Name: Alp Tuğrul Ağçalı

ID: 21801799

Section:3

CS 201 HOMEWORK 2 REPORT

1.The time complexity of algorithm 1 is O(m*n) so it's upper bound is m*n. It searches one by one whether each element of arr2 is existing in arr1 by checking all of elements of arr1. Checking all elements in arr 1 takes O(n) and while checking this checking elements in the arr2 O(m) so time complexity is O(m*n).

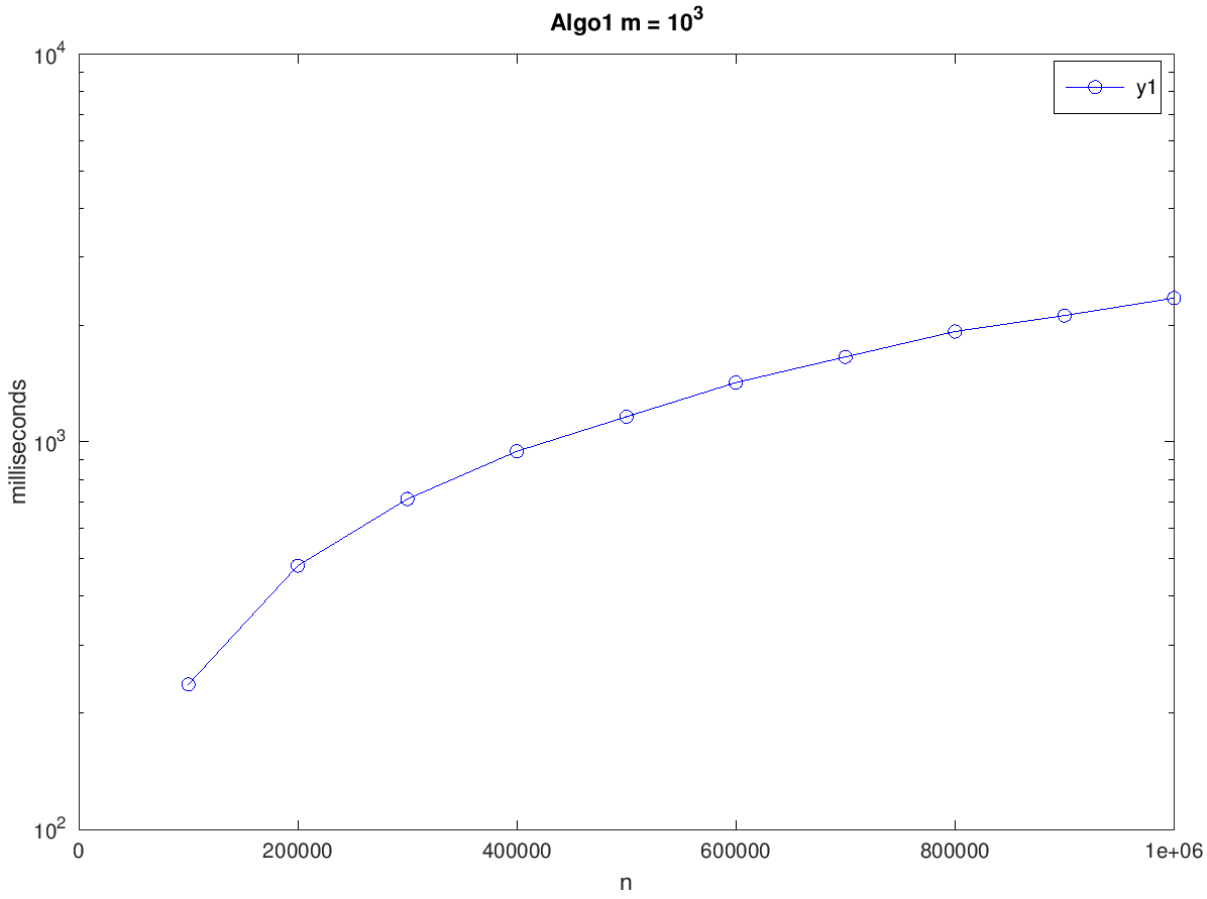
The time complexity of algorithm 2 is O(m*Log(n)) so it's upper bound is m*Log(n) (where n is the size of arr1 and m is size of arr2). In this algorithm we need sorted arr1. Because the algorithm firstly finds the middle element of arr1. Then if the value which is searched in arr1 is bigger or smaller than middle, algorithm starts to check bigger or smaller half of array with dividing this half same way and it continues until it finds. Checking the arr1 with dividing it 2 takes O(logn) and checking it for every element in arr2 takes O(m) so, time complexity is O(m*log(n)).

