Mohamed AbdelHalem_1180182

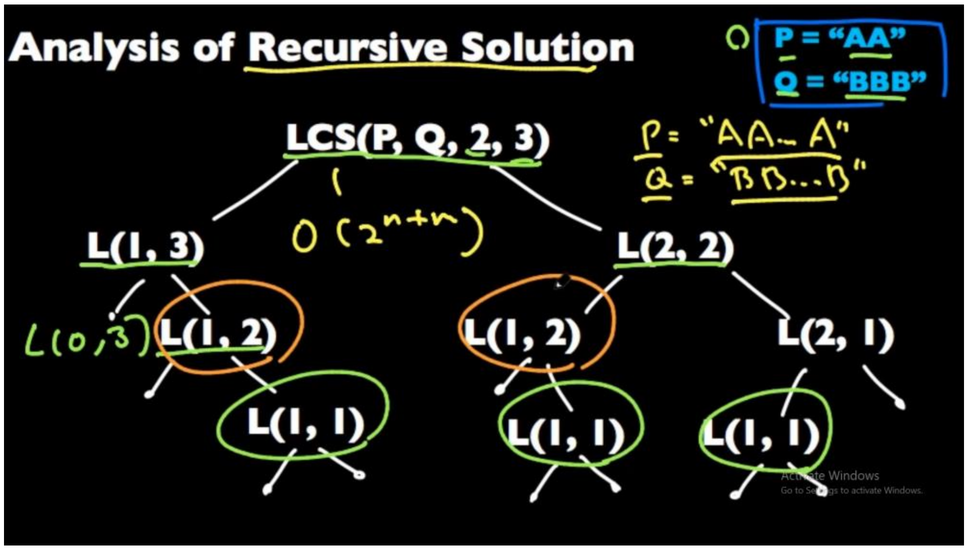
Assignment_4

Dynamic Programming Longest common subsequence:

The way I solved this problem:

On understanding Dynamic Programming I first thought of a recursive solution which depends on checking if last array items match if they do increase length by one if not get maximum of the last items in the array and on drawing the recursion tree we find repeated values that is being calculated so we need to store one of them and don't calculate each time and then use bottom up solution which is the dynamic programming solution which depends on the same idea which is the certation of the 2D array LCS which contains lengths that is being calculated to find the maximum common sub sequence the way it works is by first putting zeros in the first column and row and then check if two items match if so increase next index in the 2D array by one we found 1 match and if we don't we get maximum from array and store in current indices which is in first time if not found are zeros and after finding 1 match is 1 and so on we keep doing that N*M times until we found LCS

Recursion Tree:



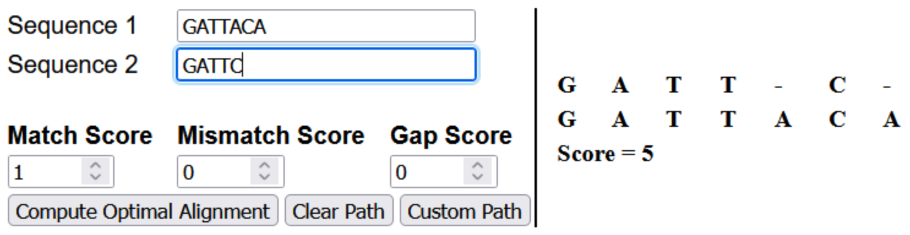
AS we can see a lot of values are being calculated more than one time which leads to a very long time and complexity is O(2^(N*M))

Pseudo Code:

A is the input array and b is the 2nd array

```
LCS(a,b)
M,N=len(a),len(b)
Create a 2D array of Size M+1*N+1>>LCS
For loop on the size of the 2D array with I and j
If I==0 or j==0
Put zeros in the first row and column of the 2D array
Else if a[I-1] ==b[j-1]:
Add one to the previous value in the LCS array and put it in current I and j
Else:
    Get previous maximum value and put in current I and j in LCS array
Return LCS[M,N] which is the longest common sequence
```

Algorithm Tracing :

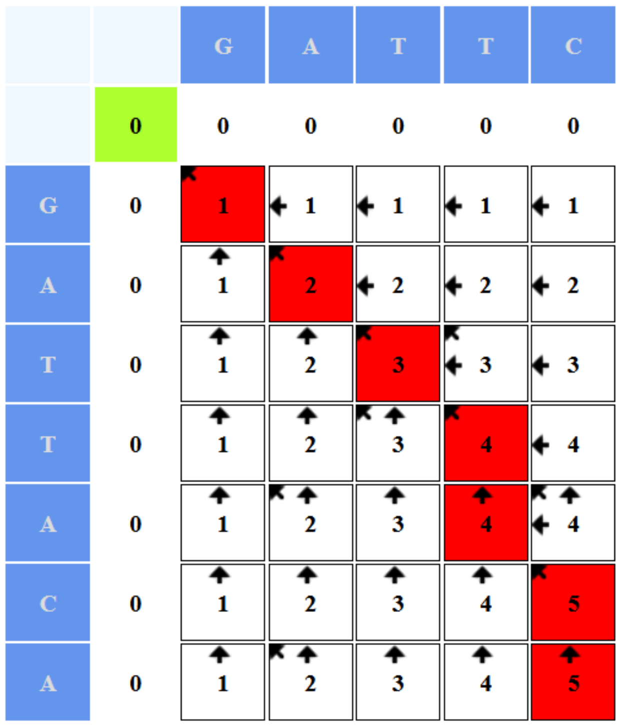


Sequence 1: GATTACA

Sequence 2: GATTC

Match Score: 1 | Mismatch Score: 0 | Gap Score: 0

```
G   A   T   T   -   C   -
G   A   T   T   A   C   A
Score = 5
```



		G	A	T	T	C
	0	0	0	0	0	0
G	0	↖ 1	← 1	← 1	← 1	← 1
A	0	↑ 1	↖ 2	← 2	← 2	← 2
T	0	↑ 1	↑ 2	↖ 3	↖ ← 3	← 3
T	0	↑ 1	↑ 2	↖ ↑ 3	↖ 4	← 4
A	0	↑ 1	↖ ↑ 2	↑ 3	↑ 4	↖ ↑ ← 4
C	0	↑ 1	↑ 2	↑ 3	↑ 4	↖ 5
A	0	↑ 1	↖ ↑ 2	↑ 3	↑ 4	↑ 5

Here it shows how we start with putting zeros and then put 1 as G=G and 2 as A=A and 3 as T=T and 4 as T=T and once reached A and T it put 4 again which is previous maximum

Complexity: O(N*M)

Sources used in this problem: https://www.youtube.com/watch?v=Qf5R-uYQRPk

Image source: https://bioboot.github.io/bimm143_W20/class-material/nw/