If u, v, and w are strings over an alphabet Σ, then w is a *shuffle* of u and v provided there are (possibly empty) strings x_i and y_i such that $u = x_1x_2\cdots x_k$ and $v = y_1y_2\cdots y_k$ and $w = x_1y_1x_2y_2\cdots x_ky_k$. A shuffle is sometimes instead called a "merge" or an "interleaving." The intuition for the definition is that w can be obtained from u and v by an operation similar to shuffling two decks of cards. We use $w = u \odot v$ to denote that w is a shuffle of u and v; note, however, that in spite of the notation there can be many different shuffles w of u and v. The string w is called a *square* provided it is equal to a shuffle of a string u with itself, namely provided $w = u \odot u$ for some string u.

The idea behind the algorithm of is to construct a grid graph, with $(|x|+1)\times(|y|+1)$ nodes; the lower-left node is represented with $(0,0)$ and the upper-right node is represented with $(|x|,|y|)$. For any $i < |x|$ and $j < |y|$, we have the edges:

$$\begin{cases} ((i,j),(i+1,j)) & \text{if } x_{i+1} = w_{i+j+1} \\ ((i,j),(i,j+1)) & \text{if } y_{j+1} = w_{i+j+1}. \end{cases} \tag{1}$$

Note that both edges may be present, and this in turn introduces an exponential number of choices if the search were to be done naïvely.

A path starts at $(0,0)$, and the i-th time it goes up we pick x_i, and the j-th time it goes right we pick y_j. Thus, a path from $(0,0)$ to $(|x|,|y|)$ represents a particular shuffle.

For example, consider Figure 1. On the left we have a shuffle of 000 and 111 that yields 010101, and on the right we have a shuffle of 011 and 011 that yields 001111. The left instance has a unique shuffle that yields 010101, which corresponds to the unique path from $(0,0)$ to $(3,3)$. On the right, there are several possible shuffles of $011, 011$ that yield 001111 — in fact, eight of them, each corresponding to a distinct path from $(0,0)$ to $(3,3)$.

A dynamic programming algorithm computes partial solutions along the top-left to bottom-right diagonal lines in the grid graph.

The number of paths is always bounded by:

$$\binom{|x|+|y|}{|x|}$$

and this bound is achieved for $\langle 1^n, 1^n, 1^{2n}\rangle$. Thus, the number of paths can be exponential in the size of the input, and so an exhaustive search is not feasible in general.

Given the discussion in this section, propose a dynamic programming algorithm that on input w, u, v checks whether $w = u \odot v$.

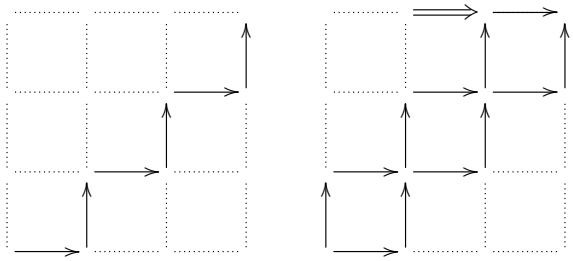

Figure 1: On the left we have a shuffle of 000 and 111 that yields 010101, and on the right we have a shuffle of 011 and 011 that yields 001111. The double arrow in the right diagram is there to show that there may be other arrows beside the solid arrows; the double arrow is $((1,3),(2,3))$ and it is there because $x_{1+1} = x_2 = 1 = w_5 = w_{1+3+1}$. The edges are placed according to (1)