Q. Given 2 string A & B. ($N \geq m$) Check how many occurences of permutation of B are present in A as a substring

N m

B : a b a c

A : a b c a a c b b c a a

ans = 4

Permutation of B : a a b c
a b a c
a b c a
b a a c
b a c a
b c a a
⋮

Q. Given 2 A & B. Check if X is a permutation of Y.

X = a a b c
Y = a b a c

1. Sort & compare

TC : O(N log N)
SC : O(1)

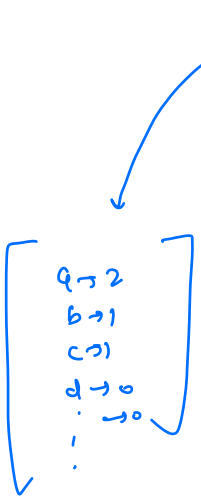
Better ✓ 2. check freq of each char

int freq1[26]
int freq2[26]

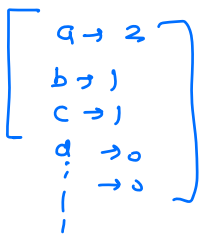
TC : O(N)
SC : O(26.2) : O(1)

B = a b a c

A = a b c a a c b b c a a



Cnt = 1



Sliding window [maintain hm of each substring]

& keep comparing it with hm of string B.

B.size → m

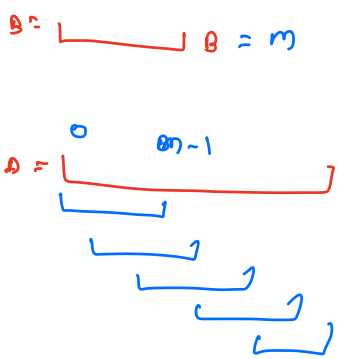
A.size → N

```
int freqB[26] = {0}
for(i=0; i< m ; i++)
{   freqB[ B[i] - 'a' ]++
```

M

```
int freqA[26] = {0}
for(i=0; i<m; i++)
{   freq[ A[i] - 'a' ]++
```

M



```
ans = 0
if ( compare( freqA, freqB ) == T )  ans++

s = 1   e = M

while ( e < N )
{
    freqA [ A[e] - 'a' ] ++
    freqA [ A[s-1] - 'a' ] --
    if ( compare( freqA, freqB ) == T )  ans++
    e++, s++
}

return ans
```

N - M

B : a b a c

A : a b c a a c b b c a a

freqA : a → 2, b → 1, c → 1, d → 0, ...

freqB : a → 2, b → 1, c → 1, d → 0, e → 0, ...

ans = 1 2 3 4

TC: O(m + ~~m~~ + N - ~~m~~)

TC: O(N + m)

SC: O(26) : O(1)

Q. String A & B. Find no of occurrence of P in S as substring.

P → m len, S → N len

Text S = a b c b a b x a

Pattern P = c b a

ans = 1

A-1 Compare each substring of m len with Pattern

```
s = 0 ; e = m - 1
while ( e < N)
{
    if( S.substr(s, e) == P)  ans++;
    s++, e++
}
```

