Problem

Find the index of the first occurrence of needle in haystack

Example

haystack = "sadbutsad"

needle   = "sad"

Output   = 0

3️⃣ KMP Algorithm (Optimal)

⚠️ This is where most beginners panic — but once dry-run, it clicks.

💡 Idea (Core Insight)

When a mismatch happens:

- Don’t restart from scratch
- Use LPS (Longest Prefix Suffix) info

▯ Step 1: Build LPS Array

Needle = "sad"

Index: 0 1 2

Char : s a d

LPS : 0 0 0

Why?

- No prefix == suffix at any point

▯ Step 2: Matching Dry Run

haystack = sadbutsad

needle   = sad

Variables

i → haystack pointer

j → needle pointer

Matching

i=0, j=0 ➔ s == s ✔

i=1, j=1 ➔ a == a ✔

i=2, j=2 ➔ d == d ✔

j == needle.length ➔ MATCH FOUND

index = i - j + 1 = 0

☑ Java Code (Clean & Interview-Ready)

```java
class Solution {
    public int strStr(String haystack, String needle) {
        if (needle.length() == 0) return 0;

        int[] lps = buildLPS(needle);
        int i = 0, j = 0;

        while (i < haystack.length()) {
            if (haystack.charAt(i) == needle.charAt(j)) {
                i++;
                j++;
                if (j == needle.length()) {
                    return i - j;
                }
            } else {
                if (j > 0) {
                    j = lps[j - 1];
                } else {
                    i++;
                }
            }
        }
        return -1;
    }

    private int[] buildLPS(String pattern) {
        int[] lps = new int[pattern.length()];
        int len = 0, i = 1;

        while (i < pattern.length()) {
            if (pattern.charAt(i) == pattern.charAt(len)) {
                len++;
                lps[i] = len;
                i++;
            } else {
                if (len > 0) {
                    len = lps[len - 1];
                } else {
                    lps[i] = 0;
                    i++;
                }
            }
        }
        return lps;
    }
}
```

✅ C++ Code

```
class Solution {
```

```
public:
    vector<int> buildLPS(string pat) {
        vector<int> lps(pat.size(), 0);
        int len = 0, i = 1;

        while (i < pat.size()) {
            if (pat[i] == pat[len]) {
                lps[i++] = ++len;
            } else if (len > 0) {
                len = lps[len - 1];
            } else {
                i++;
            }
        }
        return lps;
    }

    int strStr(string haystack, string needle) {
        if (needle.empty()) return 0;

        vector<int> lps = buildLPS(needle);
        int i = 0, j = 0;

        while (i < haystack.size()) {
            if (haystack[i] == needle[j]) {
                i++; j++;
                if (j == needle.size()) return i - j;
            } else if (j > 0) {
                j = lps[j - 1];
            } else {
                i++;
            }
        }
        return -1;
    }
};
```

✅ Python Code

```
class Solution:
    def strStr(self, haystack: str, needle: str) -> int:
        if not needle:
            return 0

        lps = self.buildLPS(needle)
        i = j = 0

        while i < len(haystack):
            if haystack[i] == needle[j]:
                i += 1
                j += 1
                if j == len(needle):
                    return i - j
            elif j > 0:
                j = lps[j - 1]
            else:
                i += 1
        return -1
```

```
 def buildLPS(self, pat):
     lps = [0] * len(pat)
     length = 0
     i = 1

     while i < len(pat):
         if pat[i] == pat[length]:
             length += 1
             lps[i] = length
             i += 1
         elif length > 0:
             length = lps[length - 1]
         else:
             i += 1
     return lps
```

⏱ KMP Complexity

Time	O(n + m)
Space	O(m)

🔥 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)	✅