CS 3310 Project 1 Report

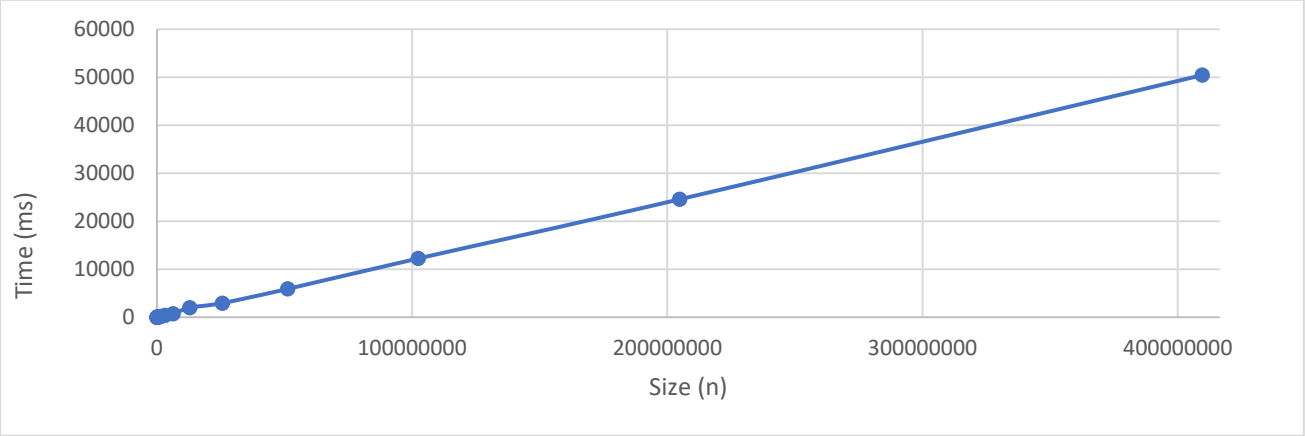
Task1: Sorting Algorithms

Merge Sort

The time complexity for Merge Sort algorithm is T(n) =2 T(n/2) + n -1, and by master theorem we will get a = 2, b = 2, d = 1. The big O notation will be O(n lg(n)).

Below is the test result I got from my merge sort. I convert all the results from nanosecond to the millisecond.

N	10000	20000	50000	100000	200000	400000	800000	1600000
ms	2.126	2.300	7.6331	15.655	35.890	65.043	84.5855	174.288
N	3200000	6400000	12800000	25600000	51200000	102400000	204800000	409600000
ms	369.848	694.629	1983.719	2942.866	5891.899	12245.599	24559.243	50441.346

