Substring Searching

In 1974, a very fast string searching method was proposed by the name of KMP algorithm with linear run-time complexity. Your task here is to code this (or any similar) algorithm in a functional language.

Given two strings *text* and *pat*, find whether *pat* exists as a substring in *text*.

Input
First line will contain an integer, *T*, which represents total number of test cases. Then *T* test cases follow. Each case will contains two lines each containing a string. First line will contain *text* while the second line will contain *pat*.

Output
For each case print `YES` if *pat* is a substring of *text* otherwise `NO`.

Constraints
$1 \leq T \leq 10$
$1 \leq |pat| \leq |text| \leq 100000$
All characters in *text* and *pat* will be lowercase latin character ('*a*'-'*z*').

Sample Input

```
4
abcdef
def
computer
muter
stringmatchingmat
ingmat
videobox
videobox
```

Sample Output

```
YES
NO
YES
YES
```

Explanation
Test Case #00: Here *"def"* is present at the end of *"abcdef"*.
Test Case #01: Though *"muter"* is a subsequence here, but we need it to be asubstring.
Test Case #02: _"ingmat"_ is present at index *3* and *11*.
Test Case #03: Both strings are same.