Austin Keelin
Yilmaz
COMP 3270
4/5/2018

The code for this assignment was written and compiled using the jGRASP IDE, version 2.04_04

I certify that I wrote the code I am submitting. I did not copy whole or parts of it from another student or have another person write the code for me. Any code I am reusing in my program is clearly marked as such with its source clearly identified in comment.

Algorithm-1

Step	Cost of each execution	Total # of times executed
1	1	1
2	1	n + 1
3	1	∑i=1 to n (i+1)
4	1	∑i=1 to n (i)
5	1	∑i=1 to n ∑j=1 to i (j+1)
6	6	∑i=1 to n ∑j=1 to i (j)
7	7	∑i=1 to n (i)
8	2	1

Multiply col.1 with col.2, add across rows and simplify
$T(n) = 2 + n + n(n+1)/2 + n + n(n+1)/2 + (n^3 + 6n^2 + 5n)/6 + (n^3 + 3n^2 + 2n) + 7(n(n+1)/2) + 2$
$= (n^3/6 + n^3) + (n^2/2 + n^2/2 + n^2 + 3n^2 + 7n^2/2) + (n + n/2 + n + n/2 + 5n/6 + 2n + 7n/2) + (2 + 2)$
$= 7n^3/6 + 17n^2/2 + 28n/3 + 4$
$= O(n^3)$

Algorithm-2

Step	Cost of each execution	Total # of times executed
1	1	1
2	1	n + 1
3	1	n
4	1	∑i=1 to n (i+1)
5	6	∑i=1 to n (i)
6	7	∑i=1 to n (i)
7	2	1

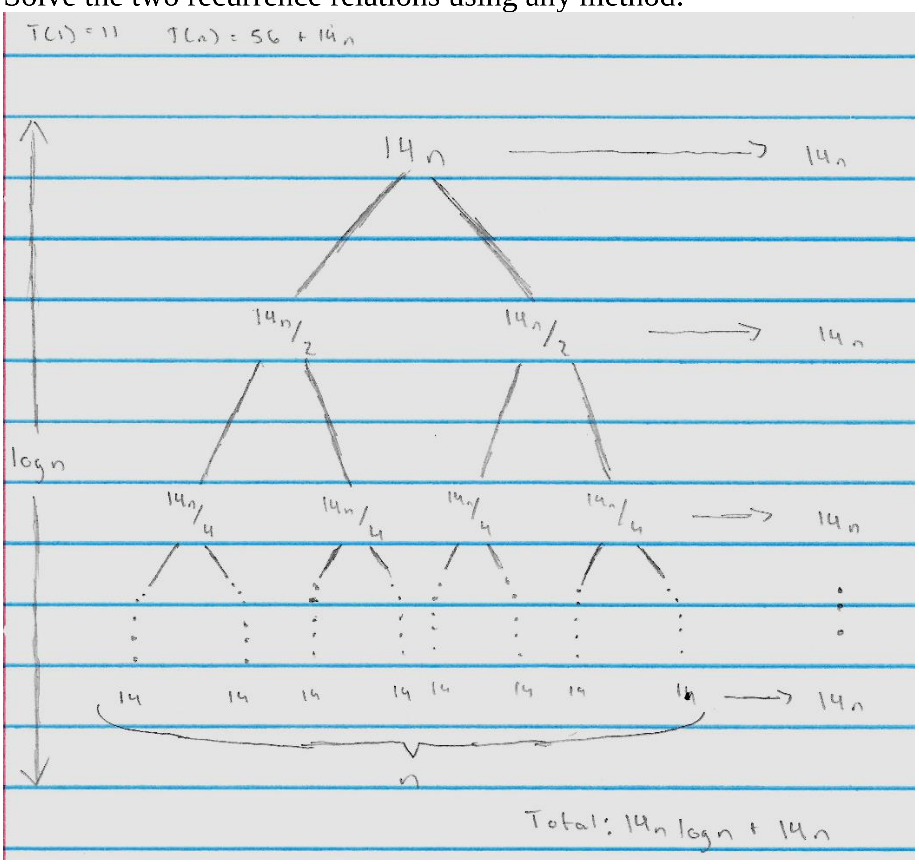
Multiply col.1 with col.2, add across rows and simplify
T(n) = 1 + (n + 1) + n + ∑i=1 to n (i + 1) + 6∑i=1 to n (i) + 7∑i=1 to n (i) + 2
= 3 + 2n + 14∑i=1 to n (i) + ∑i=1 to n (1)
$= 3 + 3n + 7(n(n + 1)) = 2 + 10n + 7n^2$
$= O(n^2)$

