Algorithm-I

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

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

$T1(n) = 3 + n + (n^2+n)/2 + (n^2 - n)/2 +$
$1/6(n^3+3n^2+2n) + (n^3-n) + 4(n^2 - n)$

$= n^3+9/2(n^2-n)+1/2(n^2+n)+1/6(n^3+3n^2+2n)+3$

$= 7n^3/6 + 11n^2/2 – 11n/3 + 3$

$=$ **$O(n^3)$**

Algorithm-2

Step	Cost of each execution	Total # of times executed
1	1	1
2	1	n+1
3	1	n
4	1	$\sum_{i=1}^{n}(n-i+1)$
5	6	$\sum_{i=1}^{n}(n-i)$
6	5	$\sum_{i=1}^{n}(n-i)$
7	1	1

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

$T2(n) = 1 + 1 + n + (n+1) + \sum_{i=1}^{n}(n-i+1) + 6\sum_{i=1}^{n}(n-i) + 5\sum_{i=1}^{n}(n-i)$

$= 3+2n+ \frac{n^2+n}{2} + 11\frac{n^2-n}{2}$

$= n(6n – 3) + 3$

$= 6n^2-3n+3$

$=$ **$O(n^2)$**

Algorithm-3

Step	Cost of each execution	Total# of times executed in any single recursive call
1	4	1
2	10	1
Steps executed when the input is a base case: **2 steps**		
First recurrence relation: T(n=l or n=0) = **O(1)** = 23		
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	6	n/2
11	8	n/2
12	4	1

Algorithm-I

13	T(n/2)	1
14	T(n/2)	1
15	8	1
Steps executed when input is NOT a base case: **15**		
Second recurrence relation: T(n>1) = **2T(n/2) + 15n + 23**		
Simplified second recurrence relation (ignore the constant term): **T(n>1) = 2T(n/2) + O(n)**		

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

William Abbot

$$T(n)\begin{cases} 23 & n=0,1 \\ 2T(n/2)+15n \end{cases}$$

Build Solution

$$T(n) = 2T(n/2) + 15n$$
$$= 2\left[2T(n/2^2) + 15(n/2)\right] + 15n$$
$$= 2^2T(n/2^2) + 15n + 15n$$
$$= 2^2\left[2T(n/2^3) + 15(n/2^2)\right] + 15n + 15n$$
$$= \underbrace{2^3T(n/2^3)}\ + \underbrace{15n + 15n + 15n}$$
$$= 2^kT\left(\tfrac{n}{2^k}\right) + k15n$$
$$\vdots$$
$$= 2^{\lg n}T(1) + 15n\lg n$$

↳ to get $T(\frac{n}{2^k})$ to be 1, you make $(k = \lg_2 n)$

$$= 2^{\lg n}\cdot 23 + 15n\lg n$$
$$= n\cdot 23 + 15n\lg n$$
$$= \boxed{23n\ \ 15n\lg n}$$
$$\therefore O(n\lg n)$$

Expand search

$$T(n/2) = 2T(n/2^2) + 15(n/2)$$
$$T(n/2^2) = 2T\left(\tfrac{n}{2^3}\right) + 15(n/2^2)$$

$T_3(n) = 23n + (15n*\log_2 n)$

$= \mathbf{O(n \log_2 n)}$

Algorithm-I

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

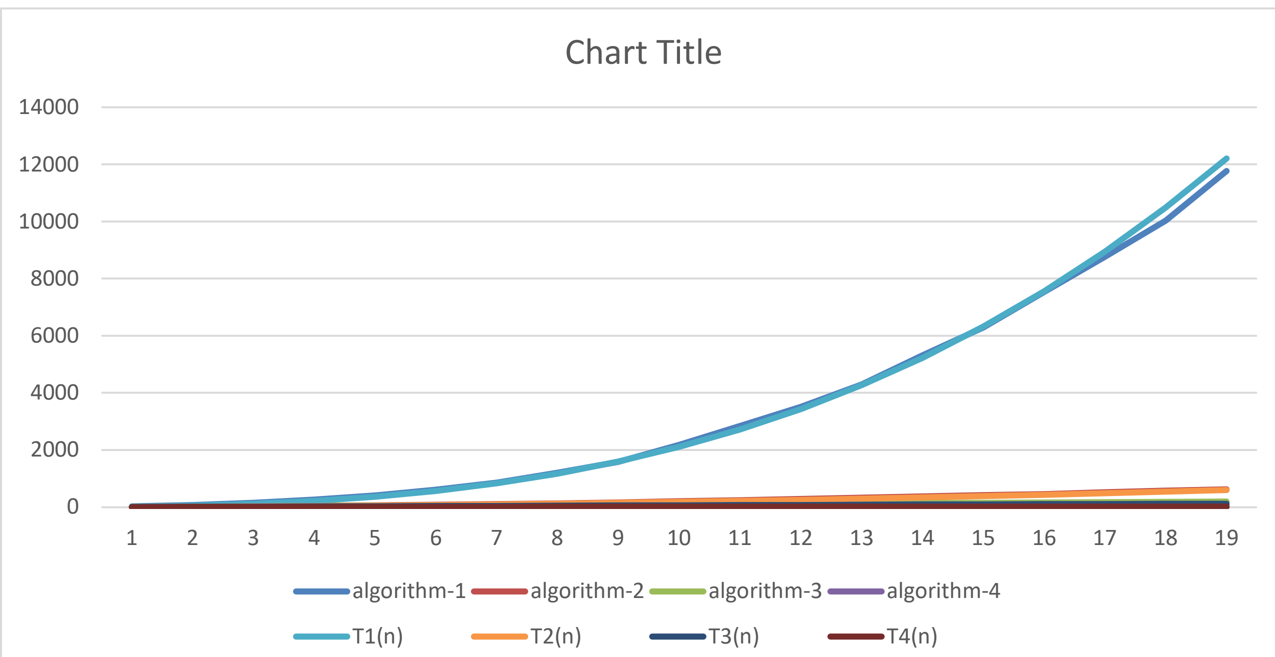
Multiply col.1 with col.2, add across rows and simplify

T4(n) = 1+1+n+1+n+n+1

= 20n + 4

= **O(n)**

Conclusions:



As the graph shows, my actual time was pretty close to my theoretical time for each algorithm except, partially, Algorithm 3 which was slightly off from the theoretical but still close (and obviously on the same order of magnitude).

Algorithm 1 and T1(n) are so on the order of n^3 because of the 3 nested loops in the algorithm, and the same follows for algorithm 2 except for n^2. Algorithm 3 is a recursive divide and conquer algorithm that splits the input size in half (roughly) each recursion, resulting in the O(nlgn). Algorithm 4 is the quickest and only has one loop, making its' time complexity O(n).

Algorithm-4