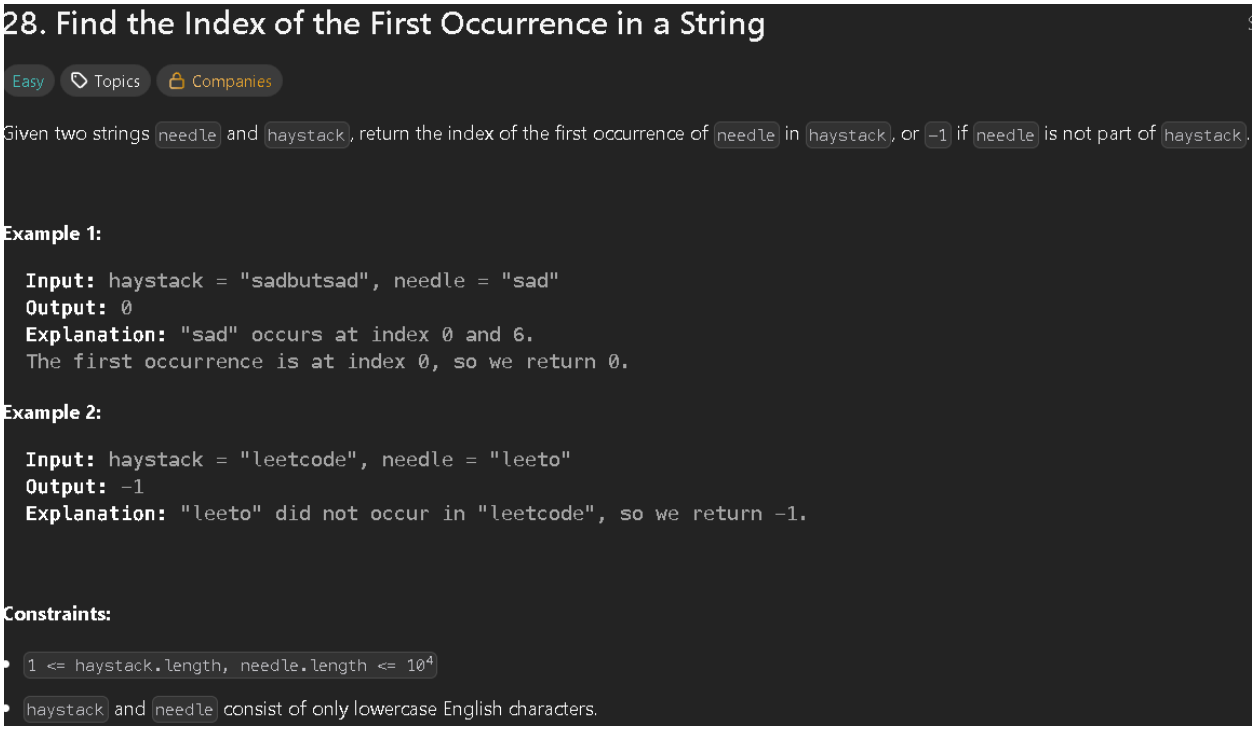

Python:

```
class Solution:
    def strStr(self, haystack: str, needle: str) -> int:
        # Use Python's built-in find() method
        return haystack.find(needle)
        # n, m = len(haystack), len(needle)

        # # Loop through haystack until the point where needle can still fit
        # for i in range(n - m + 1):
        #     if haystack[i:i+m] == needle:
        #         return i
        # return -1
```

JavaScript:

```
/**
 * @param {string} haystack
 * @param {string} needle
 * @return {number}
 */
var strStr = function(haystack, needle) {
    // Edge case: if needle is empty, return 0 (as per problem definition)
    if (needle.length === 0) return 0;

    // Loop through haystack
    for (let i = 0; i <= haystack.length - needle.length; i++) {
```

```
        // Take substring of haystack with the same length as needle
        if (haystack.substring(i, i + needle.length) === needle) {
            return i; // return the first index where it matches
        }
    }

    return -1; // not found
};
```

Java:

```
class Solution {
    public int strStr(String haystack, String needle) {
        // Edge case: if needle is empty, return 0
        if (needle.isEmpty()) {
            return 0;
        }

        int hLen = haystack.length();
        int nLen = needle.length();

        // Loop through haystack until remaining substring is at least needle's length
        for (int i = 0; i <= hLen - nLen; i++) {
            // Check substring from i to i+nLen
            if (haystack.substring(i, i + nLen).equals(needle)) {
                return i; // Found first occurrence
            }
        }

        // Not found
        return -1;
    }
}
```