ANALYSIS & DESIGN OF ALGORITHIM

PRACTICAL-2

190031968

Kinthali praneeth

IN-LAB:

1)

```
def search(pat, txt):
  M = len(pat)
  N = len(txt)

  # A loop to slide pat[] one by one */
  for i in range(N - M + 1):
    j = 0

    # For current index i, check
    # for pattern match */
    while(j < M):
      if (txt[i + j] != pat[j]):
        break
      j += 1

    if (j == M):
      print("Pattern found at index ", i)

# Driver Code
if __name__ == '__main__':
  txt = "AABAACAADAABAAABAA"
  pat = "AABA"
  search(pat, txt)
```

PYTH 3.6 (Python 3.6)

```python
def search(pat, txt):
  M = len(pat)
  N = len(txt)

  # A loop to slide pat[] one by one */
  for i in range(N - M + 1):
    j = 0

    # For current index i, check
    # for pattern match */
    while(j < M):
      if (txt[i + j] != pat[j]):
        break
      j += 1

    if (j == M):
      print("Pattern found at index ", i)

# Driver Code
if __name__ == '__main__':
  txt = "AABAACAADAABAAABAA"
  pat = "AABA"
  search(pat, txt)

```

Status Successfully executed **Date** 2021-08-04 16:05:43 **Time** 0.03 sec **Mem** 17.968 kB

Output

```
Pattern found at index  0
Pattern found at index  9
Pattern found at index  13
```

2)

```
# Python program for KMP Algorithm
def KMPSearch(pat, txt):
 M = len(pat)
 N = len(txt)
 lps = [0]*M
 j = 0 # index for pat[]
 computeLPSArray(pat, M, lps)
 i = 0
 while i < N:
  if pat[j] == txt[i]:
   i += 1
```

```
   j += 1

  if j == M:
   print ("Found pattern at index " + str(i-j))
   j = lps[j-1]

  elif i < N and pat[j] != txt[i]:

   if j != 0:
    j = lps[j-1]
   else:
    i += 1
def computeLPSArray(pat, M, lps):
 len = 0 # length of the previous longest prefix suffix

 lps[0] # lps[0] is always 0
 i = 1
 while i < M:
  if pat[i]== pat[len]:
   len += 1
   lps[i] = len
   i += 1
  else:

   if len != 0:
    len = lps[len-1]

    # Also, note that we do not increment i here
   else:
    lps[i] = 0
    i += 1

txt = "b c m a l m n x y z"
pat = " m a l"
KMPSearch(pat, txt)
```

PYTH 3.6 (Python 3.6)

```python
# Python program for KMP Algorithm
def KMPSearch(pat, txt):
 M = len(pat)
 N = len(txt)

 lps = [0]*M
 j = 0 # index for pat[]

 computeLPSArray(pat, M, lps)

 i = 0
 while i < N:
  if pat[j] == txt[i]:
   i += 1
   j += 1

  if j == M:
   print ("Found pattern at index " + str(i-j))
   j = lps[j-1]

  elif i < N and pat[j] != txt[i]:

   if j != 0:
    j = lps[j-1]
   else:
    i += 1

def computeLPSArray(pat, M, lps):
```

PYTH 3.6 (Python 3.6)

```python
    j = lps[j-1]
   else:
    i += 1

def computeLPSArray(pat, M, lps):
 len = 0 # length of the previous longest prefix suffix

 lps[0] # lps[0] is always 0
 i = 1

 while i < M:
  if pat[i]== pat[len]:
   len += 1
   lps[i] = len
   i += 1
  else:

   if len != 0:
    len = lps[len-1]

    # Also, note that we do not increment i here
   else:
    lps[i] = 0
    i += 1

txt = "b c m a l m n x y z"
pat = " m a l"
KMPSearch(pat, txt)
```

Status Successfully executed **Date** 2021-08-04 16:09:28 **Time** 0.02 sec **Mem** 17.968 kB

Output

```
Found pattern at index 3
```