Name → Akash Sharma

Section → I.T. (B.Tech)

Enrol/University Roll No → ~~B.Tech~~ 19021732

Date → 28-Oct-2021

Subject → D.A.A

Class Roll No → 08

University Roll No → 2015488

Semester → 5th Semester

Akash Sharma

Naive - string Matching Algorithm.

The Naive approach test all the possible placement of Pattern P[1...m] relative to text T[1... n]. we try shift S = 0,1 ..n-m, successively and for each shift s. Compare T [s+1.... s+m] to P[1...m].

The Naive algorithm finds all valid shifts using a loop that checks the condition P[1...m] = T[s+1... s+m] for each of the n-m+1 possible value of s.

Naive-string - Matcher (T, P)

1. n ← length[T]
2. m ← length[P]
3. for s ← 0 to n-m
4. do if P[1...m] = T[s+1.... s+m]
5. Then print "Pattern occurs with shift" s.

Rabin - Karp - Algorithm

The Rabin- Karp string matching algorithm calculates a hash value for the pattern, as well as for each M-character subsequences of text to be compared. If the hash values are unequal, the algorithm will determine the hash value for next M-character sequence. If the hash values are equal the algorithm will analyze the pattern and M- character sequence. In this way, there is only one comparision per text subsequence and character matching is only required when the hash value match.

Akash Sharma

Rabin-Karp-Matcher (T, P, d, q)

1. $n \leftarrow$ length [T]
2. $m \leftarrow$ length [P]
3. $h \leftarrow d^{m-1} \bmod q$
4. $p \leftarrow 0$.
5. $t_0 \leftarrow 0$
6. for $i \leftarrow 1$ to m
7. do $p \leftarrow (dp + P[i]) \bmod q$
8. $t_0 \leftarrow (dt_0 + T[i]) \bmod q$
9. for $s \leftarrow 0$ to $n-m$
10. do if $p = t_s$.
11. then if $P[1 \ldots m] = T[s+1 \ldots s+m]$
12. then "Pattern occurs with shift" s
13. If $s < n-m$
14. then $t_{s+1} \leftarrow (d(t_s - T[s+1]h) + T[s+m+1]) \bmod q$.

Akash Sharma

Knuth - Morris - Pratt Algorithm

Knuth - Morris and Pratt introduce a linear time algorithm for the string matching problem. A matching time of $O(n)$ is achieved by avoiding comparision with an element of 's' that have previously been involved in comparision with some element of the pattern 'p' to be matched. i.e backtracking on the string 's' never occurs.

components of KMP.

The Prefix Function (π) = The Prefix Function π for a pattern encapsulates knowledge about how the pattern matches against the shift of itself. This information can be used to avoid a useless shift of the pattern 'p'. In other words, this enable avoiding backtracking of the string 's'.

```
m ← length[P]
π[1] ← 0
k ← 0
for q ← 2 to m
do while k > 0 and P[k+1] ≠ P[q]
do k ← π[k]
If P[k+1] = P[q]
then k ← k+1
π[q] ← k
return π.
```

Akash Sharma

The KMP Matcher.

The KMP Matcher with pattern 'p', the string 's' and prefix function 'Π' as input, finds a match of p in S. Following pseudo code compute the matching component of KMP Algorithm.

1. $n \leftarrow$ length [T]
2. $m \leftarrow$ length [P]
3. $\Pi \leftarrow$ Compute-Prefix-function(P)
4. $q \leftarrow 0$
5. for $i \leftarrow 1$ to n
6. do while $q > 0$ and $P[q+1] \neq T[i]$
7. do $q \leftarrow \Pi[q]$
8. If $P[q+1] = T[i]$
9. then $q \leftarrow q+1$
11. then print "Pattern occurs with shift" $i-m$
12. $q \leftarrow \Pi[q]$.

Akashbhoumik