Kutay Şenyiğit
21902377

CS 201 Homework 2

1)

For given two arrays arr1 and arr2 with a size n and m respectively,

Algorithm 1 (Simple algorithm using linear search):

Nested two for loops are used for this algorithm. In the worst case, the first for loop will be iterated n times while the second will be iterated m times. Therefore, it's time complexity is O(n*m). (It's multiplication because for loops are nested)

Algorithm 2 (Using sorted arrays with binary search):

For a given sorted array in this algorithm, since binary search will be handled into n elements of array arr1, the binary search's time complexity will be O(logn). It is because, in the binary search algorithm, elements will be divided into 2 groups and then will be decided to which piece will be looked for. It requires a log(n) times process to find the key value to be wanted in the worst case. Moreover, to find out whether an array arr2 is an subset of an array arr1, we need to check search every elements of arr2 in arr1 so that we can understand whether array arr2 is an subset of array arr1.Since we have to know what will be searched, every elements of array arr2 should be iterated. Its time complexity will be O(m). Therefore, since for every element we use binary search algorithm 2's time complexity will be O(m*log(n)).(It's multiplication because they are nested)

Algorithm 3 (Using frequency tables):

In this algorithm, first of all, for loop is used to determine how many times the elements occur in arr1. To do that we should linearly take every element and increase its frequency. This process's time complexity is O(n) because arr1 has n elements.Secondly, another for loop is used to iterate all elements in arr2 to check arr1 has those elements. To do so, we iterate each element in arr2 and we decrease its frequency in arr1. This process's time complexity is O(m), because arr2 has m elements. Therefore, since two processes handled separately algorithm's time complexity will be O(n + m). (It's summation because they are separate)

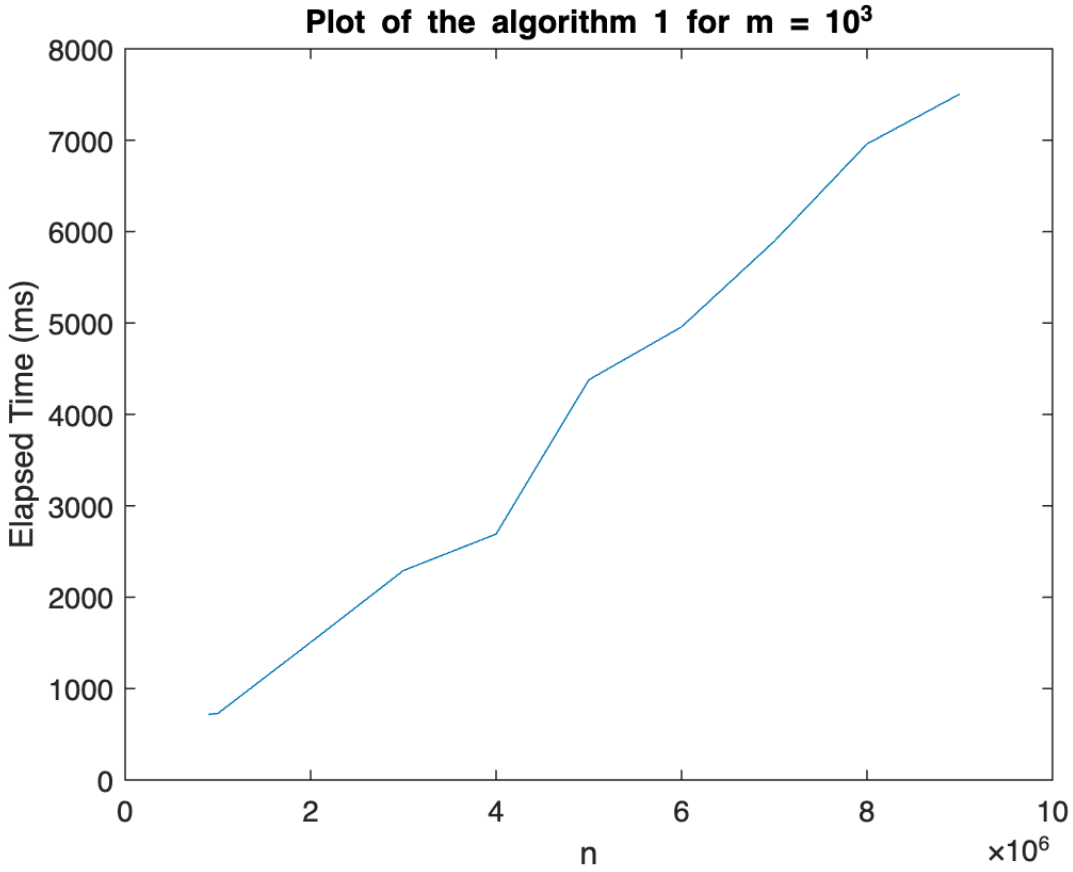
2)

