Q1) Infinite stream of characters, Google.

After a new character comes, check if the current stream of characters forms a palindromic string or not.

a	✓
ab	✗
abc	✗
abcb	✗
abcba	✓
⋮	⋮

Idea-1 → After every new character, check if current string is a palindrome or not.

palidrome. $\boxed{s = rev(s)}$

map

integer value ?

Rolling hash can help?

a b c b a.

forward hash : p^4 p^3 p^2 p^1 p^0 $\{a*26^4 + b*26^3 + c*26^2 + b*26^1 + a*26^0\}$

backward hash p^0 p^1 p^2 p^3 p^4 $\{a*26^4 + b*26^3 + c*26^2 + b*26^1 + a*26^0\}$

$\{F.H == B.H\}$ ⇒ string is palindrome.

$p = 26$

	F.H.	B.H.
a.	$a * p^0$	$a * p^0$
ab	$a * p^1 + b * p^0$	$a * p^0 + b * p^1$
abc	$a * p^2 + b * p^1 + c * p^0$	$a * p^0 + b * p^1 + c * p^2$

$$\boxed{F.H_{new} = F.H_{old} * p + \text{new\_character}}$$

$$\boxed{B.H_{new} = B.H_{old} + \text{new\_character} * p^{length-1}}$$

$F.H_{new} == B.H_{new} \Rightarrow$ palindrome.

---

if there are N characters $\rightarrow$ T.C $\rightarrow O(N)$

Q: Find length of longest substring which contains all unique / distinct elements.

ch[] →	a	e	b	c	a	b	g	e	b	@	#	g	b	k	d	b	#
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

[ans → 6]

<u>idea-1.</u> Consider all the substrings & find longest substring with distinct characters.

$$\frac{N(N+1)}{2} * N \Rightarrow T.C \rightarrow O(N^3)$$

<u>idea-2.</u>

```
for( i = 0; i < N; i++){
    HashSet<char> hs, l = 0
    for( j = i; j < N; j++){
        if( s[j] is already present in the hs){
            update ans & break
        }
        else{
            l++
            hs.insert( s[j])
        }
    }
}
```

[T.C → O(N²).]

N log N → sorting ✗ → B.S

N

idea:3

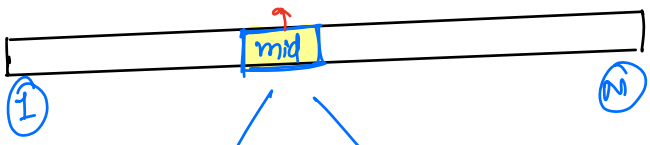
Binary Search (on length)

→ target = length of longest substring containing unique characters

→ Search space = length → [1 to N].

→ Condition =

check → [S.W.] using hashmap → O(N)



Substring with len=mid is not possible which contains all the unique elements.

Go to left:

right = mid-1

Substring with len=mid is there which contains all the unique elements.

update ans.

Go to right.

left = mid+1

{ T.C → O(N log N)
S.C → O(N) }

idea.4

ch[] →	a	e	b	c	a	b	g	e	b	@	#	g	b	K	d	b	#
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

$\boxed{\cancel{a}, \cancel{e}, b, \cancel{c}, \cancel{g}, \cancel{e}, \cancel{@}, \cancel{\#}, K, d, \#}$

ans → ~~4~~ ~~4~~ ~~5~~ ~~5~~ 6

pseudo-code.

```
i = 0 ,  j = 0 ,  Hashset<char> hs;
ans = 0
while ( j < n ){
        if ( s[j] is already present in hs){
                ans = Max (ans, j-i)
                while ( s[i] != s[j]){
                        remove s(i) from hs
                        i++
                }
                i++ , j++
        }
        else {
                add s[j] to hs , j++
        }
}
ans = Math.max( ans , j-i);
return ans;
```

$\left\{ \begin{array}{l} T.C \to O(N) \\ S.C \to O(N) \end{array} \right\}$

{ Break → 11:43 → 11:50 AM }

Q: Count all permutations of A in B as substrings.

