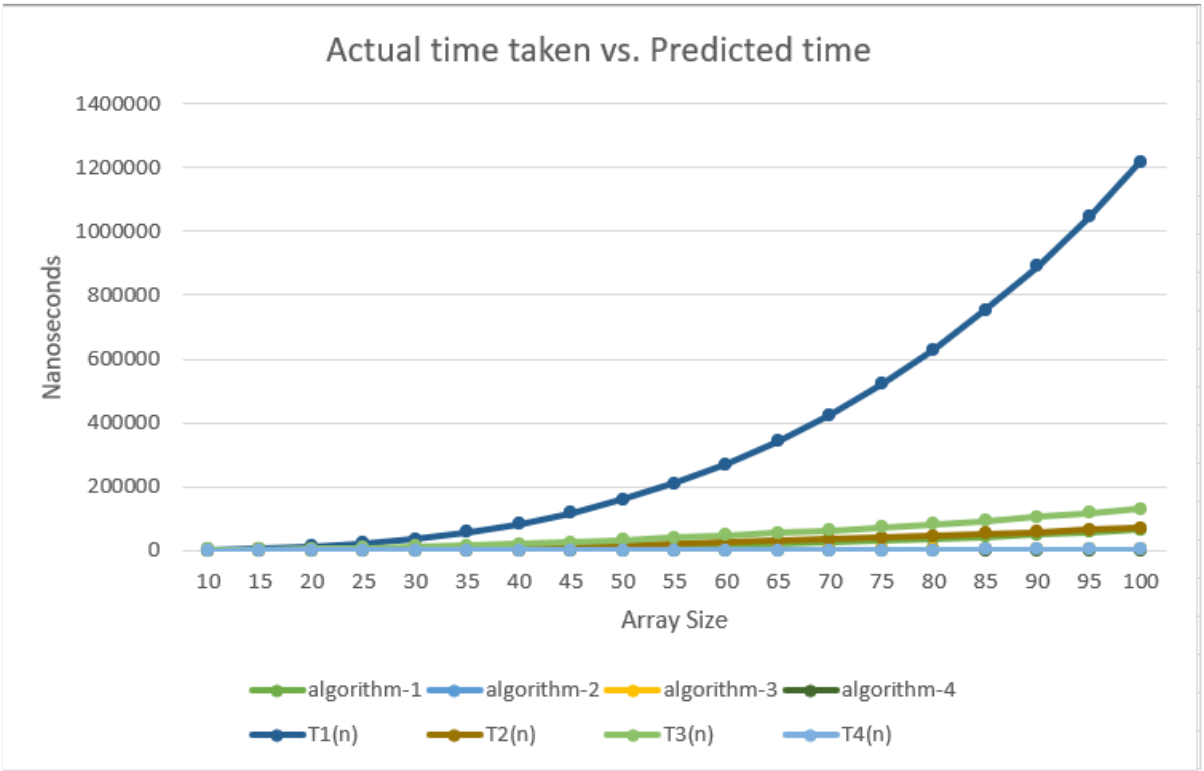
Lyna Capuano
Austin Smith
COMP 3270 – Algorithms
11/18/22

Actual time taken and predicted time (the complexity order) for each algorithm.



Explanation:
The actual time taken does match up the predicted time complexity for each respective algorithm. The calculated ceiling(T(n)) are the worst-case times, so this acts as the upper bound for the algorithms. With smaller values of input size n, we see that the Actual time matches Predicted time. But as the input size increases from 10-100 we notice that the Actual time does not match Predicted time, but this is as expected.

We have included separate graphs under each time complexity calculation which shows only the algorithm and T(n) for that particular algorithm. This allows you to have a closer look at the relation between the two lines.

