CS301 - HW1

Tahir Turgut - 27035

March 19, 2021

1 Problem 1

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

2 Problem 2

(a)

(i)

Worst case has to include the substrings with no common letter. Therefore, in each iteration, even though some subproblems are called more than once, example: (X,Y,i-1,j-1) -one from (X,Y,i-1,j) and other one from (X,Y,i,j-1)- program will enter the statement of "else" which contains two call for "lcs" function, without considering it is overlapping or not, and max operation which compares 2 element is: $O(1)$. Thus recurrence relation of first algorithm: $T(n+m) = T(m+n-1) + T(m-1+n) + O(1)$

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

For Upper Bound: Assume that $T(k) \leq c2^{m+n}$ for all $k < n$:
$T(n+m) = 2c(2^{m+n-1}) + O(1)$
$T(n+m) = c2^{m+n} + O(1)$
$T(n+m) = O(2^{m+n})$

For Lower Bound: Assume that $T(k) \geq c2^{m+n}$ for all $k < n$:
$T(n+m) = 2c(2^{m+n-1}) + O(1)$
$T(n+m) = c2^{m+n} + O(1)$
$T(n+m) = \Omega(2^{m+n})$

It is shown that $T(n+m) = O(2^{m+n}) = \Omega(2^{m+n})$
Therefore, $T(n+m) = \Theta(2^{m+n})$

(ii)

Unlike the first algorithm, Algorithm 2 (Figure 2) will store the solutions of subproblems, thanks to memoization. Because of that, a subproblem will never be solved more than once. In addition to that, in a top-down fashion, all matrix indices are visited, only once. Since one side of matrix is m and other one is n (lengths of strings), asymptotic worst-case running time of algorithm 2 is:

$$\Theta(m \cdot n)$$

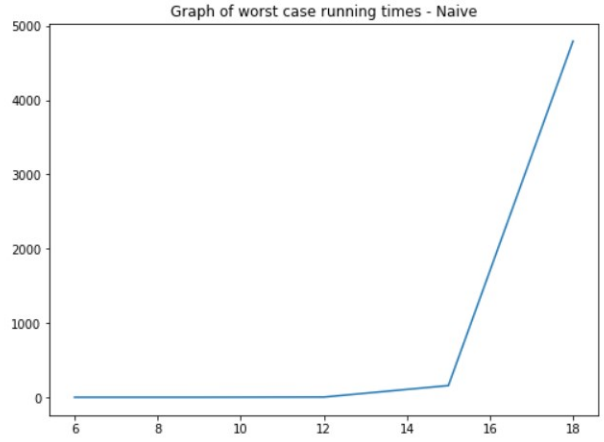
(b)

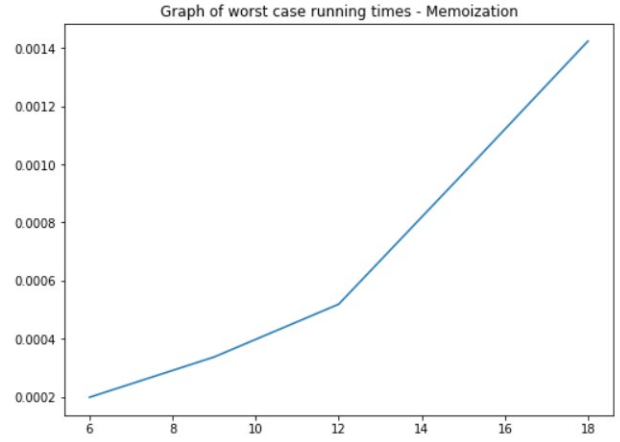
(i)

Computer properties: Windows10 OS, 16GB RAM, intel i7-770HQ CPU (2.8 gHz)

Algorithm	m = n = 6	m = n = 9	m = n = 12	m = n = 15	m = n = 18
Naive	0.0009435	0.0661942	2.9667595	157.58111	4793.8076
Memoization	0.0001986	0.0003371	0.0005187	0.0009687	0.0014233

(ii)





(iii)

Seems like memoization grows linearly which is predicted $\Theta(m \cdot n)$ -practically $\Theta(n)$, since string lengths are equal (2n).

