Rabin-Karp

Rabin-Karp Algorithm

- Key idea:
  - think of the pattern P[0..m-1] as a key, transform (hash) it into an equivalent integer *p*
  - Similarly, we transform substrings in the text string T[] into integers
    - For s=0,1,…,n-m, transform T[s..s+m-1] to an equivalent integer t_s
  - **The pattern occurs at position s if and only if $p=t_s$**
- If we compute p and t_s quickly, then the pattern matching problem is reduced to comparing p with n-m+1 integers

Rabin-Karp

- The Rabin-Karp string searching algorithm calculates a **hash value** for the pattern, and for each M-character subsequence of text to be compared.
- If the hash values are unequal, the algorithm will calculate the hash value for next M-character sequence.
- If the hash values are equal, the algorithm will do a Brute Force comparison between the pattern and the M-character sequence.
- In this way, there is only one comparison per text subsequence, and Brute Force is only needed when hash values match.

3

Rabin-Karp Algorithm

- Assume each character is digit in radix-d notation (e.g. d=10)
- p = decimal value of pattern
- t_s = decimal value of substring T[s+1..s+m] for s = 0,1...,n-m
- Strategy:
  - compute p
  - compute all t_i values
  - find all valid shifts s by comparing p with each t_s
- Compute p in O(m) time using Horner's rule:
  - **$p = P[m] + d(P[m-1] + d(P[m-2] + ... + d(P[2] + dP[1])))$**
- Compute t_0 similarly from T[1..m]
- Compute remaining t_i's
  - $t_{s+1} = d(t_s - d^{m-1}T[s+1]) + T[s+m+1]$

Rabin-Karp Algorithm

pattern is M characters long
hash_p=hash value of pattern
hash_t=hash value of first M letters in body of text
do
    if (hash_p = = hash_t)
        brute force comparison of pattern
        and selected section of text
        hash_t= hash value of next section of text, one character over
while (end of text or brute force comparison == true)

5

Rabin-Karp Math Example

- Let's say that our alphabet consists of 10 letters.
- our alphabet = a, b, c, d, e, f, g, h, i, j
- Let's say that "a" corresponds to 1, "b" corresponds to 2 and so on.

The hash value for string "cah" would be

3*100 + 1*10 + 8*1 = 318

6

Rabin-Karp Algorithm

- Let us assume $\Sigma=\{0, 1, 2,\ldots, 9\}$, so that each character is a decimal digit.
- Given a pattern P [1..m], let p denote its corresponding decimal value
- Given a text T[1..n], let t_s denote the decimal value of the length-m substring T[s+1..s+m], for s=0, 1,….., n-m.
- ts ==p, if and only if T[s+1..s+m],= P[1..m]; thus, s is a valid shift if and only if t_s=p

---

- How to compute p ?
- Using horner's rule

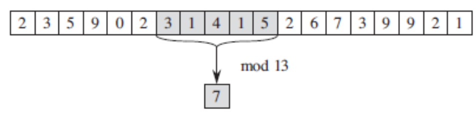
$$p = P[m] + 10\,(P[m-1] + 10(P[m-2] + \cdots + 10(P[2] + 10P[1])\cdots)) .$$

Rabin-Karp Mods

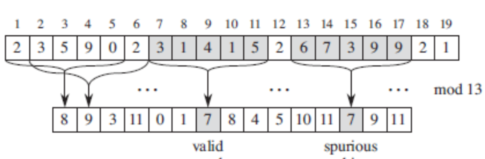
- If M is large, then the resulting value will be enormous. For this reason, we hash the value by taking it **mod** a prime number **q**.

9

The Rabin-Karp algorithm



Each character is a decimal digit, and we **compute values modulo 13.**
(a) A text string. **A window of length 5** is shown shaded.
The numerical value of the shaded number, computed modulo 13, yields the value 7.

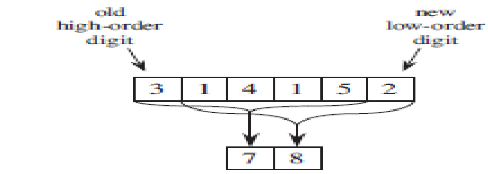


The same text string with values computed modulo 13 for each possible position of a **length-5 window.**
Assuming the **pattern P =31415, we look for windows whose value modulo 13 is 7, since 31415 mod 13= 7**

- We can compute t_0 from T[1..m]
- t_{s+1} from t_s in constant time

$$t_{s+1} = 10(t_s - 10^{m-1}T[s+1]) + T[s+m+1] .$$

- Subtracting 10^{m-1}T[s+1] removes the high-order digit from t_s ,
- Multiplying the result by 10 shifts the number left by one digit position,
- and adding T [s+m+1]brings in the appropriate low-order digit.
- if m=5 and t_s=31415
- we wish to remove the high-order digit T [s+1]=3 and bring in the new low-order digit (suppose it is T[s+5+1]= 2)
- Ts= 10(31415 – 10000 *3)+2
- =14152



old high-order digit — shift — new low-order digit

$$
\begin{aligned}
14152 &\equiv (31415 - 3 \cdot 10000) \cdot 10 + 2 \pmod{13} \\
&\equiv (7 - 3 \cdot 3) \cdot 10 + 2 \pmod{13} \\
&\equiv 8 \pmod{13}
\end{aligned}
$$

RABIN-KARP-MATCHER(T, P, d, q) *//d is radix q is modulus*

$n = T.length$

$m = P.length$

$\Theta(m)$ in $\Theta(n)$ → $h = d^{m-1} \bmod q$ *//high-order digit position for m-digit window*

$p = 0$

$t_0 = 0$

$\Theta(m)$:

for $i = 1$ **to** m // preprocessing

  $p = (dp + P[i]) \bmod q$

  $t_0 = (dt_0 + T[i]) \bmod q$

$\Theta((n-m+1)m)$:

for $s = 0$ **to** $n - m$ // matching

  if $p == t_s$

  $\Theta(m)$:

    if $P[1..m] == T[s+1..s+m]$

      print "Pattern occurs with shift" s

  if $s < n - m$

    $t_{s+1} = (d(t_s - T[s+1]h) + T[s+m+1]) \bmod q$

worst-case running time is in $\Theta((n-m+1)m)$