Algorithm-1

Step	Cost of each execution	Total # of times executed
1 maxSoFar = 0	1	1
2 **for** L = P to Q	1	n + 1
3 **for** U = L to Q	1	$\sum_{i=0}^{n}(\mathrm{i}) + 1 \approx \frac{n(n+1)}{2} + 1$
4 sum = 0	1	$\sum_{i=0}^{n}(\mathrm{i}) \approx \frac{n(n+1)}{2}$
5 **for** I = L to U	1	$\sum_{j=0}^{n}(\sum_{i=0}^{i}(\mathrm{i}))+1 \approx \frac{n(n+1)(n+2)}{2} + 1$

6 sum = sum + X[I]	6	$\sum_{j=0}^{n}(\sum_{i=0}^{i}(i)) \approx \frac{n(n+1)(n+2)}{2}$
7 maxSoFar = max(maxSoFar,sum)	7	$\sum_{i=0}^{n}(i+1) \approx \frac{n(n+1)}{2}$
8 return maxSoFar	2	1

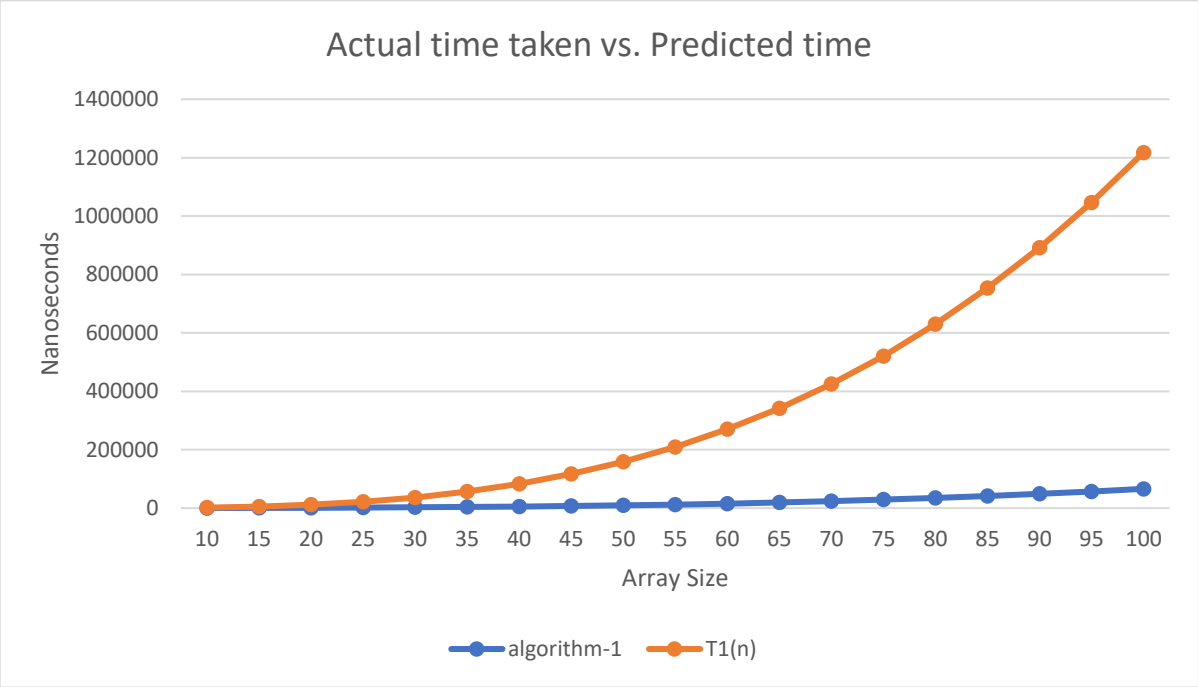
Multiply col.1 with col.2, add across rows and simplify

$T_1(n) = (1)(1) + (1)(n+1) + (1)(\frac{n(n+1)}{2} + 1) + (1)\left(\frac{n(n+1)}{2}\right) + (1)\left(\frac{n(n+1)(n+2)}{2} + 1\right) + (6)\left(\frac{n(n+1)(n+2)}{2}\right) + (7)\left(\frac{n(n+1)}{2}\right) + (2)(1)$

$T_1(n) = (n+1) + (9)(\frac{n(n+1)}{2}) + (7)(\frac{n(n+1)(n+2)}{2}) + 6$

$\mathbf{T1(n) = \left(\frac{7n^3+30n^2+23n}{6}\right) + n + 7}$

$\mathbf{T_1(n) \approx O(n^3)}$



Included a separate graph showing only Algorithm-1 and T1(n). For full explanation please see first page of this document.

