CJ Young (cty0008)
Shanti Upadhyay (spu0004)
COMP 3270 Programming Assignment – Fall 2022

Algorithm-1

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

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

$T_1(n) = 1 + n + 1 + (n^2/2) + (3n/2) + (n^2/2) + (n/2) + (n^3/2) + (5n/2) + 6(n^3/2) + 6(3n/2) + 4(n^2/2) + 4(n/2) + 2 = (7n^3/2) + 3n^2 + (33/2)n + 3$ **$= O(n^3)$**

Algorithm-2

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

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

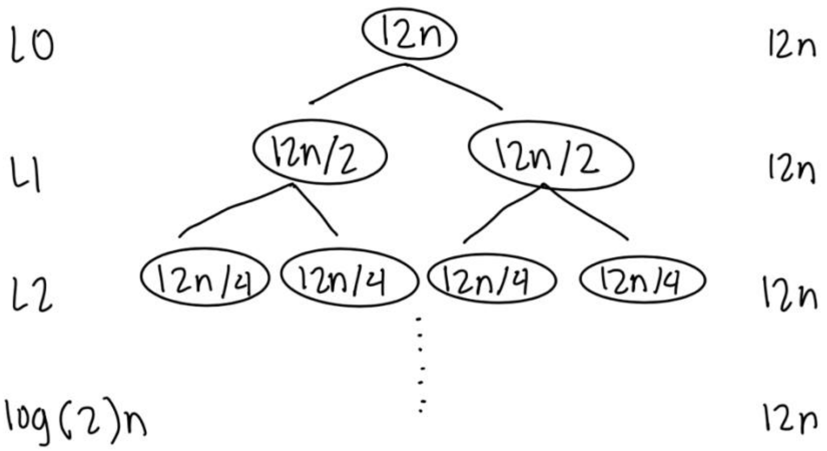
$T_2(n) = 1 + n + 1 + n + (n^2/2)+(3n/2) + 3n^2 + 3n + 2n^2 + 2n + 2 = n^2/2 + 3n/2 + 7n + 4 = n^2/2 + 17n/2 + 4$ **$= O(n^2)$**

Algorithm-3

Step	Cost of each execution	Total # of times executed in any single recursive call
1	3	1
2	3	1
Steps executed when the input is a base case: 1- 2		
First recurrence relation: T(n=1 or n=0) = 12		
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	5	(cost excluding the recursive call) 1

14	6	(cost excluding the recursive call) 1
15	5	1
Steps executed when input is NOT a base case: 1, 3-15		
Second recurrence relation: T(n>1) = 2T(n/2) + 12n + 37		
Simplified second recurrence relation (ignore the constant term): T(n>1) = 2T(n/2) + 12n		