TC: $O(N, M)$

- order of string
- comparsion

Comparsion of 2 strings = $O(len)$

Comparsion of 2 int = $O(1)$

String ⟶ int

97 98 99

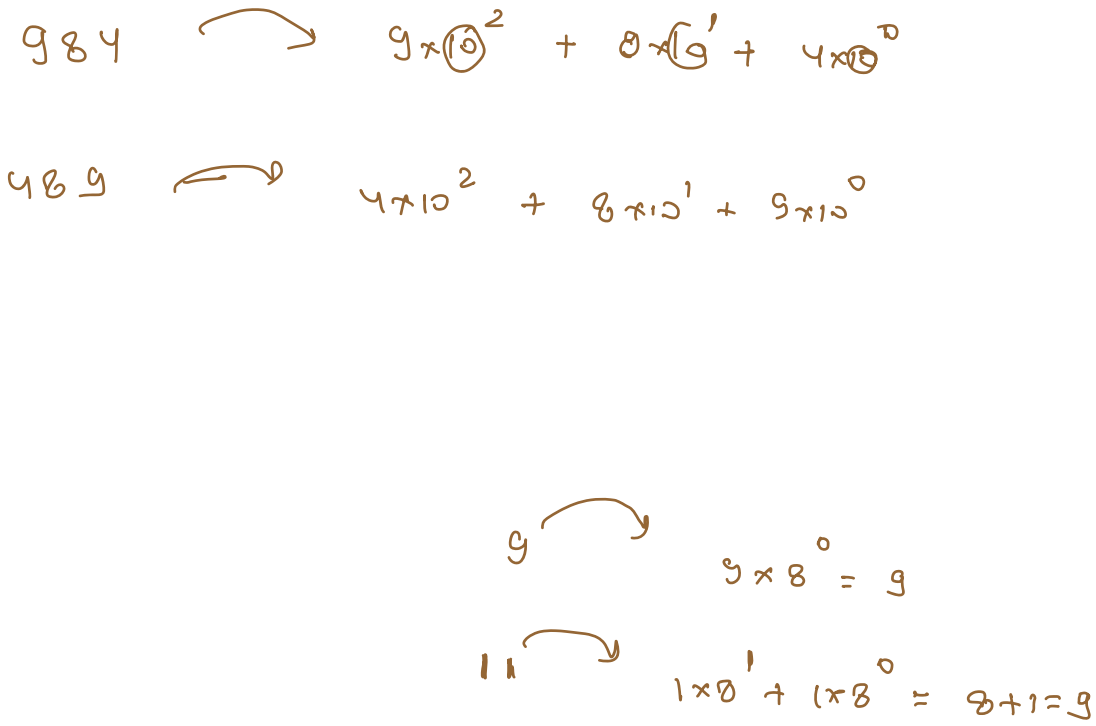
a b c = a + b + c

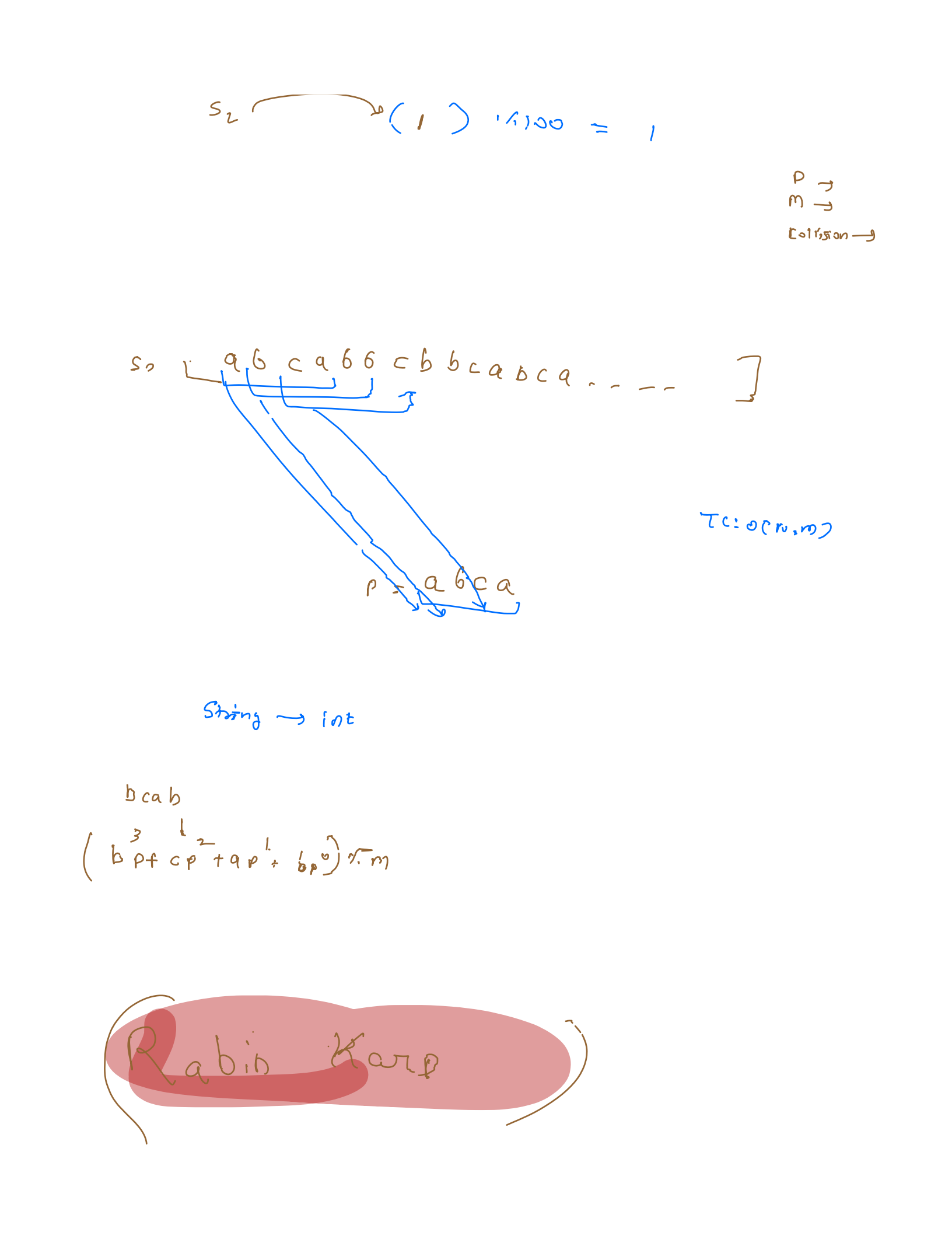
= 97 + 98 + 99

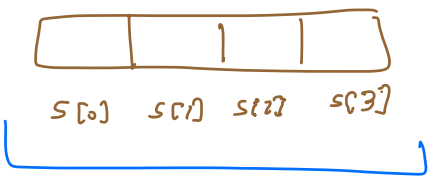
= 294

c b a = 299

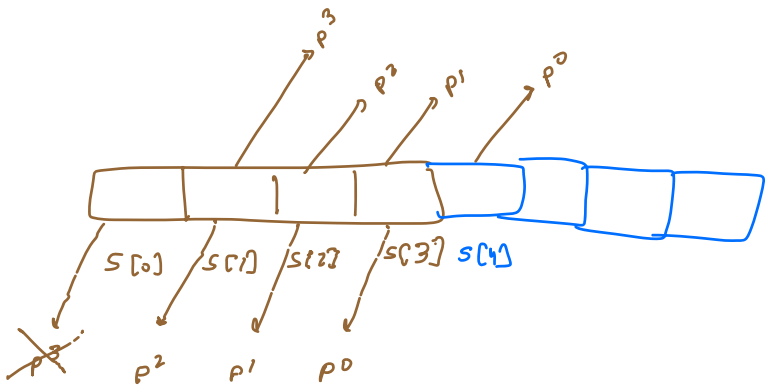
b c a = 299


984
9×10^2 + 8×10^1 + 4×10^0
489
4×10^2 + 8×10^1 + 9×10^0
9
9×8^0 = 9
11
1×8^1 + 1×8^0 = 8+1=9
a b c
a p^2 + a p^1 + a p^0 %m = int
Rolling Hash
bcdef → (b p^4 x c p^3 + d p^2 + e p^1 + f p^0) % m
S1
(101) % 100 = 1


p
m
Collision
Tc: O(n,m)
String → int
Rabin Karp



int HV = (s[0]p^3 + s[1]p^2 + s[2]p^1 + s[3]p^0) % M



HV' = ([HV - s[0]p^3] p + s[4]p^0) % M

Hash value of P = HP

Hash value of first window = HC

If (x == HP) ans++

s = 1 e = M

while (e < n)

{

x = ([x - first value] * p + new value) % m

if (x == hp) ans++

else ans

}

TC :

return ans

hp = 0

curr = m-1 :

for (i = 0 ; i < m ; i++)

{

hp = (hp + p[i] * p^curr) % m

curr--

}

m

0 1 2 ... m-1

p^(m-1) p^2 p^1 p^0

p

X = 0

curr = m-1 :

for (i = 0 ; i < m ; i++)

{

X = (X + s[i] * p^curr) % m

curr--

}

m

Side manner

ans = 0

if (x == H ans++

s = 1 e = M

while(e < M)

{

x = [(x - s[s-1] p^(m-1)) * p + s[e] * p^0] % m

if(x == hp) ans++

s++ e++

}

N - M

return ans

0 - 9 → 10

a - z (26)

P → ≥ 26

P = 29

TC: O(M + M + N - M)

TC: O(M + N)

SC: O(1)

[without cross check]

m → [0, m-1]

m = 100 [0 - 99]

m = 2 [0 - 1]

$m = 10^9 + 7$ or 999...

