ABSTRACT:

This is a project on string matching where you can find the position of the pattern and the time required for finding it by using various algorithms. We have used the following in our program:
Naive sting matching
Horspool string matching
Rabin Karp
Knuth Morris Pratt (KMP)

In Naive string matching, it checks each letter in the text with the pattern's first latter and if it matches, it then checks for the remaining pattern letters with the text.

In Horspool, a shift table is created where all letters except the pattern letters are given the length of the pattern to shift while the pattern's letters are given a shift value of m-1-j where m is the pattern length and j is a counter. With the help of the shift table string matching is done.

In Rabin Karp, a hash value is created and it can be made up by various methods. We have chosen to create the hash value with the help of a prime number. It first creates a hash value for the pattern and then for each window in the text of the pattern length. If both the hash value matches then each letter in both text and pattern is matched.

In KMP, a pi table is used to find a partial string match within the pattern to avoid recomputing the matches .The pi table function preprocesses the pattern to find matches of the prefixes of the pattern with the patern itself . it also how much of the last comparison can be reused if it fails. The algorithm avoids backtracking on the text string .

Out of all the algorithms we have used, KMP algorithm is the best and also it has a efficiency of O(n) to compare the pattern to the text in its worst case.

ALGORITMS:

1.NAVIE STRING MATCHING

Algorithm NaiveStringMatching(T[0..n-1],P[0..m-1])

```
//Implements brute-force string matching.
//Input: An array T[0..n-1] of n chars representing a text
// and an array P[0..m-1] of m chars representing a pattern.
//Output: The index of the first character in the text
//that starts a matching substring
//or -1 if the search is unsuccessful.
for i ← 0 to n-m
        j ← 0
        while (j < m) and (P[j] = T[i+j]) do
                j ← j + 1
        endwhile
        if (j = m) return i
return -1
```

2.HORSPOOL MATCHING

Algorithm HorspoolMatching(T[0..n-1], P[0..m-1])

```
        Table[alphabet size] ← ShiftTable(P[0..m-1])
        i ← m-1
        while (i < n)
                j ← 0
                while (j < m and T[i-j] = P[m-1-j])
                        j ← j + 1
                if (j = m) return i-(m-1)
                i ← i + Table[ T[i] ]
        return -1
```

Algorithm ShiftTable(P[0…m-1])

//Fills the shift table used by Horspool's algorithms
//Inputs: Pattern P[0….m-1] and an alphabet of the possible characters
//Output: Table[0…size-1] indexed by the alphabet of the possible characters and
Filled with shift sizes.

```
for i←0 to size -1 do Table[i] ←m
for j←0 to m-2 do Table[P[j]] ←m-1-j
return Table
```

3.RABIN KARP

Algorithm RabinKarp(P[0…..m-1] , T[0…..n-1] , int q)

//Matches the given pattern with the text using hash codes.
//Inputs: Pattern P[0…m-1] , Text T[0…..n-1 and a prime no q
//Output: The index of the **first** character of the pattern in the text that starts a matching
//substring or -1 if the search is unsuccessful

```
phash ←0  //pattern hash value
thash  ←0 //text hash value
        h  ←1
for i ←0 to m-2 do
      h ← (h*d)%q  //calculates the base value
for i←0 to m-1 do
      phash ← (d*phash + P[i])%q //calculate hash value of pattern
      thash ←  (d*thash + T[i])%q //calculate hash value of first m characters
                                          //of the text
for i←0 to n-m do
      if phash = thash do
            for j←0 to m-1 do
                  if  T[i+j] != P[i] do
                        break
            if j=m return i
      if i < n-m do
            //Deleting the hash value of the first character of the previous block and     //add the
            hash value of the last character in the current block
            thash ← (d*(thash – T[i]*h) + T[i+m])%q
            //Assign the prime value to text hash value if negative
```

```
            if thash < 0  do  (thash ←thash+q)
return  -1
```

4. KNUTH MORRIS PRATT

Algorithm KMP(T[0…n-1],P[0….m-1])

```
//Matches the given pattern with the text using pi table.
//Inputs: Pattern P[0…m-1] , Text T[0…..n-1]
//Output: The index of the first character of the pattern in the text that starts a matching
// substring or -1 if the search is unsuccessful
Pi[m] ← Pi_Table(P[0…m-1])
     k  ← -1
for i←0 to n-1 do
       while k < -1 and P[k+1] != T[i]
              k ← Pi[k]
       if T[i] = P[k+1] do
              k=k+1
       if k = m-1do    return i-k
return -1
```

Algorithm Pi_Table(P[0….m-1])

```
//Fill the Pi table used in KMP algorithm
//Inputs: Pattern P[0….m-1]
//Output: Pitable[0…size–1] indexed by the alphabet of the pattern filled with prefixes

k ← -1
i  ← 1
PiTable[0] = k
for i ← 1 to m-1 do
       while k > -1 and P[k+1] != P[i] do
              k ← PiTable[k]
       if P[i] = P[k+1] do
              k ← k+1
       PiTable[i] ← k
return PiTable
```

OUTPUT:
1.NAVIE STRING MATCHING

```
jyosna@jyosna:~/Desktop/daaprojectP2$ ./a.out

Enter Text: Hi this is Jyosna and Rachana . This is our DAA project StringMatching library.

Enter pattern : DAA


STRING MATCHING LIBRARY MENU
1.Naive
2.Horspool
3.Rabin Karp
4.Knuth Morris prath
5.EXIT


Enter your option:      1
The pos is : 44
0.000001 sec
```

2.HORSPOOL MATCHING

```
jyosna@jyosna:~/Desktop/daaprojectP2$ ./a.out

Enter Text: Hi this is Jyosna and Rachana . This is our DAA project StringMatching library.

Enter pattern : DAA


STRING MATCHING LIBRARY MENU
1.Naive
2.Horspool
3.Rabin Karp
4.Knuth Morris prath
5.EXIT


Enter your option:      2
The position is:44
0.000015 sec
```

3.RABIN KARP

```
jyosna@jyosna:~/Desktop/daaprojectP2$ ./a.out

Enter Text: Hi this is Jyosna and Rachana . This is our DAA project STringMatching library.

Enter pattern : DAA


STRING MATCHING LIBRARY MENU
1.Naive
2.Horspool
3.Rabin Karp
4.Knuth Morris prath
5.EXIT


Enter your option:      3
The position is:44
0.000002 sec
```

4. KNUTH MORRIS PRATT

```
jyosna@jyosna:~/Desktop/daaprojectP2$ ./a.out

Enter Text: Hi this is Jyosna and Rachana . This is our DAA project StringMatching library.

Enter pattern : DAA


STRING MATCHING LIBRARY MENU
1.Naive
2.Horspool
3.Rabin Karp
4.Knuth Morris prath
5.EXIT


Enter your option:      4
The position is:44
0.000003 sec
```