INDIAN STATISTICAL INSTITUTE

M. Tech (CS) - I Year, 2016-2017 (Semester - II)

Design and Analysis of Algorithms

Tutorial-3 Date: 14.02.2017

(Q1) Given a set of $\mathcal{U} = \{u_1, \ldots, u_n\}$ of n objects with the corresponding sizes being $s_1, \ldots, s_n$, $s_i \in \mathbb{Z}^+$, and the profits being $v_1, \ldots, v_n$, $v_i \in \mathbb{Z}^+$, and a *knapsack capacity* $\mathcal{C} \in \mathbb{Z}^+$, find a subset of $\mathcal{U}$ whose total size is bounded by $\mathcal{C}$ and the total profit is maximized.

(Q2) The following problem is known as the *edit distance* problem. Given two strings S_1 of m characters and S_2 of n 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. Insertion and deletion takes unit cost, overwrite takes a cost of two. Design and analyze an efficient algorithm that minimizes the number of edits (and hence the time) required. The characters belong to the same alphabet set.

[Hints: Can you find appropriate overlapping subproblem?]

(Ans:) The *edit distance* between two strings S_1 and S_2 is the minimum number of character insertions, character deletions and character substitutions required to transform S_1 to S_2.

Let $C(i, j)$, $(1 \leq i \leq m)$ and $(1 \leq j \leq n)$, be the *edit distance* between the first i characters of S_1 and the first j characters of S_2. Thus, the final answer is surely $C(m, n)$. As to the overlapping sub-problem, note that $C(i, j)$ can be built from subproblems.

We would hope for, as again in *LCS*, filling up a table of m rows and n columns. In writing the recursion, first, look at the base cases, i.e. the first row and the first column. $C(i, 0) = i$ and $C(0, j) = j$. Now, we treat insertion, deletion and overwrite separately.

For insertion, we insert one character into the i-th position of S_1 thus generating i characters of S_1. So, for the cost, we have $C(i, j) = C(i - 1, j) + 1$.

For deletion, we delete one character from the j-th position of S_2. So, for the cost, we have $C(i, j) = C(i, j - 1) + 1$; as the j-th character in S_2 is deleted.

For substitution, if the i-th character of S_1 is the same as the j-th character of S_2, then there is no cost involved, i.e. $C(i, j) = C(i - 1, j - 1)$; else $C(i, j) = C(i - 1, j - 1) + 2$.

Obviously, $C(i, j)$ is the minimum of these three quantities. Thus, we have the following recurrence.

$$C(i,j) = \begin{cases} i & \text{if } j = 0 \\ j & \text{if } i = 0 \\ \min & \begin{cases} C(i-1,j)+1 \\ C(i,j-1)+1 \\ C(i-1,j-1) & \text{if } S_1[i] = S_2[j] \\ C(i-1,j-1)+2 & \text{if } S_1[i] \neq S_2[j] \end{cases} \end{cases}$$

Filling up the $m \times n$ table using this recurrence takes $O(mn)$ time.

As an exercise, find out the relation between LCS and edit distance. Is LCS going to be the same as the edit distance between two strings with insertion and deletion as the only edit operations with unit cost?

(Q3) Prove or disprove the following statement. *In a graph, the sub-path of a longest path is also the longest path.*

(Q4) Design efficient algorithms to find out the shortest and longest paths between two vertices in a directed acyclic graph (DAG).

(Q5) Prove the following. If C and C' are strongly connected components, and there is an edge from a node in C to a node in C', then the highest postdfn number in C is bigger than the highest postdfn number in C'.

(Q6) A vertex v in an undirected graph $G = (V, E)$, $|V| > 2$, is called an *articulation* point if its removal along with the removal of its incident edges forms a disconnected subgraph of G. A graph is called *biconnected* if it is connected and has no *articulation* points. Design an efficient algorithm (in $O(|V| + |E|)$ time complexity) to find the articulation points of G.

[Hints: See David Mount's lecture notes – the reference (W3) in the course web-page.]