Algorithm-3

Step	Cost of each execution	Total # of times executed in any single recursive call

1	4	1
2	11	1
Steps executed when the input is a base case: 1 or 2		
First recurrence relation: T(n=1 or n=0) = 4 for n=0; 11 for n=1		
3	5	1
4	2	1
5	1	(n / 2) + 1
6	6	n / 2
7	7	n / 2
8	2	1
9	1	(n / 2) + 1
10	6	n / 2
11	7	n / 2
12	4	1
13	4	(cost excluding the recursive call) 1
14	5	(cost excluding the recursive call) 1
15	17	1
Steps executed when input is NOT a base case:1 – 15		
Second recurrence relation: T(n>1) = 56 + 14n		
Simplified second recurrence relation (ignore the constant term): T(n>1) = 14n		

Solve the two recurrence relations using any method:



T(n) = 14nlogn + 14n
= O(n log n)

Algorithm-4

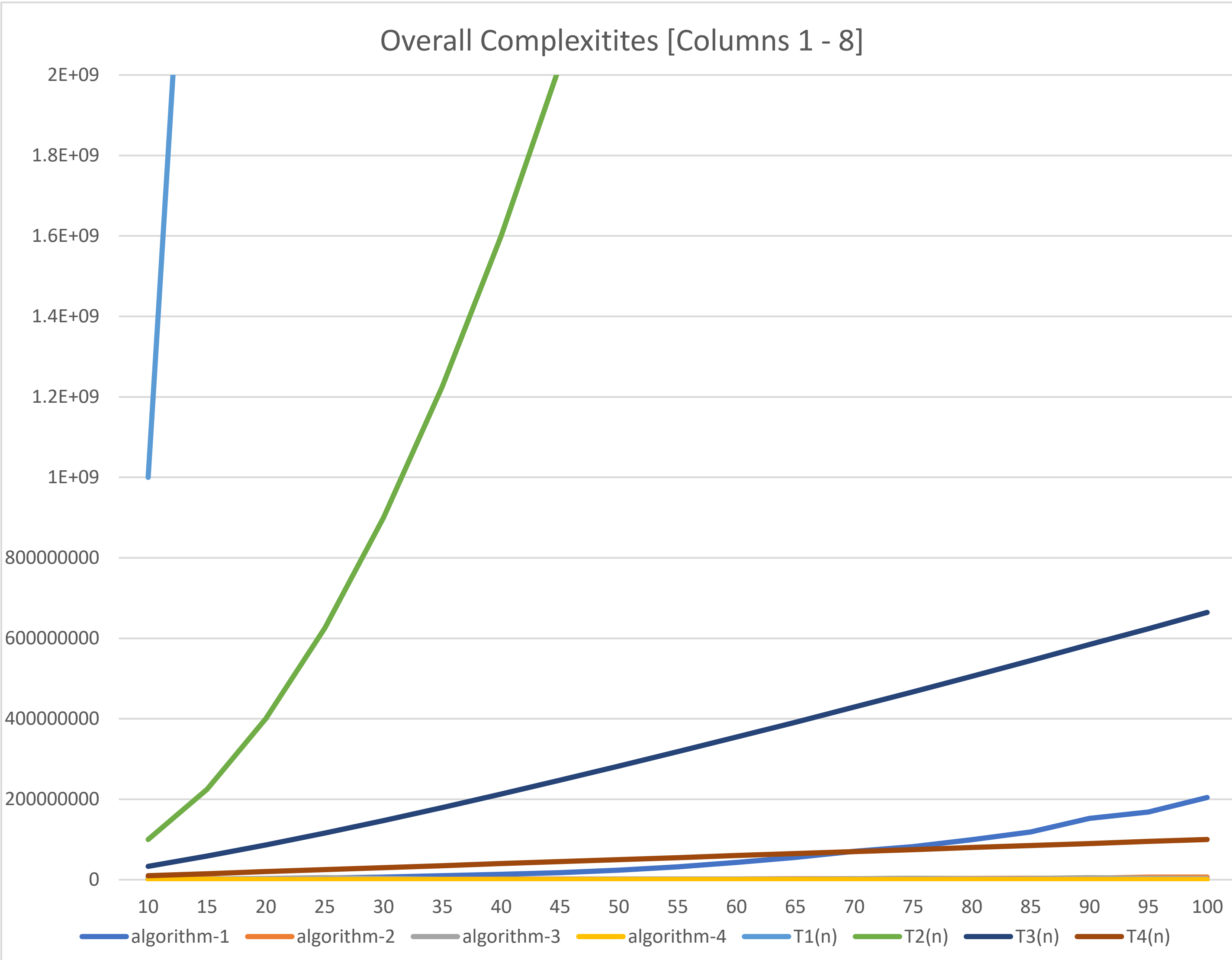
Step	Cost of each execution	Total # of times executed
1	1	1
2	1	1
3	1	n + 1
4	10	n
5	7	n
6	2	1