Algorithm-2

Step	Cost of each execution	Total # of times executed
1 maxSoFar = 0	1	1
2 **for** L = P to Q	1	n + 1
3 sum = 0	1	n
4 **for** U = L to Q	1	$\sum_{i=0}^{n}(i) + 1 \approx \frac{n(n+1)}{2} + 1$
5 sum = sum + X[U]	6	$\sum_{i=0}^{n}(i) \approx \frac{n(n+1)}{2}$

6 maxSoFar = max(maxSoFar,sum)	7	$\sum_{i=0}^{n}(i) \approx \frac{n(n+1)}{2}$
7 return maxSoFar	2	1

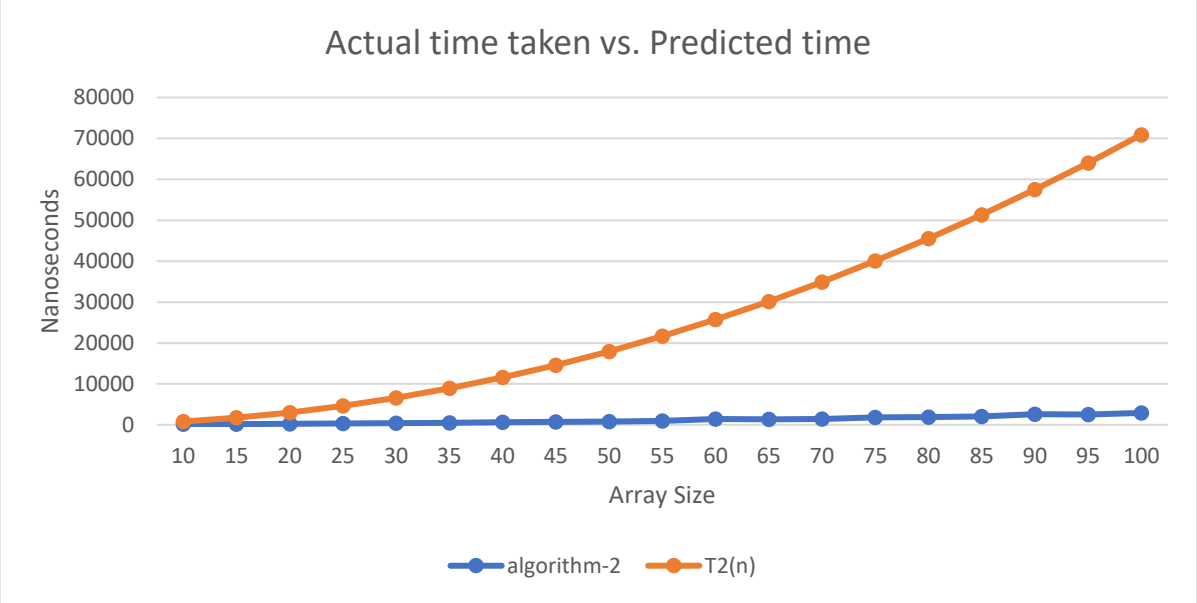
Multiply col.1 with col.2, add across rows and simplify

$T_2(n) = (1)(1) + (1)(n + 1) + (1)(n) + (1)(\frac{n(n+1)}{2} + 1) + (6)(\frac{n(n+1)}{2}) + (7)(\frac{n(n+1)}{2}) + (2)(1)$

$T_2(n) = (n+1) + n + (14)(\frac{n(n+1)}{2}) + 4$

$T_2(n) = 7n^2 + 9n + 5$

$T_2(n) \approx O(n^2)$



Included a separate graph showing only Algorithm-1 and T1(n). For full explanation please see first page of this document.

Algorithm-3

Step	Cost of each execution	Total # of times executed in any single recursive call
1 if L > U then return 0	Cost with return: 4 Cost without return: 3	1
2 if L=U then return max(0,X[L])	Cost with return: 9 Cost without return: 3	1
Steps executed when the input is a base case: step 1 Second base case: if L=U then return max(0,X[L])		
First recurrence relation: T(n=1 or n=0) = cost of T(0) = 4 cost of T(1) = 12		
3 M = (L+U)/2	5	1
4 sum = 0; maxToLeft = 0	2	1
5 **for** I = M downto L do	1	(n/2) +1
6 sum = sum + X[I]	6	n/2
7 maxToLeft = max(maxToLeft,sum)	6	n/2
8 sum=0; maxToRight=0	2	1
9 **for** I = M+1 to U	1	(n/2) + 1
10 sum = sum + X[I]	6	n/2

