



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

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



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

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 $s = 0, 1, \dots, n-m$, 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:

Text : A A B A A C A A D A A B A A B A

Pattern : A A B A

A	A	B	A									A	A	B	A
A	A	B	A	A	C	A	A	D	A	A	B	A	A	B	A
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
												A	A	B	A

Pattern Found at 0, 9 and 12



Algorithm:

THE NAIVE ALGORITHM

The naive algorithm finds all valid shifts using a loop that checks

the condition $P[1\dots m] = T[s+1\dots s+m]$ 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** $P[1\dots m] == T[s+1\dots s+m]$
- 5) **printf** "Pattern occurs with
 shift " s

Implementation:

```
#include <stdio.h>
#include <string.h>
int match(char [], char []);
int main() {
    char a[100], b[100];
    int position;
    printf("Enter some text\n");
    gets(a);
    printf("Enter a string to find\n");
    gets(b);
```



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

```
position = match(a, b);
if (position != -1) {
    printf("Found at location: %d\n", position + 1);
}
else {
    printf("Not found.\n");
}
return 0;
}

int match(char text[], char pattern[]) {
    int c, d, e, text_length, pattern_length, position = -1;
    text_length = strlen(text);
    pattern_length = strlen(pattern);

    if (pattern_length > text_length) {
        return -1;
    }
    for (c = 0; c <= text_length - pattern_length; c++) {
        position = e = c;
        for (d = 0; d < pattern_length; d++) {
            if (pattern[d] == text[e]) {
                e++;
            }
            else {
                break;
            }
        }
        if (d == pattern_length) {
```



Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

```
    return position;  
}  
}  
return -1;  
}
```

Output:

```
Enter some text  
aabbccddaabbdhgaaabbcc  
Enter a string to find  
aabbcc  
Found at location: 1  
  
=== Code Execution Successful ===|
```

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.