Homework 1

Ege Aktemur

October 2022

Question 1

(a) $T(n) = 2T(n/2) + n^3$
Answer: $\theta(n^3)$

(b) $T(n) = 7T(n/2) + n^2$
Answer: $\theta(n^{\log_2 7})$

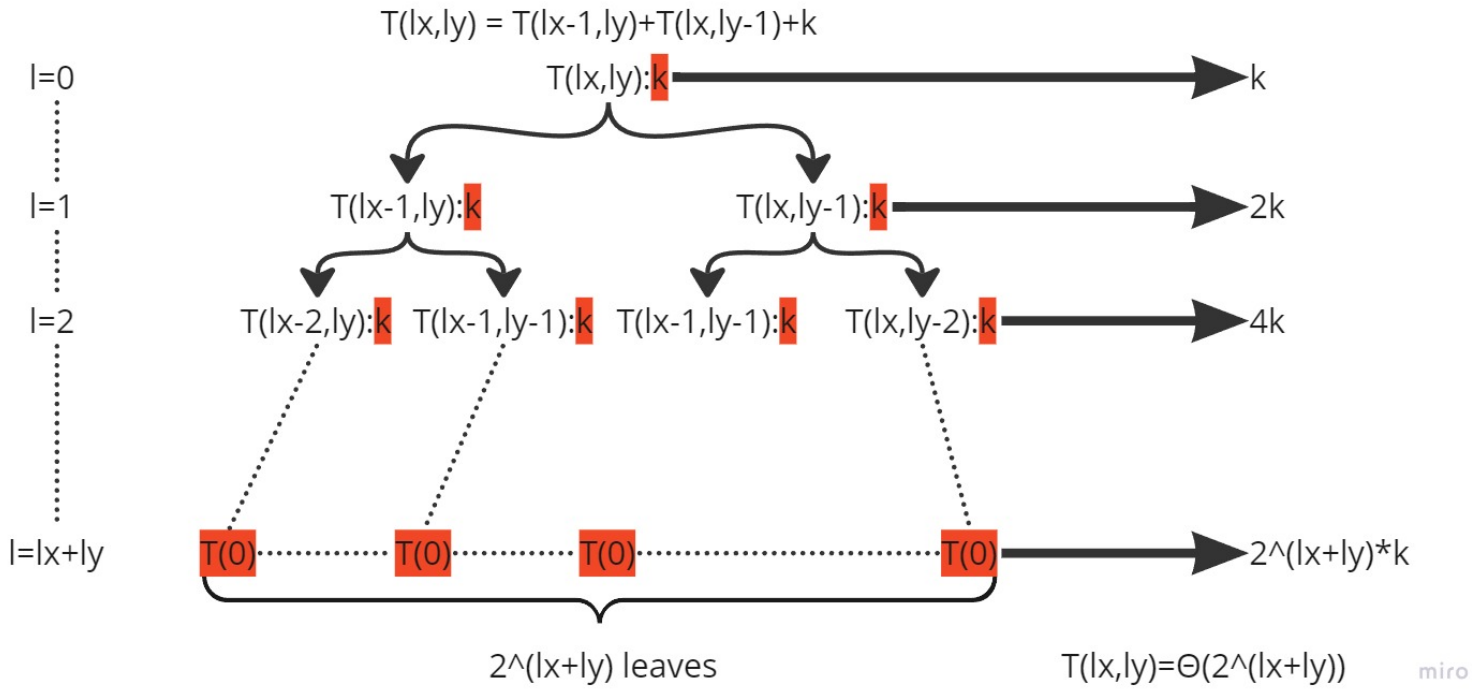
(c) $T(n) = 2T(n/4) + \sqrt{n}$
Answer: $\theta(\sqrt{n} * \log n)$

(d) $T(n) = T(n-1) + n$
Answer: $\theta(n^2)$

Question 2

Question 2a1 Analysis of Naive Algorithm

There are 2 conditions that makes naive algorithm running in worst case. Firstly, there must not be any common characters between the two strings since it increases the amount of recursions. Second condition is their length should be the same since it also increases the amount of recursive calls.
Recursive function of T(lx,ly) = T(lx-1,ly) + T(lx,ly-1) + O(1) where lx and ly are lengths of x and y respectively.
To solve this recursion, recursion tree method will be used and k constant will be used for 0(1) for simplicity.



Therefore, complexity of the Naive Algorithm is $T(lx, ly) = \theta(2^{lx+ly})$.

