**Experiment 3.3**

**Aim:** *Develop a program and analyze complexity to find shortest paths in a graph with positive edge weights using Dijkstra’s algorithm.*

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

**Input/Apparatus Used:** *VS CODE*

# 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.*

# Code:

# *#include <iostream>*

# *#include <string>*

# *#include <vector>*

# *using namespace std;*

# *void computeLPSArray(const string &pat, vector<int> &lps) {*

# *int M = pat.length();*

# *int len = 0;*

# *lps[0] = 0;*

# *int i = 1;*

# *while (i < M) {*

# *if (pat[i] == pat[len]) {*

# *len++;*

# *lps[i] = len;*

# *i++;*

# *} else {*

# *if (len != 0) {*

# *len = lps[len - 1];*

# *} else {*

# *lps[i] = 0;*

# *i++;*

# *}*

# *}*

# *}*

# *}*

# *void KMPSearch(const string &pat, const string &txt) {*

# *int M = pat.length();*

# *int N = txt.length();*

# *vector<int> lps(M);*

# *int j = 0;*

# *computeLPSArray(pat, lps);*

# *int i = 0;*

# *while (i < N) {*

# *if (pat[j] == txt[i]) {*

# *j++;*

# *i++;*

# *}*

# *if (j == M) {*

# *cout << "Found pattern at index " << i - j << endl;*

# *j = lps[j - 1];*

# *} else if (i < N && pat[j] != txt[i]) {*

# *if (j != 0) {*

# *j = lps[j - 1];*

# *} else {*

# *i = i + 1;*

# *}*

# *}*

# *}*

# *}*

# *int main() {*

# *string txt = "ABABDABACDABABCABAB";*

# *string pat = "ABABCABAB";*

# *KMPSearch(pat, txt);*

# *return 0;*

# *}}*

# Observations/Outcome :

# 

# Time Complexity:

* *Time Complexity: Computing the LPS array takes O(M) time. The time complexity of the KMP algorithm is O(N + M). Here ‘N’ is the length of the string, and ‘M’ is the length of the pattern.*