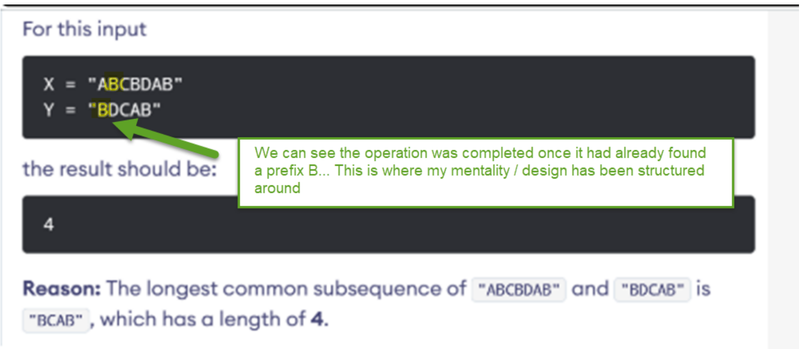
This test case triggered lots inquisitiveness and it made me realise that even in the example given in Programiz, it had performed deletion from within String Y once it had found its first match...



For this input

```
X = "ABCBDAB"
Y = "BDCAB"
```

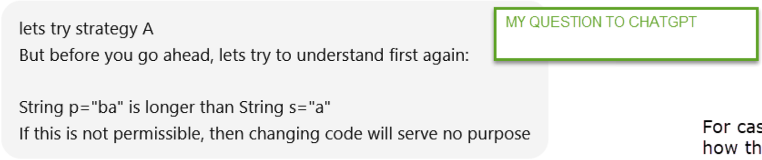
the result should be:

```
4
```

Reason: The longest common subsequence of `"ABCBDAB"` and `"BDCAB"` is `"BCAB"`, which has a length of **4**.

We can see the operation was completed once it had already found a prefix B... This is where my mentality / design has been structured around

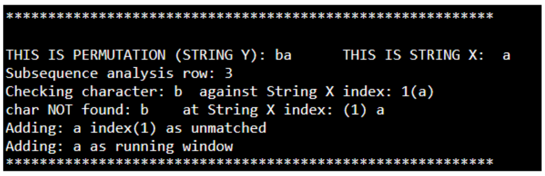
So I find myself in the exact same boat with my design, which is actually called ‘longest prefix subsequence’



MY QUESTION TO CHATGPT

lets try strategy A
But before you go ahead, lets try to understand first again:

String p="ba" is longer than String s="a"
If this is not permissible, then changing code will serve no purpose

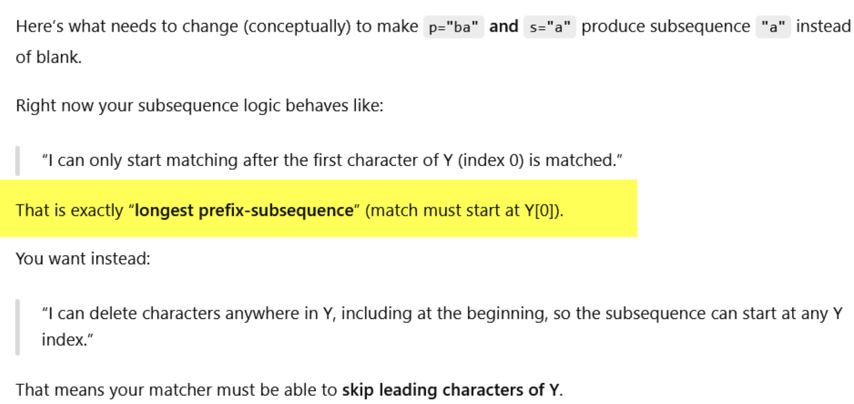


```
**********************************************************


THIS IS PERMUTATION (STRING Y): ba      THIS IS STRING X:  a
Subsequence analysis row: 3
Checking character: b  against String X index: 1(a)
char NOT found: b    at String X index: (1) a
Adding: a index(1) as unmatched
Adding: a as running window
**********************************************************
```

THE SITUATION WOULD HOWEVER BE SAVAGED WHEN THE PERMUTATION ABOVE BECOMES ab

For cases like this, we can see how this will be challenging in my code.. It would have reached end of the do while loop having processed single character in String X.
We would all of a sudden have to control what is written inside subsequences array. It is extremely challenging and chatGPT has identified that my code



Here’s what needs to change (conceptually) to make `p="ba"` **and** `s="a"` produce subsequence `"a"` instead of blank.

