Game:

Nishant | Parvesh — Both Pick Same Numbers?

[1 2 3 4]

Numbers: [1 4]

1 1	1 2	1 3	1 4
2 1	2 2	2 3	2 4
3 1	3 2	3 3	3 4
4 1	4 2	4 3	4 4

→ $\frac{4}{16}$ → $1/4$

		Probability
[1 5]	→	1/5
[1 10]	→	1/10
[1 10^9]	→	$1/10^9$, very very less

increasing range

probability decreases

Note: To Insert a String (is has a key) of len (N) in a Hashset / Hashmap $\boxed{TC \to O(N)}$

Q1) Given Text (T) & Word (W), check if Word (W) is present in T as substring → { Idea = Subarray, Continuous part of String }

	0	1	2	3	4	5	6	7	8
Text : (N)	a	b	c	b	a	g	d	d	a
Word : (k)	b	a	g	d					

Idea
1) for all substrings of len = k in Text
comp == Word

TC: $(N-k+1) * (k)$

Q) 2 Strings, len == k
TC: O(k)

$k = 1$: $(N-1+1)(1)$ → O(N) (TC)
$k = N/2$: $(N/2+1)(N/2)$ → $O(N^2)$
$k = N$: $(N-N+1)(N)$ → O(N)

TC : $O(N^2)$ SC : O(1)

Ex2:
Text : a b c a d
# of freq same,
Word : a a c b
Arrangement is changing

↳ Keep : (In public chat)

Ex:
Text : a a a a a a b
Word : a a a b
Idea2:
2 pointer approach
$O(N^2)$

obs: Comparing 2 Strings taking time → TC: O(N)

Idea: $N_1 == N_2$ → O(1)

↦ Integer

String → Number [≈ $2*10^9$] → 9 digits

→ f(   )

S = abc = f(abc) =

Idea1: Replace characters with numbers & store } won't work

We have to store N Numbers

$f_1(abc) = a+b+c$

$f_1(acb) = a+c+b$

idealy

$S_1 \neq S_2$ $f(S_1) \neq f(S_2)$

$$f_2(\overset{0}{a}\,\overset{1}{b}\,\overset{2}{c}) = \{a*1 + \cancel{b*2} + c*\cancel{3}\} = \{\overset{97}{a} + \overset{99}{c}\} \to 196$$

$$f_2(b\,c\,b) = \{b*1 + \cancel{c*2} + b*\cancel{3}\} \qquad \{2b\} \to 2*98 = 196$$

// Decimal → (7894) = $(7*10^3 + 8*10^2 + 9*10^1 + 4)$

↳ weightage of position is more

obs: If we keep pos in a power, weightage will incre.

// $p \Rightarrow$ raising position to power of $p = 1 +$

Note: In general p prime number $\{2, 3, 23, 47, 59, 101 \ldots$

$$f_3 = \begin{array}{cccc} 0 & 1 & 2 & 3 \\ (a & b & a & d) \end{array} = \left\{ a * p^0 + b * p^1 + a * p^2 + d * p^3 \right\}$$

$MOD = \{10^9 + 7\}$ / $MOD = \{\underbrace{10^{18} + 7}\}$ $\times$ {overflow}

$$f_3 = \begin{array}{ccccc} 0 & 1 & 2 & 3 & 4 \\ (c & d & a & d & e) \end{array} = \left\{ c + dp^1 + ap^2 + dp^3 + ep^4 \right\} \% (Mod)$$

Mod = 5 $\Rightarrow$

$(\underbrace{\qquad}) \% 5 \Rightarrow 0, 1, 2, 3, 4 \Rightarrow 1/5$

$(\qquad) \% 10 \Rightarrow \underbrace{\qquad\qquad} \Rightarrow 1/10$

$(\qquad) \% (10^9 + 7) \Rightarrow \underbrace{\qquad\qquad} \Rightarrow$ (very very small)

$MOD = 10^{18} + 7$

$a \% MOD = [0, \ldots 10^{18} + 6]$

$(a \% MOD * b \% MOD) \approx 10^{36}$ *

$MOD = 10^9 + 7$

$a \% MOD \Rightarrow [0, 10^9 + 6]$

$(a \% MOD * b \% MOD)$

$\Rightarrow [10^{18})$

String S_N :

$\{10^9+7\}$

$$f(s) = [S[0]*p^0 + S[1]*p^1 + S[2]*p^2 + \dots S[N-1]p^{N-1}] \% MOD$$

$$\boxed{f(s) = \left[\sum_{i=0}^{N-1} (S[i] * p^i) \% MOD\right] \% MOD}$$

Note: Proven by obs, if p is a prime number, probability of 2 diff strings having same functional value is further decreased

$$f(s) = \left[\sum_{i=0}^{N-1} (S[i] * p^i) \% MOD\right] \% MOD$$

$p = 23 \quad f_1(s_1) \quad == \quad f_1(s_2) = 1/10^9$

$(1/10^{18})$

$p = 59 \quad f_2(s_1) \quad == \quad f_2(s_2) : 1/10^9$

$p = 101 \quad f_3(s_1) \quad == \quad f_3(s_3) \to 1/10^9$

/ we cannot do / with %, but we can do mod with *

$$\begin{array}{lcccc} & 0 & 1 & 2 & 3 \\ \text{Word} : & b & a & g & d \end{array} \qquad f(w) = \{ b + ap + gp^2 + dp^3 \} \% MOD$$