11 maxToRight = max(maxToRight, sum)	6	n/2
12 maxCrossing = maxToLeft + maxToRight	4	1
13 maxInA = maxSum(X, L, M)	1+T(n/2)	(cost excluding the recursive call) log(n)
14 maxInB = maxSum(X,M+1,U)	1+T(n/2)	(cost excluding the recursive call) log(n)
15 return max(maxCrossing, maxInA, maxInB)	11	1
Steps executed when input is NOT a base case: steps 1-15		
Second recurrence relation: T(n>1) = 2T(n/2) + c		
Simplified second recurrence relation (ignore the constant term): T(n>1) = $(2^{\log(n)})\left(T\left(n/(2^{\log(n)})\right)\right) + (2^{\log(n)} - 1)$		

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

$c = (3)(1) + (3)(1) + (5)(1) + (2)(1) + (1)(\frac{n}{2} + 1) + (6)(\frac{n}{2}) + (6)(\frac{n}{2}) + (2)(1) + (1)\left(\frac{n}{2} + 1\right) + (6)(\frac{n}{2}) + (6)(\frac{n}{2}) + (4)(1) + (11)(1)$

$c = 13n + 32$

$T_3(n) = 2T(\frac{n}{2}) + c$

$T(\frac{n}{2}) = 2T(\frac{n}{4}) + c$

$T_3(n) = 2(2T(\frac{n}{4}) + c) + c$

$T(\frac{n}{4}) = 2T(\frac{n}{8}) + c$

$T_3(n) = 2(2(2T(\frac{n}{8}) + c) + c) + c$

…

$T_3(n) = (2^{\log(n)})\left(T\left(\frac{n}{2^{\log(n)}}\right)\right) + (2^{\log(n)} - 1)(13n + 32)$

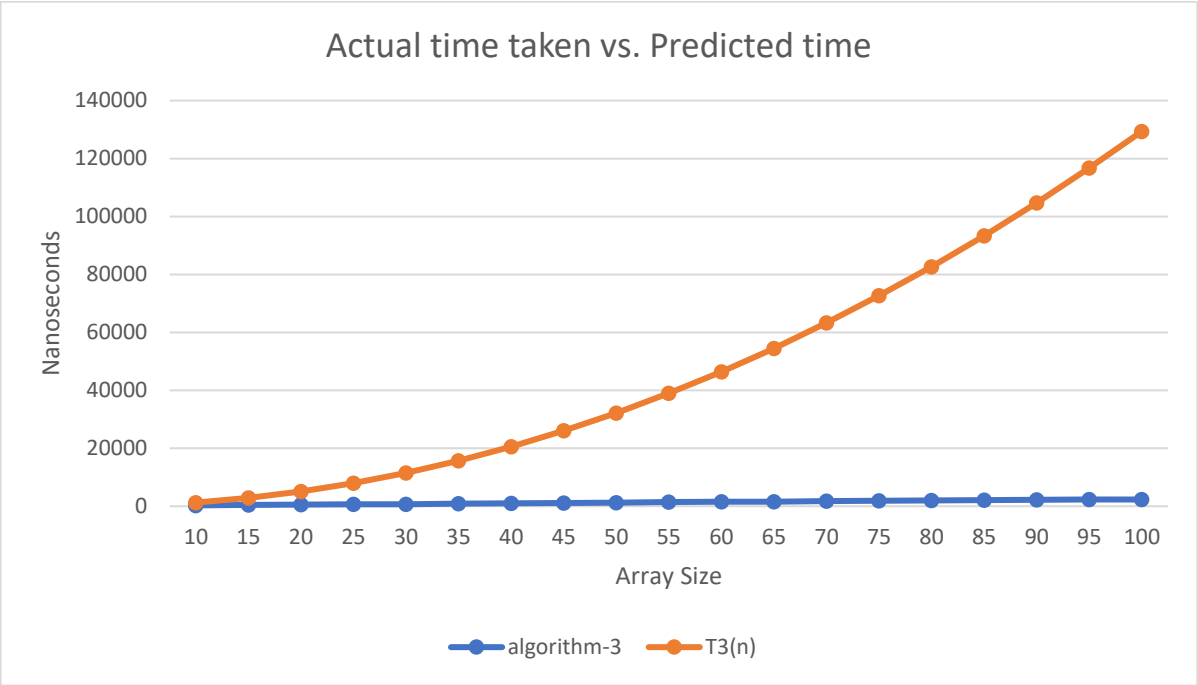
$T_3(n) = n(T(1)) + (n - 1)(13n + 32)$

$T(1) = 12$

$T3(n) = 12n + 13n^2 + 19n - 32$

$T3(n) = 13n^2 + 31n - 32$

$\mathbf{T_3(n) \approx O(n^2)}$



Included a separate graph showing only Algorithm-3 and T3(n). For full explanation please see first page of this document.

