CS301 Assignment 1

Ahmet Bilal Yıldız

23 October 2022

1 Question 1

Part a: $\Theta(n^3)$
Part b: $\Theta(n^{\log_2 7})$
Part c: $\Theta(\sqrt{n}.\log n)$
Part d: $\Theta(n^2)$

2 Question 2-a

(i)

Best asymptotic worst case for the Naive algorithm:
The worst case happens when there is no common sub-sequence occur in the given two sequences like '*aaaaaa*' and '*cccccc*'.
Since there is nothing in common the recursion always do the third option in the algorithm and it becomes:

$T(m, n) = T(m-1, n) + T(m, n-1) + O(2)$

$O(2)$ comes from the $max()$ operation since it compares two elements.

The recursion tree can be seen below.

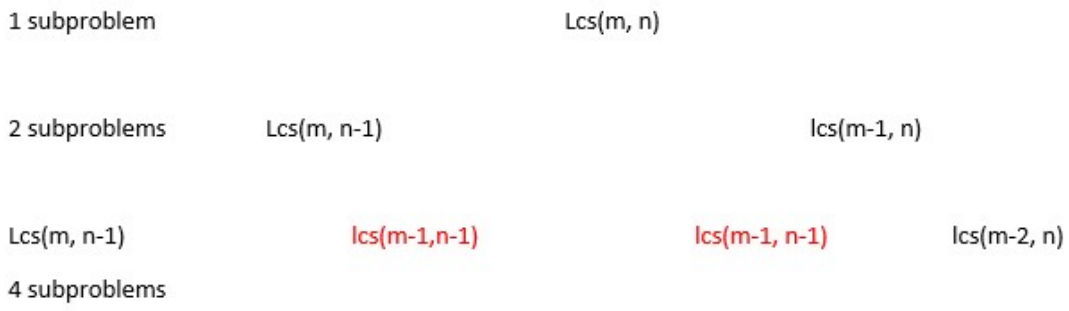

Figure 1: Recursion tree

In this approach, there are lots of repeated calculations which are represented with the red color.
As it is clear with recursion tree in each recursive call, the number of sub-problems goes 1,2,4 ... which is: $2^0, 2^1, 2^2, 2^3, ..., 2^(m+n)$
Therefore, cost of levels goes like $2^0c, 2^1c, 2^2c, 2^3c, ..., 2^(m+n-1)c$ and cost level of leafs are $T(0) = 0$. Therefore, total costs of levels is $2^(m+n) - 1$.
So, the best asymptotic running time of the Naive algorithm is $\Theta(2^(m+n))$.

(ii)

In the memoization approach, to represent all sub-problems we use a 2D array whose size is $(m+1) * (n+1)$ where m is length of first sequence and n is length of second sequence. In this approach we do not calculate the longest common sub-sequences of previously calculated sub-sequences and so we avoid repetition.

Therefore, the worst case happens when the total number of sub-sequences $= m * n$. Since the len() operation takes $O(1)$ and max() operation takes $O(2) = O(1)(actually)$, the best asymptotic running time of the Naive algorithm is $O(m * n)$.

3 Question 2-b

(i)

Algorithm	m=n=3	m=n=6	m=n=9	m=n=12	m=n=15
Naive	0.000032	0.000982	0.041217	1.788187	99.887053
Memoization	0.000017	0.000042	0.000086	0.000186	0.000205

Properties of the machine:

CPU: Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz 2.00 GHz
RAM: 16.0 GB
OS: Windows11

