**Complexity of Algorithm 1:**

In the worst case, we have to traverse arr1 for arr2's size times. arr1's size is n and arr2's size is m. Thus, the time complexity becomes O(n*m), as m will be traversed n times.

**Complexity of Algorithm 2:**

The time complexity of binary seearch algorithm is O(logN). In Algorithm 2 in the worst case, we have to binary search for arr2's size, which happens to be m. Thus, the time complexity becomes O(m*logn)

**Complexity of Algorithm 3:**

In this algorithm,we traverse both arrays separately, ones time complexity is O(n) and others is O(m). Since they are seperately traversed, the time complexity becomes O(m + n).

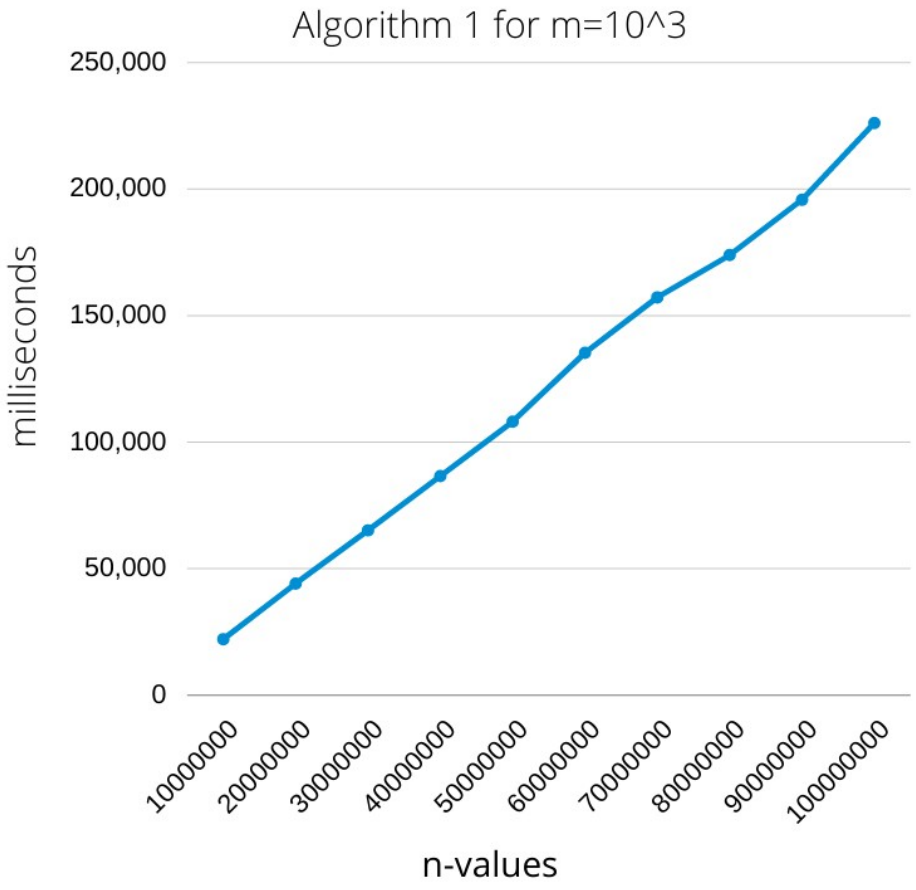
**Computer Parameters**

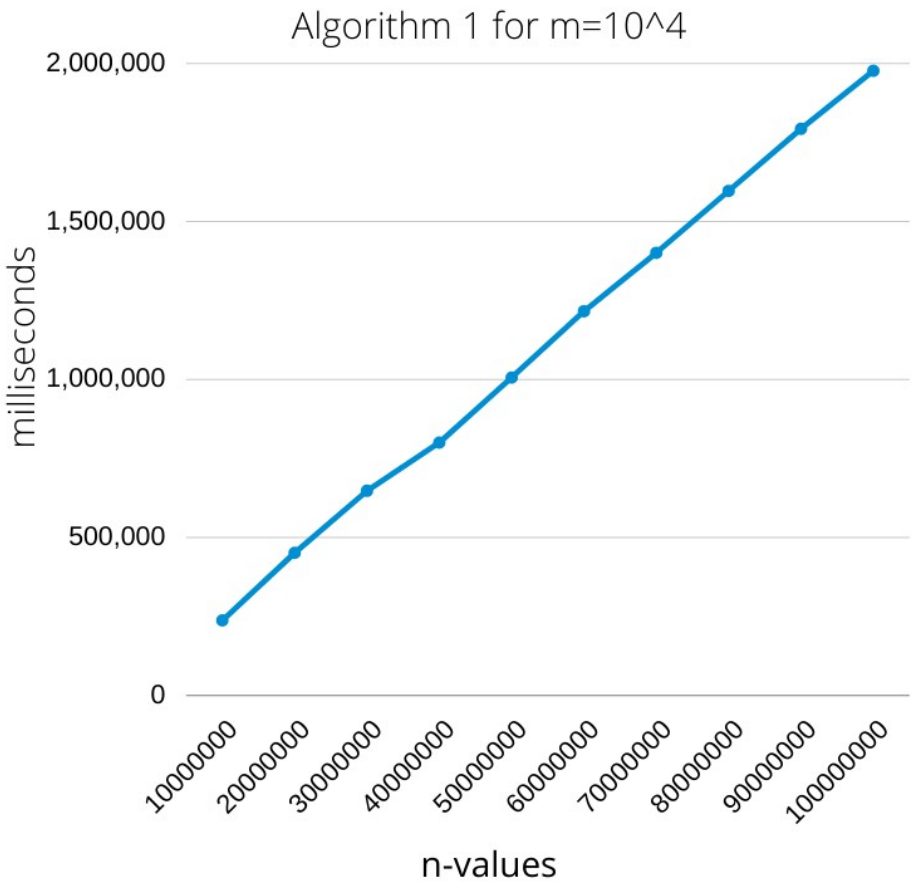
**RAM:** 15.4 GiB

**Processor:** 12 x Intel Core i7-10750H CPU @ 2.60GHz

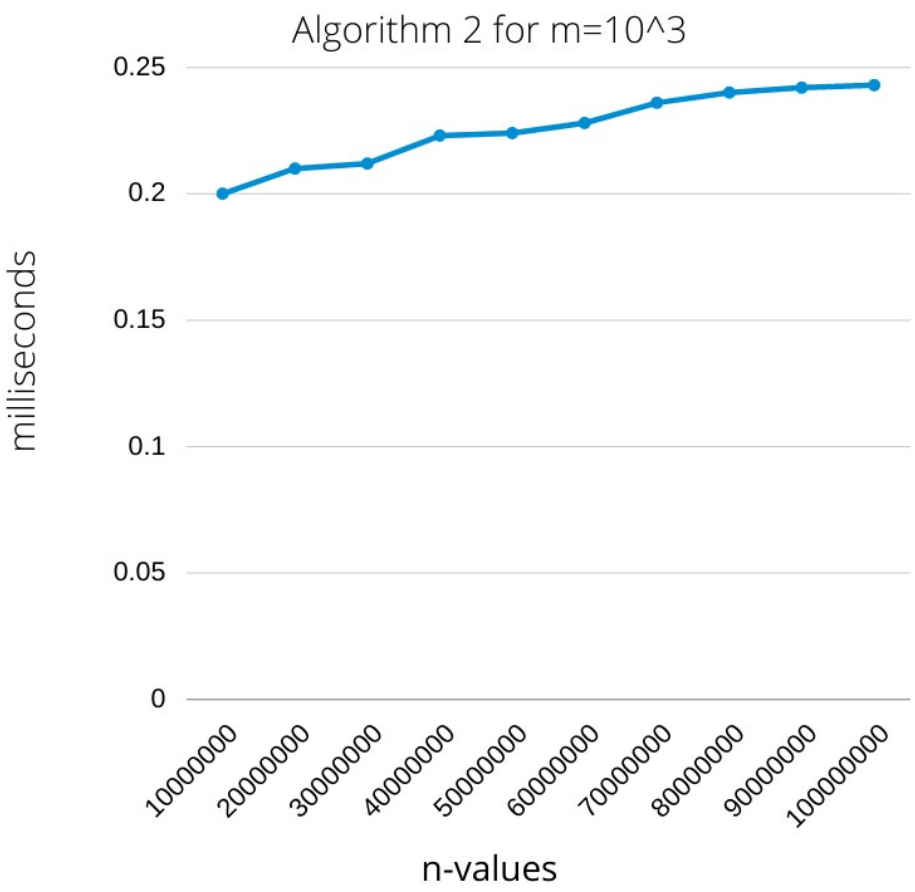
| | Algorithm 1 | Algorithm 1 | Algorithm 2 | Algorithm 2 | Algorithm 3 | Algorithm 3 |
|---|---|---|---|---|---|---|
| n | $m=10^3$ | $m=10^4$ | $m=10^3$ | $m=10^4$ | $m=10^3$ | $m=10^4$ |
| $10^7$ | 22155.9ms | 237228ms | 0.2ms | 1.987ms | 84.527ms | 85.74ms |
| $2\times10^7$ | 44126.7ms | 451126ms | 0.210ms | 2.167ms | 165.998ms | 168.704ms |
| $3\times10^7$ | 65149.3ms | 647464ms | 0.212ms | 2.104ms | 249.082ms | 249.531ms |
| $4\times10^7$ | 86555.1ms | 800134ms | 0.223ms | 2.224ms | 331.962ms | 332.011ms |
| $5\times10^7$ | 108082ms | 1006050ms | 0.224ms | 2.234ms | 414.681ms | 422.667ms |
| $6\times10^7$ | 135291ms | 1216000ms | 0.228ms | 2.236ms | 502.447ms | 513.288ms |
| $7\times10^7$ | 157209ms | 1400790ms | 0.236ms | 2.357ms | 581.467ms | 591.414ms |
| $8\times10^7$ | 173895ms | 1596690ms | 0.240ms | 2.359ms | 665.647ms | 686.06ms |
| $9\times10^7$ | 1195778ms | 1793220ms | 0.242ms | 2.361ms | 745.046ms | 777.635ms |
| $10\times10^7$ | 226109ms | 1976750ms | 0.243ms | 2.362ms | 826.61ms | 847.75ms |

