Agenda

1. Longest substring without Repeating Characters
2. First non repeating Element
3. Subarray with sum 0
4. Subarray with sum k

old $\rightarrow$ New ✓

M	18 Nov M $\rightarrow$ Break
W	20 Nov W $\rightarrow$ class
Next Fri $\rightarrow$ Break (22 Nov)	22 Nov F $\rightarrow$ class

1. Given a String s, find length of the longest substring without repeating characters.

str : " abcabcbb"   ans : 3

str : " bbbbb"   ans : 1

str : " pwwkew"   ans : 3

Brute Force : Go to all substrings, check if substr is valid (without duplicates), compare its length with ans and keep max in ans.

```
≈ N² substr            int ans=0
                       for (s ————— ) <
                           for (e ————— ) <
                               // s   e
                               for ( i=s ; i ≤ e ; i++ )
                                   put all chars in hs

                               if (hs.size() == e-s+1)
                                   ans = max(ans, e-s+1)
                           >
                       >
```

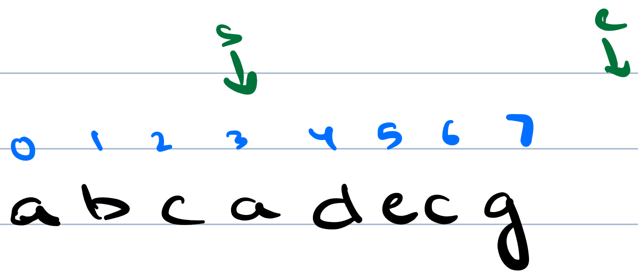
$[a \ b] = b - a + 1$

TC : $O(N^3)$

SC : $O(N)$

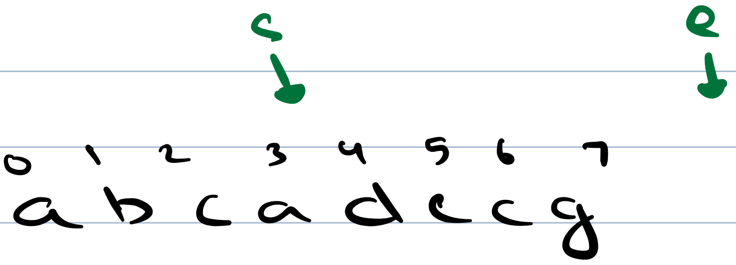
Optimized :

ans = 5 bcade

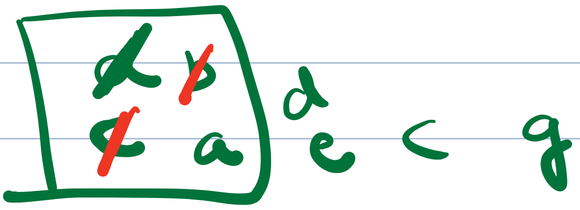


ans = ~~1~~ ~~2~~ ~~3~~ ~~4~~ 5

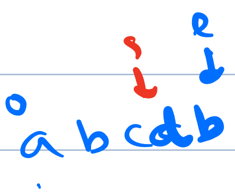
s = 0



ans = ~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~4~~ 5



```
// str, int n
int ans = 0
Hashset <char> hs
int s = 0
for (e = 0 ; e < n ; e++) {
    while (hs.contains (str[e] == true) {
        hs.remove (str[s]
        s++
    }
    hs.add (str[e])
    ans = max (ans, hs.size())
                        substr size
return ans
```



TC : O(N)

Every char can be processed twice, added once and removed once from hashset

SC : O(min(N, M)) str → a to z

M → size of character set
(ASCII → 128 chars)

---

2. Find the first $\overset{\text{unique}}{\text{non-repeating}}$ element. from start

Ex 1 ar[6] = < 1 2 3 1 2 5 > ans=3

Ex 2 ar[8] = < 4 3 3 2 5 6 4 5 > ans=2

Ex 3 ar[7] = < 2 6 8 4 7 2 9 > ans=6

Idea:

1. Insert all elements in hm
2. Iterate on hm and get 1st key with freq = 1

HM

elem	freq
1	~~1~~ 2
2	~~1~~ 2
3	1
5	1

[Note - in hashmap, insertion order is not maintained; when we print hashmap we'll get any order)

Idea:

1. Insert all elements in hm
2. Iterate on array and get elem with freq = 1

```
// int arr[], int N

Hashmap <int, int> hm
for (i=0 ; i<n ; i++) {
    if (hm. contains (arr[i]) == true)
            hm [arr[i]] ++
    else {
        hm.insert (arr[i], 1)
    }
}

for (i=0 ; i < N ; i++) {
    int freq = hm [ arr[i] ]
    if ( freq == 1)
            return arr[i]
}

return -1  // no element is
              unique
```

TC : O(N)

SC : O(N)

$\boxed{0:33}$

3. Given an array of N elements, check if there exists a subarray with sum equal to 0.

N = 10

	0	1	2	3	4	5	6	7	8	9
arr:	2	2	1	-3	4	3	1	-2	-3	2