**Department of Computer Science and Engineering**

|  |  |
| --- | --- |
| **Course Code: CSE221** | **Credits: 1.5** |
| **Course Name: Algorithms** | **Semester: Fall’18** |

**Lab 08  
Longest Common Subsequence**

1. **Topic Overview:**

Given two sequences, the students are required to find the length of longest subsequence present in both of them. A subsequence is a sequence that appears in the same relative order, but not necessarily contiguous. For example, “abc”, “abg”, “bdf”, “aeg”, ‘”acefg”, .. etc are subsequences of “abcdefg”. So a string of length n has 2^n different possible subsequences.

It is a classic computer science problem, the basis of [diff](http://en.wikipedia.org/wiki/Diff) (a file comparison program that outputs the differences between two files), and has applications in bioinformatics.

1. **Lesson Fit:**

To solve this problem, the students must have a basic idea on the following concepts:

* 1. Dynamic Programming
  2. Overlapping Subproblem
  3. Optimal Substructure

1. **Learning Outcome:**

After this lecture, the students will be able to:

* 1. Learn how to use dynamic programming to solve problems that have overlapping subproblem property. Dynamic Programming is mainly used when solutions of same subproblems are needed again and again. In dynamic programming, computed solutions to subproblems are stored in a table so that they don’t have to be recomputed. So, Dynamic Programming is not useful when there are no common (overlapping) subproblems because there is no point in storing the solutions if they are not needed again. Since, Longest Common Subsequence (LCS) problem satisfies overlapping subproblem property, dynamic programming will be particularly useful here. The students can avoid recomputations of subproblems by constructing a temporary 2-dimensional array to store solutions of subproblems.
  2. Learn how to manipulate optimal substructure property using a bottom up Tabulation method. The subproblems will be solved in the form of progressively finding out all the common subsequences of the two given sequences. That is, at first, the smallest possible common subsequence of the two sequences will be computed. Then gradually the subsequence length will increase and at the end, we will have the longest common subsequence.

1. **Anticipated Challenges and Possible Solutions**
   1. The students might fail to grasp the notion of the Tabulation method to gradually build up from smaller subproblems to solving for the final longest common subsequence.
   2. Backtracking from the end is required to find the longest common subsequence. The path of the backtracking can become confusing to the students.

**Solutions:**

1. The students will need to think in terms of the optimal substructure to overcome this challenge.
2. The students will be required to remember three directional rules and create a direction array in order to follow the correct path along the optimal subproblem solutions.
3. **Acceptance and Evaluation**

Students will show their progress as they complete each problem. They will be marked according to their class performance. There may be students who might not be able to finish all the tasks, they will submit them later and give a viva to get their performance mark. The mark distribution for the lab will be as follows:

Code: 05

Viva: 05

1. **Activity Detail**
   1. **Hour: 1  
      Explanation:**The teachers will explain the basic algorithm of Longest Common Subsequence problem. The teachers will discuss how to use a 2 dimensional array to store solutions of optimal subproblems and then use it to get solutions of other subproblems.
   2. **Hour: 2**

**Implementation:**

After explanation, the students will try to implement the concept of Longest Common Subsequence problem to solve a given problem.

**Problem Task:**

* + 1. Task 1 (Page 4-5)
  1. **Hour: 3**

**Evaluation:**

The teachers will check the status of the assigned tasks.

1. **Home tasks**
   1. Unfinished tasks

**Lab 8 Activity List**

**Task 1: Longest Common Subsequence**

**Problem Statement:**

Given two sequences, find the length of the longest subsequence present in both of them. A subsequence is a sequence that appears in the same relative order, but not necessarily contiguous.

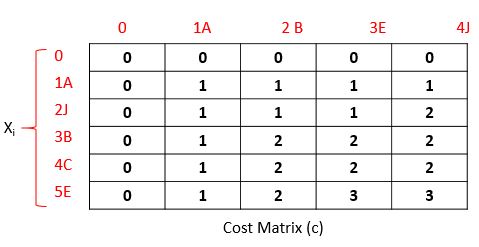
**Examples:**

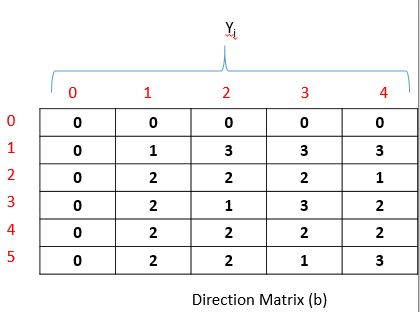
1. LCS for input sequences “ABCDGH” and “AEDFHR” is “ADH”
2. LCS for input sequences “HUMAN” and “CHIMPANZEE” is “HMAN”

**Sample Input**

Sequence 1: “ABEJ”

Sequence 2: “AJBCE”

**Sample Output:**

1. Cost Matrix:
2. Direction Matrix:
3. - Diagonal
4. - Up
5. - Left
6. Longest Common Subsequence: “ABE”
7. Length of LCS: 3