Compare Two Algorithm

January 27, 2020

Yida Wang

Code Option A

```
value = 0.5;
for (row = 0; row < N; row++)
    for (col = 0; col < N; col++) {
        A[row][col] = value;
        value = value + 1.0; }
for (row = 0; row < N; row++)
    A[row][row] = 0.0;
```

Code Option B

```
for (row = 0; row < N; row++)
    for (col = 0; col < N; col++)
        if (row == col)
            A[row][col] = 0.0;
        else
            A[row][col] = (float) ((N * row) + col) + 0.5;
```

For code option A, number of instructions executed = 2 * (N-1) * (N-1) + (N-1) = $k \cdot N^2 + N$

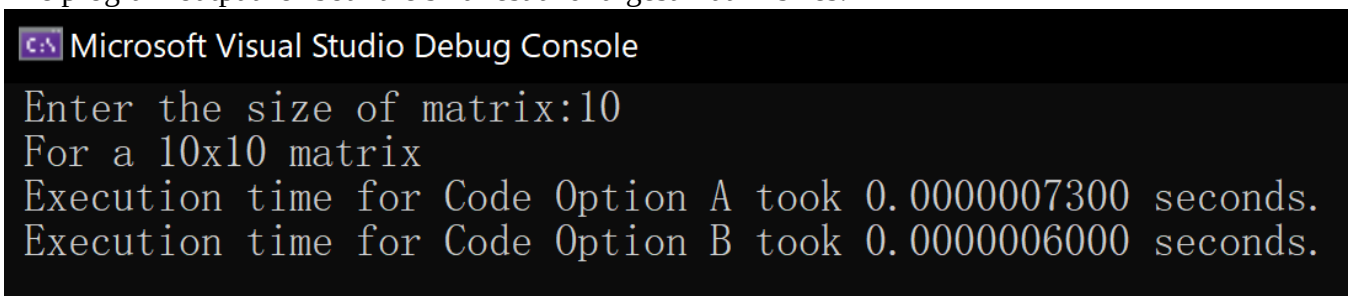
The first set of nested loops is $O(N^2)$ and the second loop is $O(N)$. This is $O(\max(N^2, N))$ which is $O(N^2)$.

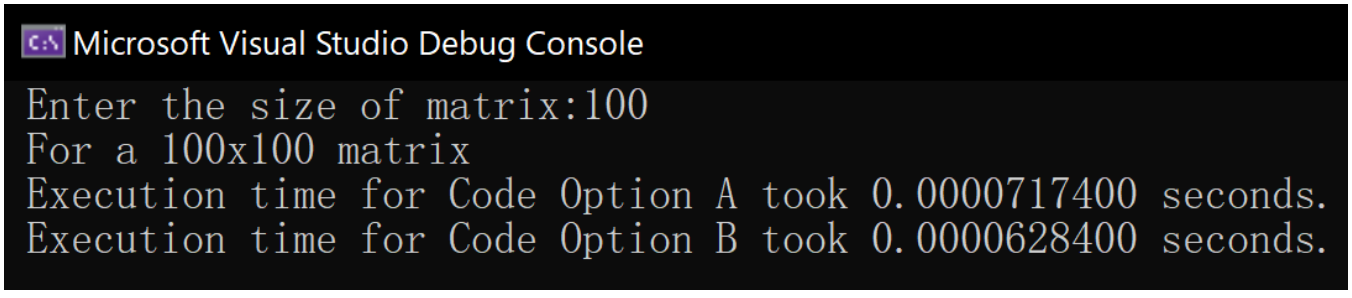
For code option B, number of instructions executed = (N-1) * (N-1) = N^2

