
In this problem you will explore some of the many possible measures of similarity between pairs of strings. Give an efficient algorithm for computing each of the following measures, with a proof of correctness and analysis of the running time in each case.

(a) Longest common subsequence. A subsequence of a string is obtained by taking a subset of its characters without changing their order. (For example, ART is a subsequence of ALGORITHM.) A common subsequence of a pair of strings \( x \) and \( y \) is a string that is a subsequence of both \( x \) and \( y \). Consider the length of a longest common subsequence of \( x \) and \( y \).

(b) Edit distance. The edit distance between \( x \) and \( y \) is the smallest number of single-character insertions, deletions, or substitutions that suffice to change \( x \) into \( y \).

(c) Edit distance with transpositions. The edit distance with transpositions between \( x \) and \( y \) is the smallest number of single-character insertions, deletions, or substitutions, or transpositions of two adjacent characters, that suffice to change \( x \) into \( y \). (For example, the edit distance with transpositions between NOSE and ONCE is 2, whereas their edit distance is 3.)

2. Processing sheet metal.

You run a company that processes sheet metal. You have a price list indicating that a rectangular piece of sheet metal of dimensions \( x_i \times y_i \) can be sold for \( v_i \) dollars, where \( i \in \{1, 2, \ldots, n\} \). Starting from a raw piece of sheet metal of dimensions \( A \times B \), you would like to make a sequence of horizontal or vertical cuts, each of which cuts a given piece into two smaller pieces, to produce pieces from your price list (and possibly some additional worthless scrap pieces). Design an algorithm that maximizes the total value of the pieces obtained by some sequence of cuts. Your algorithm should return both the maximum value that can be obtained and a description of the cuts that should be made to achieve it. Assume that the numbers \( A, B, x_i, y_i, v_i \) for \( i \in \{1, 2, \ldots, n\} \) are all positive integers. Demand for everything on your price list is high, so you can produce any number of copies of any of the pieces. Prove that your algorithm is correct and analyze its running time.

3. Network flows with vertex capacities.

(a) Let \( G = (V, E) \) be a directed graph with source vertex \( s \in V \) and sink vertex \( t \in V \). Whereas the standard network flow problem involves capacities for the edges, suppose that instead every vertex \( v \in V \) has a capacity \( c_v \geq 0 \). A vertex-capacitated flow in \( G \) is a function \( f: E \rightarrow \mathbb{R}^+ \) such that

i. (Capacity conditions) For each vertex \( v \in V \), we have
\[
\sum_{e \text{ into } v} f(e) \leq c_v \quad \text{and} \quad \sum_{e \text{ out of } v} f(e) \leq c_v.
\]

ii. (Conservation conditions) For each vertex \( v \in V \setminus \{s, t\} \), we have
\[
\sum_{e \text{ into } v} f(e) = \sum_{e \text{ out of } v} f(e).
\]
As usual, the value of a flow is $\sum_{e \text{ out of } s} f(e)$. Give an efficient algorithm to find a maximum vertex-capacitated flow in $G$ from $s$ to $t$. Establish its correctness and analyze its running time.

(b) Define a cut in a vertex-capacitated network to be a vertex subset $U \subseteq V$ such that every path from $s$ to $t$ goes through at least one vertex of $U$. The capacity of the cut $U$ is $\sum_{u \in U} c_u$. Using these definitions, state and prove an analog of the Max-Flow/Min-Cut Theorem for vertex-capacitated networks.

4. *Can you hear me now?*

Suppose you have deployed a cellular phone network with $n$ cell towers at points $(x_i, y_i)$ for $i \in \{1, \ldots, n\}$. Also suppose there are $m$ cell phones in use at points $(p_i, q_i)$ for $i \in \{1, \ldots, m\}$. Each cell tower has a range of $r$, meaning it can connect only to phones within a distance $r$ of the tower. However, the capacity of the towers is limited: each tower can connect to at most $k$ phones at a time. Given this input data, your goal is to determine the largest number of phones that can be connected to the network, and to specify which phone should be connected to which tower to achieve this optimum. Design an algorithm for this problem, prove its correctness, and analyze its running time as a function of $n$ and $m$. 