**CHANDIGARH UNIVERSITY**

**UNIVERSITY INSTITUTE OF ENGINEERING**

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**



|  |  |
| --- | --- |
| **Submitted By:** Sahil Kaundal  **Submitted To:** Neha Dutta | |
| **Subject Name** | Design and Analysis of Algorithm Lab |
| **Subject Code** | 20CSP-312 |
| **Branch** | Computer Science Engineering |
| **Semester** | 5th |

**Experiment 10**

**Student Name:** Sahil Kaundal **UID:** 21BCS8197

**Branch:** BE CSE (Lateral Entry) **Section/Group:** 616/A

**Semester:** 5th **Date of Performance:** 10/11/2022

**Subject Name:** DAA Lab **Subject Code:** 21-CSP-312

# Aim/Overview of the practical:

Code and analyze to find all occurrences of a pattern P in a given string S.

# Task to be done/ Which logistics used:

Code and analyze to find all occurrences of a pattern P in a given string S.

# Requirements:

Laptop or PC.

Operation system (Mac, Windows, Linux, or any)

Vs-Code with MinGw or any C++ Compiler

1. **Algorithm/Flowchart (For programming-based labs):**

COMPUTE-PREFIX-FUNCTION (P)

m = P. length

let π [I .. m] be a new array π [1] = 0

k = 0

for q = 2 to m

while k > 0 and P [k + l] P [q] k = π [k]

if P [k + 1] == P [q]

k = k + 1 π [q] = k return π

KMP-MATCHER(T, P)

n = T.length m = P. length

π = COMPUTE-PREFIX-FUNCTION (P)

q = 0

for i = 1 to n

while q > 0 and P [q + l] ≠ T [i] q = π [q]

if P[q + 1] == T [i]

q = q + 1 if q == m

print "Pattern occurs with shift" i - m q = π [q]

# Steps for experiment/practical/Code:

#include <iostream> using namespace std;

void findPrefix(string pattern, int m, int prefArray[])

{

int length = 0;

prefArray[0] = 0; // first place is always 0 as no prefix for (int i = 1; i < m; i++)

{

if (pattern[i] == pattern[length])

{

}

else

{

length++;

prefArray[i] = length;

if (length != 0)

{

}

else

}

}

}

length = prefArray[length - 1];

i--; // decrease i to avoid effect of increasing after

prefArray[i] = 0;

void kmpPattSearch(string mainString, string pattern, int \*locArray, int &loc)

{

int n, m, i = 0, j = 0; n = mainString.size(); m = pattern.size();

int prefixArray[m]; // prefix array as same size of pattern findPrefix(pattern, m, prefixArray);

loc = 0;

while (i < n)

{

if (mainString[i] == pattern[j])

{

i++; j++;

}

if (j == m)

{

locArray[loc] = i - j; // item found at i-j position. loc++;

j = prefixArray[j - 1]; // get the prefix length from array

}

else if (i < n && pattern[j] != mainString[i])

{

if (j != 0)

j = prefixArray[j - 1];

else

i++;

}

}

}

int main()

{

string str = "ANKNANKANNANKAN";

string patt = "ANKAN";

int locationArray[str.size()]; int index;

kmpPattSearch(str, patt, locationArray, index); for (int i = 0; i < index; i++)

{

cout << "Pattern found at location: " << locationArray[i] << endl;

}

}

# Result/Output/Writing Summary:



**Learning outcomes (What I have learnt):**

* Algorithm of Knuth Morris Pratt (KMP)
* Complexity of KMP and Prefix function

**Evaluation Grid (To be created as per the SOP and Assessment guidelines by the faculty):**

|  |  |  |  |
| --- | --- | --- | --- |
| Sr. No. | Parameters | Marks Obtained | Maximum Marks |
| 1. |  |  |  |
| 2. |  |  |  |
| 3. |  |  |  |