Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

AY: 2024-25

Class:	**SE**	**Semester:**	**IV**
Course Code:	**CSL401**	**Course Name:**	**Analysis of Algorithm Lab**

Name of Student:	**Shravani Sandeep Raut**
Roll No. :	**48**
Experiment No.:	**10**
Title of the Experiment:	**Naive String Matching algorithm**
Date of Performance:	**20/03/2025**
Date of Submission:	**27/03/2025**

Evaluation

Performance Indicator	Max. Marks	Marks Obtained
Performance	5	
Understanding	5	
Journal work and timely submission	10	
Total	20	

Performance Indicator	Exceed Expectations (EE)	Meet Expectations (ME)	Below Expectations (BE)
Performance	4-5	2-3	1
Understanding	4-5	2-3	1
Journal work and timely submission	8-10	5-8	1-4

Checked by

Name of Faculty : Mrs. Sneha Yadav

Signature :

Date:

Experiment No. 12

Title: Naïve String matching

Aim: To study and implement Naïve string matching Algorithm

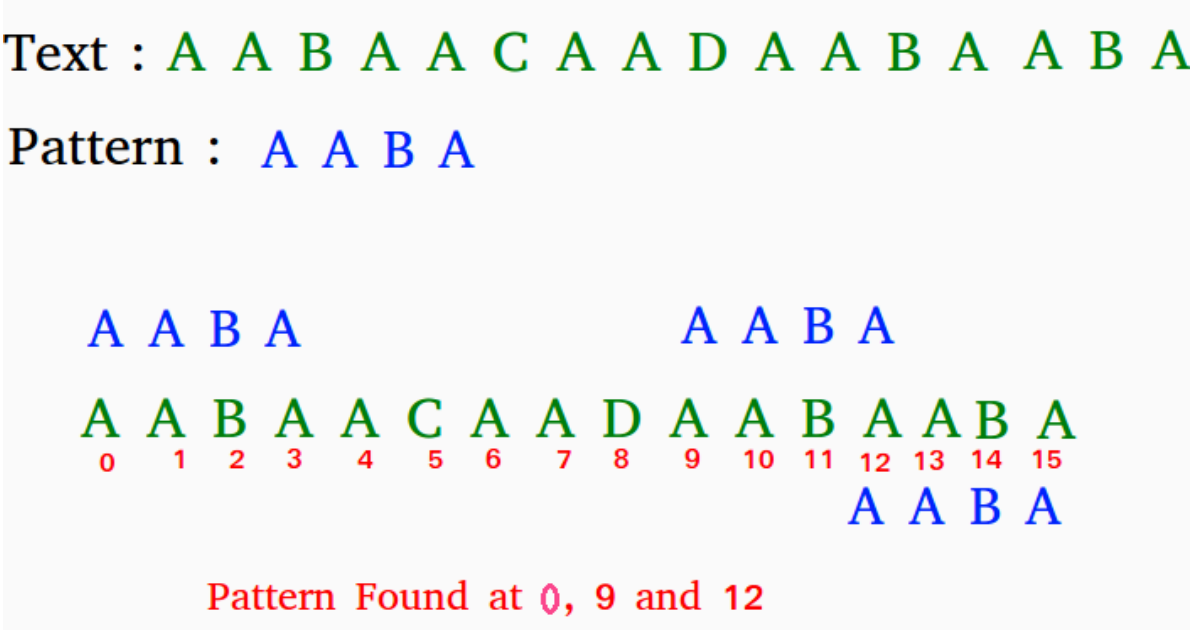
Objective: To introduce String matching methods

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)

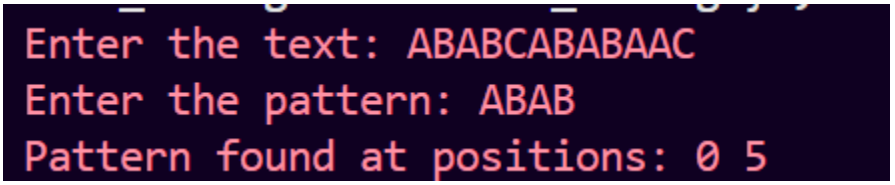
NAIVE-STRING-MATCHER(T,P)

```
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
```

Implementation:

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

char text[100], pattern[20];
int n, m, i, j;

void naiveStringMatch();

void main() {
   printf("Enter the text: ");
   gets(text);

   printf("Enter the pattern: ");
   gets(pattern);

   naiveStringMatch();
}
```

```
void naiveStringMatch() {
    n = strlen(text);
    m = strlen(pattern);

    printf("Pattern found at positions: ");
    int found = 0;

    for (i = 0; i <= n - m; i++) {
        for (j = 0; j < m; j++) {
            if (text[i + j] != pattern[j])
                break;
        }

        if (j == m) {
            printf("%d ", i);
            found = 1;
        }
    }

    if (!found)
        printf("No match found.");

    printf("\n");
}
```



Conclusion:

The naive string matching algorithm is a straightforward approach to finding a pattern within a text. It works by checking for the pattern at every possible position in the text, one by one. Although simple and easy to implement, it can be inefficient for large texts or patterns due to its time complexity of $O((n-m+1)\times m)$, where n is the length of the text and m is the length of the pattern. Despite its limitations, the naive algorithm is useful for understanding basic string processing concepts and serves as a foundation for more advanced string matching techniques.