Q) Given 2 strings A and B. len of strA → N, len of str B → m

Count all permutations of A present in B as substring.

[lowercase alphabets]

$N \leq m$

① A → xyz

B → xyxzxy [ans = 2]

② A → abcd

B → bacdgabcda [ans = 3]

③ A → aaba

B → aaababaa [ans = 3]

idea → for every substring of length N in B, compare with str A if that is a permutation of str A or not

[T.C → O(m*N)
S.C → O(1)]

str → [a b c d e f] , m = 6
(0 1 2 3 4 5)

substrings of length 1 → 'a' 'b' 'c' 'd' 'e' 'f' [M]

substring of length 2 → 'ab', 'bc', 'cd', 'de', 'ef' [m-1]

3 ——————→ [m-2]

4 ——————→ [m-3]

idea.2. Since window size is fixed, sliding window ✓

A → a b c d
0 1 2 3

B → b a c d g a b c d a
0 1 2 3 4 5 6 7 8 9
✗ ✗ ✗ ✗ ✗ ✗ ✓ ✓ ✓ ✓

count = 0 1 2 3

$fA[] \rightarrow [\ \underset{0}{1}\ \ \underset{1}{1}\ \ \underset{2}{1}\ \ \underset{3}{1}\ \ \underset{4}{0}\ \ \underset{5}{0}\ \ \underset{6}{0}\ \ -\ \ \underset{25}{0}\ ]$

$fB[] \rightarrow [\ \underset{0}{1}\ \ \underset{1}{1}\ \ \underset{2}{1}\ \ \underset{3}{1}\ \ \underset{4}{0}\ \ \underset{5}{0}\ \ \underset{6}{0}\ \ -\ \ \underset{25}{0}\ ]$

#code.→

```
fA[26], fB[26], ans = 0
for( i = 0; i < N; i++){
    ch1 = A[i] , ch2 = B[i]
    fA[ch1 - 'a'] ++;
    fB[ch2 - 'a'] ++;
}

if (compare(fA[], fB[]) == true) { ans += 1 }
```

```
for( i = N; i < M; i++){
        fB[ B[i] - 'a' ] ++;
        fB[ B[i-N] - 'a' ] --;
        if ( compare (fA[], fB[]) == true ) {
            ans += 1
        }
}
return ans;
```

T.C → O(M)
S.C → O(1)

```
boolean compare ( int[] fA, int[] fB){
        for( i = 0; i < 26; i++){
            if (fA[i] != fB[i]){
                return false;
            }
        }
        return true;
}
```

→ O(1)

Q2: Given a large text (str A of length N) and small pattern (string B of length m). [m<N]
Find the count of times B is present as substring in A.

A = "a c b a c a b c a" [N]
B = "b c a" [m]

a b c a c a b c a
b c a

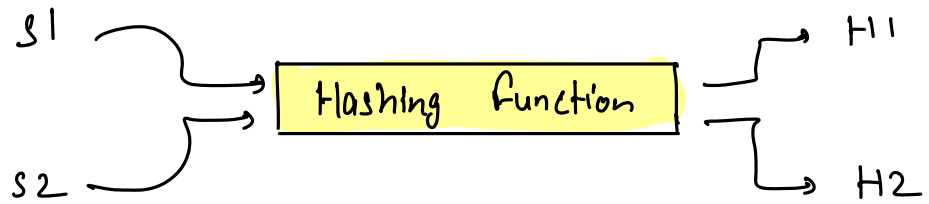
A = "a b a b c a b a b a c" (N) ans → 3
B = "a b a" [m]

idea-1. → Consider all substrings of length m in string A & check if they are equal to string B or not.

T.C → O(N*m)

Sliding Window ✗

Strings. integer



① Sum of ASCII values →

a b c

97 + 98 + 99 = 294

$$\begin{bmatrix} abc \\ acb \\ bac \\ bca \\ cab \\ cba \end{bmatrix} = 294$$

{There is a lot of collision.}

② ASCII value with position →

$a_1\ b_2\ c_3$ = (97 * 1) + (98 * 2) + (99 * 3)

