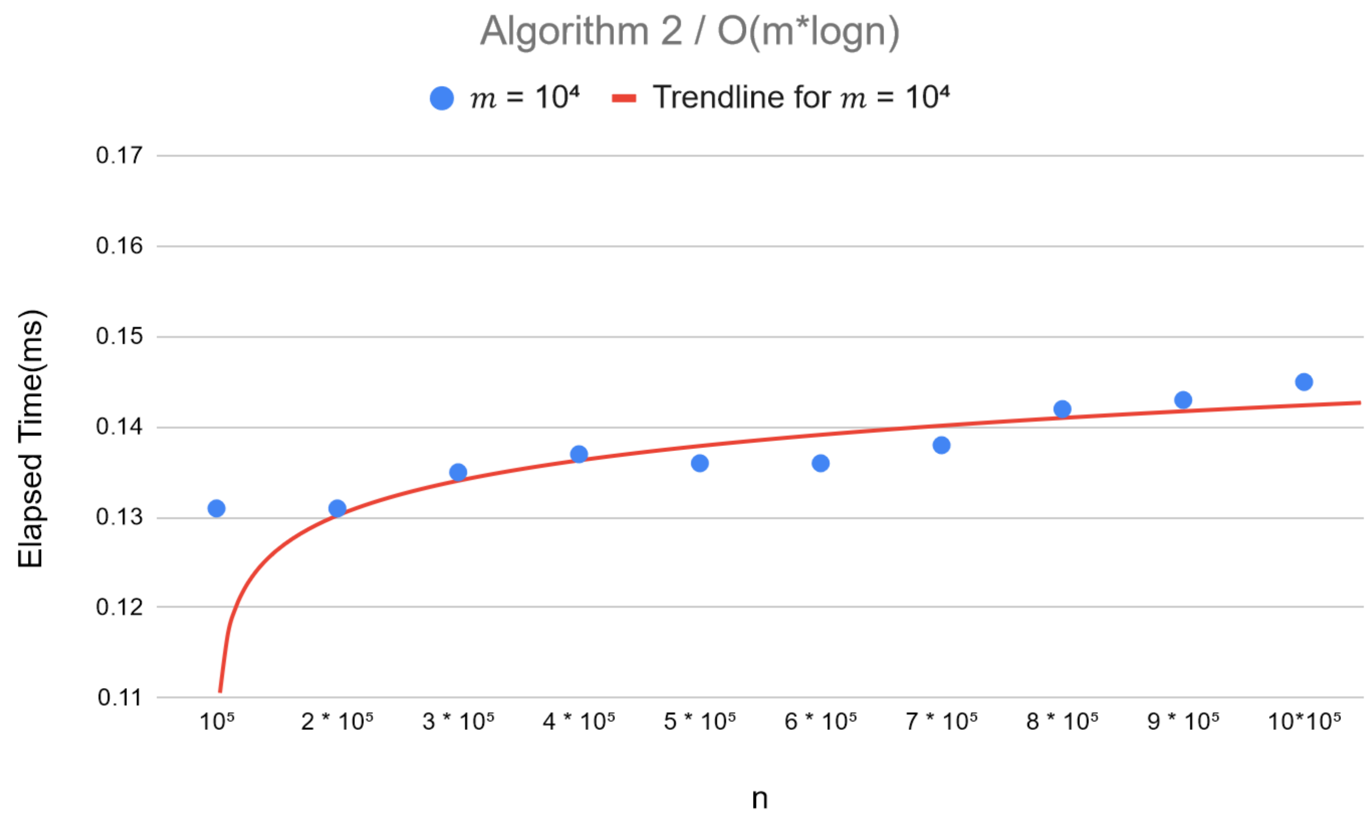
Table 1 : Running time of 3 algorithms