Processor: 1,1 GHz Dual-Core Intel Core i3
Memory: 8 GB 3733 MHz LPDDR4X

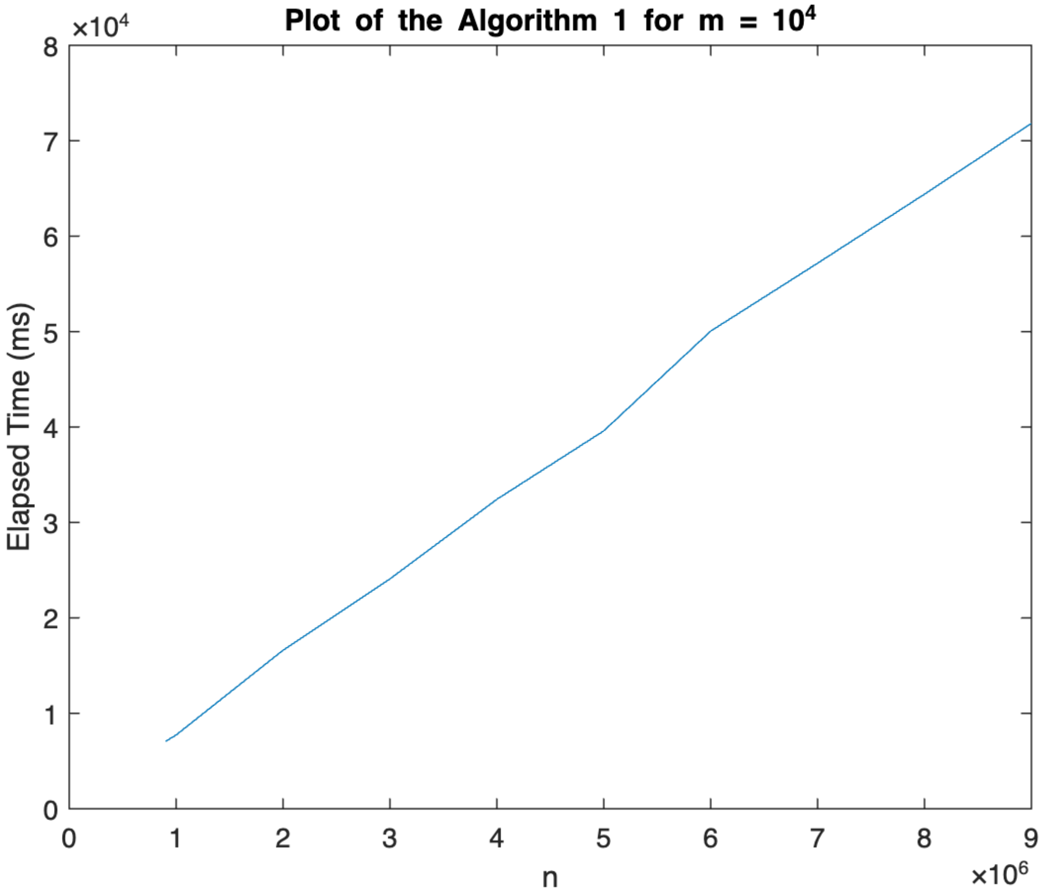
3)