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

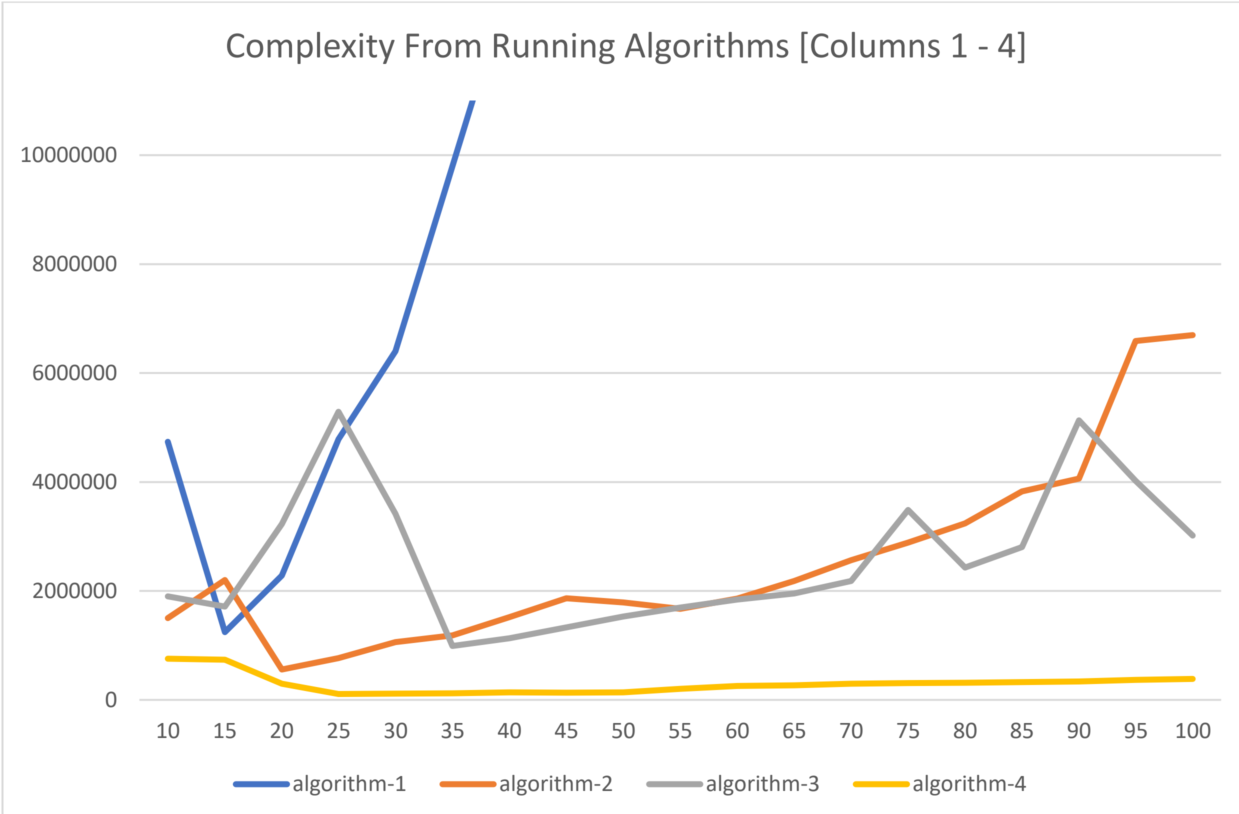
$T(n) = 1 + 1 + n + 1 + 10n + 7n + 2$

$= 18n + 5$

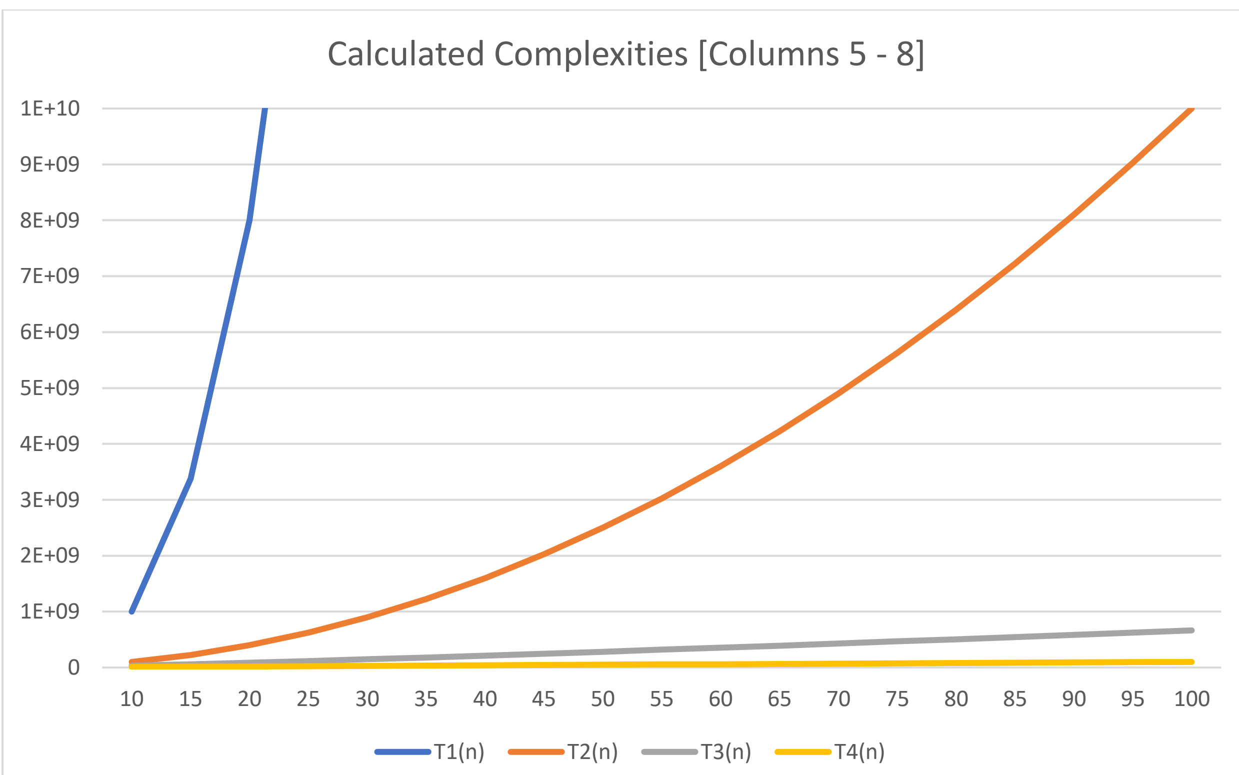
$= O(n)$



Overall Complexitites [Columns 1 - 8]
2E+09
1.8E+09
1.6E+09
1.4E+09
1.2E+09
1E+09
800000000
600000000
400000000
200000000
0
10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100
algorithm-1
algorithm-2
algorithm-3
algorithm-4
T1(n)
T2(n)
T3(n)
T4(n)



Complexity From Running Algorithms [Columns 1 - 4]
10000000
8000000
6000000
4000000
2000000
0
10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100
algorithm-1
algorithm-2
algorithm-3
algorithm-4



Calculated Complexities [Columns 5 - 8]
1E+10
9E+09
8E+09
7E+09
6E+09
5E+09
4E+09
3E+09
2E+09
1E+09
0
10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95 100
T1(n)
T2(n)
T3(n)
T4(n)

The actual time taken matched the predicted complexity for each algorithm as the input size increased. In the early runs with the smaller input sizes, where the algorithms would reach large complexity early and then dip, the data appears to suggest that the predictions were incorrect but as the algorithms continued running and reached larger input sizes, the data for the complexity of these runs each match with the respective calculated complexities.