Lab Exercise – 6 – Dynamic Programming – Longest Common Subsequence Problem

Solve the following problems:-

1. A bioinformatics researcher is studying DNA sequences to identify genetic similarities between different species. Given two DNA sequences, the researcher needs to determine the longest sequence of nucleotides (A, T, C, G) that appears in both sequences while maintaining the original order.

   Write a memoized recursive algorithm to compute the Longest Common Subsequence (LCS) of the given DNA sequences.
   Output: Function should output the LCS Sequence and its Length

   Test Case 1:
   Given DNA Sequences:
   S_1= ACCGGTCGAGTGCGCGGAAGCCGGCCGAA
   $S_{2\,=}$ GTCGTTCGGAATGCCGTTGCTCTGTAAA
   LCS is GTCGTCGGAAGCCGGCCGAA
   Length of LCS : 20

   Test Case 2:
   Given **DNA Sequences:**
   X = "ACGTGCA"
   Y = "ACTGC"
   The **LCS** is "ACTGC",
   Length: 5

   Test Case 1:

   ```
   29
   A
   C
   C
   G
   G
   T
   C
   G
   A
   G
   T
   G
   C
   G
   C
   G
   ```

```
G
A
A
G
C
C
G
G
C
C
G
A
A
28
G
T
C
G
T
T
C
G
G
A
A
T
G
C
C
G
T
T
G
C
T
C
T
G
T
A
A
A
```

Output:

```
G
T
C
G
T
C
```

```
G
G
A
A
G
C
C
G
G
C
C
G
A
A
20
```

Test Case 2:
Given DNA Sequences:
X = "ACGTGCA"
Y = "ACTGC"

```
7
A
C
G
T
G
C
A
5
A
C
T
G
C
```

Output: The **Longest Common Subsequence (LCS)** is "ACTG", with length **4**

```
A
C
T
G
4
```

(Q2 is PPS 6)

2. A **financial analyst** is analyzing the stock prices over a period to identify the **longest duration of consistent price increase**. Given an array of stock prices, implement an algorithm to compute the **longest increasing trend**.

Example:
Stock prices over 9 days:
`[10, 22, 9, 33, 21, 50, 41, 60, 80]`
The **longest increasing period** lasted **6 days**, and the sequence was `[10, 22, 33, 50, 60, 80]`.

Write an efficient algorithm to determine the longest duration of stock price increase.

Test Case 1: X = [10, 22, 9, 33, 21, 50, 41, 60, 80]
Output:
Longest Increasing Subsequence = [10, 22, 33, 50, 60, 80]
Longest Increasing Subsequence (LIS) Length = 6
Test Case 2: [3, 10, 2, 1, 20, 4, 6, 9, 7, 30]
Output:
Longest Increasing Subsequence = [3, 10, 20, 30]
Longest Increasing Subsequence (LIS) Length = 4

Explanation:
Given Sequence:
X = [10, 22, 9, 33, 21, 50, 41, 60, 80]
Initial State:

- LIS = [1, 1, 1, 1, 1, 1, 1, 1, 1]
- prev = [-1, -1, -1, -1, -1, -1, -1, -1, -1] (All set to -1 initially)

Building `LIS[]` and `prev[]` Arrays Iteratively

Index i	**Value X[i]**	**LIS Computation**	**Updated LIS[i]**	**Updated prev[i]**
0	**10**	Base case	**1**	-1
1	**22**	22 > 10 → LIS[1] = LIS[0] + 1	**2**	0
2	**9**	No increasing subsequence	**1**	-1
3	**33**	33 > 10 → LIS[3] = LIS[0] + 1 = 2	**3**	1
		33 > 22 → LIS[3] = LIS[1] + 1 = **3**		
4	**21**	21 > 10 → LIS[4] = LIS[0] + 1 = **2**	**2**	0
5	**50**	50 > 10 → LIS[5] = LIS[0] + 1 = 2	**4**	3
		50 > 22 → LIS[5] = LIS[1] + 1 = 3		
		50 > 33 → LIS[5] = LIS[3] + 1 = **4**		
6	**41**	41 > 10 → LIS[6] = LIS[0] + 1 = 2	**4**	3
		41 > 22 → LIS[6] = LIS[1] + 1 = 3		
		41 > 33 → LIS[6] = LIS[3] + 1 = **4**		
7	**60**	60 > 10 → LIS[7] = LIS[0] + 1 = 2	**5**	5
		60 > 22 → LIS[7] = LIS[1] + 1 = 3		
		60 > 33 → LIS[7] = LIS[3] + 1 = 4		
		60 > 50 → LIS[7] = LIS[5] + 1 = **5**		
8	**80**	80 > 10 → LIS[8] = LIS[0] + 1 = 2	**6**	7
		80 > 22 → LIS[8] = LIS[1] + 1 = 3		

		80 > 33 → LIS[8] = LIS[3] + 1 = 4		
		80 > 50 → LIS[8] = LIS[5] + 1 = 5		
		80 > 60 → LIS[8] = LIS[7] + 1 = **6**		

Final LIS[] and prev[] Arrays:

Index i	Value X[i]	LIS[i]	prev[i] (Previous Index)
0	**10**	**1**	-1 (No previous element)
1	**22**	**2**	0 (Comes after 10)
2	**9**	**1**	-1 (No previous element)
3	**33**	**3**	1 (Comes after 22)
4	**21**	**2**	0 (Comes after 10)
5	**50**	**4**	3 (Comes after 33)
6	**41**	**4**	3 (Comes after 33)
7	**60**	**5**	5 (Comes after 50)
8	**80**	**6**	7 (Comes after 60)

Final Step : Reconstruct LIS Using prev[]

1. **Find the maximum LIS value** → max(LIS) = 6, found at index 8
2. **Trace back using prev[]**
   - **80 (index 8)** → prev[8] = 7
   - **60 (index 7)** → prev[7] = 5
   - **50 (index 5)** → prev[5] = 3
   - **33 (index 3)** → prev[3] = 1
   - **22 (index 1)** → prev[1] = 0
   - **10 (index 0)** → prev[0] = -1 (starting point)

Final LIS sequence (Reversing the order from backtracking):
[10, 22, 33, 50, 60, 80]

Final Answer

- **Longest Increasing Subsequence = [10, 22, 33, 50, 60, 80]**
- **Longest Increasing Subsequence (LIS) Length = 6**

Test Case 1: X = [10, 22, 9, 33, 21, 50, 41, 60, 80]

```
9
10
22
9
33
21
50
41
60
80
```

Output:

```
10
22
33
50
60
80
6
```

Test Case 2: [3, 10, 2, 1, 20, 4, 6, 9, 7, 30]

```
10
3
10
2
1
20
4
6
9
7
30
```

Output:

Longest Increasing Subsequence = [3, 10, 20, 30]

Longest Increasing Subsequence (LIS) Length = 4

```
3
10
20
30
4
```