Problem

Find the index of the first occurrence of needle in haystack

Example

haystack = "sadbutsad"

needle = "sad"

Output = 0

1️⃣ Using indexOf() (Built-in Method)

💡 Idea

Let the language do the work.

🔍 Dry Run

"sadbutsad".indexOf("sad")

Internally:

- Starts scanning from index 0
- Matches sad at index 0
- Returns 0

You don't control how comparison happens.

✅ Code

Java

```
class Solution {
    public int strStr(String haystack, String needle) {
        return haystack.indexOf(needle);
    }
}
```

C++

```
class Solution {
public:
```

```
    int strStr(string haystack, string needle) {
        size_t pos = haystack.find(needle);
        return (pos == string::npos) ? -1 : pos;
    }
};
```

Python

```
class Solution:
    def strStr(self, haystack: str, needle: str) -> int:
        return haystack.find(needle)
```

⏱ Complexity

Case	Complexity
Time (worst)	O(n * m)
Space	O(1)

❌ Interview note: Usually rejected as “library usage”.

2️⃣ Two Pointers (Brute Force)

💡 Idea

Try matching needle starting from every possible index in haystack.

🧪 Dry Run (VERY IMPORTANT)

haystack = sadbutsad

needle = sad

Step 1: start at i = 0

```
s a d b u t s a d
^ ^ ^
s a d
```

✔ ✔ ✔ → MATCH → return 0

Worst-case dry run

haystack = aaaaaaa

needle = aaaab

Start i=0 → match aaaa ❌

Start i=1 → match aaaa ❌

Start i=2 → match aaaa ❌

...

That's why it becomes O(n*m).

✅ Code

Java

```
class Solution {
    public int strStr(String haystack, String needle) {
        int n = haystack.length();
        int m = needle.length();

        for (int i = 0; i <= n - m; i++) {
            int j = 0;
            while (j < m && haystack.charAt(i + j) == needle.charAt(j)) {
                j++;
            }
            if (j == m) return i;
        }
        return -1;
    }
}
```

C++

```
class Solution {
```

```
public:
    int strStr(string haystack, string needle) {
        int n = haystack.size();
        int m = needle.size();

        for (int i = 0; i <= n - m; i++) {
            int j = 0;
            while (j < m && haystack[i + j] == needle[j]) {
                j++;
            }
            if (j == m) return i;
        }
        return -1;
    }
};
```

Python

```
class Solution:
    def strStr(self, haystack: str, needle: str) -> int:
        n, m = len(haystack), len(needle)

        for i in range(n - m + 1):
            j = 0
            while j < m and haystack[i + j] == needle[j]:
                j += 1
            if j == m:
                return i
        return -1
```

⏱ Complexity

Time O(n * m)

Space O(1)

✅ Interview note: Good starting solution.

🔥 FINAL COMPARISON

Method	Time	Space	Interview
indexOf()	O(n*m)	O(1)	❌
Two Pointers	O(n*m)	O(1)	⚠
KMP	O(n+m)	O(m)	✅