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

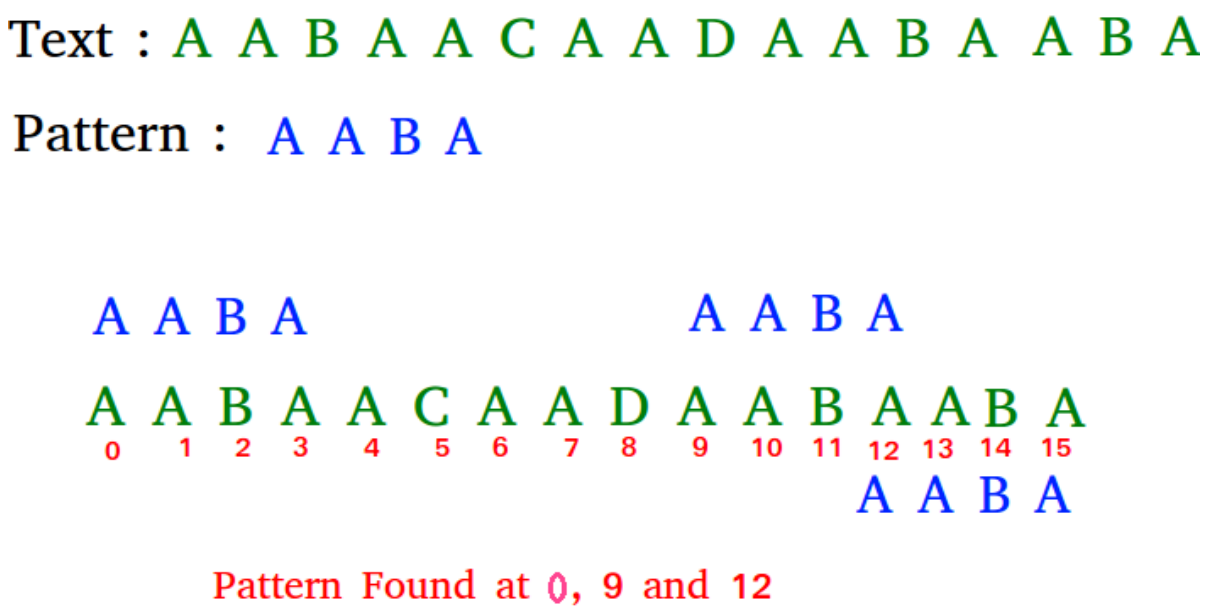
Objective: To introduce String matching methods

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)

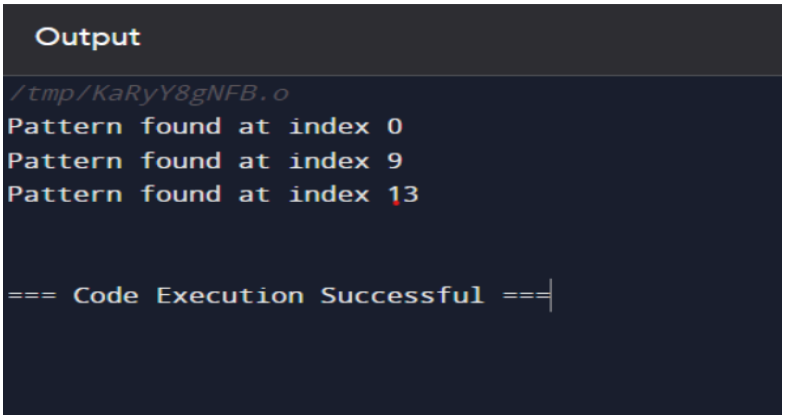
NAIVE-STRING-MATCHER(T,P)

```
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
```

Implementation:

```
#include <stdio.h> #include <string.h>
void search(char* pat, char* txt)
{
int M = strlen(pat); int N = strlen(txt);
/* A loop to slide pat[] one by one */ for (int i = 0; i <= N - M; i++) {
int j;
/* For current index i, check for pattern match */ for (j = 0; j < M; j++)
if (txt[i + j] != pat[j])
break;
if (j
== M) // if pat[0...M-1] = txt[i, i+1, ...i+M-1] printf("Pattern found at index %d \n", i);
}
}
int main()
{
char txt[] = "MAITREEPIMPLESANSKRUTIPARNIKABHAKTI";
char pat[] = "AABA";
// Function call search(pat, txt); return 0;
}
```

Output:



Conclusion:

The Naive String-Matching algorithm is a basic approach that serves as a starting point for understanding string-matching techniques. It's suitable for small-scale applications or educational purposes but may not be optimal for large-scale or performance-critical scenarios.