Submission Instructions

a. This assignment is to be solved in teams of two students. Take our warning on plagiarism very seriously. *We assume that from the moment of submission you are certifying that it is solely the work of your team, except where explicitly stated otherwise, and that each teammate can individually explain any part of the solution.*

b. Write clear answers. Justify all answers, except where explicitly not required. State any assumptions you make. Thoroughly spell-check and grammar-check your answers. Meticulously follow all instructions in this document and at the homework web-page.

c. Document each algorithm according to the coding convention of the course.

d. Name your report **LastName1.FirstName1 – LastName2.FirstName2.pdf**, but without using any special characters. The report must be in **PDF** format and must have the cover information of the demo assignment report.

e. Submit your report via the Course Manager server (whose clock may differ from yours) by the deadline given above.

f. *One* teammate must use the group submission feature of the Course Manager, and the other teammate must confirm the submission via the Course Manager before the deadline.

**Failure to follow the instructions above may result in 0 (zero) points**, and we reserve the right to process your solutions automatically.
Question 1: Search Term Replacement  

A useful feature for search engines is to suggest a replacement search term when a term given by the user is not known to the engine. In order to suggest a new search term, the engine must have some measure of the difference between the given term and possible replacement terms. We can define this difference as the minimum number of changes required between the user’s search term and the replacement term. A change can be either: (a) altering a letter in the user’s term to match a letter in the same position in the replacement term, or (b) skipping a letter in one of the two terms.

For example, if a term is a string over the alphabet \( \{A, B, \ldots, Z\} \), then \( u = DINAMCK \) can be the user’s search term, and \( r = DYNAMIC \) could be the proposed replacement term. A positioning of \( u \) and \( r \) is a way of matching up these two strings by writing them in columns, for instance:

\[
\begin{align*}
D & I & N & A & M & \quad \text{–} & C & K \\
D & Y & N & A & M & I & C & \text{–}
\end{align*}
\]

A dash (–) indicates that a letter has been skipped. The characters maintain their original ordering in each string. The difference of a positioning is the sum of the measures of resemblance of the pairs of letters in each column, as given by a resemblance matrix \( \rho[i,j] \). Note that for an alphabet of size \( a \), we have that \( \rho \) is an \( (a+1) \times (a+1) \) matrix, as it must include the skip character (–) in addition to the \( a \) characters of the alphabet. The positioning above has a difference of:

\[
\rho[D,D] + \rho[I,Y] + \rho[N,N] + \rho[A,A] + \rho[M,M] + \rho[–,I] + \rho[C,C] + \rho[K,–]
\]

For two given strings \( u[1..m] \) and \( r[1..n] \) over an alphabet of size \( a \), where \( m \) and \( n \) may be different, and given a resemblance matrix \( \rho[1..a+1,1..a+1] \), answer the following sub-questions:

a. Let \( s[i,j] \) be the minimum difference of two prefixes \( u[1..i] \) and \( r[1..j] \), where \( 0 \leq i \leq m \) and \( 0 \leq j \leq n \). Find a recursive formula that computes \( s[i,j] \).

b. Justify why dynamic programming is applicable to the problem of (i) computing the minimum difference, and (ii) returning a positioning for that minimum difference.

c. Give pseudocode for a dynamic programming algorithm to solve this problem.

d. Your algorithm should have a time complexity of \( O(m \cdot n) \): argue why it does so.

Question 2: Ring Detection in Graphs  

In an undirected graph, a path \( p = (v_0, v_1, \ldots, v_k) \) forms a ring if the vertices \( v_0 \) and \( v_k \) are equal, and all edges on the path are distinct, i.e.:

\[
\forall i,j \in \{0 \ldots k-1\}: i \neq j \implies (v_i, v_{i+1}) \neq (v_j, v_{j+1})
\]

Design an algorithm that determines the presence of rings in an undirected graph \( G = (V,E) \) and runs in \( O(|V|) \) time, independently of \( |E| \):

a. Write the pseudocode of your algorithm.

b. Explain why your algorithm has an \( O(|V|) \) time complexity.

NOTE: You cannot assume that \( G \) is connected. Furthermore, for an undirected graph:

- An edge is given by an unordered pair of vertices; in other words, the edges \( (u,v) \) and \( (v,u) \) are considered to be the same edge.

- Self-loops are forbidden; every edge consists of two distinct vertices.

Question 3: Recomputing the MST (40 points)


You are given a graph $G = (V, E)$ with non-negative edge weights, and a minimum spanning tree (MST) $T = (V, E')$ with respect to these weights; you may assume $G$ and $T$ are given as adjacency lists. Now suppose the weight of a particular edge $e \in E$ is modified from $w(e)$ to a new value $\hat{w}(e)$. You wish to update quickly the minimum spanning tree $T$ to reflect this change, without recomputing the entire tree from scratch. There are four cases:

a. $e \notin E'$ and $\hat{w}(e) > w(e)$.

b. $e \notin E'$ and $\hat{w}(e) < w(e)$.

c. $e \in E'$ and $\hat{w}(e) < w(e)$.

d. $e \in E'$ and $\hat{w}(e) > w(e)$.

In each case, discuss a linear-time algorithm for updating the tree, and for one non-trivial case provide the pseudocode of such an algorithm that updates the tree.