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



$T_3(n) = 12n\ ((\log_2 n) + 1) = 12n\log_2 n + 12n$ **= O(nlogn)**

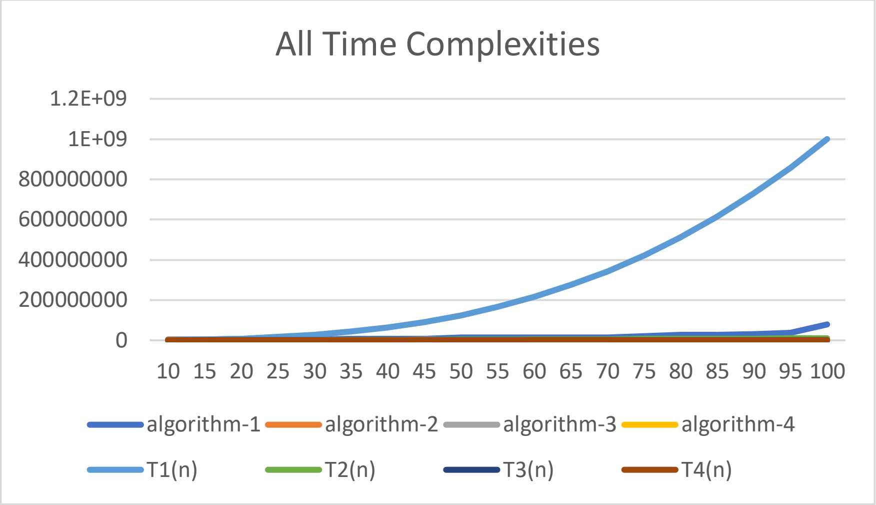
Algorithm-4

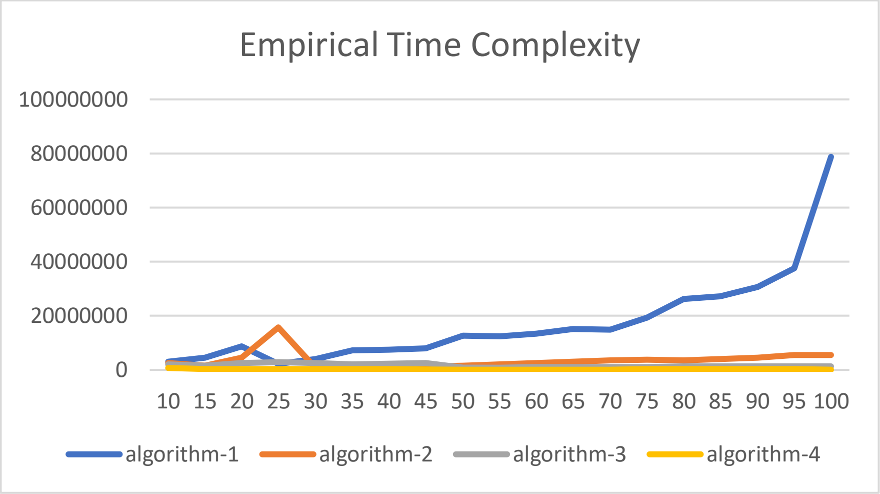
Step	Cost of each execution	Total # of times executed
1	1	1
2	1	1
3	1	n+1
4	8	n
5	4	n
6	2	1

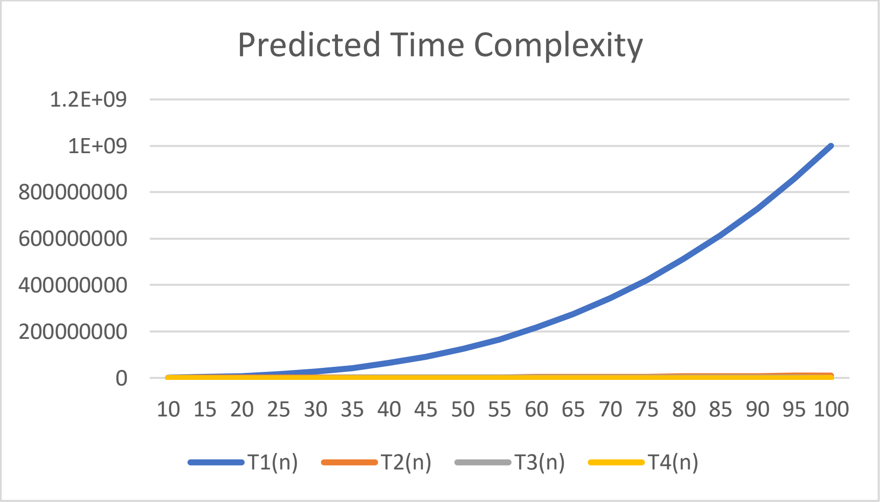
Multiply col.1 with col.2, add across rows and simplify

$T_4(n) = 1 + 1 + (n+1) + 8n + 4n + 2 = (n+1)+12n+4 = 13n+5$ **= O(n)**

After taking the data from UpadhyayYoung_phw_output.txt and displaying the data on a spreadsheet, we obtain the following graphs:







Explanation of Graphs:

The above graphs show the rate of growth for each algorithm that is being measured. The empirical and predicted time complexities represent the actual time and predicted time, respectively. There were certain variations in the algorithmic time complexity and the scaled complexity we calculated as a reference value, especially in algorithm 2. Overall, we believe that the algorithm matches the predicted values and had the same behavior as the input sizes increased from 10 to 100.