Sliding window

12/02/2026
Thursday

Longest substring without repeating characters

s = cadbzabcd

cadbz → 5

zabcd → 5

Approach-1

generate all sub-strings

like

c

ca

cad

cadb

same with a

untill d

```
for(int i=0;i<n;i++)
{       hash[256]={0};
  for(int j=i;j<n;j++)
  {
      if(hash[s[j]]==1) break;
      len=j-i+1;
      max=max(len,max);
      hash[s[j]]=1;
  }
}
print(max)
```

```
      0 1 2 3 4 5 6 7 8
S =   c a d b z a b c d
      l
      r
```

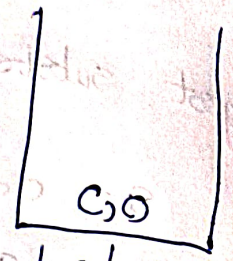
maxlen=0

first is c if it not contains then cal length

len = r - l + 1

len = 0 - 0 + 1

maxlen = 1



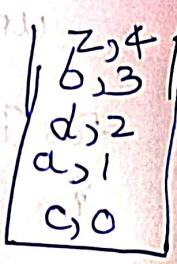
hash map
chars, index

and put it into hash map

next take a

```
      0 1 2 3 4 5 6 7 8
      c a d b z a b c d
      l   r
```

maxlen = 2




```
      0 1 2 3 4 5 6 7 8
      c a d b z a b c d
      l     r
```

maxlen = 4

```
      0 1 2 3 4 5 6 7 8
      c a d b z a b c d
      l           r
```

maxlen = 5

```
      c a d b z a b c d
      l           r
```

appears more than 1 already contains

so we take left pointer to right side of that index

7, 4
b 6
d 8
a, 5
c 7

maxlen = 0 1 2 3 4 5

we check if (mp.get(S[i]) >= l)

then only we update

l = mp.get(S[i]) + 1;

```
Function (string S)
{
    hash[256] -> -1
      n = s.size
      l=0  r=0  maxlen=0
      while (r<n)
      {
          if (hash[S[r]] != -1) {
            if (hash[S[r]] >= l)
            {
                l = hash[S[r]]+1;
            }
          }
        len = r-l+1;
        maxlen = max(len, maxlen);
        r++;
    }
    return maxlen
```

Time complexity :- O(n)

Space complexity :- O(256)