On the other hand, it is expected that Naive algorithm's function grows (2^{m+n}). However, graph booms at the change between 15 to 18, compared to other point changes. Although it is not fitted to exponential graph, it is probably because of the input sizes, so it may be considered as exponential, in some sense. The problem may arise due to input sizes, there is little change from 6 to 9, however when input size is getting larger, differences will become larger, as well.

To sum up, Memoization grants a nearly perfect fit with the prediction, whereas Naive algorithm's graph seems little bit unexpected, but the problem is input sizing, probably. Thus, in larger cases Naive algorithm will grow enormously, then Memoization should be preferred.

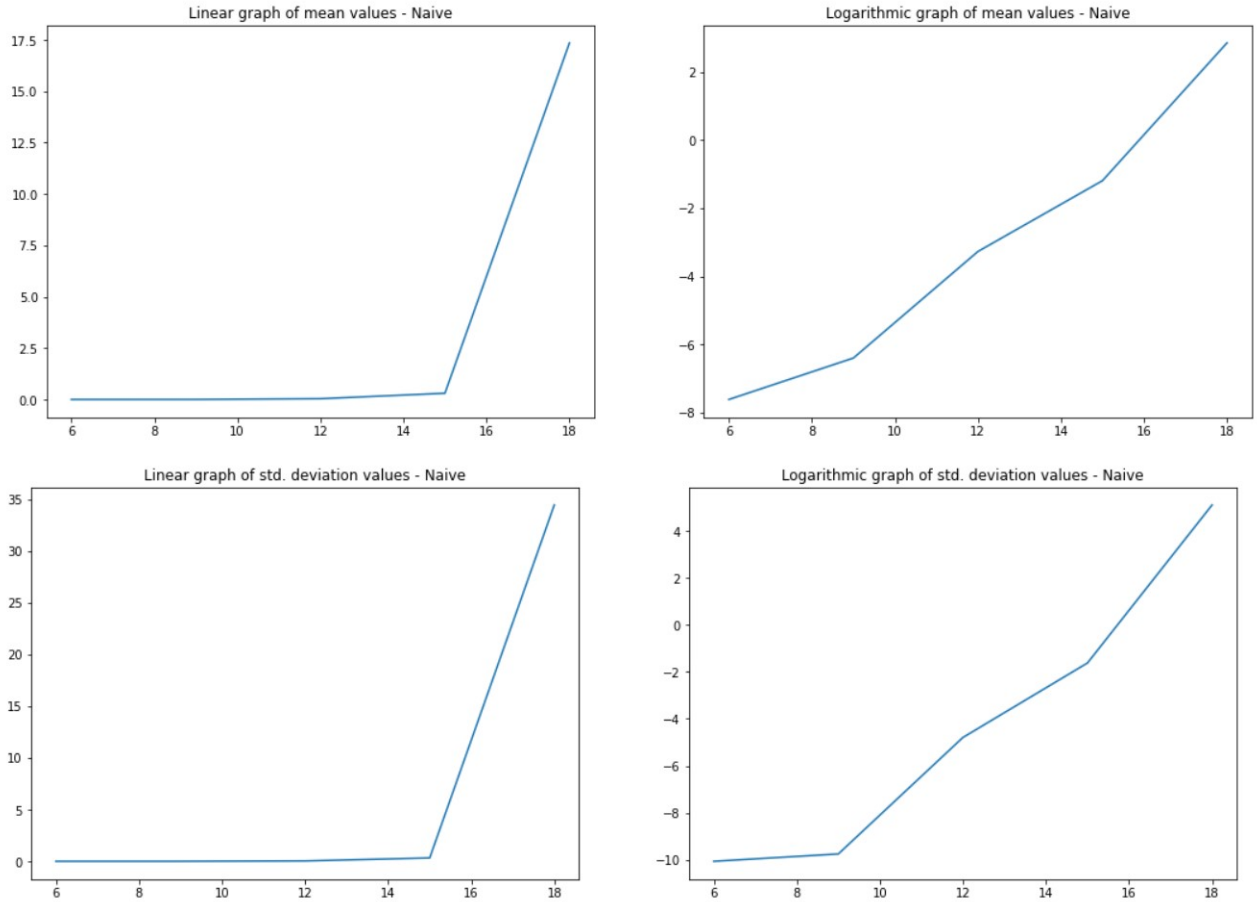
(c)

