**DAA [Day - 3]**

UID: 24MCI10204

Name: Rahul Saxena

Branch: 24MCA – AI & ML

**Question 1:** In plagiarism detection, find the length and the actual longest common subsequence of two documents.

**Answer:**

public class LCSPlagiarismDetector {

public static String findLCS(String s1, String s2) {

int m = s1.length();

int n = s2.length();

int[][] dp = new int[m + 1][n + 1];

for (int i = 1; i <= m; i++) {

for (int j = 1; j <= n; j++) {

if (s1.charAt(i - 1) == s2.charAt(j - 1)) {

dp[i][j] = dp[i - 1][j - 1] + 1; // Match found

} else {

dp[i][j] = Math.max(dp[i - 1][j], dp[i][j - 1]);

}

}

}

int i = m, j = n;

StringBuilder lcs = new StringBuilder();

while (i > 0 && j > 0) {

if (s1.charAt(i - 1) == s2.charAt(j - 1)) {

lcs.append(s1.charAt(i - 1));

i--;

j--;

} else if (dp[i - 1][j] > dp[i][j - 1]) {

i--;

} else {

j--;

}

}

return lcs.reverse().toString();

}

public static void main(String[] args) {

String doc1 = "DAABEC";

String doc2 = "ACDBECA";

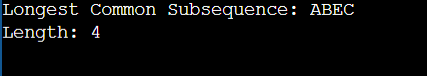
String lcs = findLCS(doc1, doc2);

System.out.println("Longest Common Subsequence: " + lcs);

System.out.println("Length: " + lcs.length());

}

}

**Output:**

**Question 2:** Given a city network, compute the shortest distances between all pairs of cities.

**Answer:**

public class AllPairsShortestPath {

final static int INF = 99999;

static void floydWarshall(int[][] graph, int V) {

int[][] dist = new int[V][V];

for (int i = 0; i < V; i++) {

for (int j = 0; j < V; j++) {

dist[i][j] = graph[i][j];

}

}

for (int k = 0; k < V; k++) {

for (int i = 0; i < V; i++) {

for (int j = 0; j < V; j++) {

if (dist[i][k] + dist[k][j] < dist[i][j])

dist[i][j] = dist[i][k] + dist[k][j];

}

}

}

System.out.println("Shortest distances between every pair of cities:");

for (int i = 0; i < V; i++) {

for (int j = 0; j < V; j++) {

if (dist[i][j] == INF)

System.out.print("INF ");

else

System.out.print(dist[i][j] + " ");

}

System.out.println();

}

}

public static void main(String[] args) {

int V = 4; // Number of cities

int[][] graph = {

{0, 5, INF, 10},

{INF, 0, 3, INF},

{INF, INF, 0, 1},

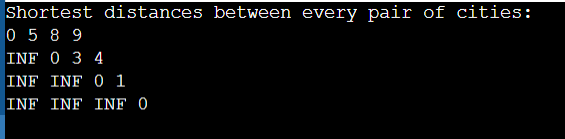
{INF, INF, INF, 0}

};

floydWarshall(graph, V);

}

}

**Output:**