***Design and Analysis of Algorithms***
***CS575, Spring 2024***

# Theory Assignment 3.1

**Due on 4/5/24 (11:59pm on Friday)**

Remember to include the following statement at the start of your answers with a signature by the side. "I have done this assignment completely on my own. I have not copied it, nor have I given my solution to anyone else. I understand that if I am involved in plagiarism or cheating I will have to sign an official form that I have cheated and that this form will be stored in my official university record. I also understand that I will receive a grade of 0 for the involved assignment for my first offense and that I will receive a grade of "F" for the course for any additional offense.

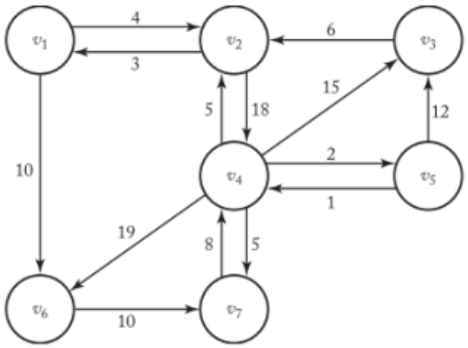
1. [40 points] The *edit distance* between two strings $S_1$ and $S_2$ is the minimum number of operations to convert one string to the other string. We assume that three types of operations can be used: Insert (a character), Delete (a character), and Replace (a character by another character). For example, the edit distance between *dof* and *dog* is 1 (one Replace), between *cat* and *act* is 2 (one Delete and one Insert or two Replace), between *cat* and *dog* is 3 (3 Replace). Design a dynamic programming algorithm to compute the edit distance between two strings by following the steps below:
   a. [10 points] Write down the principle of optimality for the minimum edit distance problem, and prove that the problem satisfies the principle of optimality.

   b. [10 points] Show the recurrence equation for computing the edit distance. (Hint: Let $d[i, j]$ be the edit distance between the substring of the first $i$ characters of $S_1$ and the substring of the first $j$ characters of $S_2$. Then consider the prefixes of the two strings in a way similar to the analysis for the LCS problem.)

   c. [10 points] Provide pesudocode for Edit-Distance($S_1$, $S_2$).

   d. [10 points] Use Edit-Distance( ) to create the table $d$ ($d[i, j]$ is defined above) for $S_1$ = cats and $S_2$ = fast. The entry at $d[4, 4]$ should show the correct edit distance between the two words.

2. [35 points] Use Floyd's algorithm to find all pairs shortest paths in the following graph.




a. [15 points] construct the matrix D, which contains the lengths of the shortest paths, and the matrix P, which contains the highest indices of the intermediate vertices on the shortest paths. Show the actions step by step. You need to show $D^0$ to $D^7$ and $P^0$ to $P^7$ (i.e. matrix P updated along with D step by step). You can use your computer program to output them or do it manually.

b. [10 points] Use the Print Shortest Path algorithm (slide 48 of the dynamic programming lecture notes) to find the shortest path from vertex v7 to vertex v3 using the matrix P you constructed from the previous step. Show the actions step by step (either trace the algorithm or show the call tree). You can take the slide 51 as an example of the call tree.

c. [10 points] Analyze the Print Shortest Path algorithm and show that it has a linear-time complexity (input size is the number of vertices in the graph).
(Hint: You can consider each array access to P[i][j] as a basic operation.)