Ece Karasu
29467

Problem 1

a- $T(n) = \theta(n^3)$ → The Master Theorem - Case 3
b- $T(n) = \theta(n^{\log_2 7})$ → The Master Theorem - Case 1
c- $T(n) = \theta(\sqrt{n}(\log n))$ → The Master Theorem - Case 2
d- $T(n) = O(n^2)$ → The Substitution Method

Problem 2

(a)

(i) According to the function, the recursive relation is the following:

$$lcs(X,Y,i,j) = \begin{cases} 0 & \text{if } i = 0 \text{ or } j = 0 \\ lcs(X, Y, i-1, j-1) + 1 & \text{if } i,j > 0 \text{ and } X[i-1] = Y[j-1] \\ \max(lcs(X, Y, i, j-1), lcs(X, Y, i-1, j)) & \text{otherwise.} \end{cases}$$

If we do the max operation in each iteration where the maximum number of iterations is the maximum number of subsequences of the longer word, we would find the worst-case running time of the algorithm.

If $n > m$ → maximum # of iterations = 2^n
If $m > n$ → maximum # of iterations = 2^m

We also have the max operation with a cost of $O(k)$. (iterates every time in the worst-case scenario)
In total, the asymptotic worst-case running time of the algorithm → **$O((2^n)*k)$**

(ii) According to the function, we have a 2D array where m and n are the numbers of rows and columns. In the worst-case scenario, each cell of the array should be visited. This operation would cost $O(m*n)$. Also, we have the max operation as in the previous algorithm. So, in total the asymptotic worst-case running time → **$O(m*n*k)$**

Ece Karasu
29467

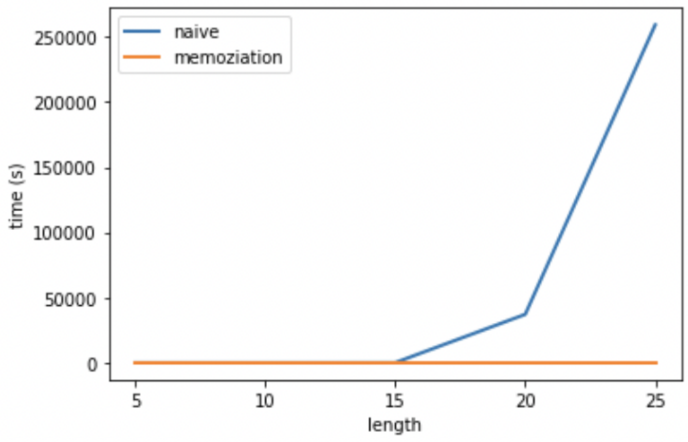
(b)

(i)

values are in seconds , M1 chip, 16GB memory, MacOS Big Sur 11.4

Algorithm	m = n = 5	m = n = 10	m = n = 15	m = n = 20	m = n = 25
Naive	7.58e-05	9.96e-02	4.17e+01	37082.75	3+ days
Memoziation	1.81e-05	4.91e-05	1e-04	1.9e-04	2.71e-04

(ii)



(iii) According to the graph results, the running time of the naive algorithm grows exponentially and the running time of the algorithm with memoization grows linearly. So, experimental results confirm the theoretical results I found in (a).

Ece Karasu
29467

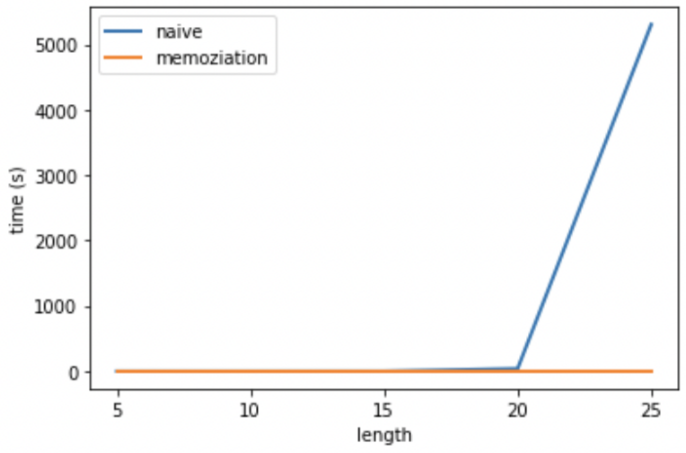
(c)

(i)

values are in seconds

	m=n=5	m=n=5	m=n=10	m=n=10	m=n=15	m=n=15	m=n=20	m=n=20	m=n=25	m=n=25
Alg.	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
Naive	1.84e-05	9.24e-06	0.004	0.004	0.15	0.09	38.29	23.57	5304.4	5491.28
Memo.	8.3e-05	0.0001	0.0004	0.0006	6.98e-05	7.31e-06	0.0001	1.03e-05	0.0001	1.1e-05

(ii)



(iii) As the input length increases, the average running time of the naive algorithm grows sharper than its running time. However, we cannot say the same thing for the algorithm with memoization, the same linearity continues for the average running time. To be more general, both algorithms does not change their behaviors when it comes to the average running times (i.e., exponential and linear).