Algorithm-4

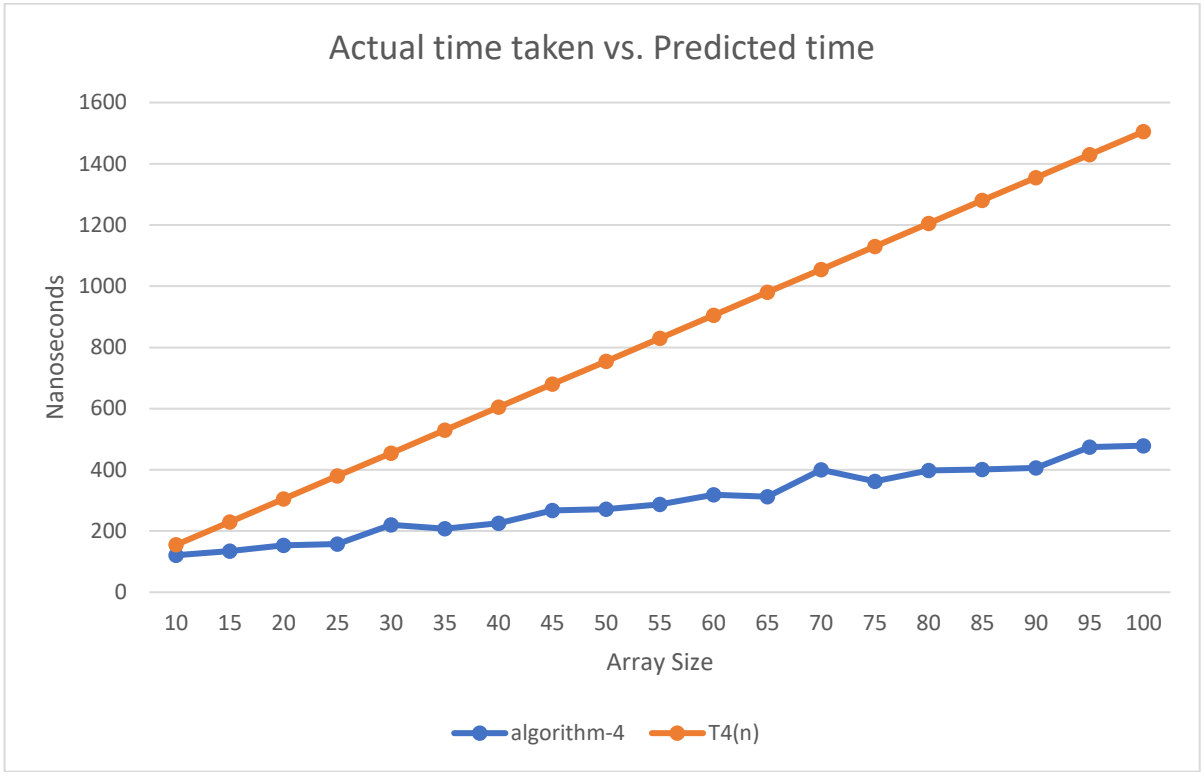
Step	Cost of each execution	Total # of times executed
1 maxSoFar = 0	1	1
2 maxEndingHere = 0	1	1
3 **for** I = P to Q	1	n + 1
4 maxEndingHere = max(0, maxEndingHere + X[I])	7	n
5 maxSoFar = max(maxSoFar, maxEndingHere)	7	n
6 return maxSoFar	2	1

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

$T_4(n) = (1)(1) + (1)(1) + (1)(n+1) + (7)(n) + (7)(n) + (2)(1)$

$T_4(n) = 15n + 5$

$T_4(n) \approx O(n)$



Included a separate graph showing only Algorithm-4 and T4(n). For full explanation please see first page of this document.