Question 2a2 Analysis of Memoization Algorithm

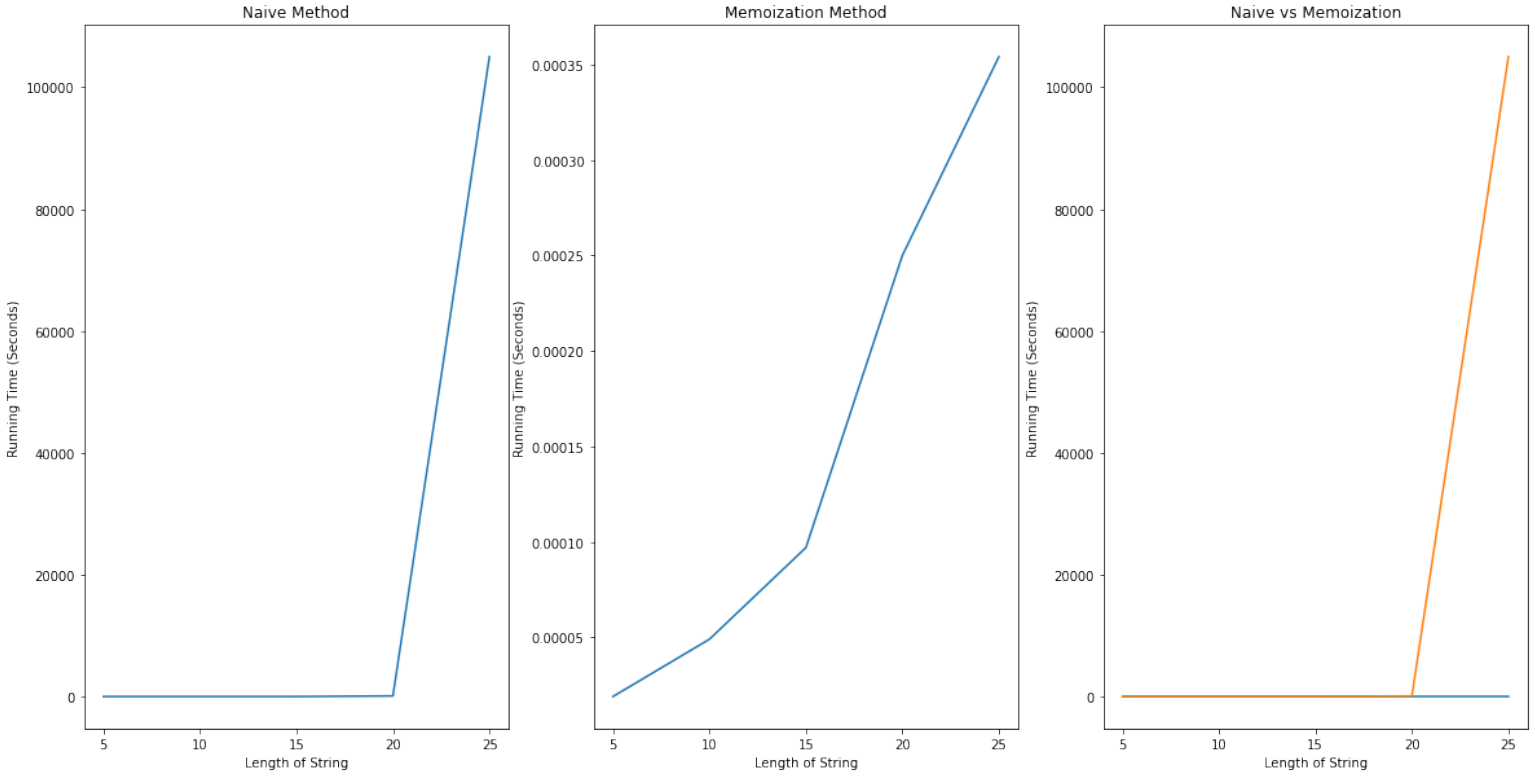
Memoization Algorithm also runs more recursive calls when there is no common characters between them and it also gets closer to worse case when two words has the same length.
Complexity of T(lx,ly) = θ(lx,ly) where lx and ly are lengths of x and y respectively. Because in the worst case algorithm must look to the every element of the "c" table in the algorithm which is sized (lx+1)(ly+1).

Question 2b1 Table of Running Time of the Algorithms

I ran the tests on machine that has i7-9750H, 16 gb RAM and SSD. This machine was running windows and I ran the tests from the cmd using python. I approximated results for Naive Method when m=n=25 because I could not get the results for them.

Algorithm	m=n=5	m=n=10	m=n=15	m=n=20	m=n=25
Naive	0.000019	0.001330	0.185070	104.314006	105000
Memoization	0.000019	0.000049	0.000097	0.000250	0.000354

Question 2b2 Graph of Running Time of the Algorithms



Question 2b3 Discussion of Results

Naive Algorithm

Even though I could not calculate the running time of m=n=25 and approximated it from testing, it can be clearly seen that algorithm is running in exponential time. This proves that complexity of the Naive Algorithm is $T(lx, ly) = \theta(2^(lx + ly))$ where lx and ly is length of x and y respectively.

