

EXPERIMENT 14

Experiment No: 14

Date: 13/06/2021

Aim: Implementation of Brute Force and BM Pattern Matching Algorithm

Theory:

Brute Force Approach:

- It is an intuitive, direct, and straightforward technique of problem-solving in which all the possible ways or all the possible solutions to a given problem are enumerated.
- Many problems solved in day-to-day life using the brute force strategy, for example exploring all the paths to a nearby market to find the minimum shortest path.
- Arranging the books in a rack using all the possibilities to optimize the rack spaces, etc.
- In fact, daily life activities use a brute force nature, even though optimal algorithms are also possible.

Boyer Moore Approach:

- Robert Boyer and J Strother Moore established it in 1977.
- The B-M String search algorithm is a particularly efficient algorithm and has served as a standard benchmark for string search algorithm ever since.
- The B-M algorithm takes a 'backward' approach: the pattern string (P) is aligned with the start of the text string (T), and then compares the characters of a pattern from right to left, beginning with rightmost character.
- If a character is compared that is not within the pattern, no match can be found by analysing any further aspects at this position so the pattern can be changed entirely past the mismatching character.

EXPERIMENT 14

- For deciding the possible shifts, B-M algorithm uses two pre-processing strategies simultaneously.
- Whenever a mismatch occurs, the algorithm calculates a variation using both approaches and selects the more significant shift thus, if make use of the most effective strategy for each case.
- The two strategies are called heuristics of B - M as they are used to reduce the search. They are:
 - Bad Character Heuristics
 - Good Suffix Heuristics

Brute Force Pattern Matching:

- In Brute Force Pattern Matching algorithm, we test all the possible placements of P relative to T.
- This algorithm consists of two nested loops.
- The outer loop indexing through all possible starting indices of the pattern in the text, and the inner loop indexing through each character of the pattern, comparing it to its potentially corresponding character in the text.
- If the pattern is found than the starting index of P in T is returned.

The Boyer-Moore Algorithm:

- In the Brute Force Pattern Matching algorithm, we need to examine every character in T in order to locate pattern P.
- But the Boyer-Moore (BM) algorithm matching algorithm can sometimes avoid comparisons between P and a sizeable fraction of the characters in T.
- The main idea is to improve the running time of the Brute-Force algorithm by adding two potentially two potentially time saving heuristics:
 - **Looking-Glass Heuristic:**
 - When testing a possible placement of P against T, start the comparisons from the end of P and move backward to the front of P.

EXPERIMENT 14

○ Character-Jump Heuristic:

- During the testing of a possible placement of P against T, a mismatch of text character $T[i] = c$ with the corresponding pattern $P[j]$ is handled in the following manner.
- If c is not present anywhere in P, then shift P completely past $T[i]$ because it cannot match any character in P.
- Otherwise shift P until an occurrence of character c in P gets aligned with $T[i]$.

Time Complexity

- The time complexity of the Brute Force Pattern Matching algorithm and Boyer-Moore algorithm is $O(nm)$ where n is the number of characters in the text string and m is the number of characters in the pattern string.

Tracing With Examples:

Brute Force Pattern Matching Algorithm

Text : $xyxz\ yxx\ yzzxy$

Pattern : $yxxyz$

x	y	x	z	y	x	x	y	z	z	x	y
---	---	---	---	---	---	---	---	---	---	---	---

1
x x x y c

2 3 4
x x x y c

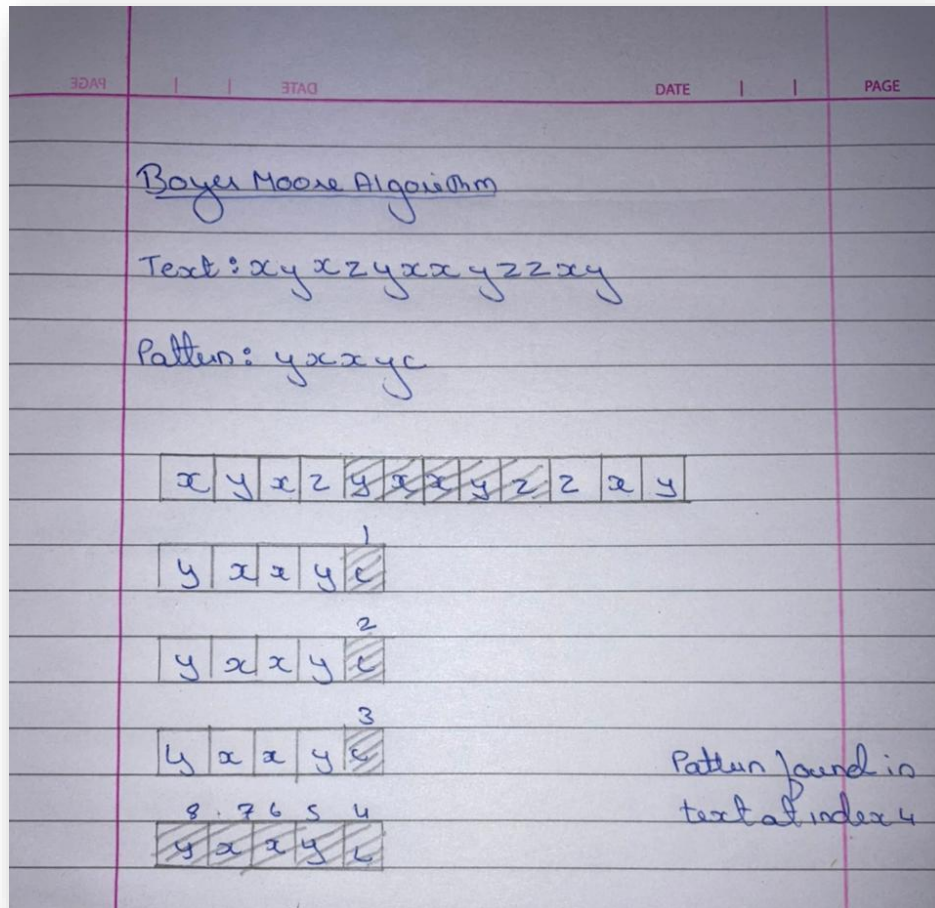
5
x x x y c

6
x x x y c

7 8 9 10 11
x x x y c

Pattern found in text at index 4

EXPERIMENT 14



Algorithm 1: Brute-Force Pattern Matching

Algorithm Brute ForceMatch (T, P):

Input: Strings T (text) with n characters and P (pattern) with m characters

Output: Starting index of the first substring of T matching P, or an indication that P is not a substring of T

for $i=0$ to $n-m$ (for each candidate index in T) do

$j=0$

 while ($j < m$ and $T[i+j]=P[j]$) do

$j=j+1$

EXPERIMENT 14

```
    if J==m then  
        return i  
  
    return "There is no substring of 7 matching P"
```

Algorithm 2: The Boyer Moore Pattern Matching

Algorithm BMMatch(T,P):

Input : Strings T with n characters and P pattern with m charcaters

Output: String index of the first substring of T matching P or an indication
that P is not a substring of T

compute finction last

i<-m-1

j<-m-1

repeat

if P[i]==p[j] then

if j==0 then

return i

else

i<-i-1

j<-j-1

else

i<- i+m-min(j,1+last(T(i)))

j<-m-1

until i>n-1

return "There is no such substring of T matching P"

EXPERIMENT 14

Program 1: Brute Force Approach

```
//BRUTE FORCE APPROACH

#include<bits/stdc++.h>

using namespace std;

int main()

{

    string a,b;

    int c,d;

    cout<<"*****"<<endl;

    cout<<"Enter The String:"<<endl;

    cin>>a;

    cout<<"*****"<<endl;

    cout<<"Enter The Pattern:"<<endl;

    cin>>b;

    cout<<"*****"<<endl;

    c=a.size();

    d=b.size();

    int x=0;

    for(int i=0;i<=c-d;i++)

    {

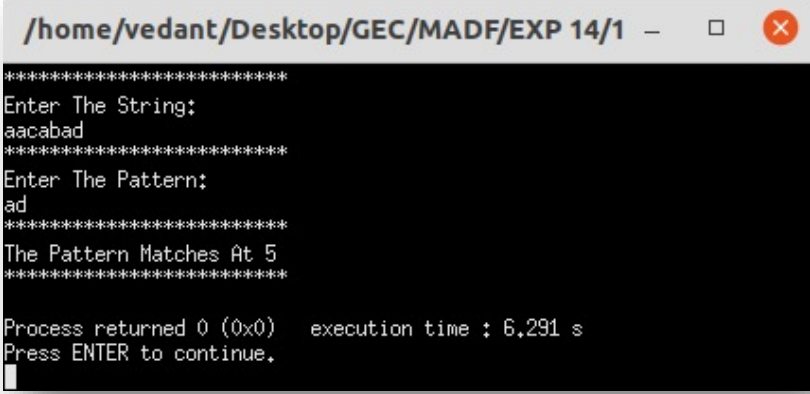
        int j=0;

        while(j<d&& a[i+j]==b[j])
```