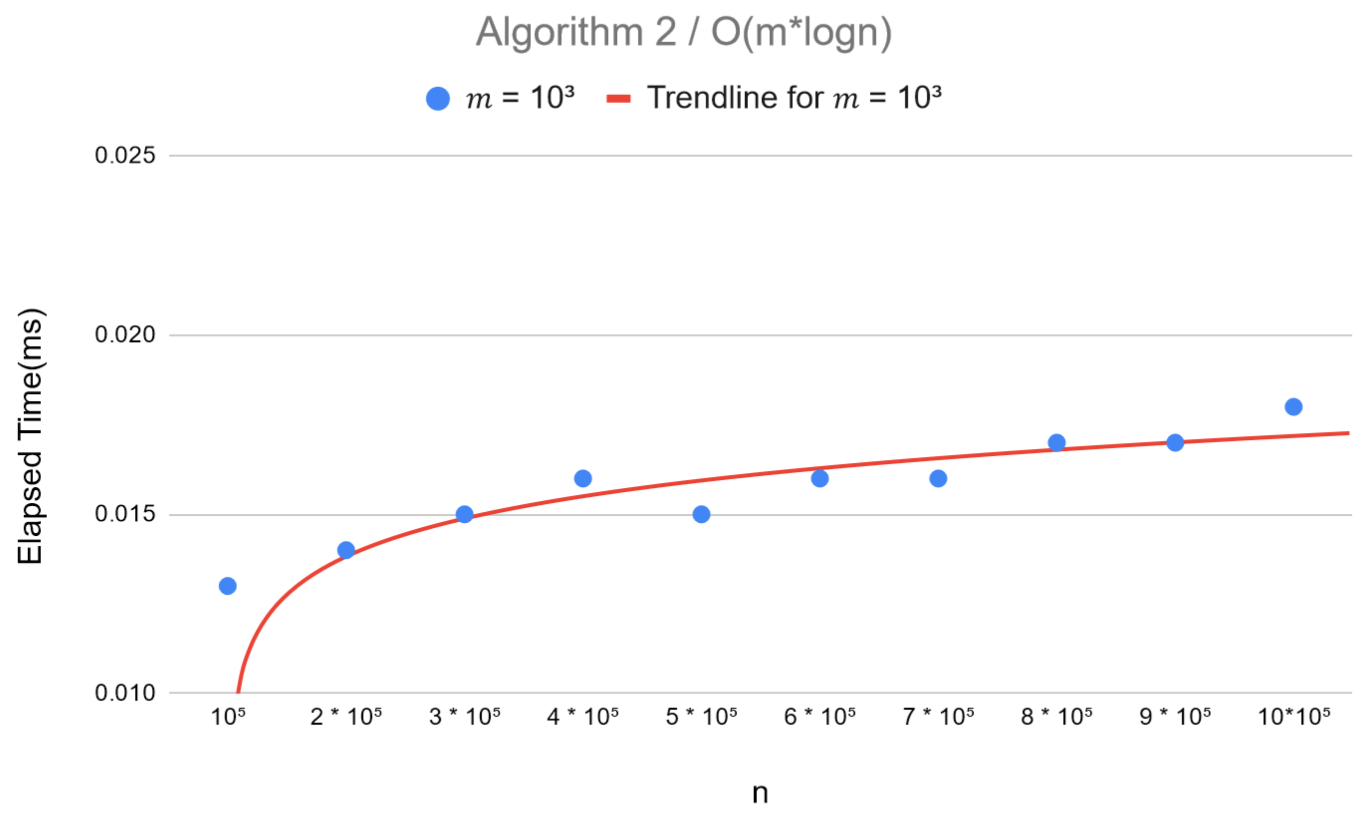
Graph 1: Time Taken vs Input Size(n) for Algorithm 1 when $m = 10^3$