= _____

{Collisions.}

Hint.→

Decimal no's. (10) → 41396

$4 \times 10^4 + 1 \times 10^3 + 3 \times 10^2 \times 9 \times 10^1 + 6 \times 10^0$

b a d e f

0 1 2 3 4

$(b) * 26^4 + (a) * 26^3 + (d) * 26^2 + (e) * 26^1 + (f) * 26^0$

$$\left\{ \sum_{i=0}^{len-1} ch[i] * p^{len-i-1} \right\}$$

$p \to 26$

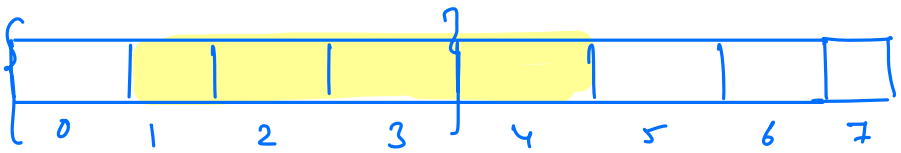
↓

very large (overflow)

$$\Rightarrow \left\{ \sum_{i=0}^{len-1} ch[i] * p^{len-i-1} \;\%\; m \right\}$$

$m \to 10^9+7.$

A →



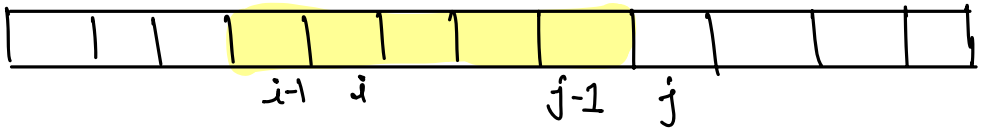
$m = 4$

$$H1 = A[0] \times p^3 + A[1] \times p^2 + A[2] \times p^1 + A[3] \times p^0$$

$$H2 = A[1] \times p^3 + A[2] \times p^2 + A[3] \times p^1 + A[4] \times p^0$$

$$H2 = (H1 - A[0] \times p^3) * p + A[4]$$

T.C → O(1)



H_{old} = hash value from $i-1$ to $j-1$ length = m.

H_{new} = hash value from i to j

H_{new} = ? in terms of H_{old}

$$H_{new} = \left(H_{old} - A[i-1] \times b^{m-1}\right) * b + A[j]$$

↳ Rolling Hash

Rabin Karp Algorithm

Disadvantage → collisions.

$mod \to 10^9+7$

% mod → $[0, 10^9+6]$

0 1 2 — ——— ——— 10^9+6

$S_1 = H_1 = 0$

$S_2 = H_2 = \frac{1}{mod}$

$S_3 = H_3 = \frac{2}{mod}$

$S_4 = H_4 = \frac{3}{mod}$

$\vdots$

$S_N = H_N = \frac{N-1}{mod}$

$N \le 10^5$

$mod = 10^9+7$

$= \frac{10^5-1}{10^9+7} \approx \frac{10^5}{10^9} \simeq 10^{-4} = 0.0001$

{ % of collision ⇒ $10^{-4} \times 10^2 = 0.01\%$ }

∴ Rabin Karp is 99.99% accurate.

≈ 100%.

[Z-Algo, KMP] 100%.

Given a large text (str A of length N) and small pattern (string B of length m). [m<N]
Find the count of times B is present as substring in A.

A = "abcbcabca" [N]
0 1 2 3 4 5 6 7 8

B = "bca" [m]
0 1 2

#code→

```
ans = 0

//1. find hash value of string B
long hb = 0, p = 26, long power = 1, mod = 10^9+7
for( i = m-1; i >= 0; i--){
        ch = B[i];
        hb = (hb + ch*power) % mod;
        power = (power * p) % mod;
}

//2. find hash value of first m characters of string A
long ha = 0, power = 1
for( i = m-1; i >= 0; i--){
        ch = A[i];
        ha = (ha + ch*power) % mod;
        power = (power * p) % mod;
}
```

```
if (ha == hb) { ans += 1 }

long largestpower = fastpower( p, m-1, mod);
for( i=m; i<N; i++){
    ha = [ha - A[i-m] * largestpower] * p + A[i]

    if (ha == hb) { ans += 1 }
}
return ans;
```

T.C → O(N)
S.C → O(log m)

Arrays
B.M
Maths
Recursion
Sorting
Searching
Hashing

Linkedlist
Stack
Queue
Trees
Trie
Heap

BackTracking
Greedy
D.P
Graphs

→ 80% problems can be expected from these topics in an interview.