EXPERIMENT 14

```
j++;  
  
if(j==d)  
{  
  
    cout<<"*****"<<endl;  
  
    cout<<"The Pattern Matches At "<<i<<endl;  
  
    cout<<"*****"<<endl;  
  
    x=1;  
  
    break;  
}  
  
}  
  
if(x==0)  
  
    cout<<"No Such Pattern Available"<<endl;  
  
}
```

Output



```
/home/vedant/Desktop/GEC/MADF/EXP 14/1  
*****  
Enter The String:  
aacabad  
*****  
Enter The Pattern:  
ad  
*****  
The Pattern Matches At 5  
*****  
Process returned 0 (0x0)   execution time : 6.291 s  
Press ENTER to continue.
```

EXPERIMENT 14

Program 2: The Boyer Moore Approach

```
#include<bits/stdc++.h>

using namespace std;

int min(int a,int b)
{
    if(a<b) return a;

    return b;
}

int main()
{
    string a,b;

    int c,d;

    cout<<"*****"<<endl;

    cout<<"Enter The String:"<<endl;

    cin>>a;

    cout<<"*****"<<endl;

    cout<<"Enter The Pattern:"<<endl;

    cin>>b;

    cout<<"*****"<<endl;

    c=a.size();

    d=b.size();

    int x=0;
```


EXPERIMENT 14

```
vector<int> last(27,-1);

for(int i=0;i<d;i++)

{

    last[int(b[i])-97]=i;

}


int i=d-1,j=d-1;

do{

    if(b[j]==a[i])

    {

        if(j==0)

        {

            x=1;

            cout<<"*****"<<endl;

            cout<<"The Pattern Matches At "<<i<<endl;

            cout<<"*****"<<endl;

            break;

        }

        i--;

        j--;

    }else{
```

EXPERIMENT 14

```
        cout<<a[i]<<b[j]<<endl;

        i=i+d-min(j,last[int(a[i]-97)]+1);

        j=d-1;

    }

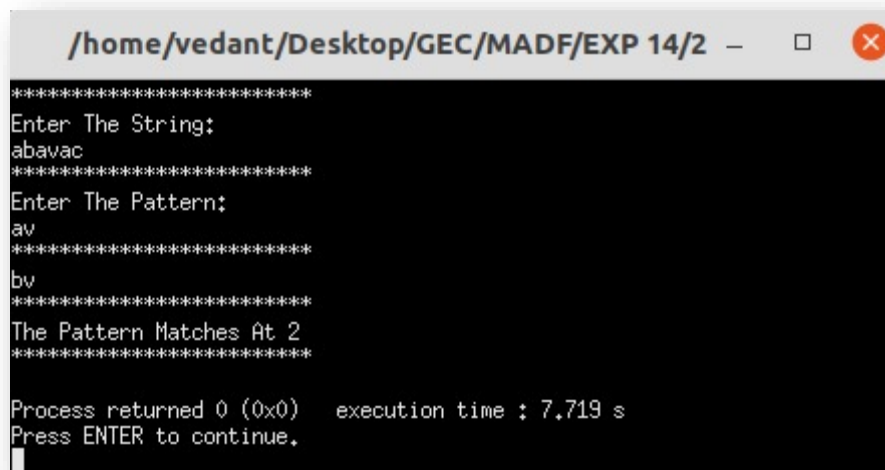
    }while(i<c);

    if(x==0)

        cout<<"No Such Pattern Available"<<endl;

}
```

Output



```
/home/vedant/Desktop/GEC/MADF/EXP 14/2
*****
Enter The String:
abavac
*****
Enter The Pattern:
av
*****
bv
*****
The Pattern Matches At 2
*****
Process returned 0 (0x0)   execution time : 7.719 s
Press ENTER to continue.
█
```

Conclusion

- Detailed concept of Brute Force Approach and Boyer Moore Approach was studied successfully.
- Programs using Brute Force Algorithm and Boyer Moore Algorithm were executed successfully.