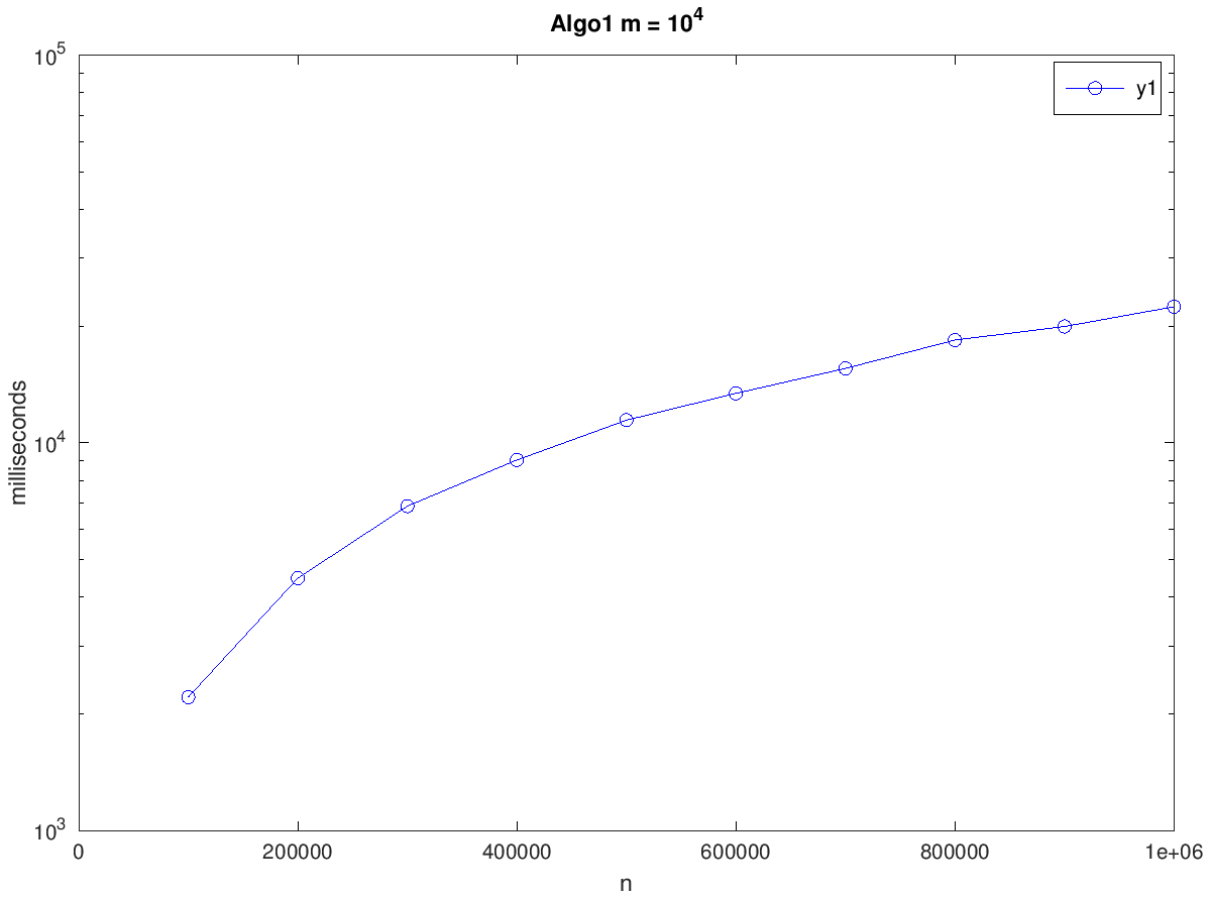
The time complexity of algorithm 3 is O(n+m) so it's upper bound is m+n. In this algorithm we create a frequency table whose size is the biggest element in arr1. Then by checking all of elements in arr1 we count the frequency of elements. After that, algorithm checks whether the elements of arr2 exists in frequency table. Checking every element in arr1 to finding their frequency takes O(n) time. After that checking every element in arr2 takes O(m) time so, time complexity is O(m+n) time.

