Experiment No. 12
Naïve String matching
Date of Performance:
Date of Submission:

Experiment No. 12

Title: Naïve String matching

Aim: To study and implement Naïve string matching Algorithm

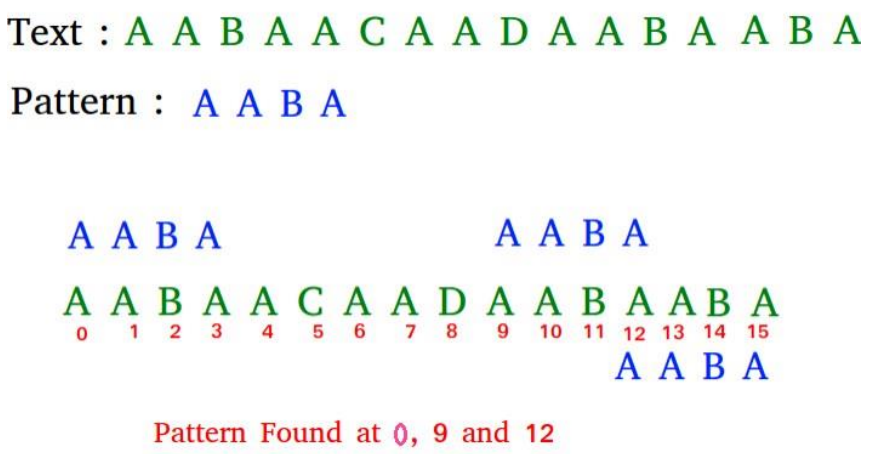
Objective: To introduce String matching methods

Theory:

The naïve approach tests all the possible placement of Pattern P [1.....m] relative to text T [1.....n]. We try shift s = 0, 1.....n-m, successively and for each shift s. Compare T [s+1.....s+m] to P [1.....m].

The naïve algorithm finds all valid shifts using a loop that checks the condition P [1.....m] = T [s+1.....s+m] for each of the n - m +1 possible value of s.

Example:



Algorithm:

THE NAIVE ALGORITHM

The naive algorithm finds all valid shifts using a loop that checks the condition P[1….m]=T[s+1…. s+m] for each of the n-m+1 possible values of s.(P=pattern , T=text/string , s=shift)

```
NAIVE-STRING-MATCHER(T,P)
1) n = T.length
2)  m = P.length
3) for s=0 to n-m
4)        if P[1…m]==T[s+1….s+m]
5)              printf" Pattern occurs with
   shift " s
```

Code :

```
#include <stdio.h>
#include <string.h>

// Function to perform naive string matching
void naiveStringMatch(char text[], char pattern[]) {
   int n = strlen(text);
   int m = strlen(pattern);
   int count = 0;

   // Iterate through each position of the text
   for (int i = 0; i <= n - m; i++) {
      int j;
      // Check if the pattern matches the substring starting at position i
      for (j = 0; j < m; j++) {
         if (text[i + j] != pattern[j])
            break; // mismatch found, break the loop
      }
      if (j == m) {
```

```
            printf("Pattern found at index %d\n", i);
            count++;
        }
    }

    if (count == 0)
        printf("Pattern not found in the text.\n");
}

int main() {
    char text[] = "AABAACAADAABAAABAA";
    char pattern[] = "AABA";

    printf("Text: %s\n", text);
    printf("Pattern: %s\n", pattern);

    printf("Occurrences of pattern in the text:\n");
    naiveStringMatch(text, pattern);

    return 0;
}
```

Output :

```
PS E:\Testing_Lang> cd "e:\Testing_Lang\" ; if ($?) { gcc test.c -o test } ; if ($?) { .\test }
Text: AABAACAADAABAAABAA
Pattern: AABA
Occurrences of pattern in the text:
Pattern found at index 0
Pattern found at index 9
Pattern found at index 13
PS E:\Testing_Lang>
```

Conclusion: Naïve string-matching algorithm has been successfully implemented. In this program, the naiveStringMatch function iterates through each position of the text and checks if the pattern matches the substring starting at that position. If a match is found, it prints the index where the match occurred.

The time complexity of the Naive String-Matching Algorithm depends on the length of the text n and the length of the pattern m. In the worst-case scenario, where the pattern matches at every position of the text, the time complexity is O((n-m+1)m). However, in typical cases, the algorithm tends to have a time complexity closer to O(nm) because the average number of comparisons made is proportional to the length of both the text and the pattern.