SNS COLLEGE OF ENGINEERING

Coimbatore-107

COURSE NAME: ANALYSIS OF ALGORITHM

II YEAR/ IV SEMESTER

UNIT – IV

STRING MATCHING ALGORITHM

Topic

Rabin Karp Algorithm

Rabin Karp Algorithm

Unit - IV

Note (X):

In Naive string algorithm, we check every substring of text of pattern's size is equal to the pattern or not.

Like Naive Algorithm, Rabin karp algorithm matches the hash value of the pattern with the hash value of the substring of text, for match.

Rabin Karp Algorithm:

Step1: choose suitable Base & Modulus
* Select prime number 'p' as modulus (This avoids overflow issues & ensures good distribution of hash values)
* choose base 'b' (prime number)
eg: 256 for ASCII characters

Step 2: Initialize Hash value to '0'.

Step 3: Calculate initial Hash value for the pattern.
⇒ For each character 'c'
⇒ position 'i'.
$h = d^{1}(m-1) \% q$
⇒ pattern length 'm'

Rolling Hash Formula:

$hash_i = ((hash_{i-1} - text[i-m] \cdot b^{m-1}) + text[i]) \bmod p.$

where:

(i) $hash_i$ ⇒ Rolling hash at index i [Start of New window]

(ii) $text[i-m]$ ⇒ character leaving window

(iii) $text[i]$ ⇒ New char. entering window

(iv) Base ⇒ b [d = 256]

(v) m ⇒ pattern Length.

(vi) q ⇒ Prime modulus (eg: q = 101)

Example:

1. Text = "9876543212345674891"
2. pattern = "54321"
3. Base 'd' = 256
4. Modulus 'q' = 101
5. pattern Length 'm' = 5

Step 1:

Compute Hashvalue 'h' $= d^{(m-1)} \% q$

$h = 256^{(5-1)} \% 101$

$= 256^4 \% 101$

Compute step by step:

(i) '256' mod 101 = 256 ÷ 101 = 2 (quotient)

2 × 101 = 202.

256 − 202 = 54

(ii) $256^2 \% 101$:

$= 256 \times 256 = 65536 \div 101 = 648$ (quo)

$= 648 \times 101 = 65448$

$65536 - 65448 = 88$

(iii) $256^3 \% 101$:

$= 256^2 \times 256 = 88 \times 256 = 22528$

$= 22528 \div 101 = 223$ (quo)

$= 223 \times 101 = 22523$

$22528 - 22523 = 5$

(iv) $256^4 \% 101$:

$= 256^3 \times 256^1 = 5 \times 256 = 1280$

$= 1280 \div 101 = 12$ (quo)

$= 12 \times 101 = 1212$

$1280 - 1212 = 68$

Step 2: Compute Hash of pattern "54321"

ASCII values for each character

5 ⇒ 53

4 ⇒ 52

3 ⇒ 51

2 ⇒ 50

1 ⇒ 49

Pathash = $(53 \times 256^4 + 52 \times 256^3 + 51 * 256^2 + 50 \times 256^1 + 49) \bmod 101$.

$= (53 \times 68 + 52 \times 5 + 51 \times 88 + 50 \times 5 + 49) \bmod 101$

$= (3604 + 260 + 4488 + 2700 + 49)$

11101 mod 101 = 92 → Steps

11101 ÷ 101 = 109 (quo)

109 × 101 = 11009

∴ 11101 − 11009 = 92. Patternhash = 92

Index	Window	
0	98765	(Shift 1 char)
1	old: 9 New: 4 87654	Shift 1 char
2	old: 8 New: 3 76543	Shift 1 char
3	65432	
4	54321	

Step 3: Compute Hash value for each index in text.

