Experiment 3.3

Aim: Develop a program and analyze complexity to find all occurrences of a pattern P in a given string S.

Objectives: Analyze to find all occurrences of a pattern P in a given string S.

Input/Apparatus Used: String is taken as input in order to find pattern from it.

Procedure/Algorithm:

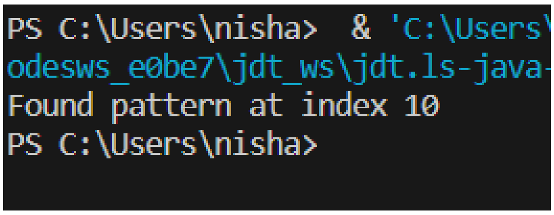
- We will first create the LPS array.
- Initialize two variables - ‘strIdx’ and ‘patIdx’ to iterate over the string and the pattern, respectively.
- If ‘pat[patIdx]’ equals ‘str[strIdx]’, we will increment both the indexes.
- When ‘patIdx’ equals the length of the pattern, this means that the pattern is found in the string. Therefore we print the index and set ‘patIdx’ = LPS[patIdx-1].
- If ‘pat[patIdx]’ is not equal to ‘str[strIdx]’, we update the patIdx witht he last index that matches with ‘str[strIdx]’ using the LPS array.

  Doing this, we will find all occurrences of the pattern in the string.

Sample Code:

```
class exp10{
   void KMPSearch(String pat, String txt)
   {
      int M = pat.length();
      int N = txt.length();

      int lps[] = new int[M];
      int j = 0;

      computeLPSArray(pat, M, lps);

      int i = 0;
      while ((N - i) >= (M - j)) {
         if (pat.charAt(j) == txt.charAt(i)) {
            j++;
            i++;
         }
         if (j == M) {
            System.out.println("Found pattern "
                                   + "at index " + (i - j));
                j = lps[j - 1];
            }

            else if (i < N
                     && pat.charAt(j) != txt.charAt(i)) {
                if (j != 0)
                    j = lps[j - 1];
                else
                    i = i + 1;
            }
        }
    }

    void computeLPSArray(String pat, int M, int lps[])
    {
        int len = 0;
        int i = 1;
        lps[0] = 0;

        while (i < M) {
            if (pat.charAt(i) == pat.charAt(len)) {
                len++;
                lps[i] = len;
                i++;
            }
            else
            {
                if (len != 0) {
                    len = lps[len - 1];
                }
                else
                {
                    lps[i] = len;
                    i++;
                }
            }
        }
    }
    public static void main(String args[])
    {
        String txt = "ABABDABACDABABCABAB";
        String pat = "ABABCABAB";
        new exp10().KMPSearch(pat, txt);
    }
}
```

Observations/Outcome :



```
PS C:\Users\nisha>  & 'C:\Users\
odesws_e0be7\jdt_ws\jdt.ls-java-
Found pattern at index 10
PS C:\Users\nisha>
```

Time Complexity: O(N + M)