A : a b c

B : a b c b a c a b c

[ans = 5].

idea: Permutations. →

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

(backtracking)

Generate all permutation & for every permutation find how many times that permutation is present in B as substring.

T.C → $O(n! * n)$

observation

[For every permutation → frequency of each character in all the permutation will remain same.]

length

(N) A : a c a

(M) B : a c a a b z c a a

freq → a b c d — — z

[2 0 1 0 0]

freq → a b c d - - - z

[2 0 1 1]

1 1 0 0

2 0 1

2 1

1 0

[ans → 3.].

pseudo-code.

```
farr [26] ;     // hashmap ✓

for ( i = 0 ; i < N ; i++) {
        ch = A[i]
        farr[ch - 'a'] ++
}                                   N

cfarr [26];

for ( i = 0 ; i < N ; i++) {
        ch = B[i]
        cfarr[ch - 'a'] ++
}                                   N

compare ( farr , cfarr) { count ++ } → 26.

s = 1 , e = N.

while ( e < M) {
        cfarr [ B[s-1] - 'a'] -- ; // excluding s-1th ch
        cfarr [ B[e] - 'a' ] ++ ; // including eth ch
        compare ( farr , cfarr) { count ++}
        s++ , e ++ ;
}                                   (M-N) * 26

return count;
```

T.C → $(M-N) * 26$.

A : a b c {N}

B : a b c b a c a b c {M}

req-frequency

a	b	c
1	1	1

curr-freq.

a	b	c
~~1~~	~~0~~	~~0~~
~~0~~	~~1~~	~~1~~
~~1~~	~~2~~	~~0~~
~~2~~	~~1~~	~~1~~
1	~~0~~	~~0~~
	1	1

need = ~~3~~ ~~1~~ ~~1~~

~~2~~ ~~0~~ ~~0~~

~~1~~ ~~1~~ ~~1~~

~~0~~ ~~0~~ 0

count = ~~1~~ ~~2~~ ~~3~~ ~~4~~ 5.

pseudo-code.

```
need = A.length(), count = 0
req_farr[26];
for( i = 0 ; i < N ; i++) {
        ch = A[i]
    req_farr[ch - 'a'] ++
}
```

Creating required frequency arr.

N

//Create the first window

```
cfarr[26];
for( i = 0 ; i < N ; i++) {
    if ( cfarr[B[i] - 'a'] < req_farr[B[i] - 'a'])
        need --
    cfarr (B[i] - 'a'] ++
}
```

//it is fulfilling our need.

N

```
if (need == 0) { count ++ }

s = 1 , e = N
while ( e < M) {   // exclude B[s-1], include B[e]
    if ( B[e] != B[s-1] ) {
        if ( cfarr[B[e] - 'a'] < req_farr[B[e] - 'a']){
            need --
        if ( cfarr [B[s-1] - 'a'] <= req_farr[B[s-1] - 'a']){
            need ++
        cfarr [B[e] - 'a'] ++ , cfarr(B[s-1] - 'a'] -- ;
    if (need == 0) { count ++ }
    s++ , e++
}
return count ;
```

M-N

S.C → O(1)

T.C → O(N+M)

H.W.

Q: find the shortest substring in A which contains all the characters of B.

A: A D O B E C O D E B A N C

B: A B C

→ prev 2 questions.

cfarr

A	B	C	D	E	- N -	O	- Z
1	1	1	1	1	1	1	1
~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	0
~~1~~	~~1~~	~~1~~	~~1~~	~~1~~	1	~~1~~	
~~0~~	~~2~~	~~0~~	~~2~~	~~2~~		~~2~~	
1	~~1~~	1	~~1~~	~~1~~		~~1~~	
			0	0		0	

need = ~~2~~

~~2~~	~~1~~
~~1~~	~~0~~
~~0~~	~~1~~ 0

req. farr

A	B	C	D
1	1	1	0

ans = ~~6~~ . ~~6~~ . 4