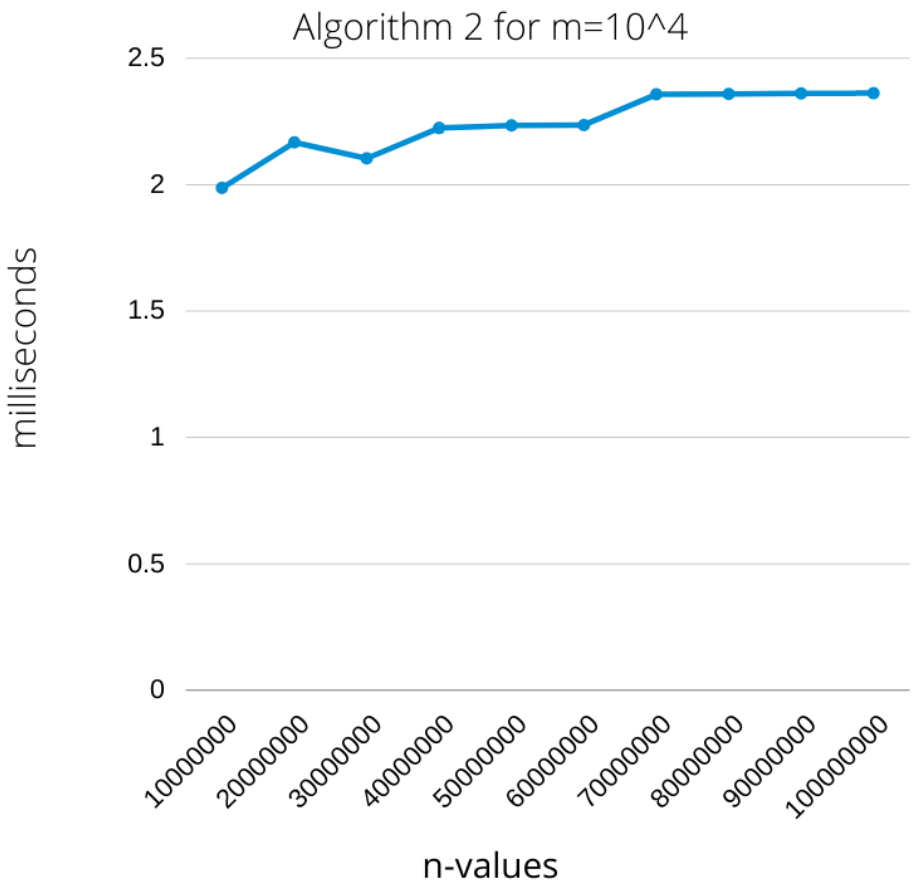
**Algorithm 1:**


Algorithm 1 for m=10^3
250,000
200,000
150,000
100,000
50,000
0
milliseconds
10000000
20000000
30000000
40000000
50000000
60000000
70000000
80000000
90000000
100000000
n-values

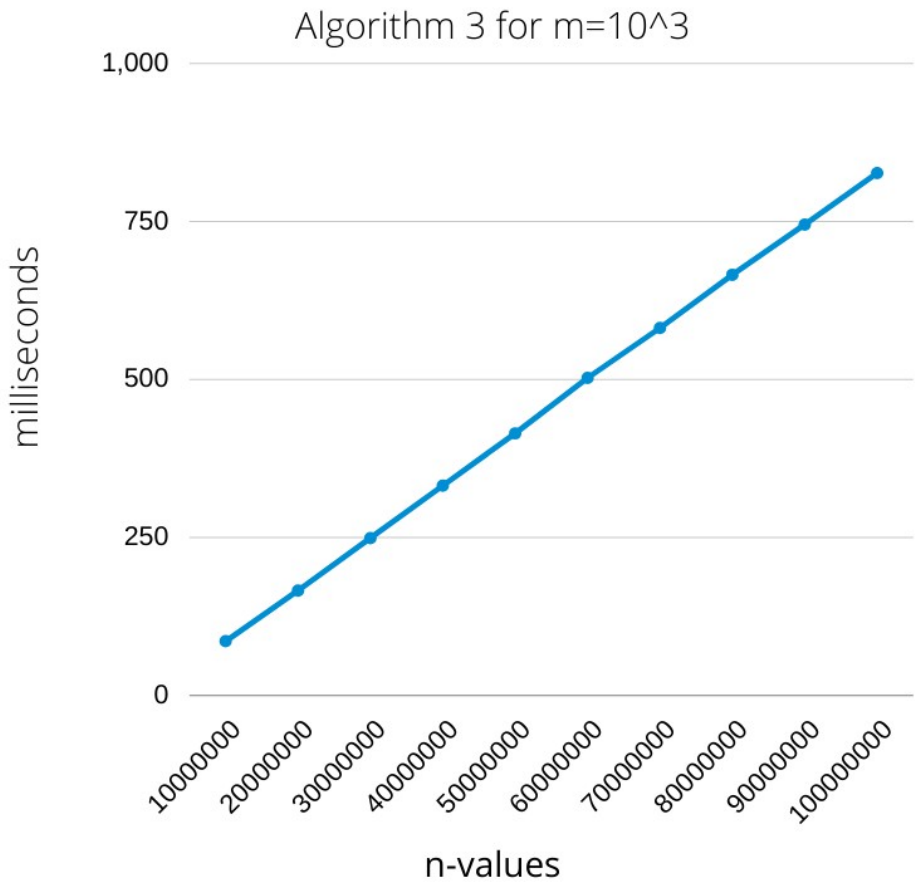

Algorithm 1 for m=10^4
2,000,000
1,500,000
1,000,000
500,000
0
milliseconds
10000000
20000000
30000000
40000000
50000000
60000000
70000000
80000000
90000000
100000000
n-values