Memoization Algorithm

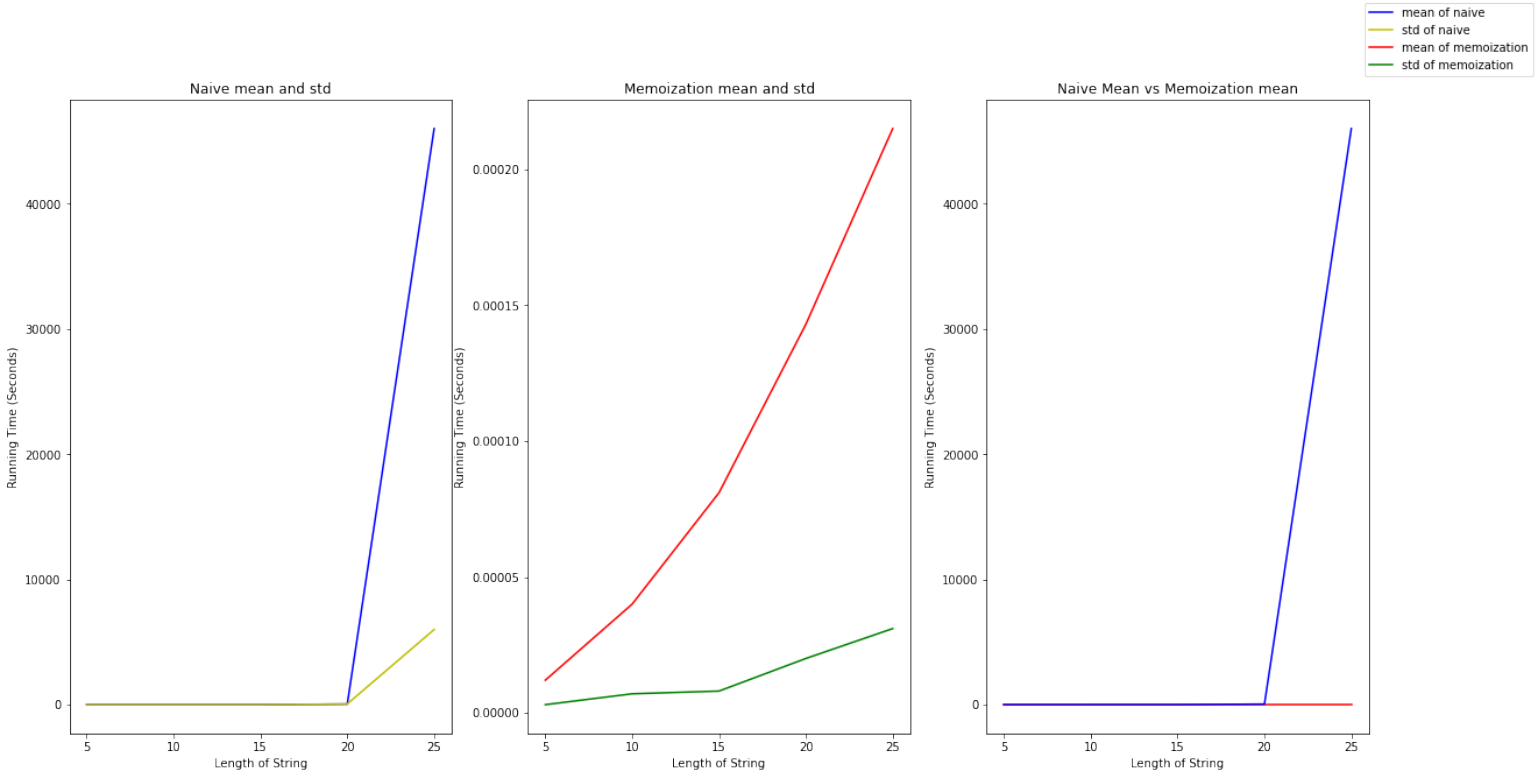
Compared to Naive Algorithm, Memoization Algorithm grows so much slower and it is more scalable. Since the values of the Memoization Algorithm grows simalr to second degree polynomials it proves my guess which is $T(lx, ly) = (2^(lx * ly))$ where lx and ly is length of x and y respectively.