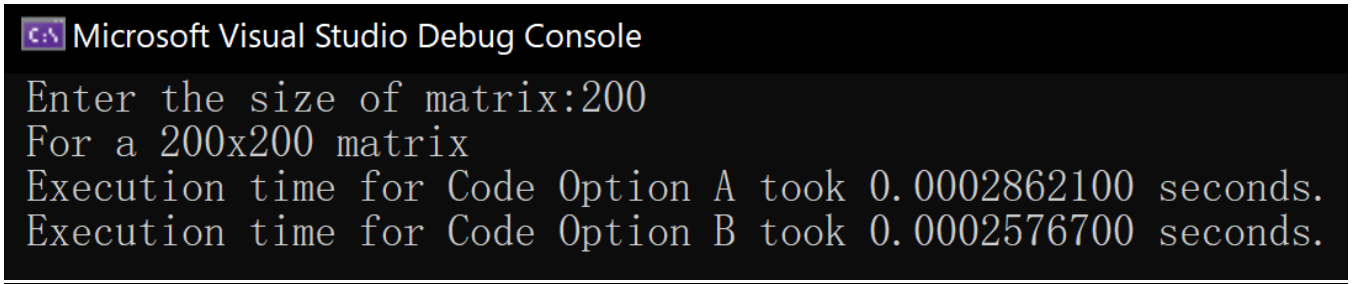
For a if/else statement, either if statement will execute or else statement will execute. Since the if statement is not a order of N condition, and the else statement is O(1), it is a O(1).

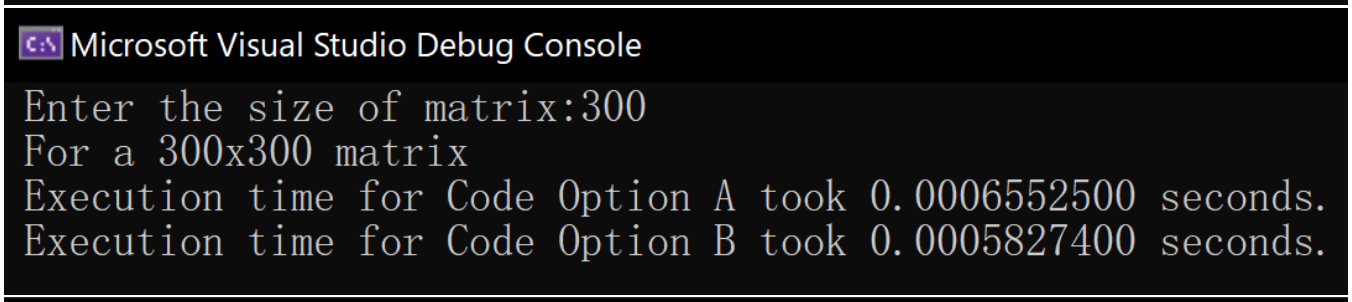
The outer loop execute N-1 times, and the inner loop also execute N-1 times, so the total is (N-1)*(N-1) times. Thus, the complexity is $O(N^2)$.

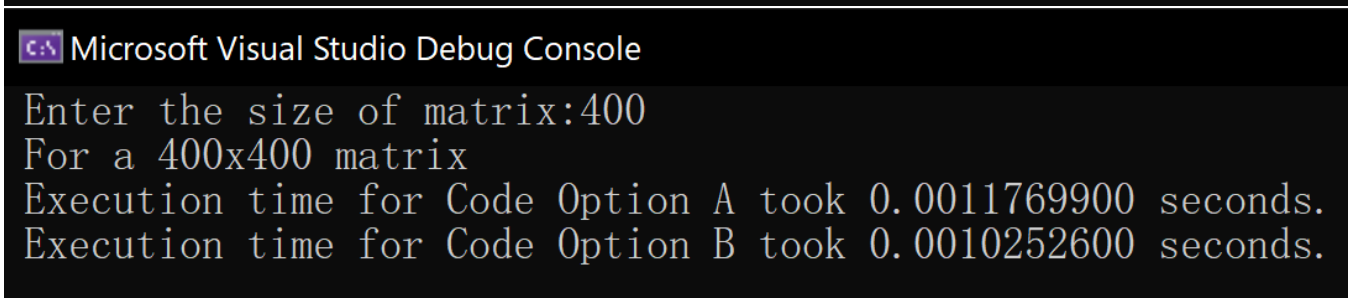
The program output for both the smallest and largest matrix sizes.

