CS21003: Algorithms-I (Theory)
Tutorial – 4 (Dynamic Programming)
Date: 13-February-2020

1. (*Edit-Distance problem*) Given two strings S_1 of n_1 characters and S_2 of n_2 characters, the problem is to modify S_1 and convert it into S_2 through a sequence of character edits. The character edits can be *insertion*, *deletion* and *overwrite*. Each character edit takes unit amount of time. Assume that, all the characters belong to the same alphabet set. Your task is to devise an algorithm that minimizes the number of edits (and hence the time) required.

   (i) Propose a recursive definition for finding the number of edits required.

   (ii) Convert the above definition into a recursive algorithm to solve the above problem. What will be the time complexity? Are there overlapping subproblems?

   (iii) Can you use memoization to reduce the time-complexity? Present the memoized algorithm to solve the same. What will be your space-complexity (additional space required to memoize)?

   (iv) Design an efficient iterative algorithm that minimizes the number of edits required. What is the time and space complexity of your iterative solution?

2. (*Longest Ascending Subsequence problem*) Given an array $A[1..n]$ of natural numbers, the problem is to determine the longest ascending subsequence of $A[\,]$. A subsequence of an array $A[\,]$ is defined as a list $A[i_1], A[i_2], \ldots, A[i_m]$ for some $1 \leq i_1 < i_2 < \cdots < i_m \leq n$. The value m is called the length of the subsequence. Such a subsequence is called ascending if $A[i_1] \leq A[i_2] \leq \cdots \leq A[i_m]$. Your task is to devise an algorithm that finds the length along with the longest ascending subsequence from the given array.

   (i) Propose a recursive definition for finding the length of the longest ascending subsequence.

   (ii) Convert the above definition into a recursive algorithm to solve the above problem. What will be the time complexity? Are there overlapping subproblems?

   (iii) Can you use memoization to reduce the time-complexity? Present the memoized algorithm to solve the same. What will be your space-complexity (additional space required to memoize)?

   (iv) Design an efficient iterative algorithm that finds the length as well as the longest ascending subsequence. What is the time and space complexity of your iterative solution?