Longest Proper Prefix Which is Also a Suffix (LPS)

1. Problem Statement

Given a string `s`, the task is to determine the length of the longest proper prefix of `s` that is also a suffix.

- A **proper prefix** of `s` is a prefix that is **not equal to** `s` **itself**.
- A **suffix** is a substring that appears at the end of `s`.
- The goal is to find the longest prefix which is also a suffix (excluding the full string itself).

---

2. Approach (Using LPS Array from KMP Algorithm)

This problem is efficiently solved using the **prefix function (LPS array)** from the **Knuth-Morris-Pratt (KMP) string matching algorithm**.

Steps:

1. Construct the **LPS (Longest Prefix Suffix) array**.
2. The last value of the LPS array gives the length of the longest proper prefix which is also a suffix.

LPS Array Construction:

- Define an array `lps` of size `n` (length of string `s`).
- `lps[i]` stores the length of the longest prefix-suffix for `s[0...i]`.
- Iterate through the string and update `lps` based on character matches and previous `lps` values.

---

3. Complexity Analysis

Time Complexity:

- The function builds the LPS array in a **single pass** of the string.
- The **for loop** runs **O(n)** iterations.

- The **while loop** runs in amortized **O(1)** time per character.
- **Overall Time Complexity: O(n).**

Space Complexity:

- The function uses an **LPS array** of size `n`, leading to **O(n) space complexity**.
- If optimized to use a single variable instead of storing the array, the space complexity can be reduced to **O(1)**.

---

4. Example Execution

Input:

```
s = "abababab"
```

Processing (LPS Array Calculation):

```
Index:   0 1 2 3 4 5
String:  a b a b a b
LPS:     0 0 1 2 3 4
```

Output:

```
4
```

Explanation:

- The longest prefix which is also a suffix is "abab" (length = 4).

---

5. Use Cases

- **Pattern Matching (KMP Algorithm)**: The LPS array is fundamental to the KMP string-matching algorithm, making substring searches more efficient.
- **String Repetition Detection**: It helps in identifying cyclic patterns in strings.
- **Automata Design**: Used in state transitions in finite state automata.

- **Plagiarism Detection**: Identifies similar prefixes and suffixes across documents.
- **Text Processing in NLP**: Used in tokenization and pattern recognition for text-based models.

---

6. Code Implementation (C++)

```
#include <iostream>
#include <vector>

using namespace std;

int longestPrefixLength(string s) {
    int n = s.size();
    vector<int> lps(n, 0);
    int j = 0;
    for(int i = 1; i < n; i++) {
        while(j > 0 && s[i] != s[j]) {
            j = lps[j - 1];
        }
        if(s[i] == s[j]) {
            j++;
            lps[i] = j;
        }
    }
    return lps[n - 1];
}

int main() {
    string s = "abababab";
    cout << longestPrefixLength(s) << endl;
    return 0;
}
```

---

7. Summary

- This method efficiently finds the longest prefix which is also a suffix using the LPS array.
- It runs in **O(n) time** and can be optimized to **O(1) space**.
- It is widely used in string processing, pattern matching, and automata-based applications.