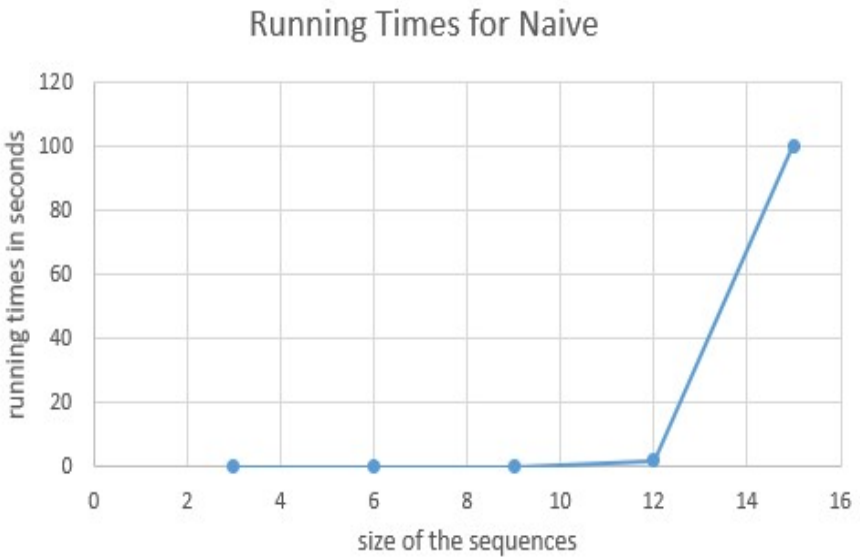

Figure 2: Running Times for Naive

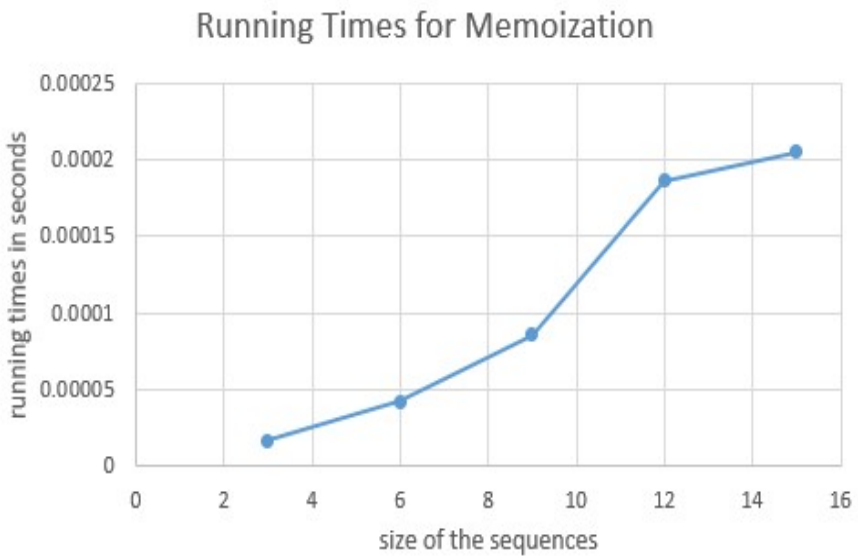

Figure 3: Running Times for Memoization

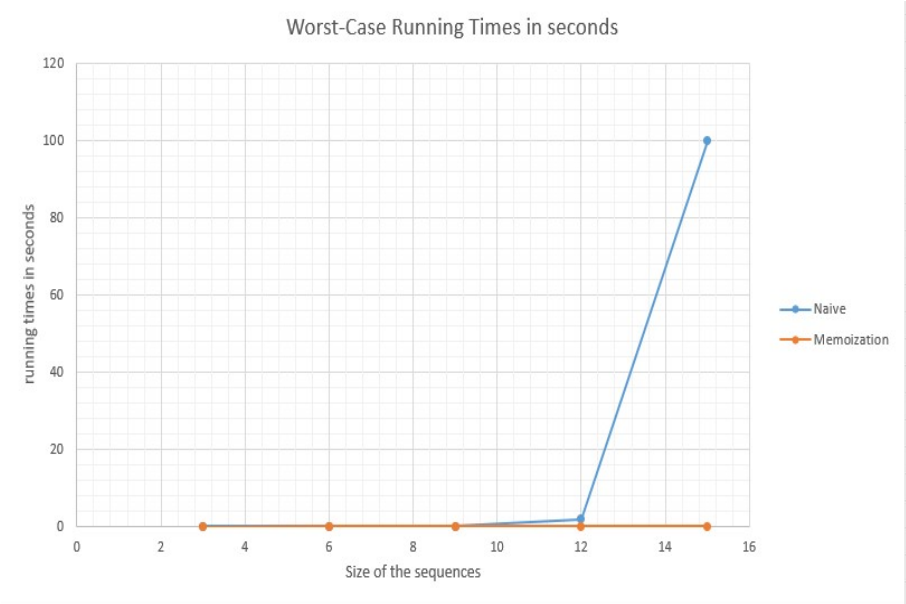

Figure 4: Worst-Case Running Times

(ii)-(iii)

