Assignment 1 : CS301

Gizem Topsakal 27924

October 23, 2022

1. **Problem 1 (Recurrences (10 points))**

   (a) $\theta(n^3)$ by Master Theorem case 3

   (b) $\theta(n^{log_2 7})$ by Master Theorem case 1

   (c) $\theta(\sqrt{n} * logn)$ by Master Theorem case 2

   (d) $\theta(n^2)$ by Substitution Method

2. **Problem 2 (Longest Common Subsequence - Python)**

   (a) i. I think the worst-case running time would occur when the lengths of the string X (m) and the string Y (n) are the same and their components are completely different. Because when their components are completely different, their last characters will be always different and this way, the algorithm will make two recursive calls for every character as in the else statement.

   To compute the running time, we know that in the worst case the function will make 2 recursive calls for each character. So we should have 2 power of each character which is m+n. Now we have $\theta(2^{m+n})$. Since we said that the worst case is when length of each string is equal we can say that m = n. As a conclusion, the running time will be $\theta(2^{2n})$.

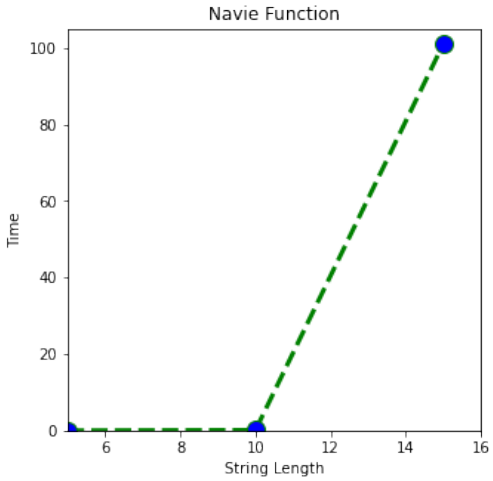
   ii. As far as I understand, with memoization, the function will contain the comparisons of each character in something like a matrix so this will cause an acceleration since our code will not have to compare things over and over again. Now, when we think of it like a 2D matrix, the worst case occur when the matrix has the most number of cells which is again the case of m = n. So the number of cells, which are the comparisons our code should do, is n^2. Since each comparison will take O(1) time, and we have n^2 cells; in total best asymptotic worst-case running time will be $\theta(n^2)$.

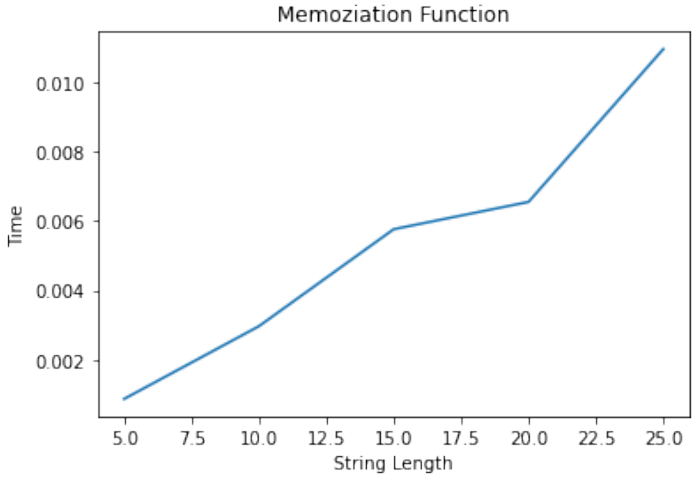
   (b) i.

Algorithm	m = n = 5	m = n = 10	m = n = 15	m = n = 20	m = n = 25
Naive	0.006235112	0.123475198	101.06170939	-	-
Memoziation	0.000865487	0.002964101	0.0057641109	0.006551045	0.0109635349

   Since the worst-case running time of the Naive algorithm is $\theta(2^{2n})$, which increases exponentially, I wasn't able to run the algorithm in Google Colab with the input sizes of 20 and 25 due to the extreme increase of the time complexity.

   Properties:
   - MacBook Pro (2019)
   - 2,3 GHz Intel Core i9
   - MacOS
   - 16 GB 2667 MHz DDR4
   - Intel UHD Graphics 630 1536 MB





iii. Like I stated in the running times, we can see that the worst-case running time of the Naive algorithm is $\theta(2^{2n})$. Since it grows so fast, we can see in the experimental results, there is a huge difference between input size 10 and 15. I couldn't even calculate 20 and 25 input size. This proves our theoretical results.
Since the Memoization algorithm grows polynomially, I was able to calculate input sizes 20 and 25 and get smaller seconds in comparison to the Naive algorithm. This means that the Memoization algorithm has more scalibity than the Naive algorithm.

