Sliding window

12/02/2026
Thursday

longest substring without repeating characters

s = cadbzabcd

cadbz   zabcd
↓   ↓
5   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
~~l~~ len= r-l+1
len= 0-0+1
and
put it into hash map

c,0
hash map
char, index

maxlen= ~~0~~ ~~1~~ 2

next take a

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

maxlen=~~3~~

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

z,4
b,3
d,2
a,1
c,0

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

maxlen=~~3~~4

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 ~~3~~ 6
d ~~2~~ 8
a, ~~1~~ 5
c ~~0~~ 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)