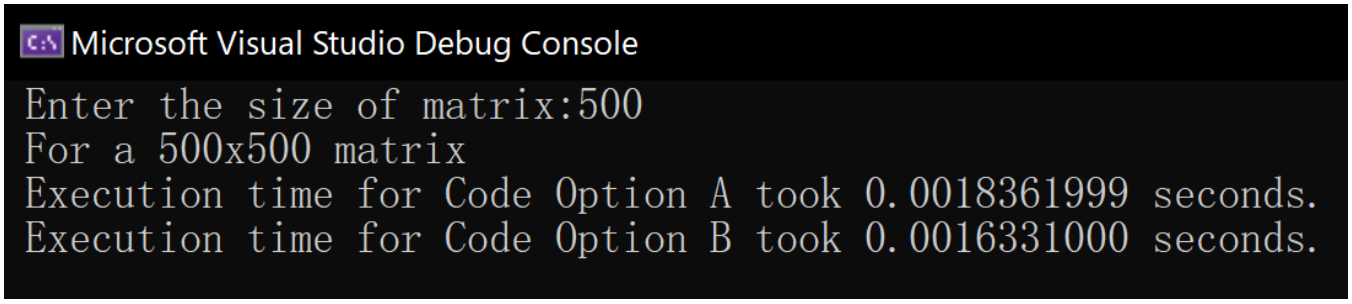


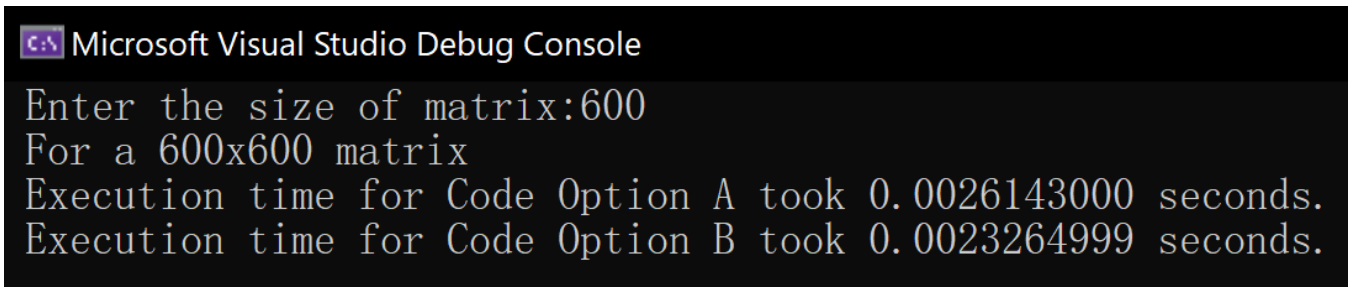


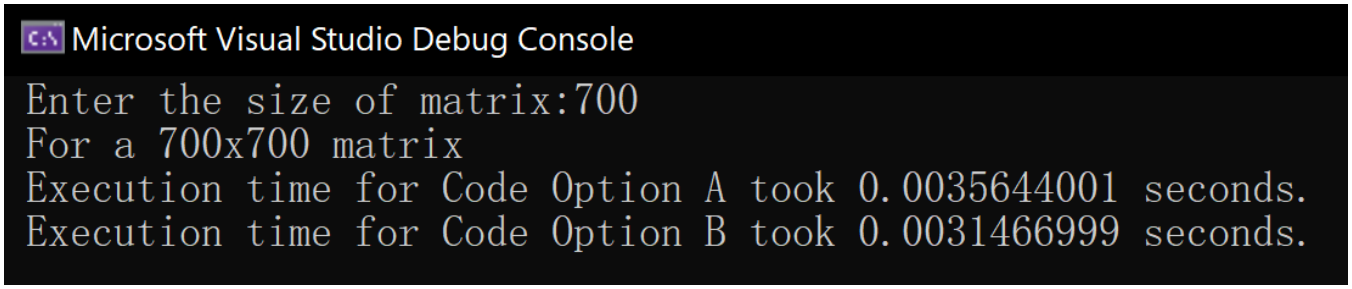

Microsoft Visual Studio Debug Console
Enter the size of matrix:200
For a 200x200 matrix
Execution time for Code Option A took 0.0002862100 seconds.
Execution time for Code Option B took 0.0002576700 seconds.