Question 2c1 Table of Running Time of the Algorithms

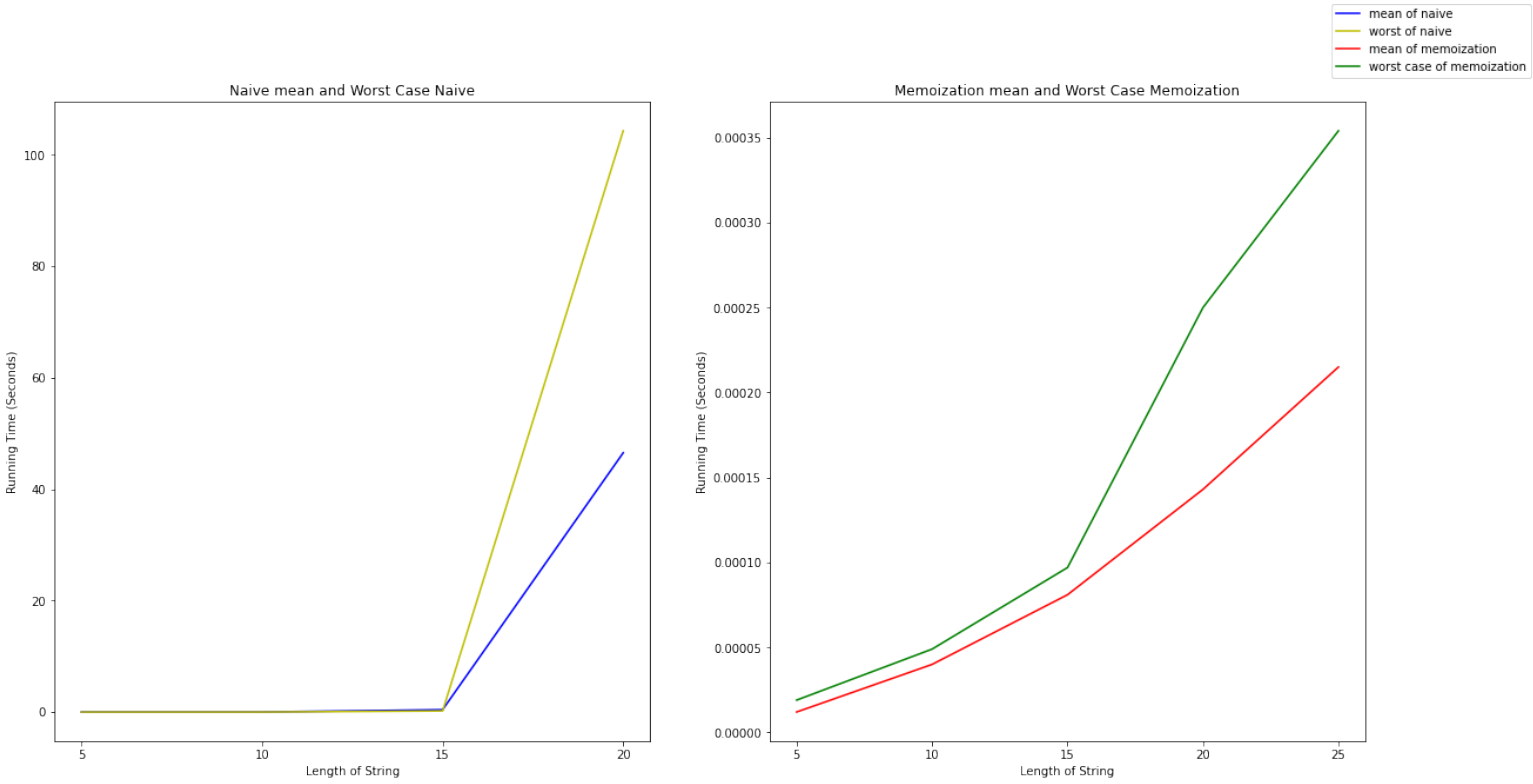
I ran the tests on machine that has i7-9750H, 16 gb RAM and SSD. This machine was running windows and I ran the tests from the cmd using python. I approximated results for Naive Method when m=n=25 because I could not get the results for them.

Algorithm	m=n=5		m=n=10		m=n=15		m=n=20		m=n=25	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
Naive	0.000027	0.000020	0.002247	0.003306	0.414240	0.639204	46.528692	57.596478	46000	6000
Memoization	0.000012	0.000003	0.000040	0.000007	0.000081	0.000008	0.000143	0.000020	0.000215	0.000031

Question 2c2 Graph of Running Time of the Algorithms



Question 2c3 Discussion of Results



Naive Algorithm

While Naive Algorithm still runs in exponential time, worst case of Naive algorithm grows more quicker compared to average scenario which can be seen in the graph above. Reason of this is the reduced recursive calls caused by the occurrence of common characters and subsequences which does not exist in worst case.

Memoization Algorithm

We can see the the differences of worst case versus average scenario more clearly in the Memoization Algorithm which is again caused by the reduced recursive calls from existence of common characters and subsequences. Additionally we can still observe the polynomial behavior of the algorithm.