Probability of collision

int = x_1 int = x_2 are different ⟶ String also different

$x_1 == x_2$ → collision different

→ !collision strings are same

Given & find particular pattern these or not.

Tc: O(M+N+NM)

Q. Given a running stream of characters

At each input check if string is Palindrome or not.

x y x y y x y x ⟶

T F T F F F F T ⟶

at Each input

check if Palindrome or not

Tc: $O(N)$ for each input

Tc: $O(N^2)$ all input

a b c b a

front = reverse

front = reverse

a b c b a

✓

✗ $a[0]p^4 + a[1]p^3 + a[2]p^2 + a[3]p^1 + a[4]p^0$

a b c b a d

✓

$$s[0]p^5 + s[1]p^4 + s[2]p^3 + s[3]p^2 + s[4]p^1 + s[5]p^0$$

$$x' = px + \text{new char}$$

a b c b a

✓ ✓

$$y \quad s[0]p^0 + s[1]p^1 + s[2]p^2 + s[3]p^3 + s[4]p^4$$

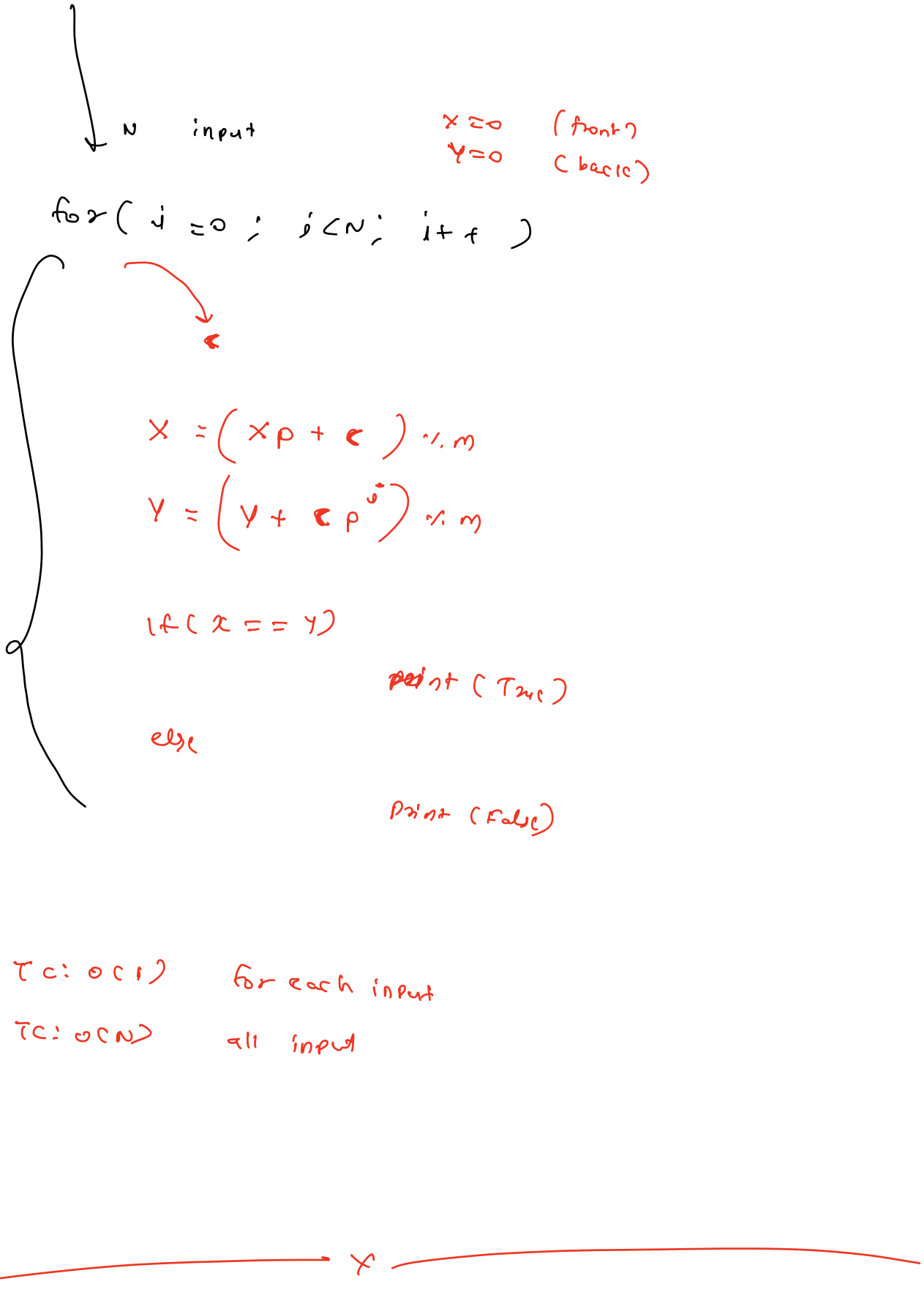
