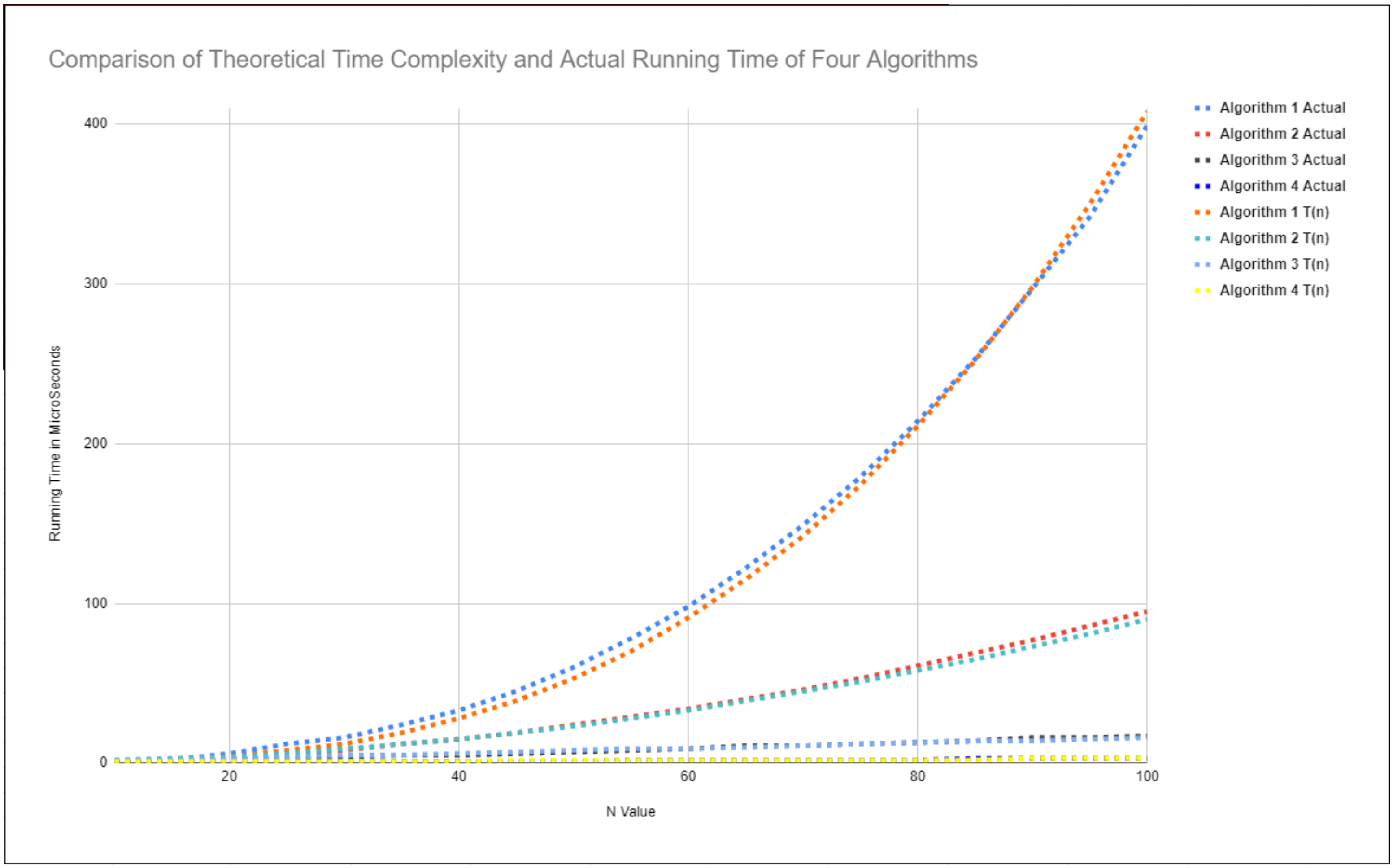


Explanation: This graph compares the calculated T(n) to the running time shown by the System Clock. The actual time and the calculated time did end up being fairly similar, especially for the faster algorithms. The slower algorithms had slightly more variance but still quite similar.

Algorithm-1

Step	Cost of each execution	Total # of times executed
1	1	1
2	1	n
3	1	n(n+1)/2
4	1	n(n+1)/2
5	1	n(n+1)(2n+1)/6
6	6	n(n+1)(2n+1)/6
7	8	n(n+1)/2
8	2	1

Multiply col.1 with col.2, add across rows and simplify

$T_1(n) = ((4n^3+15n^2+17n)/6 + 2$

Algorithm-2

Step	Cost of each execution	Total # of times executed
1	1	1
2	1	n+1
3	1	n
4	1	(n(n+1)/2) + n
5	6	n(n+1)/2
6	8	n
7	2	1

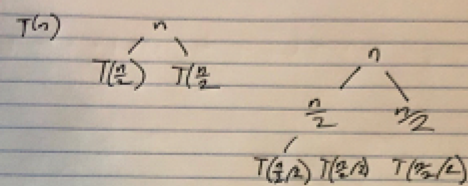
Multiply col.1 with col.2, add across rows and simplify

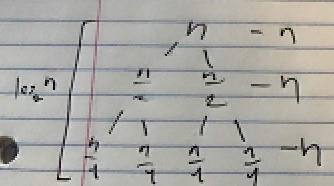
$T_2(n) = ((7n^2 + 15n)/2) + 4$

Algorithm-3

Step	Cost of each execution	Total # of times executed in any single recursive call
1	5	1
2	13	1
Steps executed when the input is a base case: 18		
First recurrence relation: T(n=1 or n=0) = 18		
3	5	1
4	2	1
5	1	n/2 + 1
6	6	n/2
7	8	n/2
8	2	1
9	1	n/2 + 1
10	5	n/2
11	8	n/2
12	4	1
13	1	(cost excluding the recursive call)
14	1	(cost excluding the recursive call)
15	6	1
Steps executed when input is NOT a base case:		
Second recurrence relation: T(n>1) = aT(n/b) + f(n) = 2T(n/2) + 18n		
Simplified second recurrence relation (ignore the constant term): T(n>1) = 2T(n/2) + n		

Solve the two recurrence relations using any method (recommended method is the Recursion Tree). Show your work below:





$$T(n) = \sum_{i=0}^{\log_2 n - 1} \left(\tfrac{1}{2}\right)^i n$$

$$\frac{1}{1-\frac{1}{2}} n$$

$2n \log n + f(n)$

$$\boxed{2n\log(n) + 18n}$$

Algorithm-4

Step	Cost of each execution	Total # of times executed
1	1	1
2	1	1
3	1	n
4	11	n
5	8	n
6	2	1

Multiply col.1 with col.2, add across rows and simplify

$T_4(n) = 20n + 4$