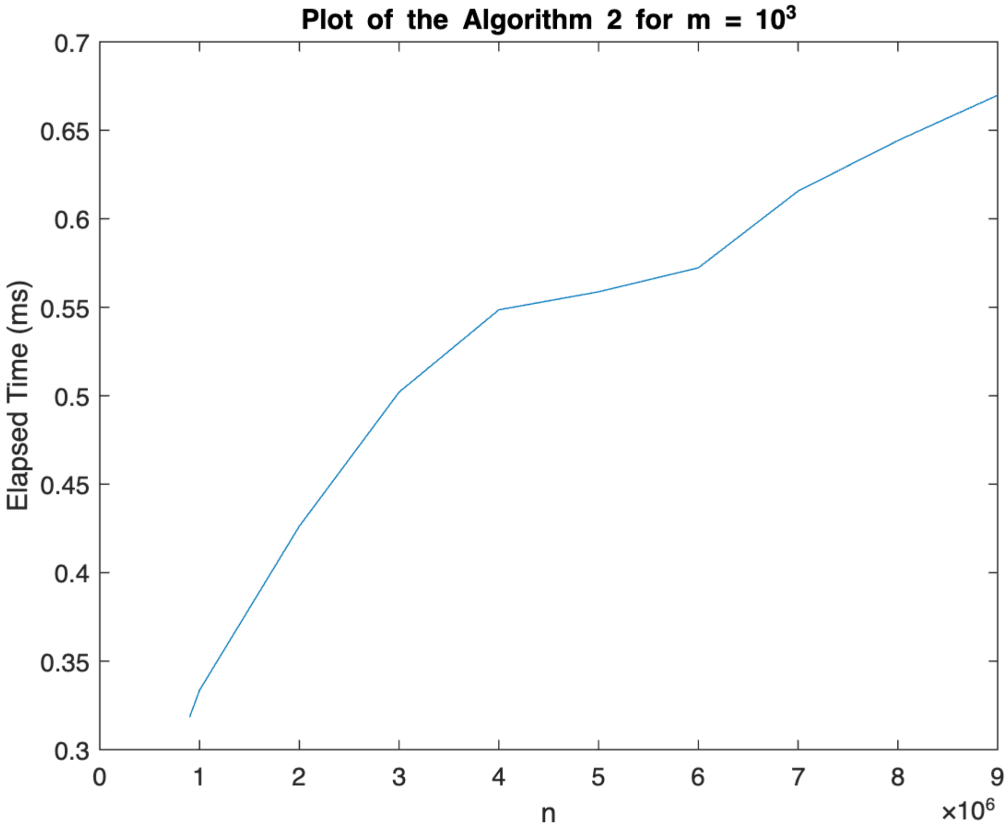
	Algorithm 1		**Algorithm 2**		**Algorithm 3**	
n	$m = 10^3$	$m = 10^4$	$m = 10^3$	$m = 10^4$	$m = 10^3$	$m = 10^4$
$9 * 10^5$	**717.33ms**	**7047.3ms**	**0.31829ms**	**4.27663ms**	**6.14568ms**	**7.71358ms**
$1 * 10^6$	**728.79ms**	**7731.3ms**	**0.33358ms**	**4.37966ms**	**6.68501ms**	**10.457ms**
$2 * 10^6$	**1506.1ms**	**16613.4ms**	**0.42625ms**	**4.96757ms**	**14.2901ms**	**18.4131ms**
$3 * 10^6$	**2291.4ms**	**24087.1ms**	**0.50201ms**	**5.20424ms**	**20.8373ms**	**28.917ms**
$4 * 10^6$	**2690.5ms**	**32425.7ms**	**0.54855ms**	**5.73866ms**	**27.9691ms**	**40.5372ms**
$5 * 10^6$	**4378.7ms**	**39601.1ms**	**0.55874ms**	**6.77574ms**	**35.3613ms**	**57.2198ms**
$6 * 10^6$	**4957.7ms**	**50053.1ms**	**0.57228ms**	**7.12205ms**	**43.646ms**	**58.6668ms**
$7 * 10^6$	**5894.4ms**	**57169.1ms**	**0.61566ms**	**7.3111ms**	**49.835ms**	**66.5179ms**
$8 * 10^6$	**6959.6ms**	**64405.5ms**	**0.64422ms**	**7.53854ms**	**56.527ms**	**75.5438ms**
$9 * 10^6$	**7505.3ms**	**71822.2ms**	**0.66981ms**	**7.98113ms**	**60.6623ms**	**86.6325ms**

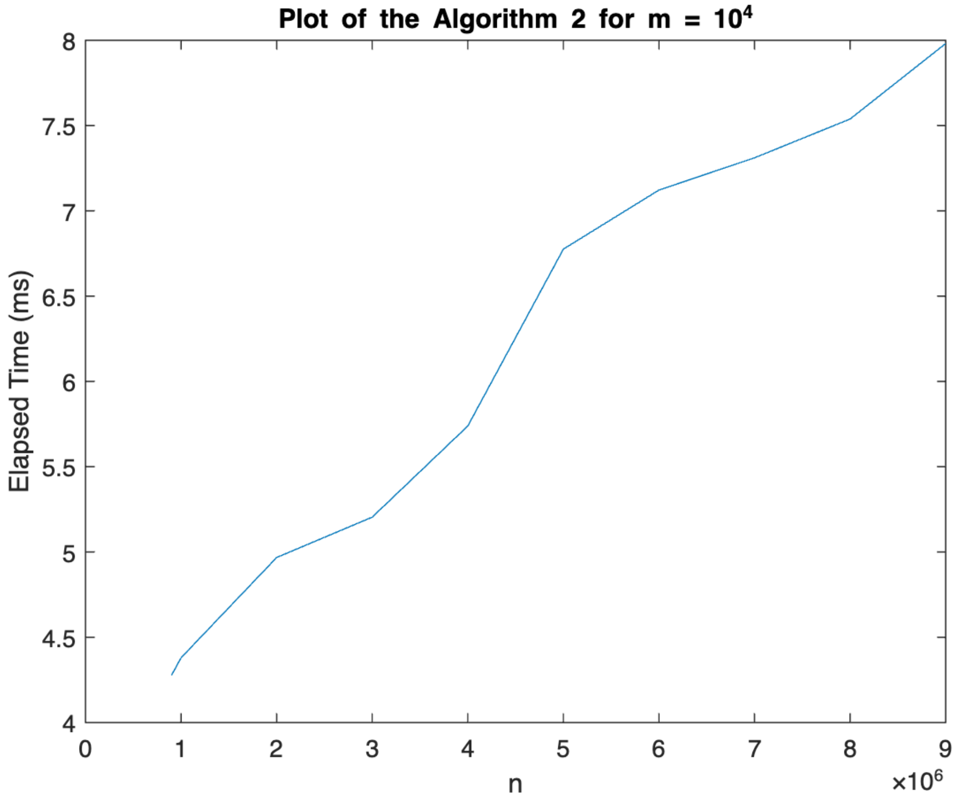
4)

