Hash

➡ **Popular Hash Functions :-**

① **Permutation Hash :-**

when order matters.

"a b b"
"a b b c"   thint there are written in a digit system of base x
"b b a"

$a = 1 \quad b = 2 \quad c = 3 \quad d = 4$

"abb" $\rightarrow$ $ax^2 + bx + c$

"abbc" $\rightarrow$ $ax^3 + bx^2 + bx + c$

"bba" $\rightarrow$ $bx^2 + bx + a$

for "a b d g e f k m p" big numbers take $\% M$.

② **Set Hash :-**

$$h(\{ a, e, b \}) = (a+k)^H + (b+k)^H + (e+k)^H \quad \% M$$

$k = 33 \; 59 \; 2 \; 19$
$H = 62 \qquad H > 50$

idea: $h(\{a, b, c\}) = k^H + c_1 (a+b+c) k^{H-1}$
$$+ c_2 (a^2 + b^2 + c^2) k^{H-2}$$
$$+ c_3 (a^3 + b^3 + c^3) k^{H-3}$$

③ **XOR :-**

$$f(\{a, b, c\}) = \left[(a+k) \wedge (b+k) \wedge (c+k) \right] \quad \text{# No } \% M$$
it'll distort XOR.

$$\text{or} \quad \left[((a+k) \% M) \wedge ((b+k) \% M) \wedge ((c+k) \% M) \right]$$

Double Hashing :-   generate pair of 2 hashes & match.

$\{ a \; b \; c \} \xrightarrow{(k_1 H_1 M_1)} x_1$
$\phantom{\{ a \; b \; c \}} \xrightarrow{(k_2 H_2 M_2)} x_2$

if $(x_1 = Y_1 \; \&\& \; x_2 == Y_2)$
very less chance
$\{ d \; e \; f \}$ can also satisfy this.

① Rolling Hash

idea :   Pattern = "ccg" $h(Pattern) = 2+2+6 = 10$
                    2 2 6

String = " $\underbrace{g\ e\ c}_{h(subS)}\ g\ d\ b\ \ldots$ "
                            $= 10$

if $h(subS) == h(Pattern)$
   check one by one $\rightarrow$ if match, return.

then; " $g\ \underbrace{e\ c\ g}\ d\ b\ \ldots$ "
          $h(subS) = 10$ check again ..
          match $\rightarrow$ return

$\rangle$ pattern = "ccg" $= 2 \times 26^2 + 2 \times 26 + 6 \times 26^0$
                         $= H(pattern)$

"$\underbrace{g\ e\ c}_{H_1}\ \overset{H_2}{g}\ b\ d\ \ldots$ "

$H_1 = 6 \times 26^2 + 2 \times 26 + 2 \times 26^0$

$H_2 = (H_1 - 6 \times 26^2) \times 26 + 6$

$\quad = (2 \times 26 + 2 \times 26^0) \times 26 + 6$

$\quad = 2 \times 26^2 + 2 \times 26 + 6 \times 26^0$.

Hash generation :-

① for (int i=0 ; i<P ; i++)
      hp = (hp + pattern[i] × pow(D, P-i-1)) % Mod

② for(int i=0 ; i<P ; i++)
      hp = (D × hp + pattern[i]) % Mod

D: no of digits (base) E.g 256 or 26 if only alphabets..
Mod : 1e9+7