Stephen Sallas
04/13/2021
COMP-3270

Programming Assignment

Algorithm-1

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

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

$T_1(n) = 1(1) + 1(n+1) + 1\left(\frac{n(n+1)}{2} + 1\right) + 1\left(\frac{n(n+1)}{2}\right) + 1\left(\frac{n(n+1)(n+2)}{6} + 1\right) + 6\left(\frac{n(n+1)(n+2)}{6}\right) + 5\left(\frac{n(n+1)}{2}\right) + 2(1) = \frac{7n^3}{6} + 7n^2 + \frac{41(n)}{6} + 6 = \theta(n^3)$

Stephen Sallas
04/13/2021
COMP-3270

Algorithm-2

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

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

$T_2(n) = 1(1) + 1(n+1) + 1(n) + 1\left(\frac{n(n+1)}{2} + 1\right) + 6\left(\frac{n(n+1)}{2}\right) + 5\left(\frac{n(n+1)}{2}\right) + 2(1) = 6n^2 + 8n + 5 = \theta(n^2)$

Stephen Sallas
04/13/2021
COMP-3270

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 or 2**		
First recurrence relation: T(n=1 or n=0) = **T(1)=13, T(0)=14**		
3	5	1
4	2	1
5	1	$\frac{n}{2}+1$
6	6	$\frac{n}{2}$
7	5	1
8	2	1
9	1	$\frac{n}{2}+1$
10	6	$\frac{n}{2}$
11	5	1
12	4	1
13	5	(cost excluding the recursive call) = 1
14	6	(cost excluding the recursive call) = 1
15	6	1
Steps executed when input is NOT a base case: **1-15**		
Second recurrence relation: T(n>1) = $\boldsymbol{2T\left(\frac{n}{2}\right)+2n+9}$		
Simplified second recurrence relation (ignore the constant term): T(n>1) = $\boldsymbol{2T\left(\frac{n}{2}\right)+2n}$		

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

Using Master Theorem… T(n) = $2T\left(\frac{n}{2}\right)+2n$

a = 2, b = 2, c = 2, k = 1, Since $a=b^k$… T(n) = $\theta(nlogn)$

T_3(n) = $\theta(nlogn)$

Stephen Sallas
04/13/2021
COMP-3270

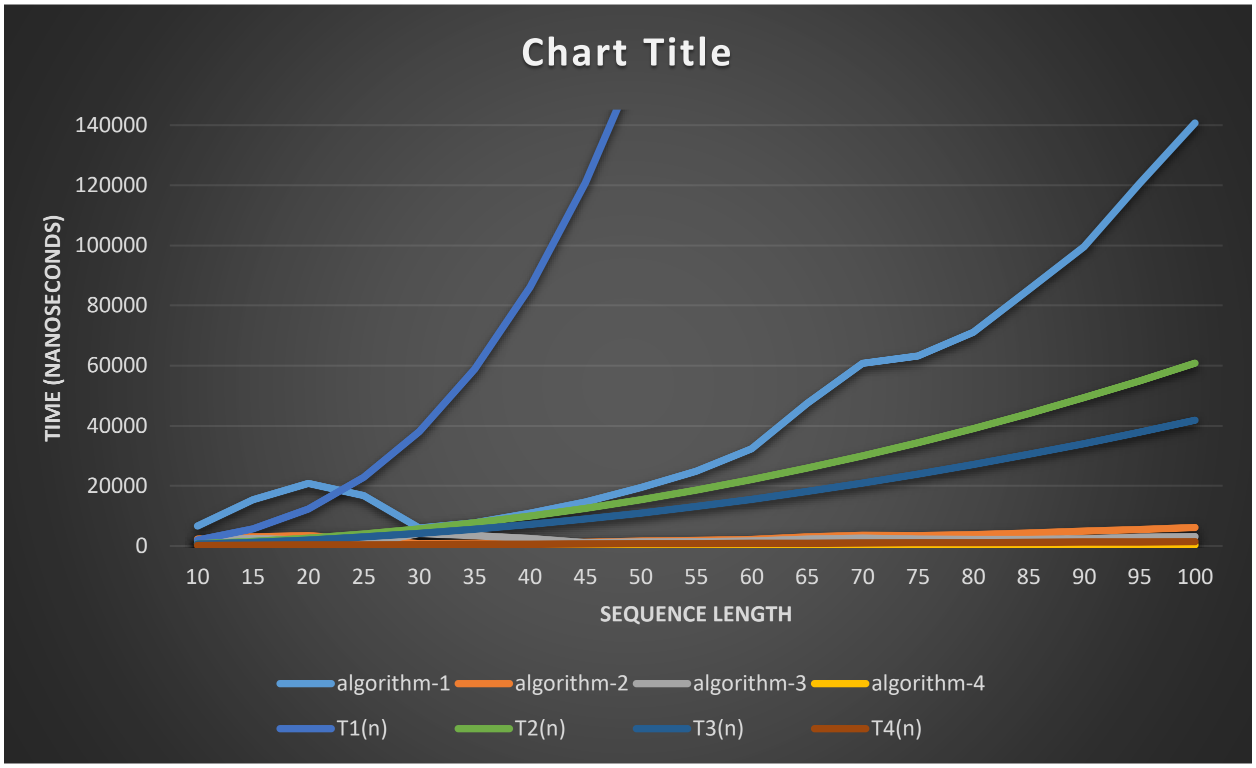
Algorithm-4

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

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

$T_4(n) = 1(1) + 1(1) + 1(n + 1) + 8(n) + 5(n) + 1(1) = 14n + 5 = \theta(n)$

Stephen Sallas
04/13/2021
COMP-3270



This line graph shows the progression of time taken by the algorithms and predictions to execute on increasingly larger sequences of integers. The graph came out almost exactly as I expected it to. The predicted complexities will always take more time than the algorithms since they are close to the worst possible time complexity. T1(n) in particular is so large I had to partially crop it out to make the rest of the graph visible. The algorithms and predictions all consistently took more time as the sequence length increased. This was expected because with more integers the longer it takes to process. This did have some exceptions, as seen in algorithm-1, because the numbers are randomly generated. The more times this graph is generated and averaged, the closer the algorithms would be to the predictions.