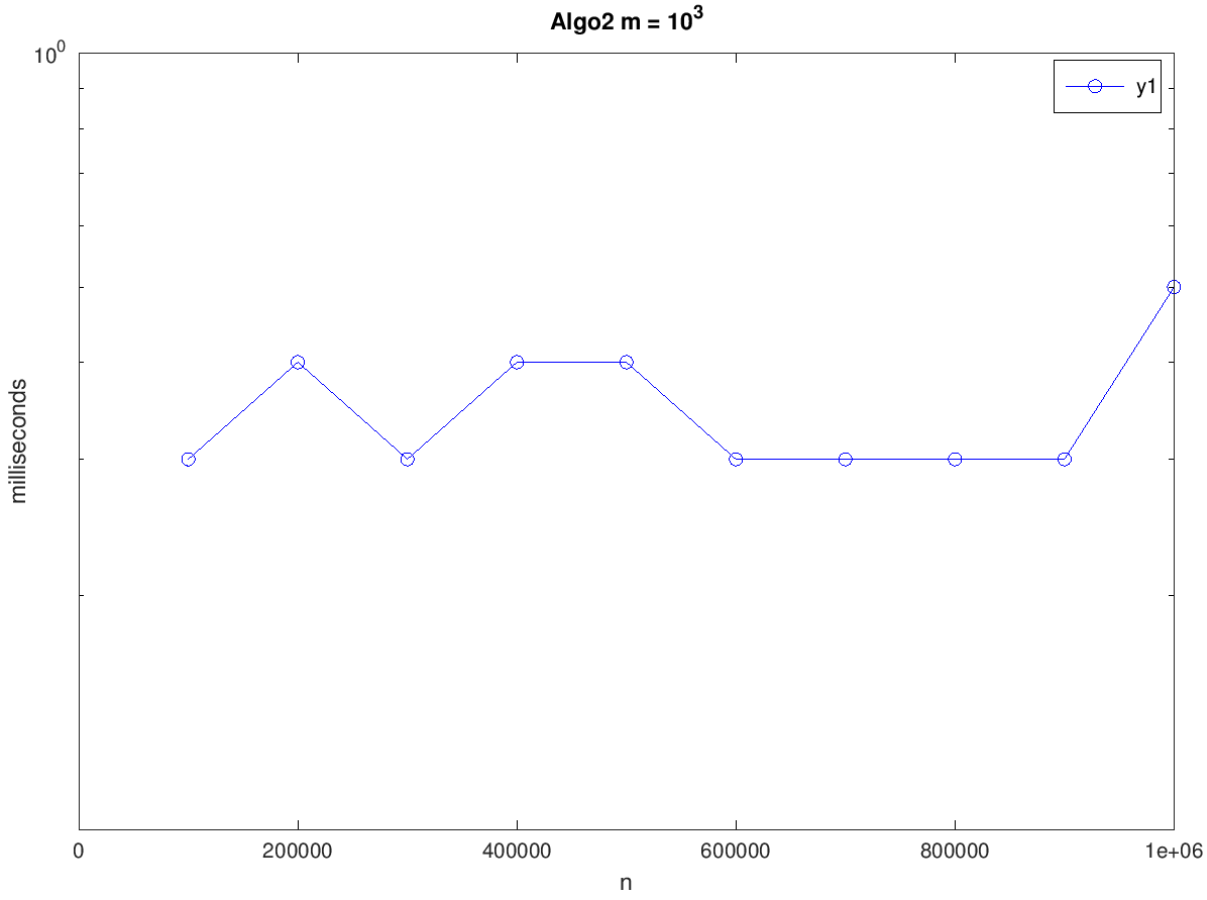
2. RAM 16GB, Processor Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz 2.81 GHz.