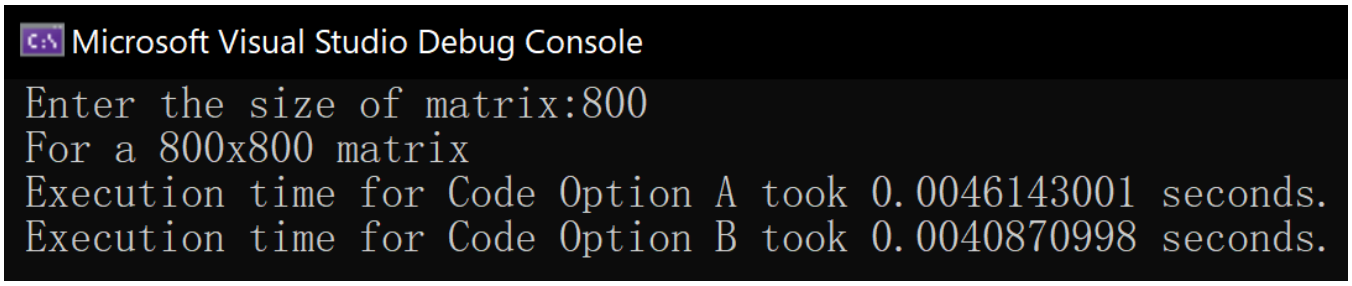

Microsoft Visual Studio Debug Console
Enter the size of matrix:300
For a 300x300 matrix
Execution time for Code Option A took 0.0006552500 seconds.
Execution time for Code Option B took 0.0005827400 seconds.

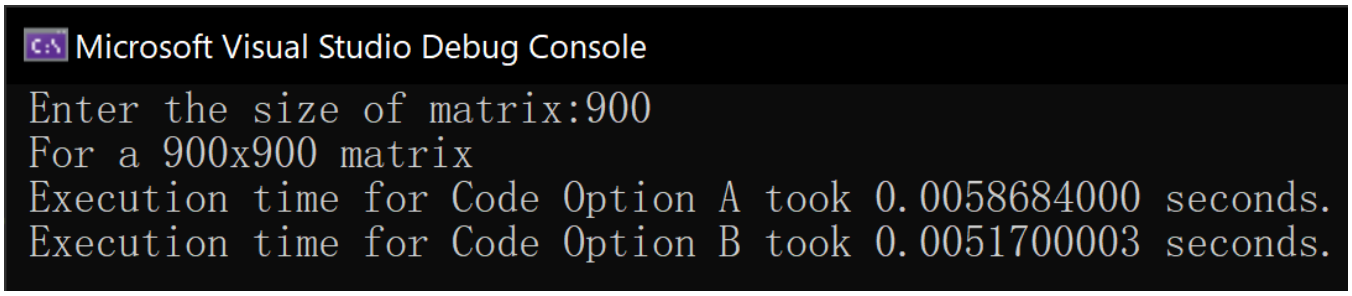

Microsoft Visual Studio Debug Console
Enter the size of matrix:400
For a 400x400 matrix
Execution time for Code Option A took 0.0011769900 seconds.
Execution time for Code Option B took 0.0010252600 seconds.

