**CONTENT BEYOND THE SYLLABUS**

**Ex.no:1**

**KNUTH-MORRIS- PRATT PATTERN MATCHING ALGORITHM**

**AIM**

To implementing Knuth-Morris- Pratt pattern matching algorithm.

**ALGORITHM**

1. Read the pattern and text.
2. Compute failure function.
3. Apply the KMP algorithm.

**Program**

public class KMPAlgorithm {

// Function to implement the KMP pattern matching algorithm

public static void KMP(String text, String pattern) {

int n = text.length();

int m = pattern.length();

// Create the longest prefix suffix (LPS) array

int[] lps = new int[m];

computeLPSArray(pattern, m, lps);

int i = 0; // index for text[]

int j = 0; // index for pattern[]

while (i < n) {

if (pattern.charAt(j) == text.charAt(i)) {

i++;

j++;

}

if (j == m) {

System.out.println("Found pattern at index " + (i - j));

j = lps[j - 1];

} else if (i < n && pattern.charAt(j) != text.charAt(i)) {

if (j != 0) {

j = lps[j - 1];

} else {

i++;

}

}

}

}

// Function to compute the LPS array

private static void computeLPSArray(String pattern, int m, int[] lps) {

int len = 0;

int i = 1;

lps[0] = 0;

while (i < m) {

if (pattern.charAt(i) == pattern.charAt(len)) {

len++;

lps[i] = len;

i++;

} else {

if (len != 0) {

len = lps[len - 1];

} else {

lps[i] = 0;

i++;

}

}

}

}

// Main function to test the KMP algorithm

public static void main(String[] args) {

String text = "ABABDABACDABABCABAB";

String pattern = "ABABCABAB";

KMP(text, pattern);

}

}

**Output**

**Found pattern at index 10**

**=== Code Execution Successful ===**

**RESULT**

Thus the Knuth-Morris- Pratt pattern matching algorithm has been implemented and the output has been verified successfully.

What is Wireshark?

Wireshark is a network packet analyzer. A network packet analyzer presents captured packet data in as

much detail as possible. You could think of a network packet analyzer as a measuring device for

examining whats happening inside a network cable, just like an electrician uses a voltmeter for examining

whats happening inside an electric cable (but at a higher level, of course)