Danyl Fernandes
2020012004 (72)
26-04-2021

AOA Experiment 9

Aim:

To implement & analyze Naive String matching algorithm:

Implementation:

```c
#include<stdio.h>
#include<string.h>
void Naive_string_matcher(char *T, char *P)
{
int m = strlen(T);
int n = strlen(P);
for(int s = 0; s ⩽ m - n; s++)
{
int x;
for(x = 0; x < n; x++)
if(T[s + x] ≠ P[x])
{
break;
}
if(x == n)
{
printf("Pattern Occurs with shift:\t%d\n", s);
}
}
}
int main()
{
char str[30], pattern[30];
printf("\nEnter a Text:\t");
scanf("%s", str);
printf("\nEnter a Pattern:\t");
scanf("%s", pattern);
Naive_string_matcher(str, pattern);
return 0;
}
```

Output:

```
Enter a Text:   moon

Enter a Pattern:        on
Pattern Occurs with shift:      2

--------------------------------
Process exited after 5.402 seconds with return value 0
Press any key to continue . . .
```

Daryl Fernandes
2020012004 (72)

Exp 09 :

Theory

- This algorithm follows the brute force approach.
- Let an array T[1:n] is the given base string of length n and an array P[1:m] is the pattern of length m that is to be searched for in T.
- The algorithm slides the pattern P across the base string T looking for a match
- Let s is the number of times P has been shifted when the algorithm find P in T, it outputs s by indicating that "Pattern occurs with shift s".
- This algorithm does not give an optimal solution since it ignores the information gained about the base string T for one value of s while considering the other values of s.
- We can find substring by checking once for the string. It does not occupy extra space to perform operations

Daryl Fernandes
20202012004 (72)

Algorithm :

- The naive algorithm finds all valid shifts using a loop that checks the condition $P[1 \ldots m] = T[s+1 \ldots s+m]$ for each of the $n-m+1$ possible values of s

```
Naive-String-Matcher (T, P)
    n = T.length
    m = P.length
    for s=0 to n-m
        if P[1...m] == T[s+1 .. s+m]
            print "Pattern occurs with shifts
```

Analysis :

- The time complexity of the naive string matching algorithm is $O((n-m+1)m)$, where $n = |T| + 1$ = the length of a string being searched & $m = |P|$ = the length of a pattern substring being compared.

Example

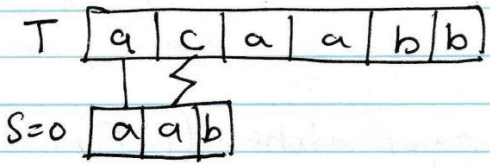
- Consider a string matching problem instance: $T = acabb$ & $P = aab$. The working of the naive string matching algorithm is as shown

- The vertical straight line connects the

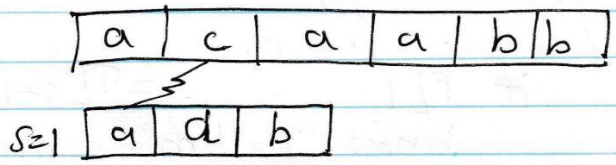
Daryl Fernandes
20200120004 (72)

matching character & a jagged line connects the first mismatching character if any. In this example, the algorithm finds the matching pattern with shift s=2

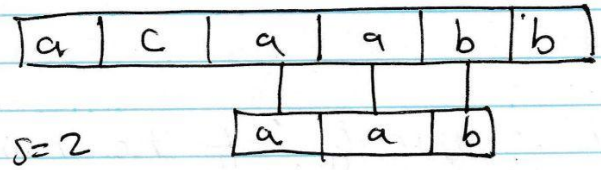
a) T



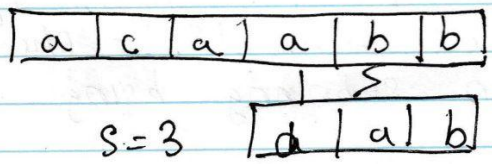
b)



c)



d)



Conclusion:

- The implementation of Naive string matching algorithm with the analysis was successful.