# CS301 Assignment 1

Alpay Naçar

10.16.2023

## 1 Problem 1

1. $T(n) = 2T\left(\frac{n}{2}\right) + n^3 = \Theta(n^3)$
2. $T(n) = 7T\left(\frac{n}{2}\right) + n^2 = \Theta(n^{\log_2(7)})$
3. $T(n) = 2T\left(\frac{n}{4}\right) + n = \Theta(\sqrt{n} \cdot \log(n))$
4. $T(n) = T(n-1) + n = \Theta(n^2)$

## 2 Problem 2

### 2.a

1. $O(2^{n+m})$, because to get worse case we need to run else part (else part has higher complexity than if and elif parts), and when we assume that the algorithm always gets in else part, it becomes, T(m+n) = 2T(m+n-1) + c, (c part is from i==0, j==0, X[i-1]==Y[i-1], and max) using induction, we can prove that the best worst time complexity is $O(2^{n+m})$.
2. O(n*m), because with memorization we don't calculate same results again and again, instead we store them and use them when we need them again. (for(1 to n) * for(1 to m), equals to O(n*m))

### 2.b

1. 

| **Algorithm** | **m=n=4** | **m=n=8** | **m=n=12** | **m=n=16** | **m=n=18** |
|---|---|---|---|---|---|
| **Naive** | 2.4e-5 | 4e-3 | 8e-1 | 1.8e2 | 2.7e3 |
| **Memoization** | 7.3e-7 | 1.5e-6 | 2.9e-6 | 5e-6 | 1.1e-5 |

   Properties of my machine
   - CPU: AMD Ryzen 7 5800H
   - RAM: 16 GB
   - OS: Windows11
   - IDE: VS Code
   - Python version: 3.11.5

2. 

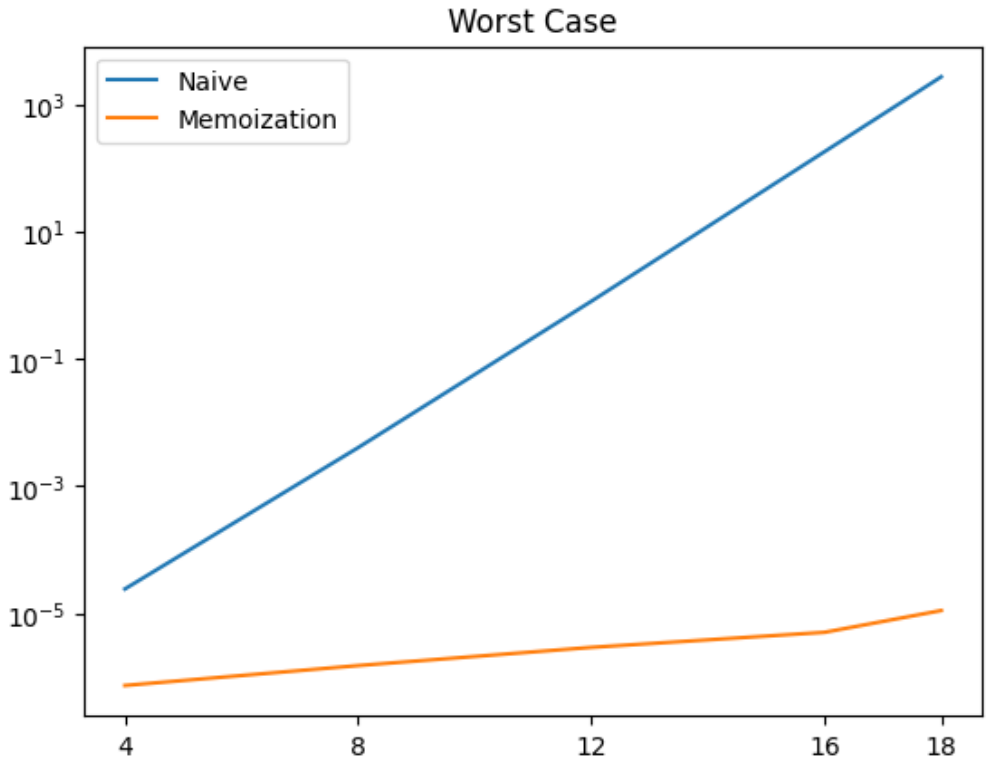


Figure 1: Worst Case

3.
- When we take a look at the plot, we can say that our theoretical expectations align with the experimental results. We can see that complexity of naive approach is logarithmic (be aware of the fact that the plots yscale is 'log'), and it aligns with our expectation of $O(2^{n+m})$. Memoization seems similar to logarithmic but when we zoom in, it is actually exponential, which is what we found in the theoretical part.
- Also, we can see that the expectation time of naive implementation increases much faster than the memoization solution, as expected. In conclusion, memorization is more scalable, because it shows only a small increase when we try larger input sizes, compared to naive solution.

