A *shuffle* of two strings, sometimes called instead a *merge* or *interleaving*, is a way of combining those two strings into one, preserving the order of the symbols in the two original strings. 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. For example, if $u = 01101110$ and $v = 10101000$, then $w = 0110110011101000$ is a possible shuffle. Note that the colors are used to show that w comes from u and v; coloring is not part of the shuffle.

More formally, we say that 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 $w = x_1y_1x_2y_2\cdots x_ky_k$. Take the example from the first paragraph; and let ε denote the empty string, then: $w = 0\varepsilon 11\varepsilon 0\varepsilon 11\varepsilon 00111010\varepsilon 00$

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.

Your Task: Design, and implement in Python 3, a dynamic programming algorithm which on input w, u, v checks whether $w = u \odot v$ (and outputs "yes" or "no").

While you are free to take any approach (as long as you explain it in your assignment solutions!), here is a possible way to design this algorithm: 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 possible dynamic programming algorithm would compute 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.

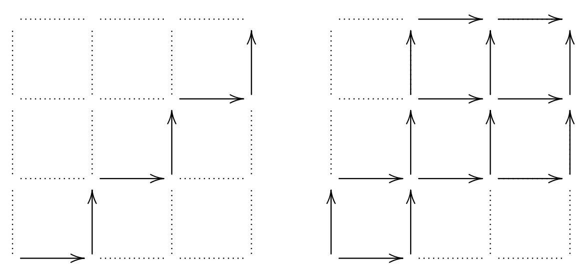

Figure 1: On the left we have a shuffle of 000 and 111 that yields 010101, and on the right we have a shuffle of 011 and 011 that yields 001111. The edges are placed according to (1)