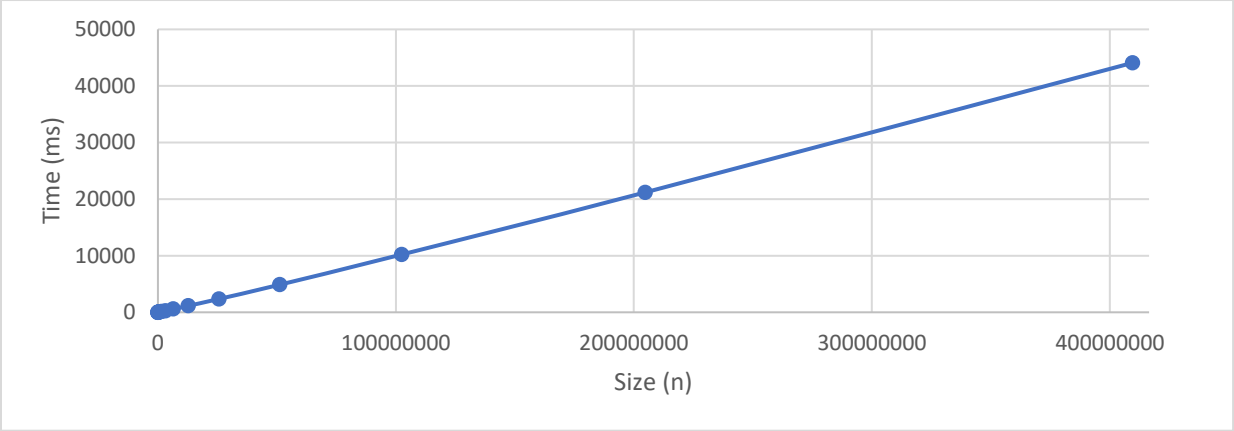


Quick Sort

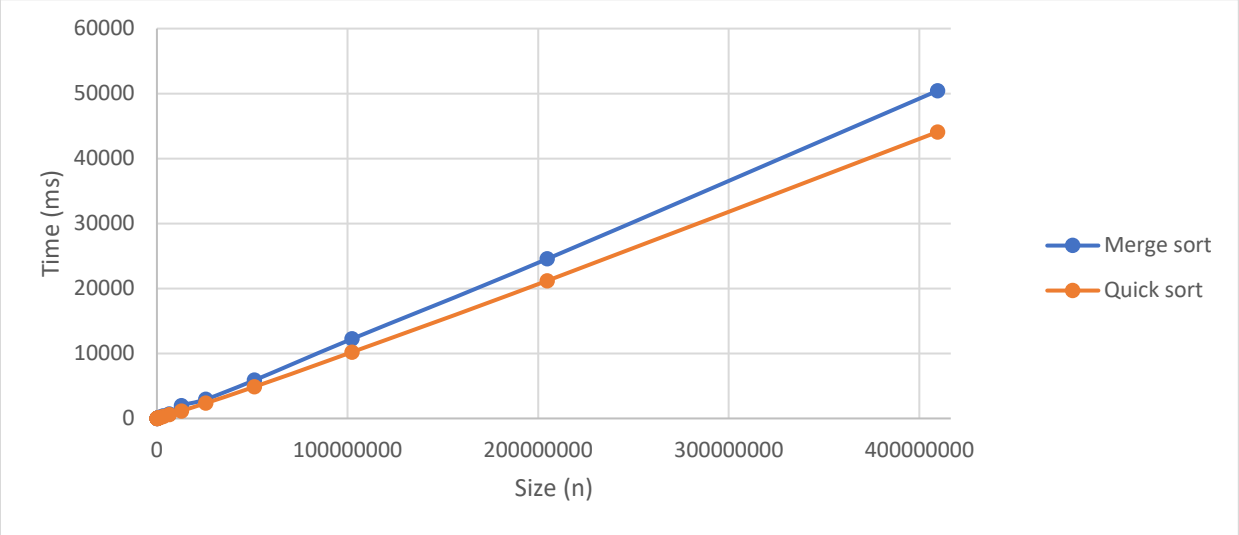
The time complexity for the Quick Sort algorithm is T(n) = T(n -1) + n -1, and by master theorem we will get a = 1, b = 1, d = 1. The big O notation will also be O(n lg(n)).

Below is the test result I got from my quick sort. I convert all the results from nanosecond to the millisecond.

N	10000	20000	50000	100000	200000	400000	800000	1600000
ms	1.525	1.330	3.565	5.977	7.413	34.0662	67.877	130.053
N	3200000	6400000	12800000	25600000	51200000	102400000	204800000	409600000
ms	277.099	572.703	1151.332	2353.422	4879.304	10213.768	21197.139	44092.817



Now let's compare both results. The blue line is merge sort, and the orange line is quick sort.



By the graph, we can see quicksort is more efficient than merge sort when the size of n increases, but in theory, merge sort is suppose to be more efficient and work faster than quicksort. I think the reason that caused the result different than theory might be because I didn't use in-place merge sort.

Task:2 Tower of Hanoi

The recurrence relation for my code will be

T(1) = 1

T(n) = 2T(n-1) +1

When there is only one disk it only requires one move. Two disks will require three moving to solve it, and three disks will need 7 moves to solve. If we plug in the n disks number into the recurrence relation we will get.

T(1) = 1

T(2) = 2T(1)+1 = 3

T(3) = 2T(2)+1 = 7

Which proof the recurrence relation match the result.

The time complexity of my algorithm will be

T(n) $= 2^n - 1$

When I plug-in the n disks number.

