Project 1

Carson Carpenter, Nathan Landino

Mathematical Analysis

Complete Exercises 3-14 (a) and 3-14 (b) from *Algorithm Design in Three Acts*.

3-14: Each of the following is a problem, and pseudocode for an algorithm solving that problem. For each algorithm, derive a complexity function for its running time T(n), then prove the efficiency class O(f(n)) that T(n) belongs to.

(a)

mean problem
input: a non-empty list L of n numbers **output:** the mean (average) of L

```
def mean(L):
    total = 0
    for x in L:
        total += x
    return total / len(L)
```

Time Complexity = T(n) = 2n + 2

By properties of O, O(T(n)) = O(2n + 2) = O(n)

So 2n + 2 ∈ O(n)

(b)	*square matrix construction*
	input: a positive integer n and number x
	output: an $n \times n$ matrix with each element equal to x

```
def construct_square_matrix(n, x):
    rows = []
    for r in range(n):
        rows.append([])
        for c in range(n):
            rows[r].append(x)
    return rows
```

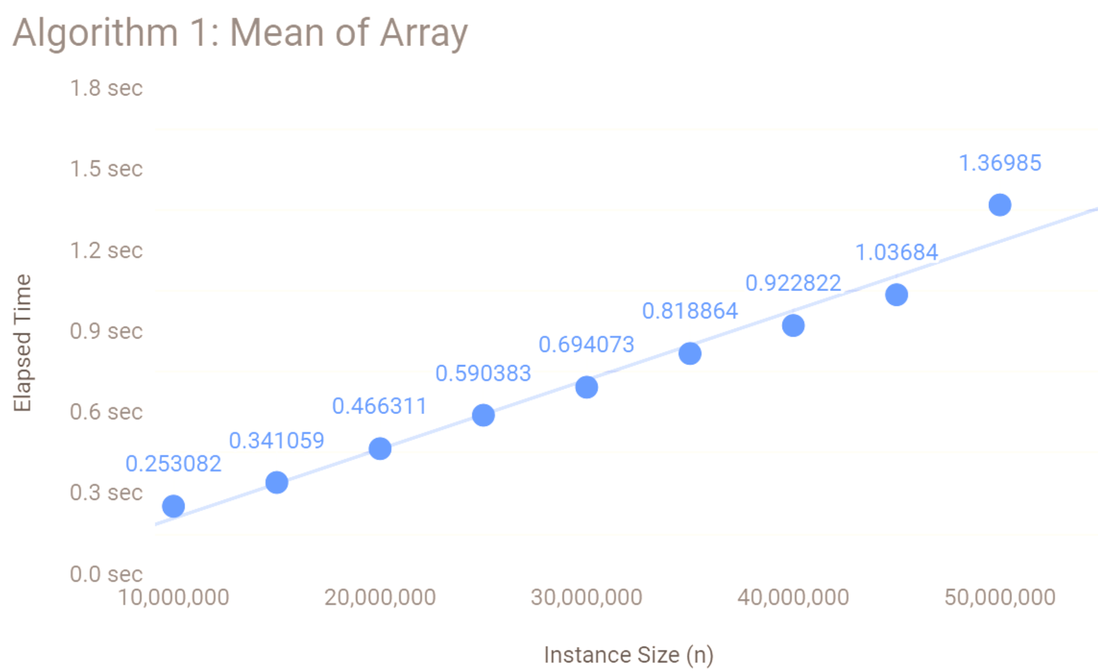
Time Complexity = T(n) = 2n^2 + 2n + 2

By properties of O, O(T(n)) = O(2n^2 + 2n + 2) = O(n^2)

