DAA ASSIGNMENT

Implementation of KMP algorithm:

Unlike other search algorithm, where we slide the pattern by one and compare all characters at each shift, we use a value from lps[] to decide the next characters to be matched. The idea is to not match a character that we know will anyway match.

How to use lps[] to decide the next positions (or to know the number of characters to be skipped)?

- We start the comparison of pat[j] with j = 0 with characters of the current window of text.
- We keep matching characters txt[i] and pat[j] and keep incrementing i and j while pat[j] and txt[i] keep matching.
- When we see a mismatch
    - We know that characters pat [0...j-1] match with txt [i-j…i-1] (Note that j starts with 0 and increments it only when there is a match).
    - We also know (from the above definition) that lps[j-1] is the count of characters of pat [0…j-1] that are both proper prefix and suffix.
    - From the above two points, we can conclude that we do not need to match these lps[j-1] characters with txt [i-j…i-1] because we know that these characters will anyway match. Let us consider the above example to understand this.

PROGRAM:

```
//KMP ALGORITHM
def KMP(s,t):
    l=[]
    for i in range(0,len(s)-len(t)+1):
        if(s[i:i+len(t)]==t):
            l.append(i)
    if(l!=[]):
        for i in l:
            print("Pattern",t,"found at index:",i)
    else:
        print("Pattern not found")
def LPS(s):
    l=[0]
    i=0
    j=1
    for k in range(1,len(s)):
        if(s[i]==s[j]):
            l.append(i+1)
            i+=1
            j+=1
        else:
            l.append(0)
            j+=1
            i=0
    print("LSP for the Given string",s,":")
    print(l)
txt=str(input("Enter the String: "))
pat=str(input("Enter the Pattern to be Searched: "))
KMP(txt,pat)
LPS(txt)
```

Output:

Enter the String: AAAAABAAABA

Enter the Pattern to be Searched: AAAA

Pattern AAAA found at index: 0

Pattern AAAA found at index: 1

LSP for the Given string AAAAABAAABA:

[0, 1, 2, 3, 4, 0, 1, 2, 3, 0, 1]

Advantages of the KMP algorithm

- A very obvious advantage of the KMP algorithm is it's time complexity. It's very fast as compared to any other exact string-matching algorithm.
- No worse case or accidental inputs exist here.

Applications of the KMP Algorithm

It's uses are:

1. Checking for Plagiarism in documents etc
2. Bioinformatics and DNA sequencing
3. Digital Forensics
4. Spelling checkers
5. Spam filters
6. Search engines, or for searching content in large databases
7. Intrusion detection system

Complexity of KMP algorithm:

Time complexity - $O(n+m)$

SOURCE LINKS:

https://www.geeksforgeeks.org/kmp-algorithm-for-pattern-searching/

KMP Algorithm | Knuth Morris Pratt Algorithm - Scaler Topics