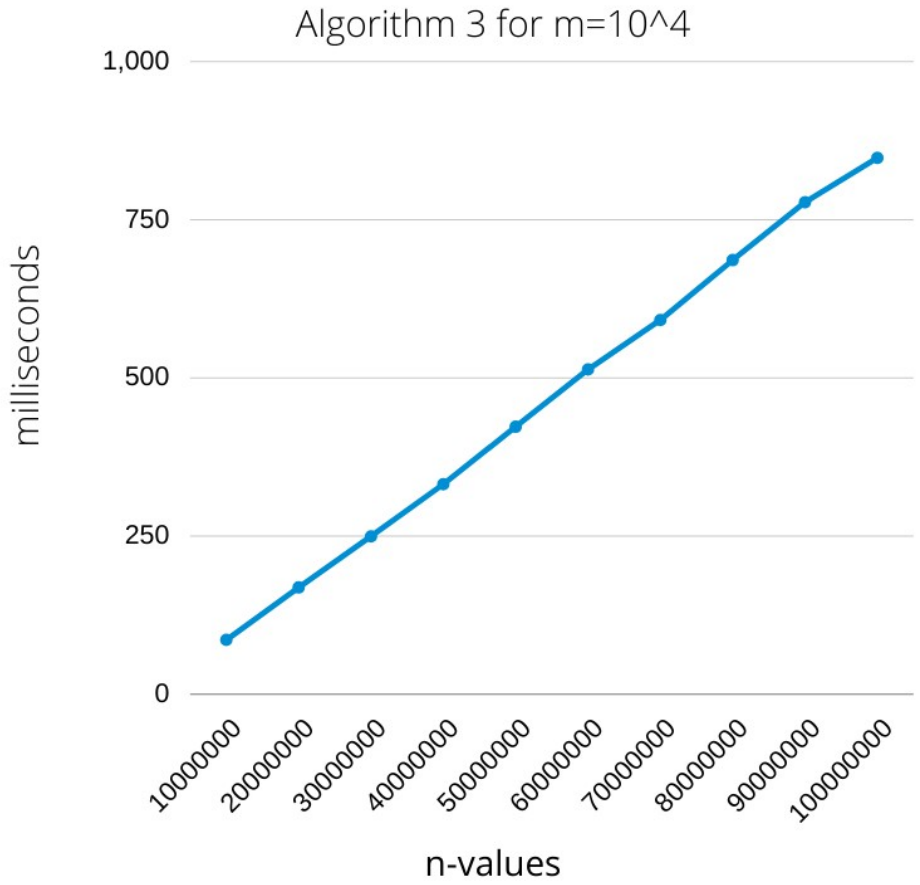
**Algorithm 2:**


Algorithm 2 for m=10^3
0.25
0.2
0.15
0.1
0.05
0
milliseconds
10000000
20000000
30000000
40000000
50000000
60000000
70000000
80000000
90000000
100000000
n-values


Algorithm 2 for m=10^4
2.5
2
1.5
1
0.5
0
milliseconds
10000000
20000000
30000000
40000000
50000000
60000000
70000000
80000000
90000000
100000000
n-values

**Algorithm 3:**


Algorithm 3 for m=10^3
1,000
750
500
250
0
milliseconds
10000000
20000000
30000000
40000000
50000000
60000000
70000000
80000000
90000000
100000000
n-values


Algorithm 3 for m=10^4
1,000
750
500
250
0
milliseconds
10000000
20000000
30000000
40000000
50000000
60000000
70000000
80000000
90000000
100000000
n-values