T(1) = 1

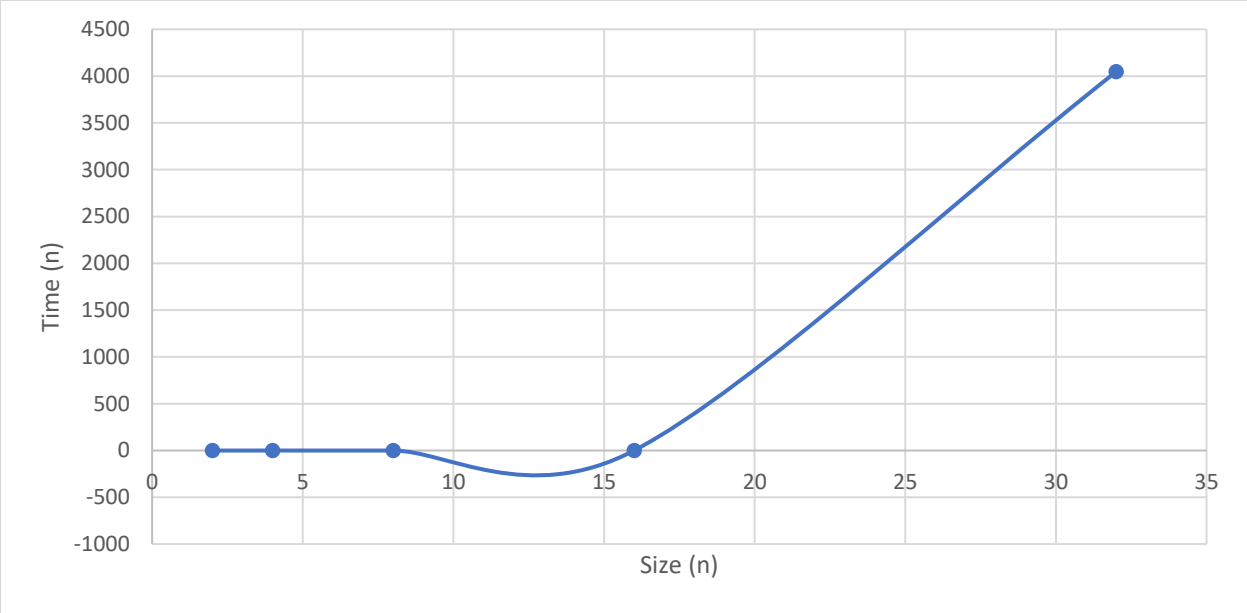
T(2) = 3

T(3) = 7

We will also get the same result as my recurrence relation, so which proof the time complexity is correct too.

Below is the test result I got from my test, and I also convert the result from nanosecond to the millisecond.

n	2	4	8	16	32
ms	0.0006	0.0007	0.0012	0.0456	4046.736



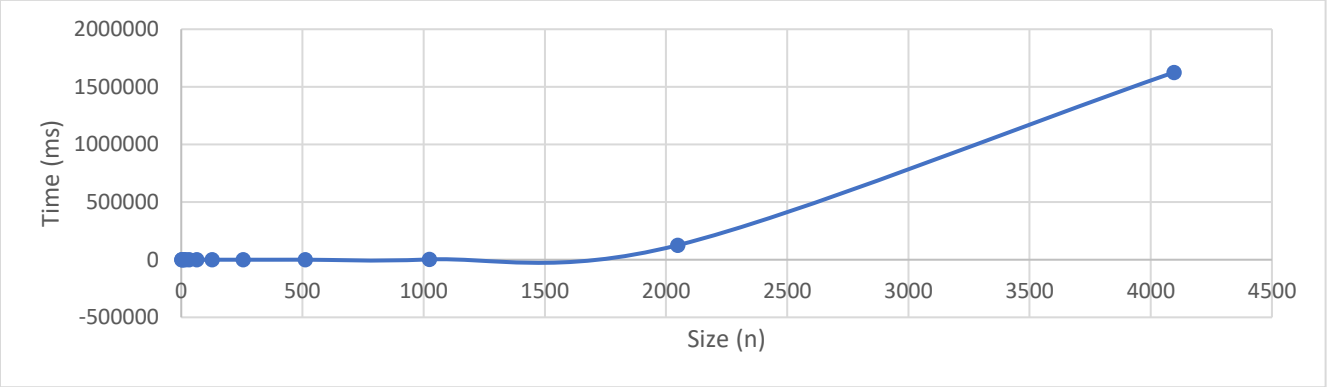
My computer failed to execute the test after n is greater than 32. From the graph, we can see the time required to run this method increase rapidly when n became larger. Simply we plug in the 32 into the time complexity formula we will get 4294967295, and 64 will be 18446744073709551615 which is 4294967297 time of the time that will be required to run this method. This algorithm will simply cause the my computer's memory overflow.

Task:3 Matrix Multiplication

Classical Matrix Multiplication

The time complexity of my classical multiplication is T(n) = $n^3 - n^2$ since I use the algorithm from the in-class exercise, and bit O notation is O(n^3).

N	2	4	8	16	32	64	128	256
ms	0.0037	0.0045	0.8	0.0188	1.0183	2.9216	12.5077	31.9441
N	512	1024	2048	4096				
ms	220.931	2630.576	125951.582	1625982.840				



