Hash Tables

Pattern Matching -

RABIN-KARP ALGORITHM

→ uses a ROLLING-HASH Technique
→ uses a window of size of the pattern, which slides over till it reaches the end of text.
→ everytime the hash of the window matches the size of the pattern, each character is checked with each character of the pattern.
→ the HASH FUNCTION used is SUPER IMPORTANT

Sliding window:

A problem-solving technique that works by maintaining a window, a contiguous part of the data. Its advantage is that if we want to calculate something about the next window, we don't need to do it from scratch, we just take the value of the actual window and update it in some way, which is faster

→ do not need to calculate from scratch everytime.

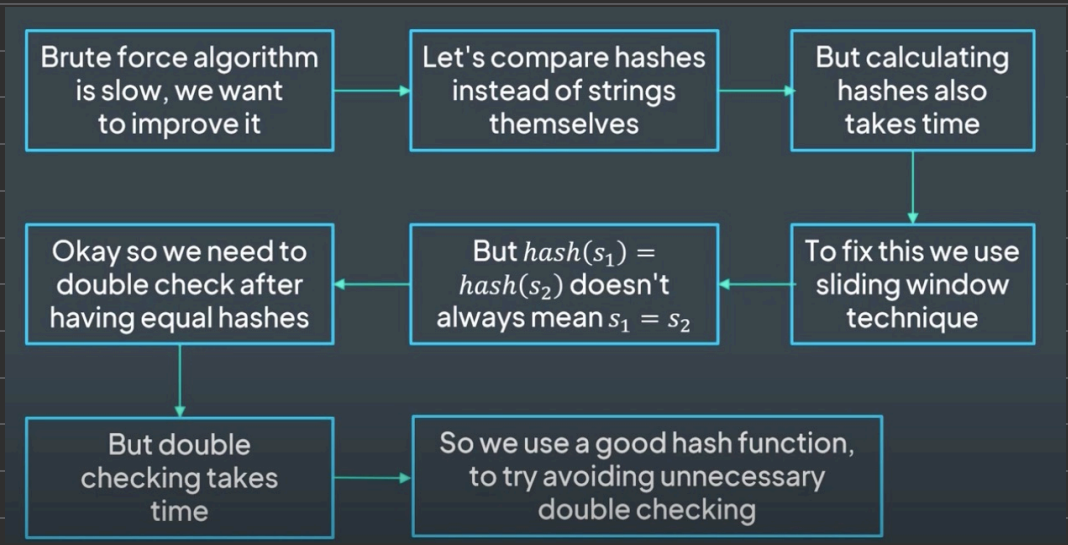
Ex -

$sum_{w_0} = 25$

$sum_{w_1} = 25 - (5) + (1) \Rightarrow 21$

previous element

new element



BUILDING THE HASH FUNCTION-

say, → string 'p' of size 'm'.
→ d = alphabet size → no.of diff. characters that string can take
→ q = large prime number → used modulo to avoid overflow

$$hash(p) = \left(\sum_{i=0}^{m-1} p_i * d^{m-i-1} \right) \% \, q$$

$$= (p_0 * d^{m-1} + p_1 * d^{m-2} + \cdots + p_{m-2} * d^1 + p_{m-1} * d^0) \% \, q$$

Example:

$p =$ "click"
$m = |p| = 5$
$d = 26$
$q = 101$

$$hash(p) = (\text{"c"} * 26^4 + \text{"l"} * 26^3 + \text{"i"} * 26^2 + \text{"c"} * 26^1 + \text{"k"} * 26^0) \% 101$$
$$= (99 * 456976 + 108 * 17576 + 105 * 676 + 99 * 26 + 107 * 1) \% 101$$
$$= 47212493 \% 101 = 43$$

Can cause an overflow

$$\left(\sum_{i=0}^{m-1} p_i * d^{m-i-1}\right) \% \, q$$

```
hash_p = sum([ord(p[i])*(d**(m-i-1)) for i in range(m)]) % q
```

Safe ✔

```
hash_p = 0
for i in range(m):
    hash_p = (d*hash_p + ord(p[i])) % q
```