(1) Window ("98765"; index : 0)

Compute ASCII value : 9→57, 8→56, 7→55, 6→54, 5→53

$H_0 = (57\times68) + (56\times5) + (55\times88) + (54\times54) + 53) \bmod 101$

∴ we have $= (57\times256^4 + 56\times256^3 + 55\times256^2 + 54\times256^1 + 53) \bmod 101$

$= (3876 + 280 + 4840 + 2916 + 53) \bmod 101$

$= 11965 \bmod 101$

11965 ÷ 101 = 118 (quo)

118 × 101 = 11918

11965 − 11918 = 47

∴ Ans = 47

Step 1: Slide the window & Apply Rolling Hash Function

hash i = [(hash i-1 − old char * h) + new char)] mod q

(i) Compute for index 1: Window 6

old char: 9 → Ascii → 57

New char: 4 → ASCII → 52

Apply in formula (Rolling Hash)

$h_1 = [(47 - 57 * 68) \cdot 256 + 52] \% 101$

1. 57 * 68 = 3876
2. 47 − 3876 = −3829
3. −3829 × 256 = −9,80,224
4. −9,80,224 + 52 = −980172
5. −980172 % 101 = −9704

−9704 * 101 = −980104

−980172 + 980104 = −68 $h_1 = -68$

To make it positive, Add Hash with 101

101 − 68 = 33

$\therefore h_1 = 33$ No match with pat

(ii) Compute for index 2: Window "76543"

old char: 8 → ASCII → 56

New char: 3 → ASCII → 51

Apply in formula (Rolling Hash):

$h_2 = [(33 - 56 * 68) \cdot 256 + 51] \% 101$

1. $56 * 68 = 3808$
2. $33 - 3808 = -3775$
3. $-3775 * 256 = -966400$
4. $-966400 + 51 = -966349$
5. $-966349 \% 101 = -9567$

$9567 * 101 = -966267$

$-966349 + 966267 = -82$; $h_2 = -82$

To make it positive, Add Hash with 101

$101 - 82 = 19$

$\therefore h_2 = 19$ → No match with Pattern

(iii) Compute for index 3: Window "65432"

old char: 7 ; ASCII → 51

New char: 2 ; ASCII → 50

Apply in formula (Rolling Hash):

$h_3 = [(19 - 55 * 68) \cdot 256 + 50] \% 101$

1. $55 * 68 = 3740$
2. $19 - 3740 = -3721$
3. $-3721 \times 256 = -952576$
4. $-952576 + 50 = -952526$
5. $-952526 \% 101 = -9430$

$9430 * 101 = 952430$

$-952526 + 952430 = -96$; $h_3 = -96$

To make it positive; $101 - 96 = 5$; $h_3 = 5$

Add with 101.

No Match with Pattern

(iv). Compute for index 4; Window "5432"

old char : 6 → ASCII Value : 54

New char : 1 → ASCII Value : 49

$h_4 = [(5 - 54 * 68) * 256 + 49] \% 101$

1. $54 * 68 = 3672$
2. $5 - 3672 = -3667$
3. $-3667 * 256 = -938752$
4. $-938752 + 49 = -938703$
5. $-938703 \% 101 = -9294$

   $9294 * 101 = -938694$

   $-938703 + 938694 = -9 \rightarrow h_4$

To make it positive; Add Hash with 101

$101 - 9 = 92$

$\therefore h_4 = 92$

(x) Pattern hash & hash value of text (index 4) got matched (x)

Algorithm RabinKarp()

```
{ for(i=0; i<m; i++) // calculate initial hash of pat.
pathash = (d * pathash + p[i]) % q
Texthash = (d * texthash + p[i]) % q
// Slide pattern over text
for( i=0; i<=n-m;i++)
{
if(pathash == texthash)
```

```
for(j=0; j<m; j++)
{ if (T[i+j] != P[j])
}
if (j==m)
match found at index i;
}
if (i<n-m)
{ hash i = (hash[i-1] - old char * h) * d + newchar
                                                mod q
 if (texthash < 0)
  texthash += q
}
}
```

Time complexity

Best case : O(n+m) (Happens at No Hash collisions

Worst case : O(n)*m) → All hash collision.

Average Case : O(n+m) Collisions are rare

Space Complexity :

O(1) → only few integer variables & constants are stored.