Strassen's Matrix Multiplication

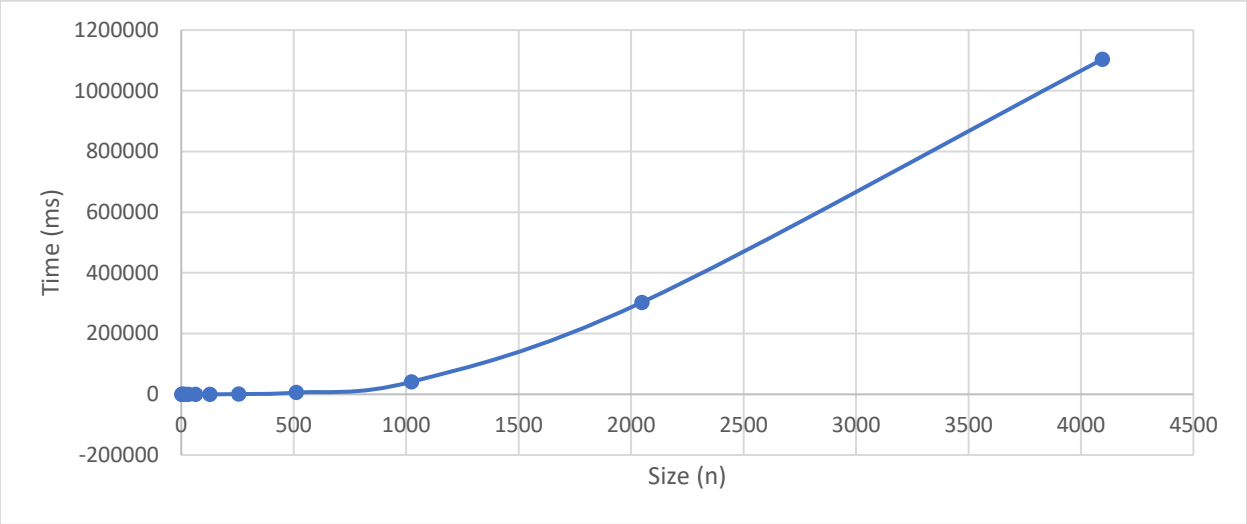
The time complexity of my Strassen's multiplication is

T(1) = 0

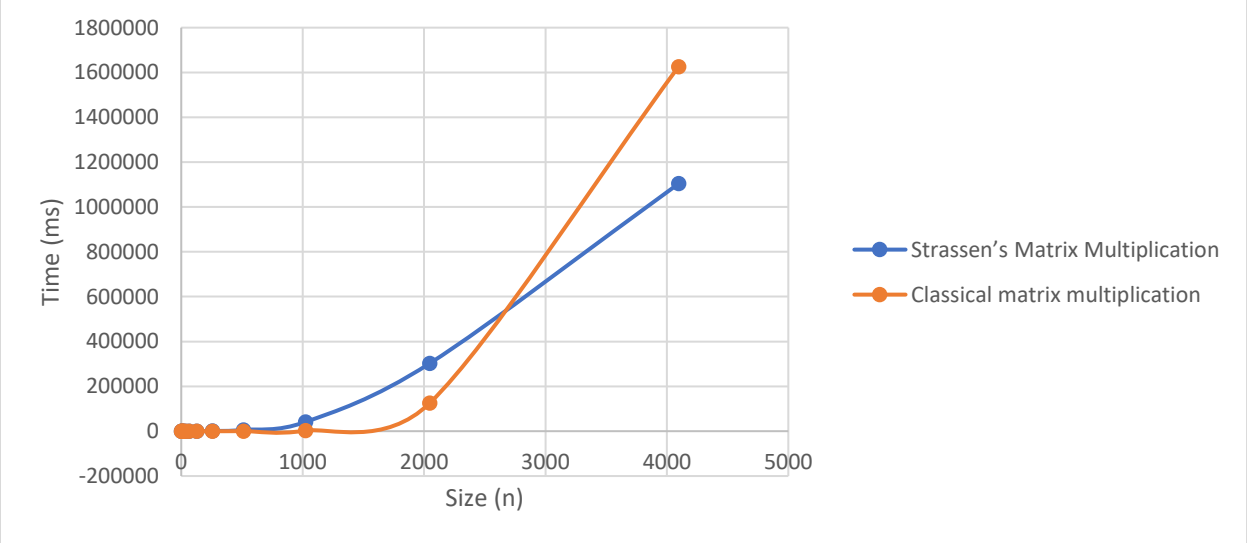
$T(n) = 7T\left(\frac{n}{2}\right) + 18(\frac{n}{2})^2$

By mater theorem, we ill get the big O notation O(n^{lg7})

N	2	4	8	16	32	64	128	256
ms	0.0037	0.0045	0.8	0.0188	1.0183	2.9216	12.5077	31.9441
N	512	1024	2048	4096				
ms	220.931	2630.576	125951.582	1625982.840				



Now let's compare both results. The blue line is Strassen's multiplication, and the orange line is classical multiplication.



From the above graph, we can see classical multiplication is more efficient when the n number is small. However, when n number passed 2700 the Strassen's efficiency is lower than the classical one. The graph actually proof that Strassen Big $O(n^{lg7})$ is less than the classical $O(n^3)$.