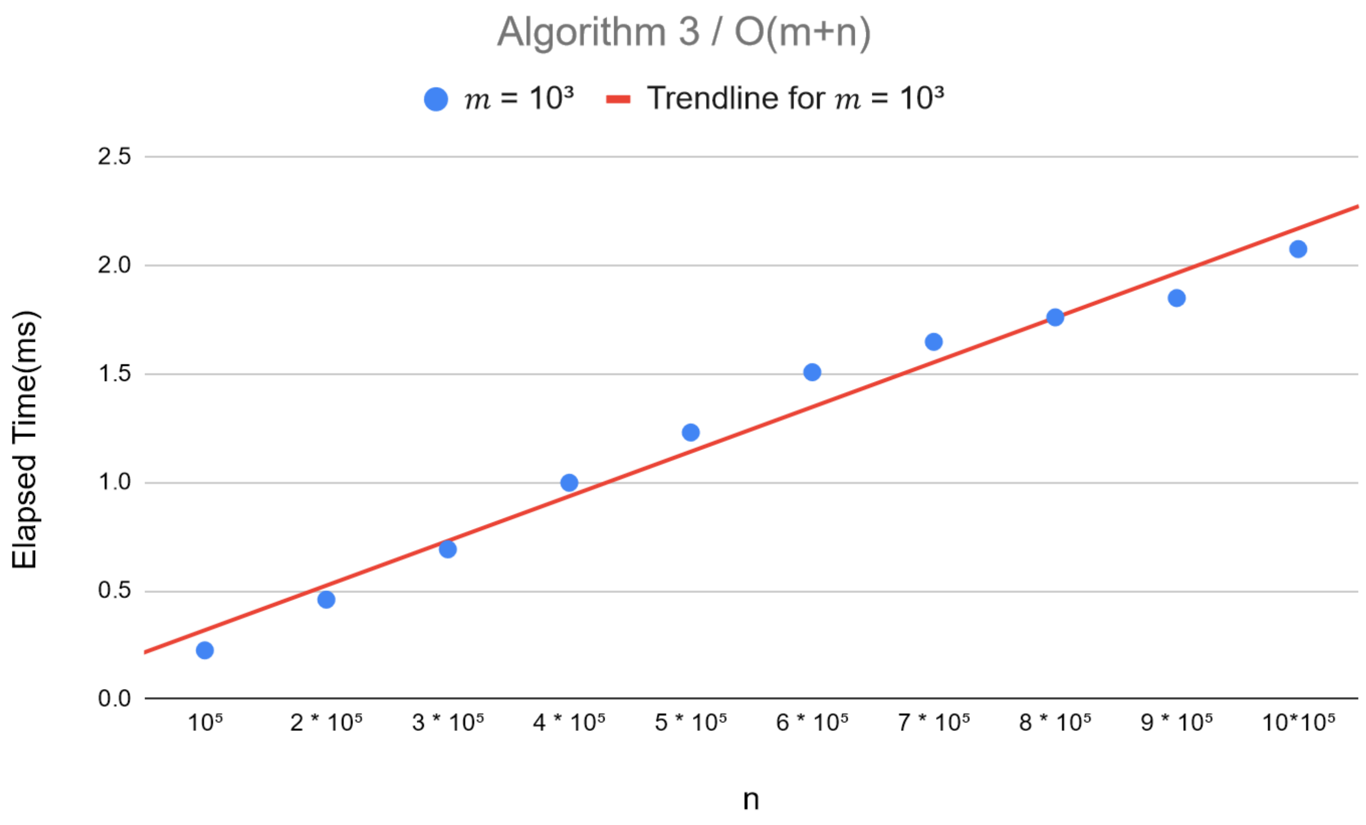
Graph 2: Time Taken vs Input Size(n) for Algorithm 1 when $m = 10^4$



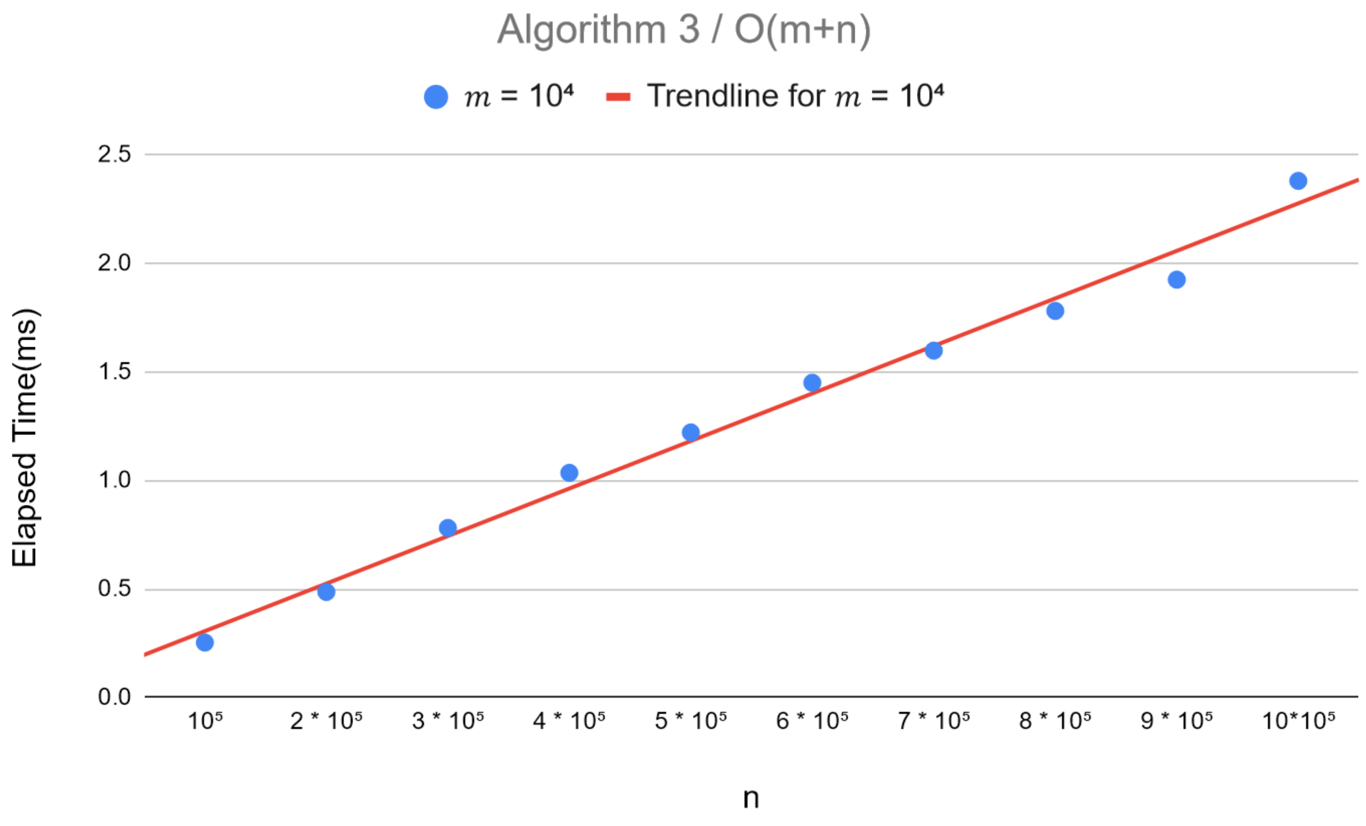
Graph 3: Time Taken vs Input Size(n) for Algorithm 2 when $m = 10^3$



Graph 4: Time Taken vs Input Size(n) for Algorithm 2 when $m = 10^4$



Graph 5: Time Taken vs Input Size(n) for Algorithm 3 when $m = 10^3$



Graph 6: Time Taken vs Input Size(n) for Algorithm 3 when $m = 10^4$

Results:

The algorithm results were inline with the time complexity of each algorithm.

The fastest was Algorithm 2, which uses sorted arrays with binary search. As the size of array 1(n) increases, the time taken by the algorithm remains almost the same with small logarithmic growth. The most significant increase came with an increase in the size of array 2(m), which can be expected as the complexity of this algorithm is O(m*log n), and increases in 'm' affect the time relatively much more than n. This time complexity is derived from the algorithm's for-loop that loops over the array 2, which gives the 'm'. Nested within the for-loop is a line of code that repeatedly divides by a constant. This gives the algorithm the 'log n,' and we get O(m*log n) since it's nested.

Algorithm 3(frequency tables) was the second-fastest, and algorithm 1(linear search) was the slowest. This is also inline with their time complexities O(n+m) and O(n*m), respectively. An increase in the size of any array increases the time taken for algorithm 1 quite significantly more than the change in algorithm 2. Since Algorithm 3 has two independent for-loops, we get O(n+m), and in algorithm 1, the second loop is nested hence O(n*m).

To sum it up, the effectiveness of algorithms can be summed up as O(m*log n) > O(n+m) >O(n*m).

Computer Specification
Lenovo Legion Y520
Processor: Intel(R) Core(TM) i5-7300HQ CPU @ 2.50GHz 4 Core(s)
Memory: 16GB DDR4
Graphics: Nvidia GTX1060 Max-Q
Storage: 512GB SSD