x = 1;	1
for i = 1:n	from i = 1 to n+1 ∑1
for j = 1:n	from I = 1 to n ∑ (from j = 1 to n+1 ∑(1))
x = x + 1;	from I = 1 to n ∑ (from j = 1 to n ∑(1))

the for loop check is checked 1 more time than the contents of the for-loop, hence the n+1.
Total runtime is:

1.

1 + n+1 + n(n+1) + n*n

Polynomial of function:
$2 * 10^{-9}n^2 + 5 * 10^{-7}n + 0.0014$

3.

Upper bound: n^2
Lower bound: n

Big-O(n^2)
Big-Omega(n) less precisely: Big-Omega(1)
Big-theta(n^2)

4.

If I was reading the textbook correctly, n_0 is in big theta at n_0, f(n) lies between c1g(n) and c2g(n), meaning n_0 lies within the upper and lower bounds of the function. I picked n = 12500 as this lies right on the curve that fits to my data, hence it would be within the upper and lower bounds of big theta.

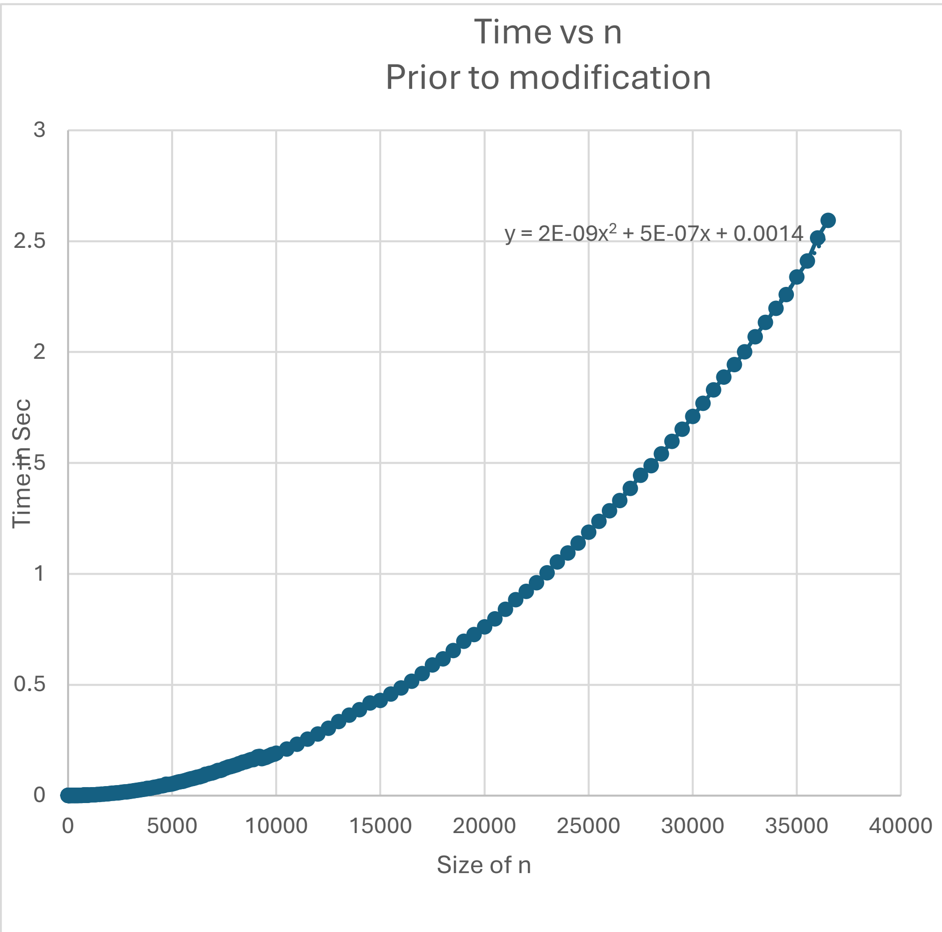
Modified function:

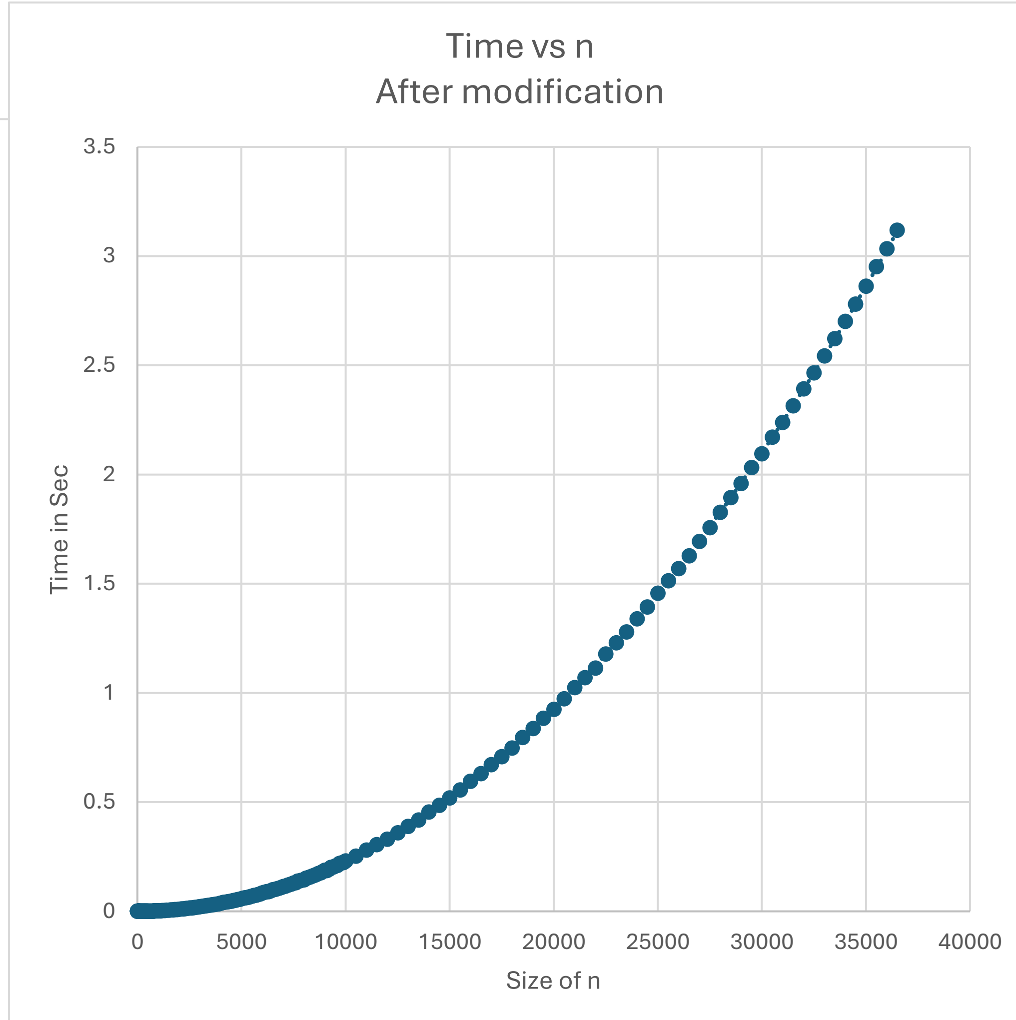
If you modified the function, it would take the algorithm longer to run:

5.

If you modified the function, it would increase the total run time adding an additional n*n so:

1 + n+1 + n(n+1) + n*n + n*n


Time vs n
Prior to modification
y = 2E-09x2 + 5E-07x + 0.0014
Time in Sec
Size of n
0
0.5
1
1.5
2
2.5
3
0
5000
10000
15000
20000
25000
30000
35000
40000


Time vs n
After modification
Time in Sec
Size of n
0
0.5
1
1.5
2
2.5
3
3.5
0
5000
10000
15000
20000
25000
30000
35000
40000