T_4

$$\begin{array}{lccccccccc} & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ \text{Text} : & a & b & c & b & a & g & d & d & a \end{array}$$

substrings

going coming $f(w) =$

$[0 \quad 3] = \{ a + bp + cp^2 + bp^3 \} \% M \quad = \quad \{ \underline{b + ap + gp^2 + dp^3} \} \% MOD$

$[1 \quad 4] = \{ \cancel{a} + bp + cp^2 + bp^3 \} \% M - \cancel{a} + ap^4$

$= \{ b p + cp^2 + bp^3 + ap^4 \} \% M \quad = \quad p * \{ b + ap + gp^2 + dp^3 \} \% MOD$

$= \{ b p + cp^2 + bp^3 + ap^4 \} \% M \quad = \quad (\underbrace{bp + ap^2 + gp^3 + dp^4}) \% MOD$

$[2 \quad 5] = \{ \{ \cancel{b}p + cp^2 + bp^3 + ap^4 \} \% M - \cancel{b}p + gp^5 \} \% M$

$= \{ cp^2 + bp^3 + ap^4 + gp^5 \} \% M \quad = \quad p * (bp + ap^2 + gp^3 + dp^4) \% MOD$

$\{ cp^2 + bp^3 + ap^4 + gp^5 \} \% M \quad = \quad (bp^2 + ap^3 + gp^4 + dp^5) \% MOD$

$[3 \quad 6] = \{ \cancel{c}p^2 + bp^3 + ap^4 + gp^5 \} \% M - c \cancel{*} p^2 + dp^6 \} \% M$

$= \{bp^3 + ap^4 + gp^5 + dp^6\} \% M = p * (bp^2 + ap^3 + gp^4 + dp^5) \% MOD$

$\{bp^3 + ap^4 + gp^5 + dp^6\} \% M == (bp^3 + ap^4 + gp^5 + dp^6) \% MOD$

$p^3 \Rightarrow p^3 \% M$

$p^4 \Rightarrow p^4 \% M$

$p^{100} \Rightarrow p^{100} \% M$

↳ fast power exponent with modules

// . Rabin - karp pattern matching algorithm

```
int search( string T, string w ){

    M = (10^9 + 7),  N = T.size(),  k = w.size()
    fT = 0,  fw = 0,  P = ___ ,  c = 0

    i = 0; i < k; i++){                                   obs 1
        fw = [fw + w[i] * p^i] % M                        on the fly update
        fT = [fT + T[i] * p^i] % M                        powers of p^i
    }
    if( fw == fT ){ c = c+1 }                             obs 2
    s = 1,  e = k                                         on the fly update
    while( e < N ){                                       p^(s-1) & p^e
        // substring in T[s  e]
        // going out T[s-1]   coming in T[e]
        fT = ((fT - T[s-1] * p^(s-1) + T[e] * p^e) % M + M) % M
        fw = (fw * p) % M
        if( fw == fT ){ c = c+1 }
        s = s+1,   e = e+1
    }
}

return c;      TC: O(N)   SC: O(1)
```

Hint

Ex: $S = \{ a \quad b \quad c \quad \overset{1}{b} \quad \overset{0}{a} \}$

$H_f \Rightarrow \{ a + bp + cp^2 + bp^3 + ap^4 \}$

$H_r \Rightarrow \{ ap^4 + bp^3 + cp^2 + bp + a \}$

Obs: If String is palindrome,

Hashvalue from forward

= Hashvalue from Backward

google!!

IMrc:

Given ∞ stream of character, each character is added 1 by 1, after each character, check if entire string is palindrome or not?

H_f $\qquad$ H_r

Ex:

a ✓	a	a ✓
a b	$a + bp$	$(a*p + b)$
a b b	$a + bp + bp^2$	$ap^2 + bp + b$
a b b a	$a + bp + bp^2 + ap^3$	$ap^3 + bp^2 + bp + a$ ✓

Say we are adding i^{th} character ch

$$H_f = (H_f + ch * p^i) \% M$$

[calculate p^i on the fly]

$$H_r = ((H_r)*p + ch) \% M$$

// Say we want to reduce collisions ⇒

p =	13	29	41	47	101
	H_{f_1}	H_{f_2}	H_{f_3}	H_{f_4}	H_{f_5}
	H_{r_1}	H_{r_2}	H_{r_3}	H_{r_4}	H_{r_5}

- Try (1hr)
- TA (_)
- Rate to me (Doubt Session)
- BFS / DFS / Dijkstra /

- Saturday → (Pattern - 2) ✓
- Sunday → (More Probl) ✓ ⇒
- Tuesday → (Holiday) ✓
- Thursday → (Holiday) ✓