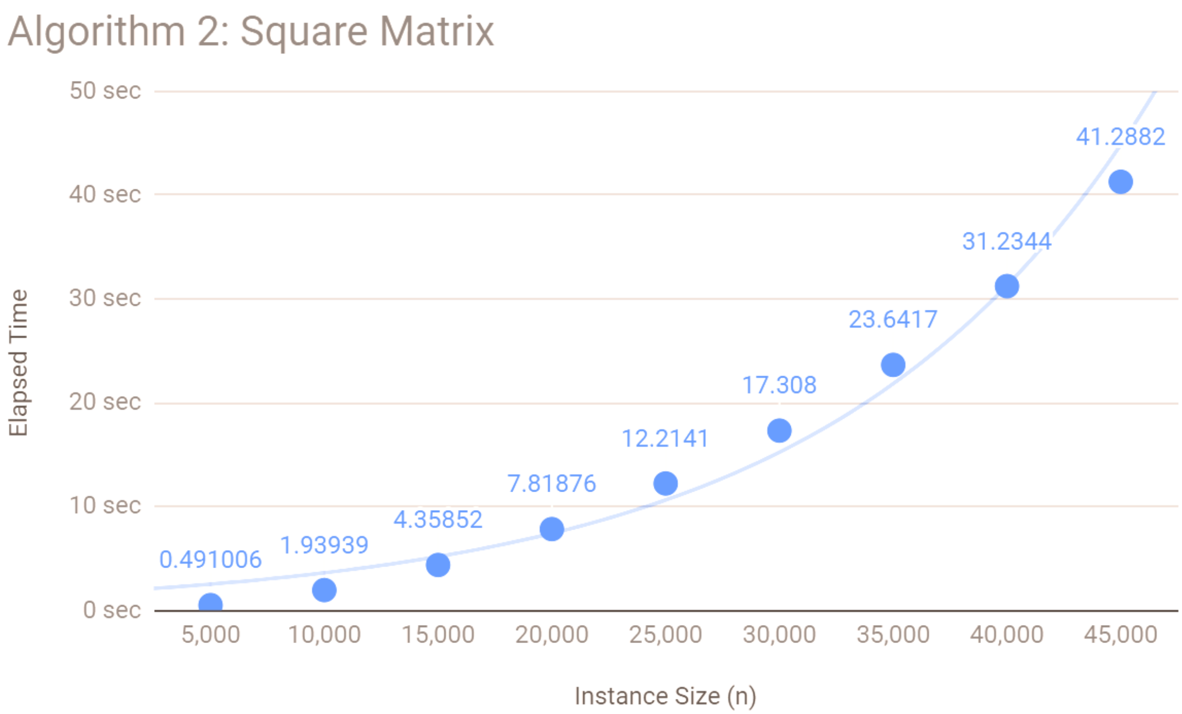
So 2n^2 + 2n + 2 ∈ O(n^2)

Empirical Analysis

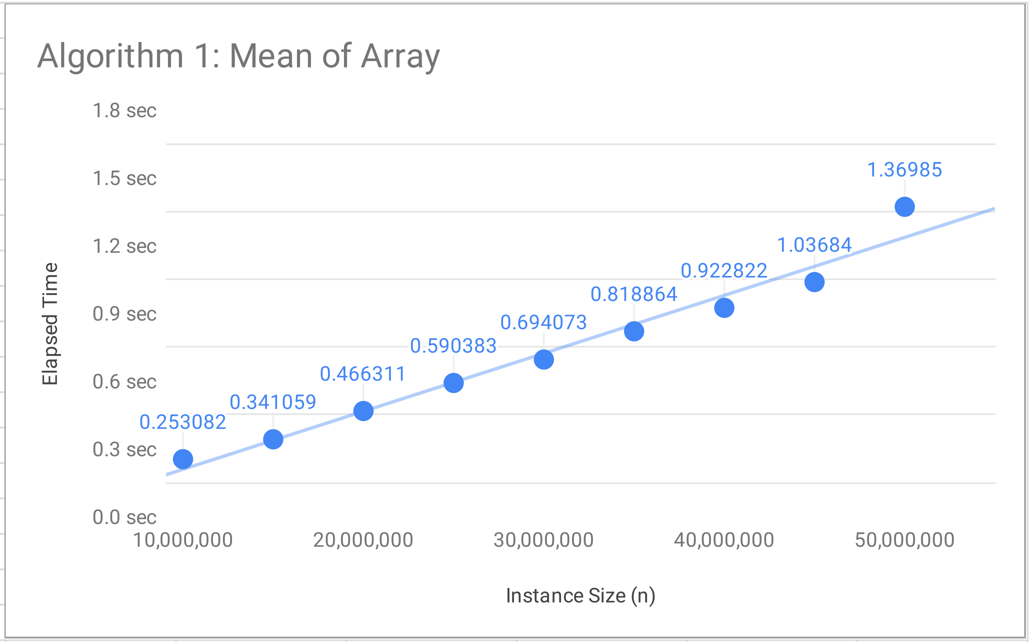
These results were found by timing how long it took to calculate the mean of an array of varying size. The resulting graph is linear, which matches the result from 3-14(a) which predicted an O(n) algorithm.



These experimental results were found by timing how long it took to initialize a square matrix of varying size. The resulting graph is exponential, which matches the result from 3-14(b) which predicted an O(n^2) algorithm.



Instance Size (n)	Elapsed Time (sec)
10,000,000	0.253082
15,000,000	0.341059
20,000,000	0.466311
25,000,000	0.590383
30,000,000	0.694073
35,000,000	0.818864
40,000,000	0.922822
45,000,000	1.03684
50,000,000	1.36985



Instance Size (n)	Elapsed Time (sec)
5,000	0.491006
10,000	1.93939
15,000	4.35852
20,000	7.81876
25,000	12.2141
30,000	17.308
35,000	23.6417
40,000	31.2344
45,000	41.2882