## 2.c

1.

| Algorithm | m=n=4 | | m=n=8 | | m=n=12 | | m=n=16 | | m=n=18 | |
|---|---|---|---|---|---|---|---|---|---|---|
| | $\mu$ | $\sigma$ | $\mu$ | $\sigma$ | $\mu$ | $\sigma$ | $\mu$ | $\sigma$ | $\mu$ | $\sigma$ |
| Naive | 9e-6 | 5e-6 | 3e-4 | 2.6e-4 | 1.4e-2 | 1.1e-2 | 7.2e-1 | 6.9e-1 | 7.5 | 6.5 |
| Memoization | 5.6e-6 | 2.2e-6 | 2e-5 | 6.5e-6 | 4.8e-5 | 9.4e-6 | 6.7e-5 | 7.5e-6 | 1.2e-4 | 2.1e-5 |

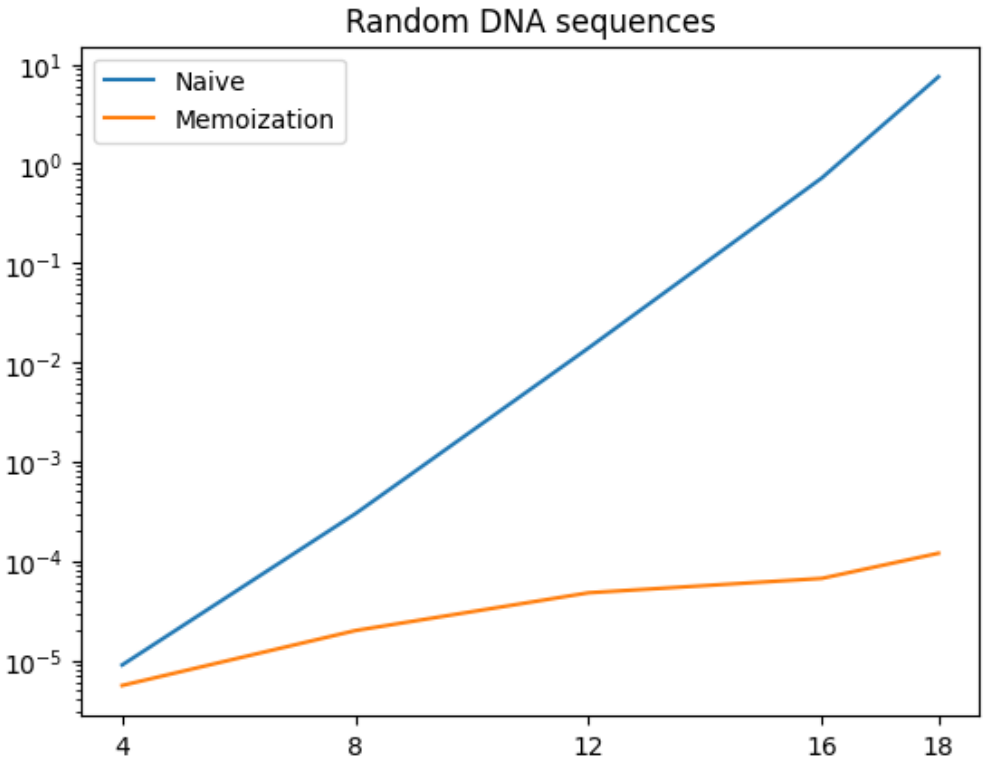


Figure 2: Average Case

2.

3.
   - Average running times show that both of these algorithms run significantly faster than their worst-case scenarios, and when the input size increases, difference between worst-case and average-case increases.
   - In conclusion, (I compared them by putting two visuals on the same plot, and) average running times are significantly low compared to worst-case running times.