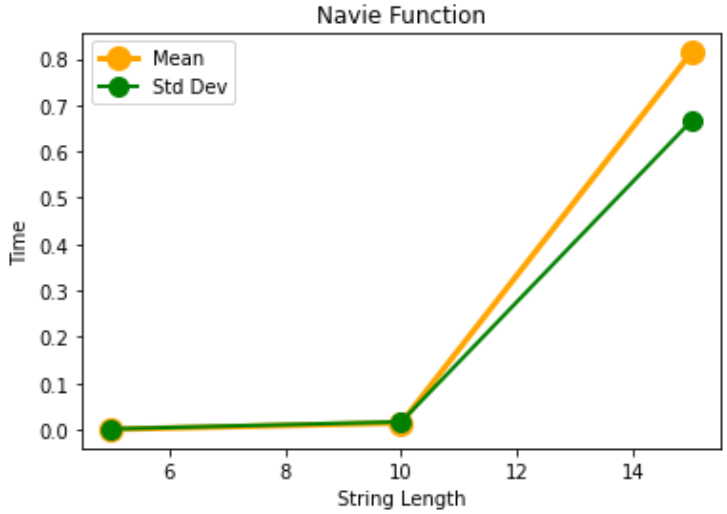
(c) i.

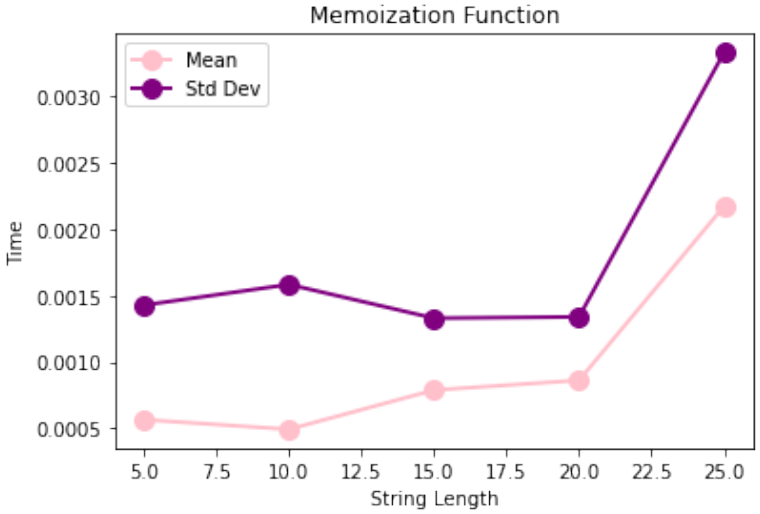
Algorithm	m = n = 5 (μ)	m = n = 5 (σ)	m = n = 10 (μ)	m = n = 10 (σ)
Naive	0.000603679	0.0014733368	0.01434096	0.0168203
Memoziation	0.000566445	0.0014283582	0.00049399	0.0015835

Algorithm	m = n = 15 (μ)	m = n = 15 (σ)	m = n = 20 (μ)	m = n = 20 (σ)
Naive	0.81348425	0.6658089	-	-
Memoziation	0.00078933	0.0013300	0.0008624863	0.00134002

Algorithm	m = n = 25 (μ)	m = n = 25 (σ)
Naive	-	-
Memoziation	0.0021725237	0.0033288936

ii.





iii. In this question, I generated two txt files, and inserted 30 randomly generated DNA sequences into the txt files. I used Google Colab to read the txt files and calcuta their means and standart deviations. Compared to the worst case results I found in the b section, these random running times were much better. It is because this time there were some common characters in most of the DNA sequences since they were generated randomly. So in this case the algorithm was not have to call 2 recursions every time. But still unfortunately, my machine was not able to run input sizes 20 and 25 with naive algorithm in this case either. Since sequences were generated randomly even if it has a so little possibility, I may get the worst case, or some case close to the worst case.
Since the Memoization's asymptotic running time is quadratic, I was able to collect the running times and they were still better than Naive algorithm and the worst case experimental results because in this case some sequences were containing common characters.