Challenge: Longest Common Subsequence

In this lesson, we will look at another classic dynamic programming problem: the longest common subsequence problem.

We'll cover the following

- Problem statement
- Input
- Output
- Problem statement

Problem statement

Given two strings, find the length of the longest common subsequence between them. A common subsequence in a pair of strings is a sequence that follows the same order of characters, but the sequence does not necessarily have to be contiguous. For example, two strings `two` and `too` have a common subsequence of `to`. Even though `to` does not appear contiguously in `two`, the order of characters is still preserved, i.e., `o` follows `t`. While these strings had smaller subsequences as well, such as `t` and `o`, in the context of this problem, we are only interested in finding the length of the longest subsequence.

Input

Your algorithm will take two strings, i.e., `str1` and `str2`, as input. Strings can be of variable length, even empty too.

```
str1 = "two"
str2 = "too"
```

Output

Your algorithm should return an integer representing the length of the longest common subsequence.

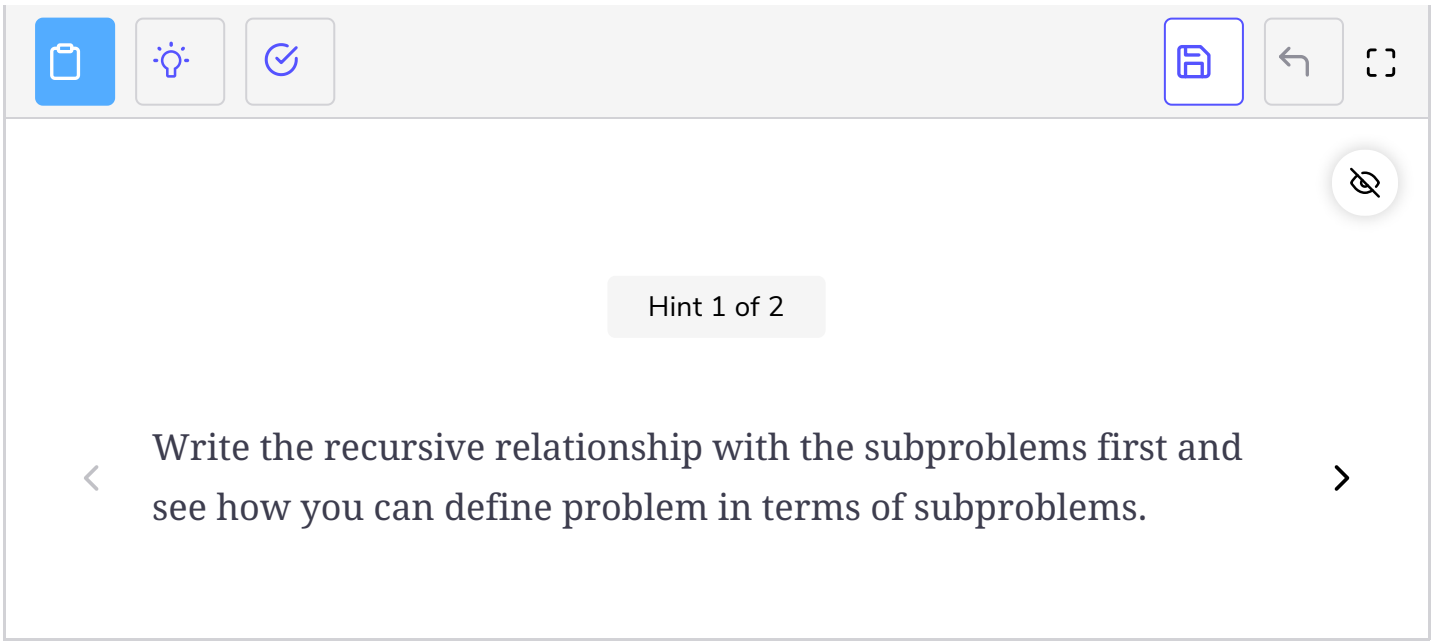
```
LCS("two", "too") = 2
```

Problem statement

You have already solved a similar problem of the longest common substring in the previous chapter. This problem is slightly different because we are finding subsequence instead of substring here. Think about a simple recursive solution first and then build on that to write a dynamic programming solution. Best of luck!

```
def LCS(str1, str2):
  # write your code here
  return -1

stressTesting = True
```



In the next lesson, we will see some solutions to this problem.