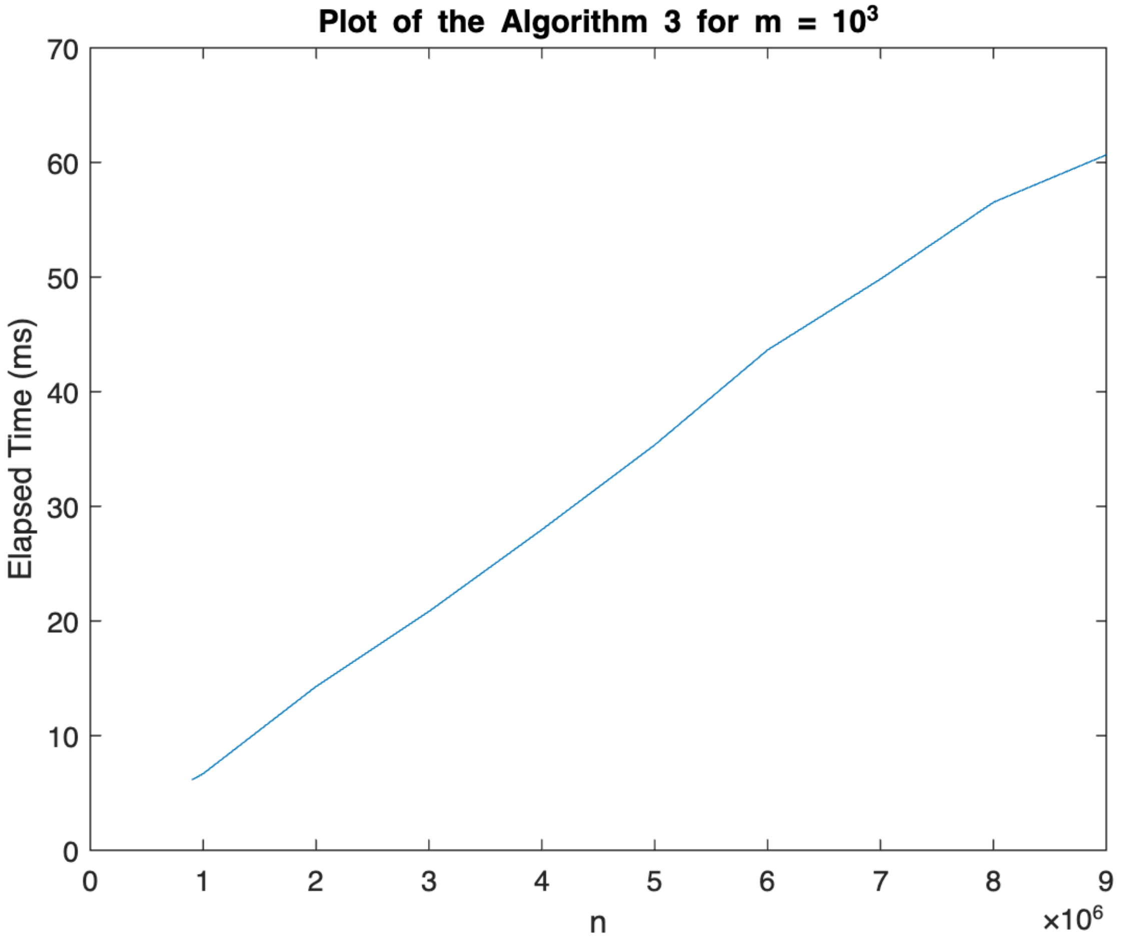


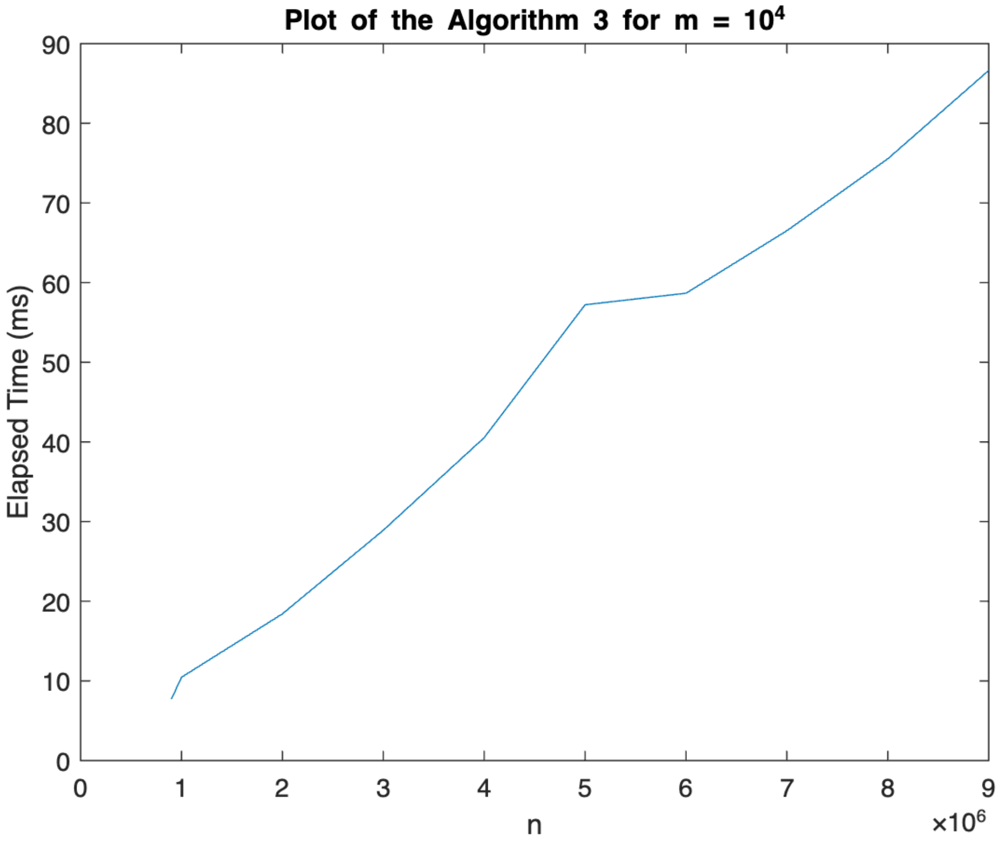
Plot of the algorithm 1 for m = 10^3
Elapsed Time (ms)
8000
7000
6000
5000
4000
3000
2000
1000
0
0
2
4
6
8
10
n
×10^6


Plot of the Algorithm 1 for m = 10^4
×10^4
8
7
6
5
4
3
2
1
0
Elapsed Time (ms)
0
1
2
3
4
5
6
7
8
9
n
×10^6


Plot of the Algorithm 2 for m = 10^3
Elapsed Time (ms)
0.7
0.65
0.6
0.55
0.5
0.45
0.4
0.35
0.3
0
1
2
3
4
5
6
7
8
9
n
×10^6


Plot of the Algorithm 2 for m = 10^4
Elapsed Time (ms)
8
7.5
7
6.5
6
5.5
5
4.5
4
0
1
2
3
4
5
6
7
8
9
n
×10^6


Plot of the Algorithm 3 for m = 10^3
Elapsed Time (ms)
0
10
20
30
40
50
60
70
n
0
1
2
3
4
5
6
7
8
9
×10^6


Plot of the Algorithm 3 for m = 10^4
Elapsed Time (ms)
0
10
20
30
40
50
60
70
80
90
n
0
1
2
3
4
5
6
7
8
9
×10^6