y a b c b a d

✓ ✓

$$s[0]p^0 + s[1]p^1 + s[2]p^2 + s[3]p^3 + s[4]p^4 + s[5]p^5$$

$\longrightarrow n^{th}$ input

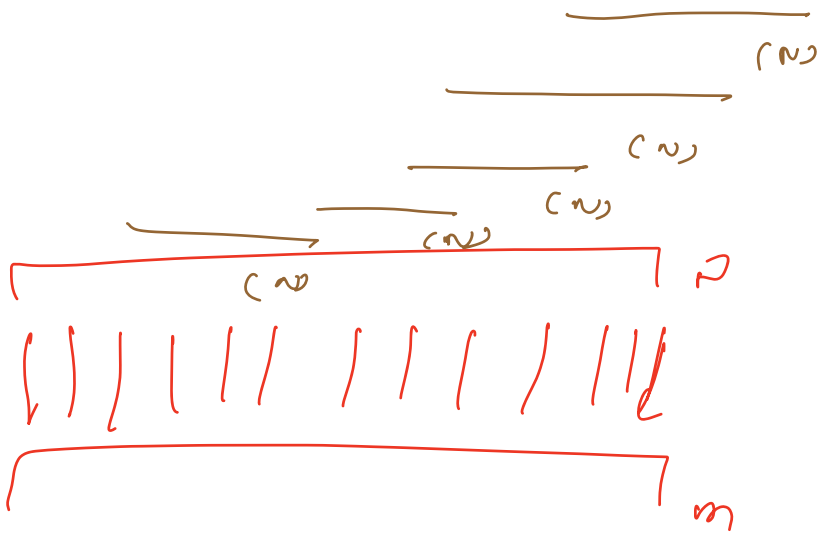
$$y' = y + \text{new char } p^n$$

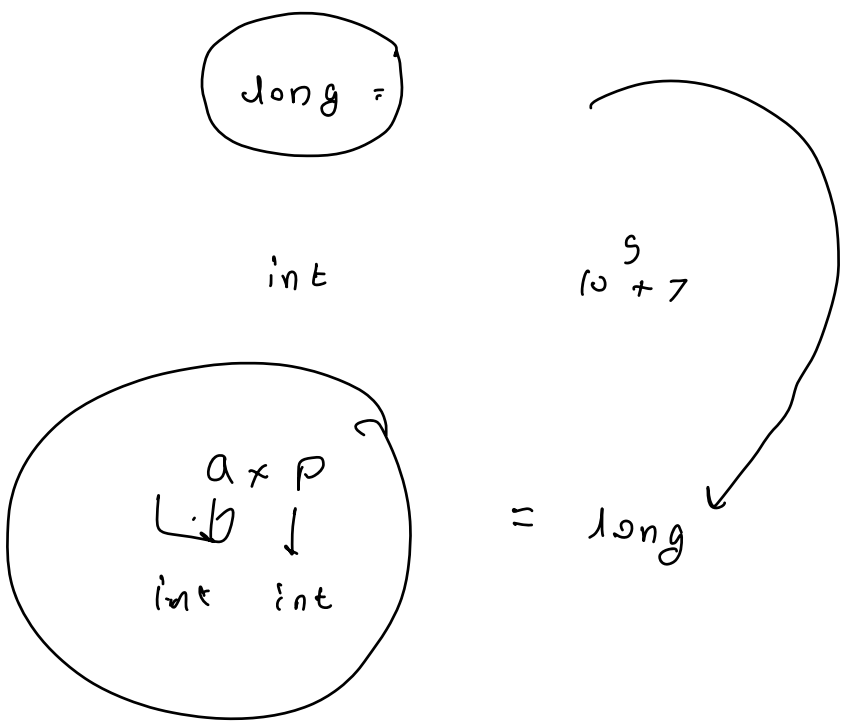


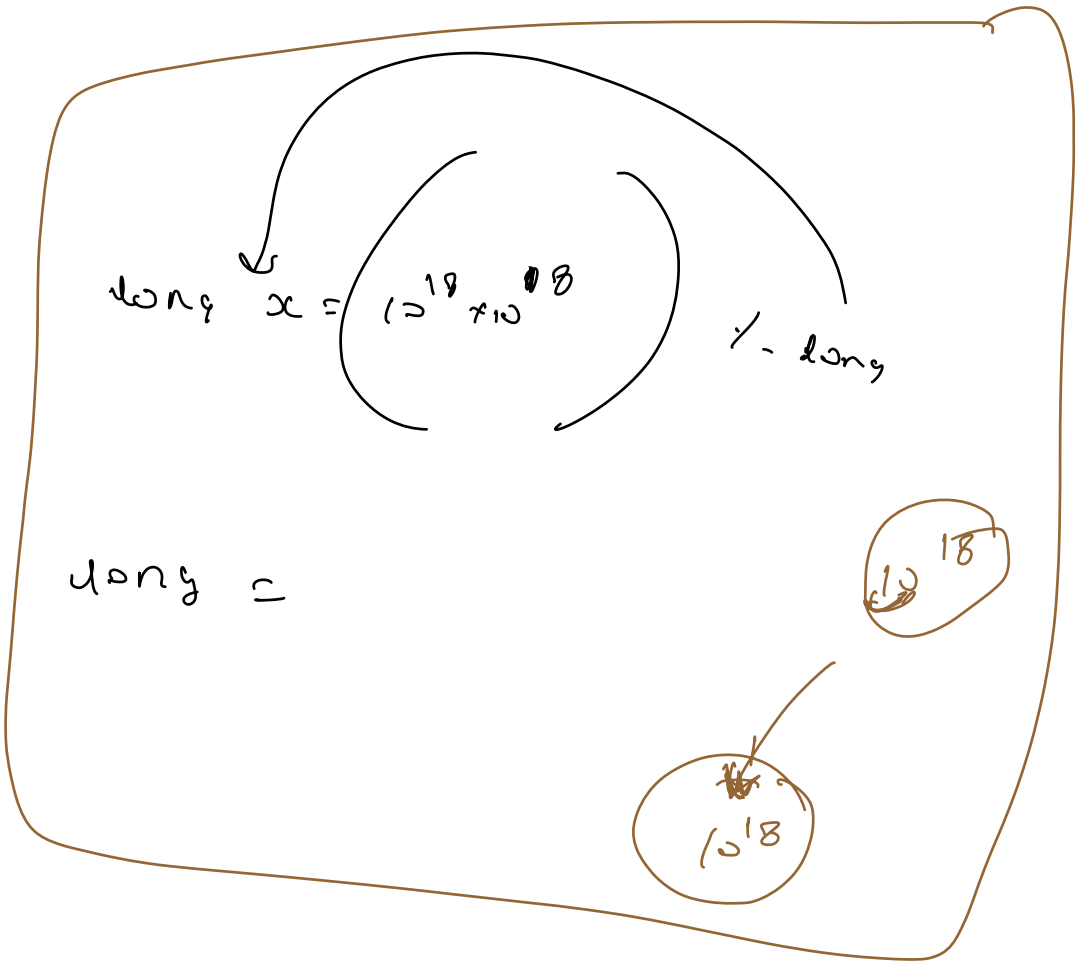
TC: O(1) for each input

TC: O(N) all input

Doubt.






long x = (10^18 * 10^18) % long
long =
10^18
10^18