Right now your subsequence logic behaves like:

> “I can only start matching after the first character of Y (index 0) is matched.”

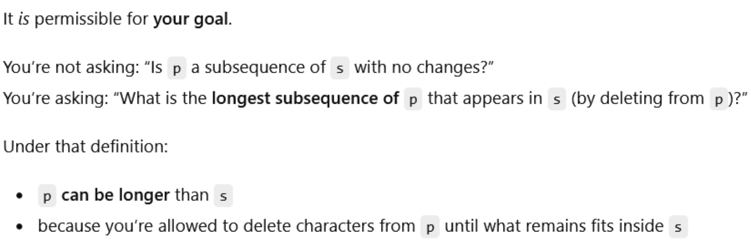
That is exactly “**longest prefix-subsequence**” (match must start at Y[0]).

You want instead:

> “I can delete characters anywhere in Y, including at the beginning, so the subsequence can start at any Y index.”

That means your matcher must be able to **skip leading characters of Y**.

chatGPT is correct since I used mechanism hasFirstMatch to ensure that characters are only removed from String Y ONLY if it has found first match. And to make issues even more trickier, I am confronted with a single character in String s.



It *is* permissible for **your goal**.

You’re not asking: “Is `p` a subsequence of `s` with no changes?”
You’re asking: “What is the **longest subsequence of** `p` that appears in `s` (by deleting from `p`)?”

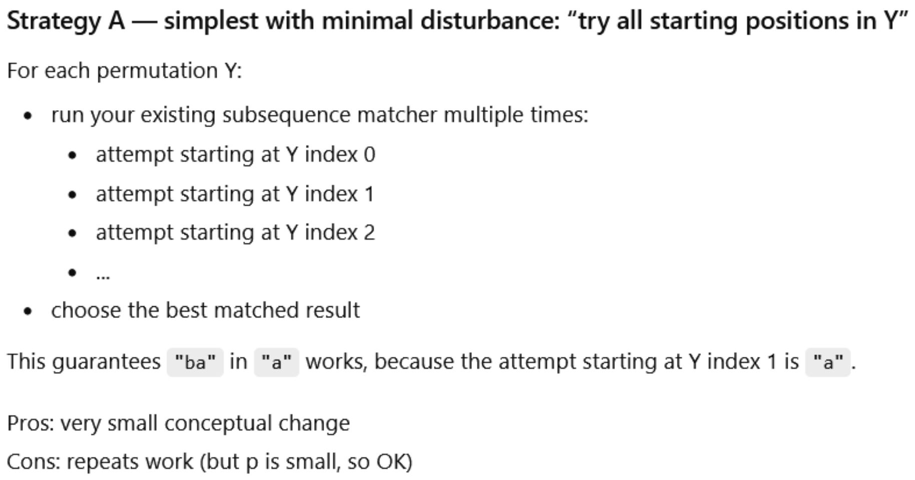
Under that definition:

- `p` **can be longer** than `s`
- because you’re allowed to delete characters from `p` until what remains fits inside `s`

So ChatGPT has given me an insight into longest subsequence of p.

But it becomes more confusing, since I know that with:
<u>prefix longest subsequence (my design)</u>
String p =”ba” and String s = “a”
I would explore ba and ab
So, I would find the length subsequence = 1 using String p = ab String s = a
I would find length subsequence = 0 using String p = ba

With
<u>longest subsequence (suggested by ChatGPT)</u>
I would explore ba and ab
So, I would find the length subsequence = 1 using String p = ab
I would find length subsequence = 0 using String p = ba
BUT using resilience, it would also identify length subsequence = 1 using String pa = ba (starting at index 1)



Strategy A — simplest with minimal disturbance: “try all starting positions in Y”

For each permutation Y:

- run your existing subsequence matcher multiple times:
  - attempt starting at Y index 0
  - attempt starting at Y index 1
  - attempt starting at Y index 2
  - ...
- choose the best matched result

This guarantees `"ba"` in `"a"` works, because the attempt starting at Y index 1 is `"a"`.

Pros: very small conceptual change
Cons: repeats work (but p is small, so OK)

So this raised a very interesting question, why would we want this to appear given that prefix approach can identify String p = ab String s = a (length subsequence = 1)
We have to remember that I use permutations of String p to facilitate all this…

Here is question to ChatGPT and response: