SNS COLLEGE OF ENGINEERING

Coimbatore-107

COURSE NAME: ANALYSIS OF ALGORITHM

II YEAR/ IV SEMESTER

UNIT – IV

STRING MATCHING ALGORITHM

Topic

Naive String Matching Algorithm

Unit-4

Naive String Matching Algorithm

It is the simplest method to find pattern (substring) in a text.

Consider:

1). T = Text of Length 'n'
2). P = Pattern of Length 'm'.

⇒ Search from beginning of the text.
⇒ For each position 'i' from 0 to n-m:
* Compare the substring T[i...i+m-1] with 'p'.
* If they match, record the position.

Example:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Text:	A	B	A	B	A	B	C	A	B	A	B	A	B	C	A	B	A	B

Pattern: "ABABC"

Step 1:
Compare pattern with position '0' → No Match.

Step 2: At position '2':
Compared & matched. (9-6)

Step 3: At position '9':
Compared & matched (9-13)

Algorithm:

```
for (i=0; i<=n-m; i++)
{
for (j=0; j<m; j++)
{
if (T[i+j] != P[j])
break;
}
if (j==m)
print ("patter found at index", i);
}
```

Analysis:

⇒ Time Complexity:

- In worst case, it compares all 'm' characters for each position 'j'
  * Almost 'm' comparisons at each n-m+1 positions leads to

O(n*m)

In Best case: Algorithm compares only the first character at each position & fails, lead to 'n' comparison.

O(n)

In Average case: pattern & text are randomly distributed & don't match lead to O(n)

Space Complexity:

It uses few variables (i, j) & No extra array or Data Structure needed.

$S(n) = O(1)$ constant.