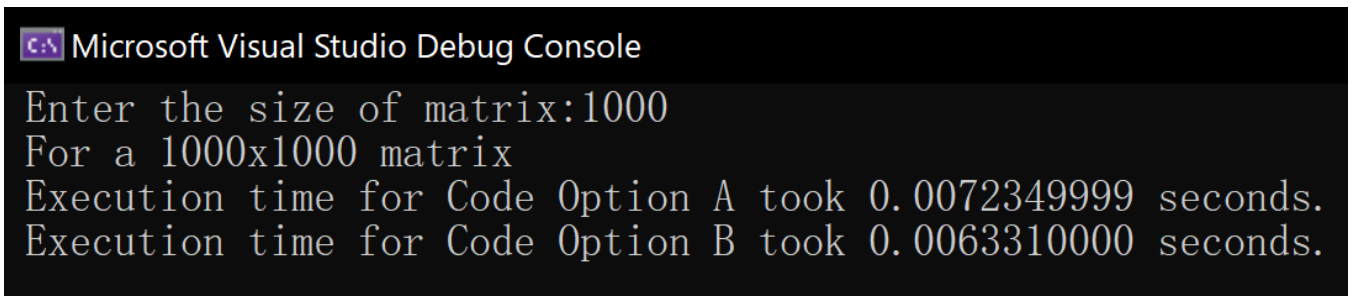

Microsoft Visual Studio Debug Console
Enter the size of matrix:500
For a 500x500 matrix
Execution time for Code Option A took 0.0018361999 seconds.
Execution time for Code Option B took 0.0016331000 seconds.

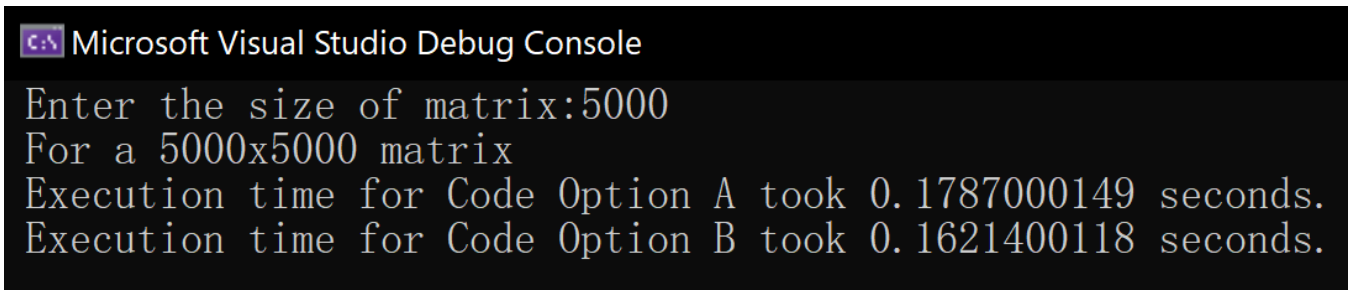

Microsoft Visual Studio Debug Console
Enter the size of matrix:600
For a 600x600 matrix
Execution time for Code Option A took 0.0026143000 seconds.
Execution time for Code Option B took 0.0023264999 seconds.


Microsoft Visual Studio Debug Console
Enter the size of matrix:700
For a 700x700 matrix
Execution time for Code Option A took 0.0035644001 seconds.
Execution time for Code Option B took 0.0031466999 seconds.









A table summarizing the execution times for both algorithms for ALL matrix sizes analyzed.

Size of matrix	Execution time for Code Option A	Execution time for Code Option B
10x10	0.00000073	0.00000060
100x100	0.00007174	0.00006284
200x200	0.00028621	0.00025767
300x300	0.00065525	0.00058274
400x400	0.00117699	0.00102526
500x500	0.00183620	0.00163310
600x600	0.00261430	0.00232650
700x700	0.00356440	0.00314670
800x800	0.00461430	0.00408710
900x900	0.00586840	0.00517000
1000x1000	0.00723500	0.00633100
5000x5000	0.17870001	0.16214001

