CS301 Assignment 1 Asnwers

Uğur Öztunç 28176

Problem 1

(a) = $\theta(n^3)$

(b) = $\theta(n^{\log_2 7})$

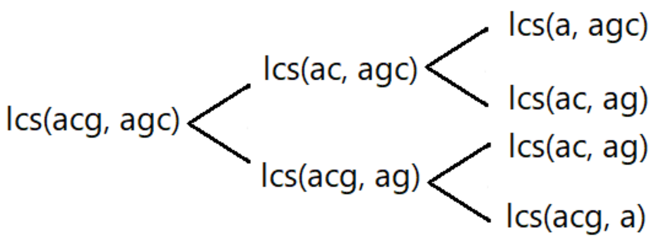
(c) = $\theta(\sqrt{n}\log_2 n)$

(d) = $\theta(n^2)$

Problem 2

a)

(i) In this case all substrings of both strings needed to be found and each substring must be compared with the second string. Finding all substrings will take $O(2^n)$ time. As for the comparison part, it will depend on the length of the second string; however, $O(2^n)$ is clearly predominant against comparison time. As seen in the figure below after each comparison that there is no match, run time is doubled. Thus, the answer is $O(2^n)$.



(ii) With memoization algorithm instead of wasting time by finding the results of exact same sub-problems encountered, once the result of a sub-problem is found, it is recorded into a 2D matrix. In cases where the same sub-problem is encountered again, the result is called and used without losing time, since the necessary information is in this matrix. If we call how many rows and columns does the matrix have as 'n' and 'm' respectively, the best asymptotic worst-case running time of the algorithm is $O(n * m)$.

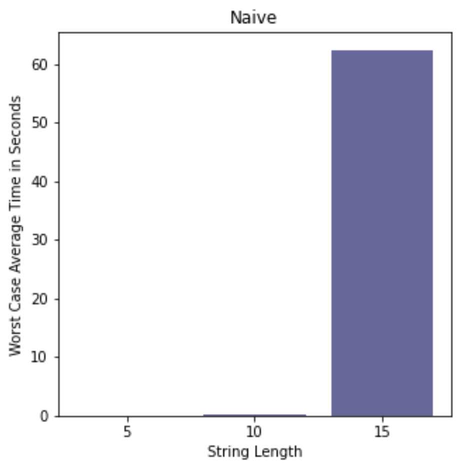
b)

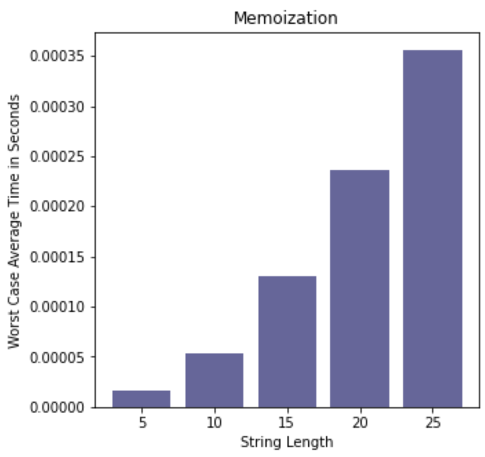
(i) CPU: Intel Core i5-8265U 1.60Ghz 8 cores, RAM: 8GB 2133Mhz, OS: Windows 10 Home 64bit

Algorithm	m=n=5	m=n=10	m=n=15	m=n=20	m=n=25
Naive	0.000109704732895	0.078989020983378	62.33158675829569	n/a	n/a
Memoization	0.000016616980200	0.000053298473400	0.000129844347636	0.000235924720764	0.000355798403422

!- It took so long that it was unable to record a proper average run time for length 20 and 25 with naive algorithm, therefore there are no entries for them.

(ii)





(iii) Based on the results above, it cannot be said that there is a significant difference between the average times of the two algorithms with very short strings, such as with length of 5. However, it is clear that the scalability of naive algorithm is far more poor than memoization, since as seen on graphs the average time of naive algorithm increases quite exponentially as the string length increases, while the average time of memoization algorithm shows more stable and proportional increase as the string length increases. Ultimately, it can be said that experimental findings correlate with theoretical ones: $O(n * m)$ is much better than $O(2^n)$.

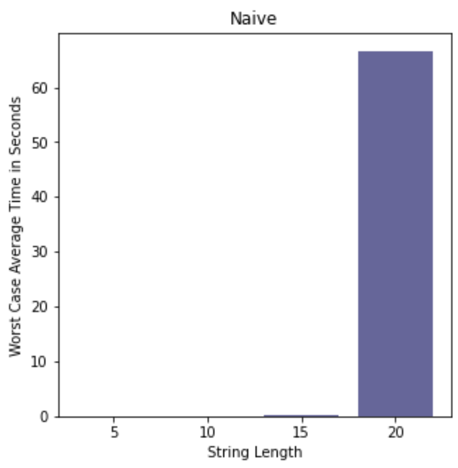
c)

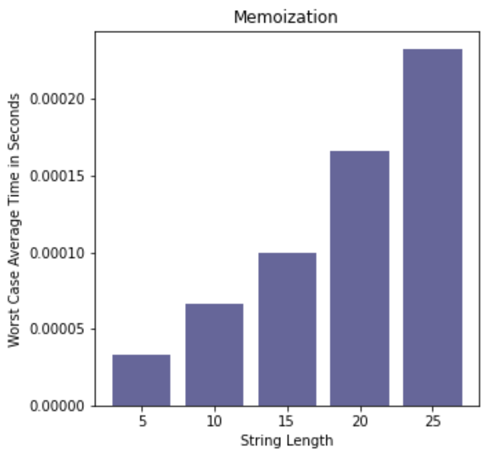
(i) CPU: Intel Core i5-8265U 1.60Ghz 8 cores, RAM: 8GB 2133Mhz, OS: Windows 10 Home 64bit

	Naive		Memoization	
	µ	σ	µ	σ
m=n=5	0.00003324349721	0.00018208213313	0.00003321965535	0.00018195154590
m=n=10	0.00432176589965	0.00419900419057	0.00006647109985	0.00025296407205
m=n=15	0.29873379866282	0.20705577737426	0.00009977022807	0.00030442749182
m=n=20	66.5791403532028	95.0883013119088	0.00016627311707	0.00037819236356
m=n=25	n/a	n/a	0.00023266474405	0.00042912852067

!- It took so long that it was unable to record a proper average run time for length 25 with naive algorithm, therefore there is no entry for it.

(ii)





(iii) In general both algorithms showed better results as expected, since there are no worst cases in randomly generated DNA sequences. However, it is clear that the naive algorithm is still not as scalable as memoization algorithm, especially at string lengths higher than 10. The rate of change in the increase in the running times of the two algorithms with the increasing string lengths is similar to the one in the worst case: Naive algorithm increases highly exponential, while memoization algorithm increases proportional.