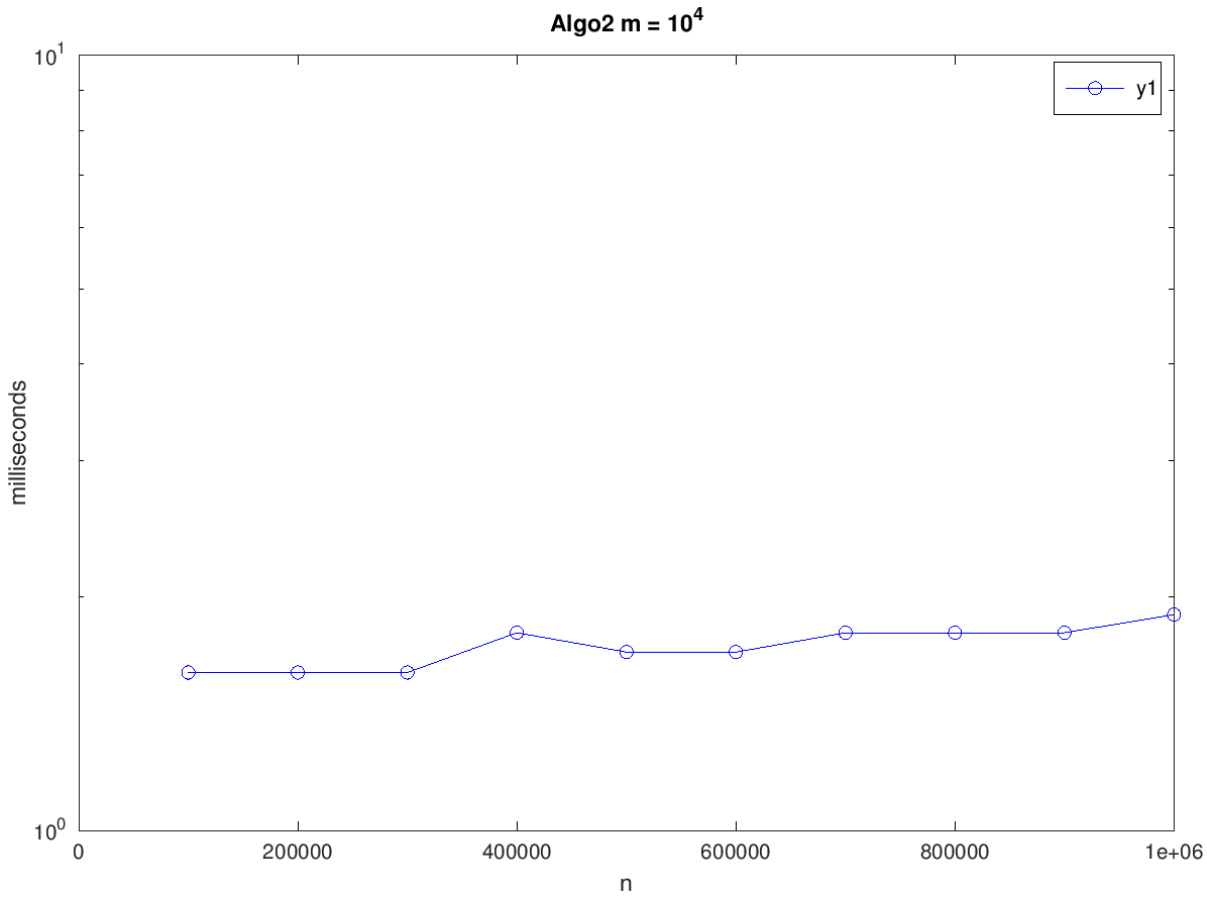
3.

