COMP3270 Programming Homework
Clayton Haley

Algorithm-1

Step	Cost of each execution	Total # of times executed
1	1	1
2	1	n + 1
3	1	$\sum_{i=1}^{n}(n-i+1)$
4	1	$\sum_{i=1}^{n}(n-i)$
5	1	$\sum_{i=1}^{n}\sum_{j=1}^{n}(j-i+1)$
6	6	$\sum_{i=1}^{n}\sum_{j=1}^{n}(j-i)$
7	4	$\sum_{i=1}^{n}\sum_{j=1}^{n}(j-i)$
8	1	1

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

$T_1(n) = 1 + (n+1) + \sum_{i=1}^{n}(n-i+1) + \sum_{i=1}^{n}(n-i) + \sum_{i=1}^{n}\sum_{j=1}^{n}(j-i+1) + 6\sum_{i=1}^{n}\sum_{j=1}^{n}(j-i) + 4\sum_{i=1}^{n}\sum_{j=1}^{n}(j-i)+1$

$= 3 + n + 2\sum_{i=1}^{n}(n-i) - 2n + (11\sum_{i=1}^{n}(\sum_{j=1}^{n}(j-i)) - 11n^2) + \sum_{i=1}^{n}1 + \sum_{i=1}^{n}\sum_{j=1}^{n}1$

$= 3 + n + 2(\frac{n(n+1)}{2}) - 2n + 11(\frac{n(n+1)(n-1)}{2}) - 11n^2 + n + n^2$

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

$= 3 - 9n^2 + \frac{11n^3}{2} + \frac{13n}{2}$

$= \frac{11n^3}{2} - 9n^2 + \frac{13n}{2} + 3$

$= O(n^3)$

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	4	$\sum_{i=1}^{n}(n-i)$
7	1	1

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

$T_2(n) = 1 + (n+1) + n + \sum_{i=1}^{n}(n-i+1) + 6\sum_{i=1}^{n}(n-i) + 4\sum_{i=1}^{n}(n-i) + 1$

$= 2n + 3 + \sum_{i=1}^{n}(n-i+1) + 10\sum_{i=1}^{n}(n-i)$

$= 2n + 3 + 11\sum_{i=1}^{n}(n-i) - 11n + \sum_{i=1}^{n}1$

$= -8n + 11(\frac{n(n+1)}{2}) + 3$

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

$= O(n^2)$

Algorithm-3

Step	Cost of each execution	Total # of times executed in any single recursive call
1	4	1
2	8	1

Steps executed when the input is a base case: 1, 2
First recurrence relation: T(n=1 or n=0) = O(1)

3	5	1
4	2	1
5	1	n/2 + 1
6	6	n/2
7	4	n/2
8	2	1
9	1	n/2 + 1
10	6	n/2
11	4	n/2
12	4	1
13	T(n/2)	1
14	T(n/2)	1
15	4	1

Steps executed when input is NOT a base case: 3,4,5,6,7,8,9,10,11,12,13,14,15
Second recurrence relation: T(n>1) = 2T(n/2) + 11n + 19
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:

$T_3(n) = \sum_{i=0}^{log_2(n-1)} (19 * 2^i) + 12 * 2^{\log_2 n}$

$< 19(\frac{2^{log_2(n-1)+1} - 1}{2-1}) + 12n$

$= 19(2^{[\log_2(n-1) + \log_2 2]} - 1) + 12n$

$= 19[2^{(\log_2 2(n-1)} - 1] + 12n$

$= 19[2(n-1)^{\log_2 2} - 1] + 12n$

$= 19[2n - 3] + 12n$

$= 38n - 47 + 12n$

$= 50n - 47$

$= O(n)$

Algorithm-4

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

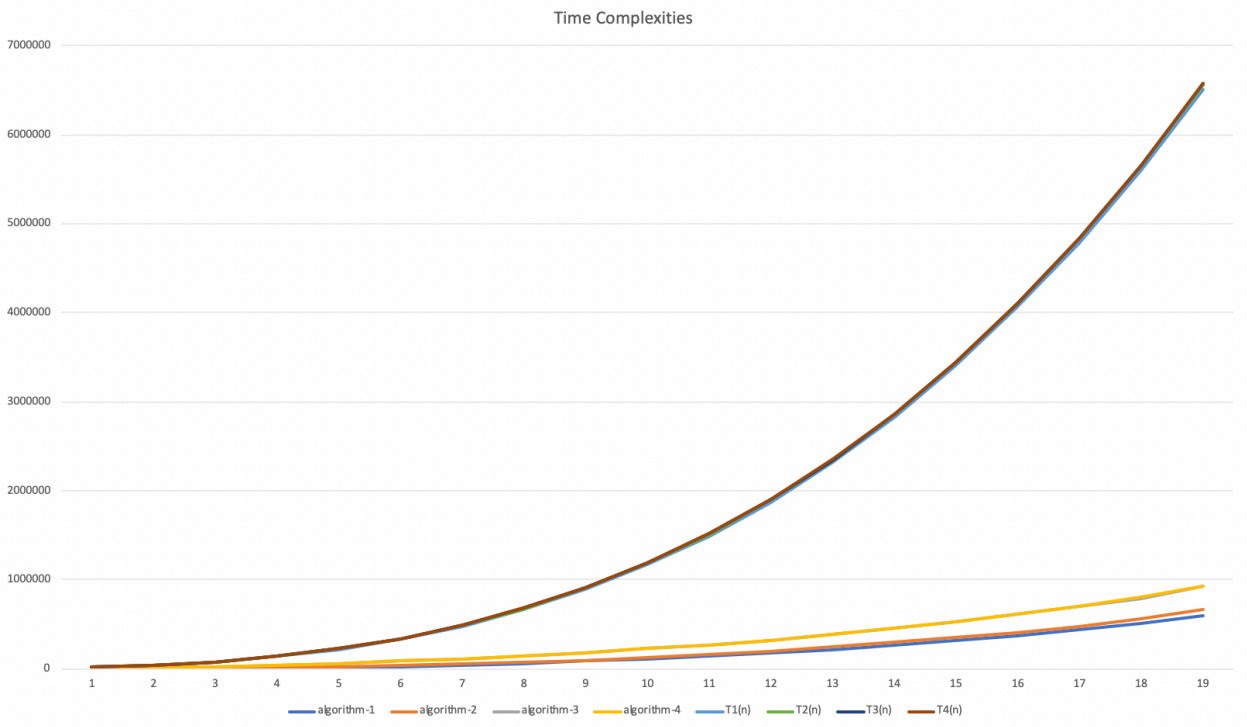
Multiply col.1 with col.2, add across rows and simplify
$T_4(n) = 1 + 1 + (n + 1) + 7n + 4n + 1$

$= 4 + 8n$

$= 12n + 4$

$= O(n)$

Graph



From what we can see from the graph, the actual running time of each algorithm did not match their predicted running times as the input size increases from 10-100. One reason for this might be because the predicted times increase at a constant rate. On the other hand, the actual running times are based on the types of integers that are in the array as well as the actual time that the algorithm takes which might be affected by different types of inputs. Overall, it is useful to try and predict time complexities in order to help us understand the actual behavior of algorithms in real time.