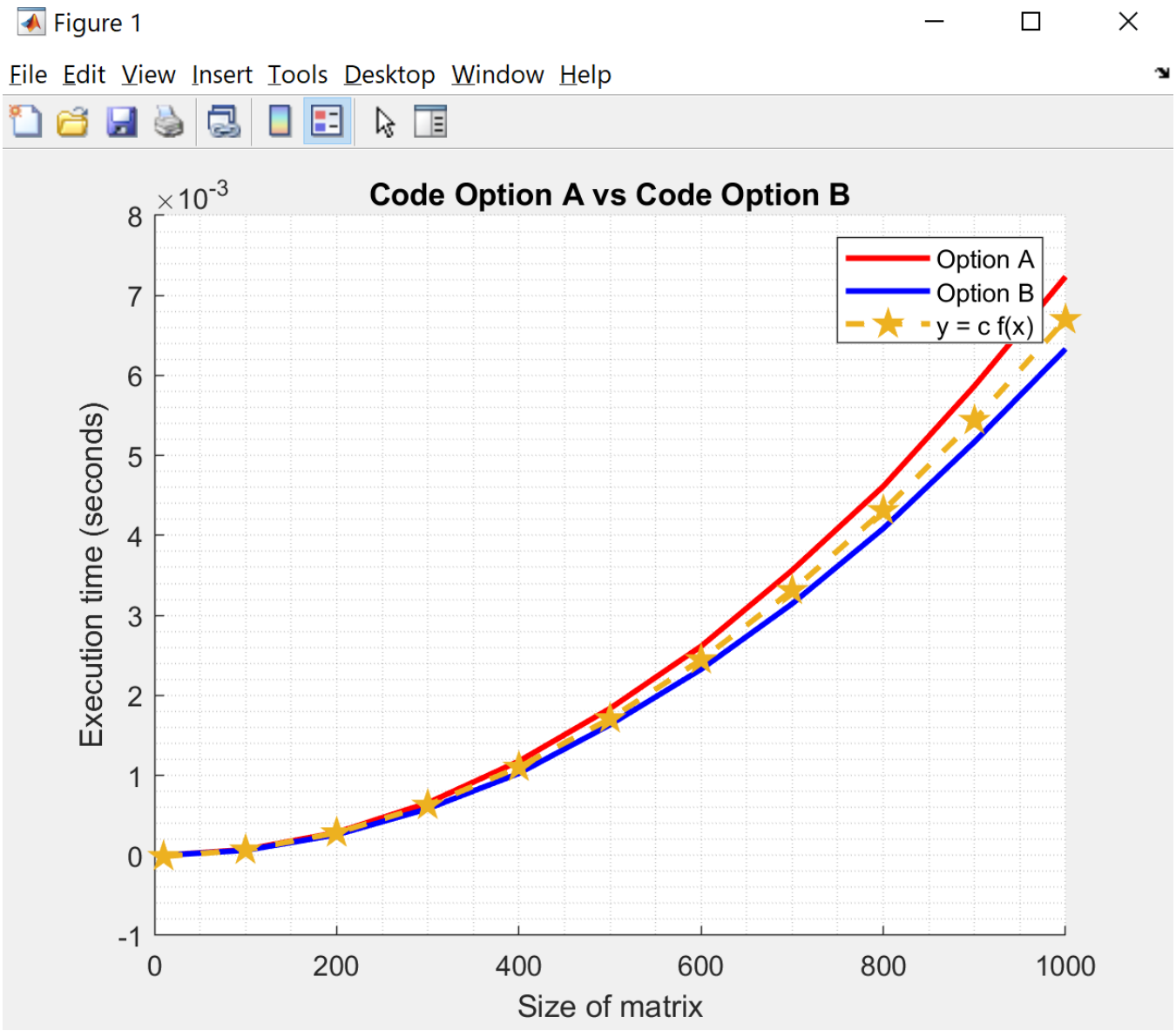


Figure 1. MATLAB graph comparing execution time versus matrix size for both algorithms.

As we can see from program output that Code Option B's algorithm is faster than the Code Option A's algorithm because Code Option B did not use an extra for loop for zeroing the A[row][row].

Instead of an extra for loop, Code Option B uses an if else statement which is an O(1) complexity compare to O(N) in the Code Option A.

Thus, the execution time is faster in Code Option B.