Experimental results can be seen in the above graph.
According to experimental results, it is accurate that running time in the Naive approach increases very largely when the sizes of the sequences increase. Therefore, the observations(figure2) supported to the theoretical results which is $O(2^n)$ (exponential) time.

Differently, running times in the Memoization approach increases like quadratic(figure3) and that also support the theoretical results which is $O(m * n)$.

Memoization algorithm performs better in terms of scalibility because in Naive approach there are lots of repeated computations and when the size of the sequences increases the performing time increases sharply. However, the increased time amount in Memoization approach is increases slightly when it is compared to the Naive approach changes (figure4). Since the performance of the Memoization approach is better for large input sizes (large sequences here), Memoization approach is better in scalibility.

4 Question 2-c

(i)

Algorithm	m = n = 3		m = n = 6		m = n = 9		m = n = 12		m = n = 15	
	µ	σ	µ	σ	µ	σ	µ	σ	µ	σ
Naive	0.000063	0.000271	0.000106	0.000048	0.001846	0.002144	0.032801	0.0342	0.47533	0.897646
Memoization	0.000013	0.000003	0.000053	0.00015	0.000054	0.000021	0.000083	0.000015	0.000133	0.000016

(ii)

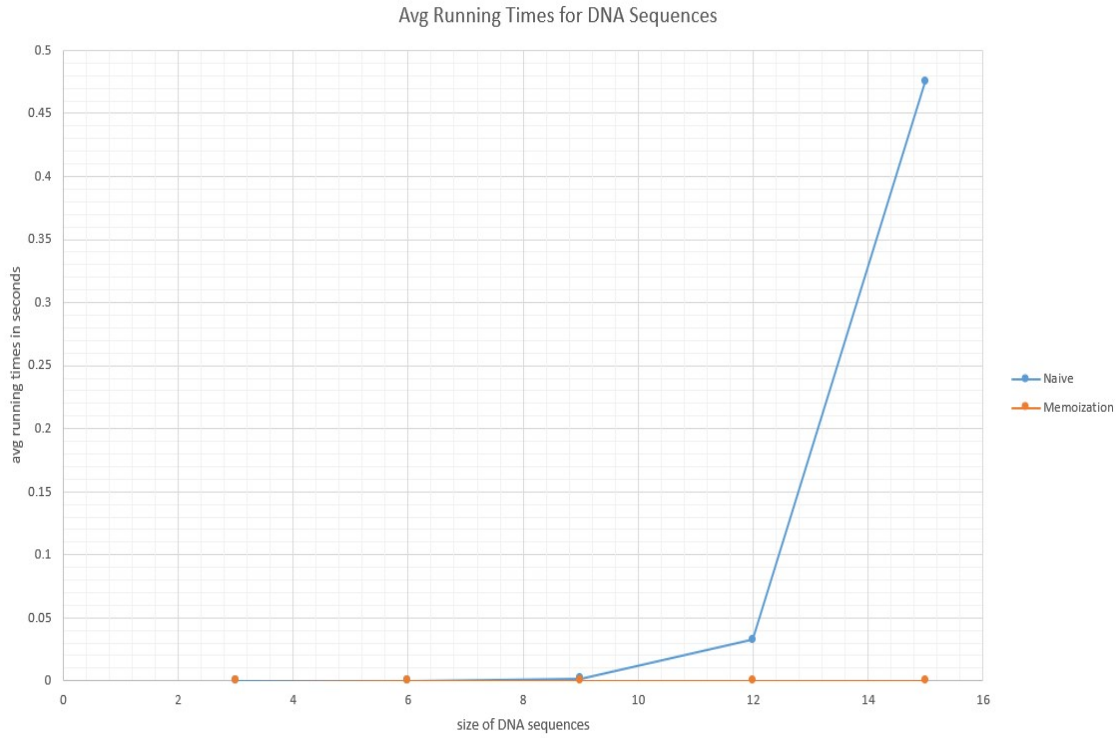

Figure 5: Avg-Running Times in DNA Sequences

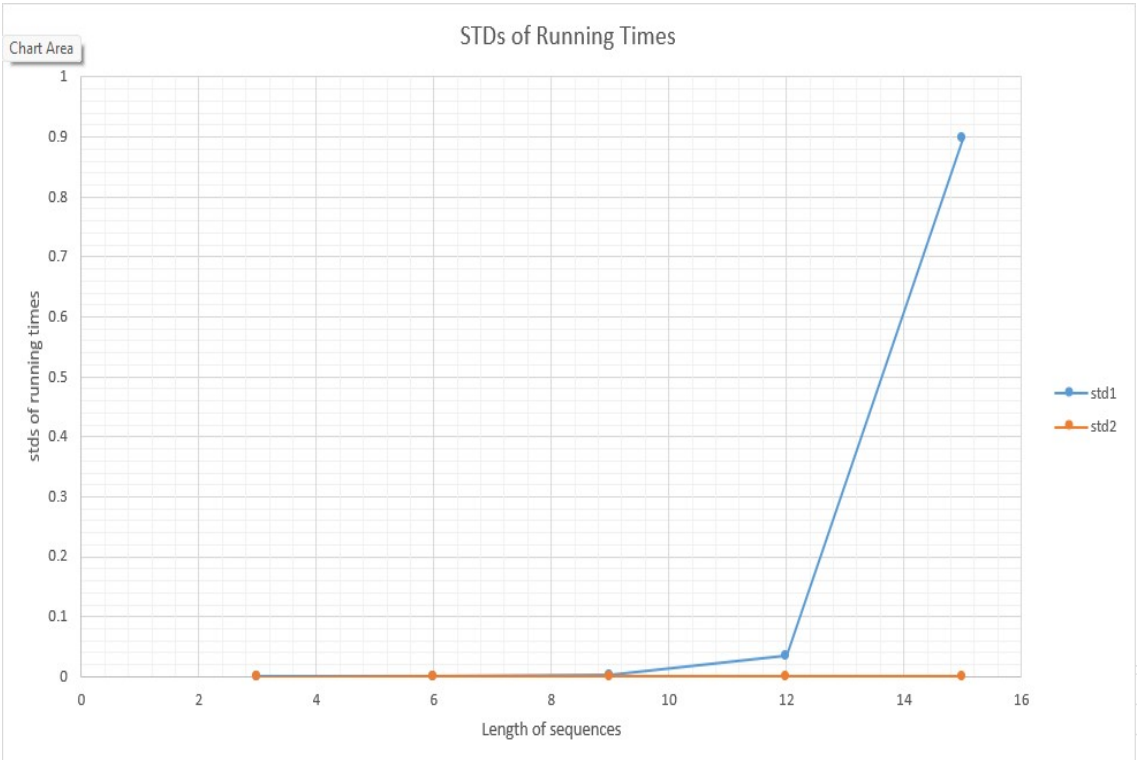

Figure 6: STD's in DNA Sequences

(iii)

When the growth of average running times considered, it is clearly seen that Naive approach grows sharply depending on the sizes of inputs, however; this growth is less than the growth in the worst case. Even the growth is less than worst-case growth, it again like exponential growth as seen in the graph.

Similarly in the Memoization approach, the growth of the average running times is less than the worst case time in general, however; the growth pattern is similar to the worst-case growth patter.

In conclusion, the average running time results for both Naive approach and Memoization approach are less than their worst case clearly, and their growths are again less than the growth in compared to their worst-case times. However, the growth patterns are similar and similar to the worst-case times it is observed that Memoization performs better with bigger inputs and so it the Memoization's better performance in terms of scalibitliy is observed again in the average running times.