|  |
| --- |
| 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

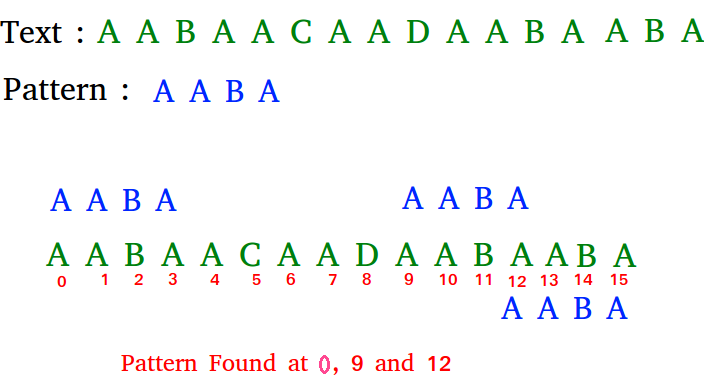
**Objective:** To introduce String matching methods

**Theory:**

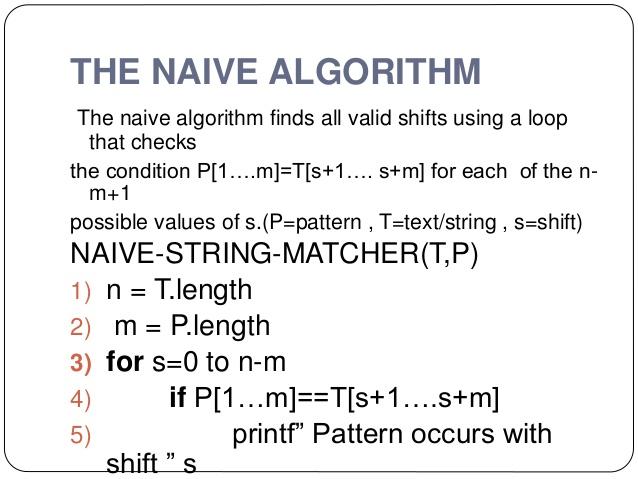
The naïve approach tests all the possible placement of Pattern P [1.......m] relative to text T [1......n]. We try shift s = 0, 1.......n-m, successively and for each shift s. Compare T [s+1.......s+m] to P [1......m].

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:**



**Implementation:**

// C program for Naive Pattern Searching algorithm

#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);

}

}

// Driver's code

int main()

{

char txt[] = "AABCCAABC";

char pat[] = "AB";

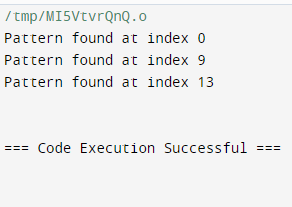
// Function call

search(pat, txt);

return 0;

}

**Output:**



**Conclusion:** Experiment underscores the utility of the naive string matching algorithm in efficiently locating occurrences of a pattern within a text. While straightforward in approach, its effectiveness in basic string searching tasks highlights its foundational significance in algorithmic design and serves as a benchmark for more complex pattern matching algorithms.