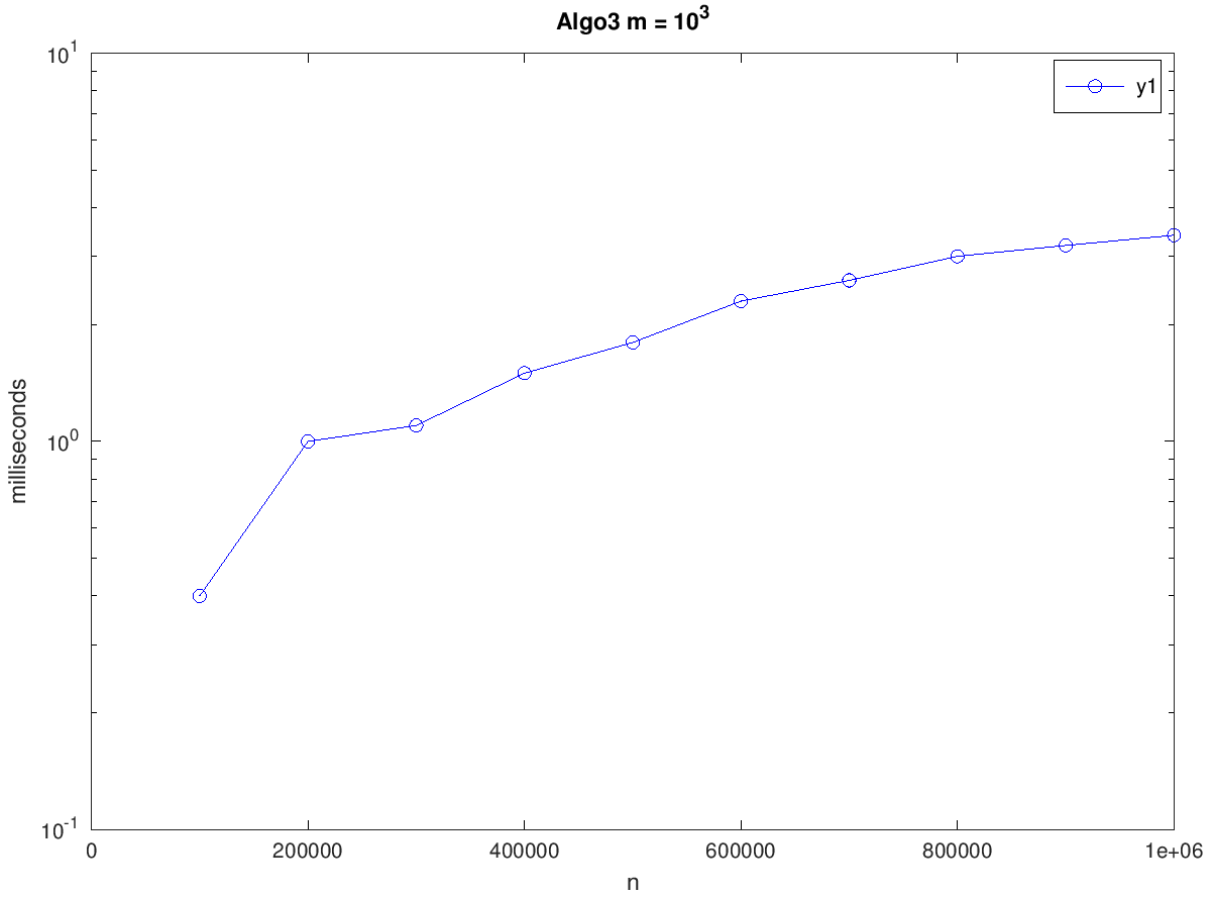
n	ALGORİTHM 1		ALGORİTHM 2		ALGORİTHM 3	
	$m = 10^3$	$m = 10^4$	$m = 10^3$	$m = 10^4$	$m = 10^3$	$m = 10^4$
10^5	237	2211	0,3	1,6	0,4	0,6
$2*10^5$	480	4484	0,4	1,6	1	1,1
$3*10^5$	713	6875	0,3	1,6	1,1	1,2
$4*10^5$	946	9040	0,4	1,8	1,5	1,6
$5*10^5$	1162	11456	0,4	1,7	1,8	2,1
$6*10^5$	1423	13433	0,3	1,7	2,3	2,5
$7*10^5$	1657	15567	0,3	1,8	2,6	2,6
$8*10^5$	1928	18429	0,3	1,8	3	2,9
$9*10^5$	2119	19975	0,3	1,8	3,2	3,3
10^6	2350	22462	0,5	1,9	3,4	3,8