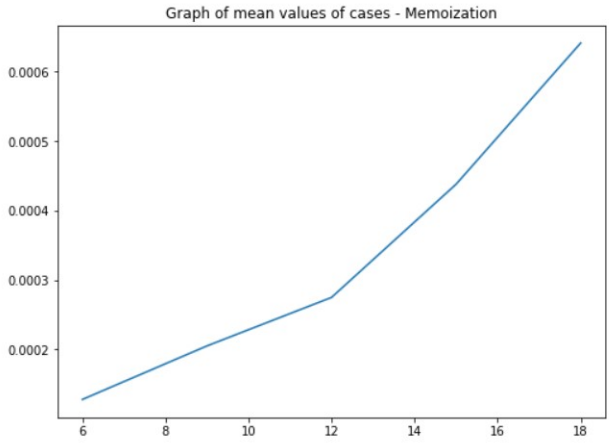
(i)

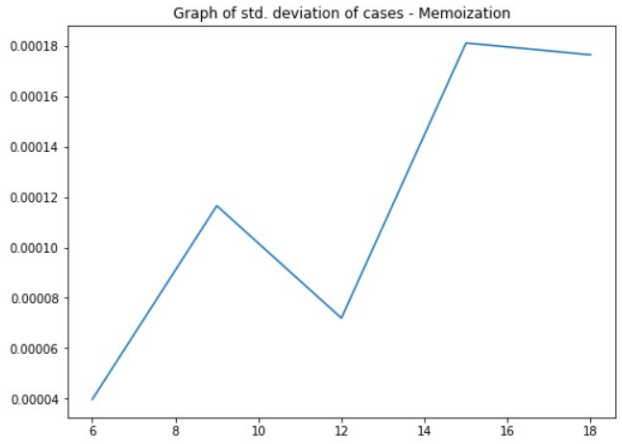
Average running time in seconds(μ)$and std. deviation$(σ)

Algorithm	m = n = 6	m = n = 9	m = n = 12	m = n = 15	m = n = 18
Naive	$\mu = 0.000294153$ $\sigma = 0.000333421$	$\mu = 0.004659973$ $\sigma = 0.003303187$	$\mu = 0.056124443$ $\sigma = 0.052853964$	$\mu = 0.878098810$ $\sigma = 0.807751477$	$\mu = 17.36093859$ $\sigma = 34.45523194$
Memoization	$\mu = 0.000127563$ $\sigma = 0.000039617$	$\mu = 0.000204747$ $\sigma = 0.000116545$	$\mu = 0.000274693$ $\sigma = 0.000071916$	$\mu = 0.000438000$ $\sigma = 0.000181180$	$\mu = 0.000641600$ $\sigma = 0.000176474$

(ii)







(iii)

First of all, Naive algorithm's linear graph has same issue like worst case graph, input sizing. That is why a $(x, log(y))$ graph is also provided. It can be observed that in logarithmic graph Naive algorithm acts like linear graph, which is expected, since log of 2^{m+n} is $(m+n)$ -acts like $\Theta(n)$. Thus, it can be derived that average running time of Naive algorithm is same as worst case -their graphs grows in same rate- and its experiment is not that far from theoretical result ($\Theta(2^{m+n})$). It can be said that the bound that is concluded is asymptotic. In addition to those, std. deviations shows that when input size becomes larger, the result becomes volatile, as well. Finally, running times are generally 1000 times smaller than worst case, as expected.

Secondly, Memoization's graph has again similarity with worst case's graph, it is growing almost linearly. Furthermore, it can be said that running times are again smaller than the worst cases, in general, but the difference level is not that hard as Naive algorithm.