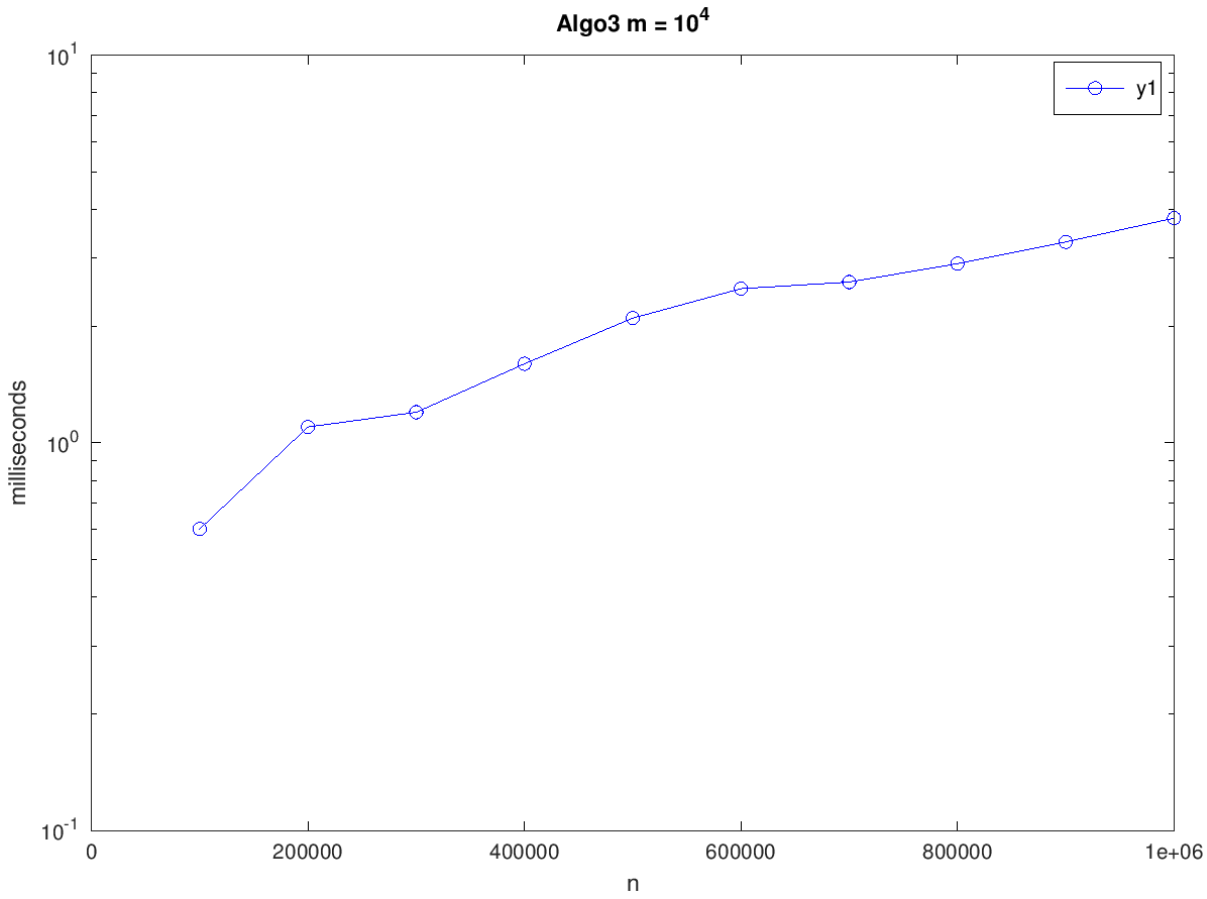

Algo1 m = 10^3
10^4
10^3
10^2
milliseconds
0
200000
400000
600000
800000
1e+06
n
y1


Algo1 m = 10^4
y1
10^5
10^4
10^3
milliseconds
0
200000
400000
600000
800000
1e+06
n


Algo2 m = 10^3
y1
10^0
milliseconds
0
200000
400000
600000
800000
1e+06
n


Algo2 m = 10^4
y1
milliseconds
n
10^1
10^0
0
200000
400000
600000
800000
1e+06


Algo3 m = 10^3
y1
milliseconds
n
10^1
10^0
10^-1
0
200000
400000
600000
800000
1e+06


Algo3 m = 10^4
y1
milliseconds
10^1
10^0
10^-1
0
200000
400000
600000
800000
1e+06
n