/\*

\* Time complexity : O(n)

\* Space complexity : O(n)

\* use hashSet to store substring

\* \*/

public int lengthOfLongestSubstring1(String s) {

int res = 0, left = 0, right = 0;

HashSet<Character> hashSet = new HashSet<>();

while(right < s.length()){

if(!hashSet.contains(s.charAt(right))){ // check if hashSet contains this character,

hashSet.add(s.charAt(right++)); // if not, add this character and move right to next

res = Math.max(res, hashSet.size());

}

else

hashSet.remove(s.charAt(left++)); // if hashSet already contains this character, then remove this

}

return res;

}

/\*

\* Time complexity : O(n)

\* Space complexity : O(n)

\* use a 256 integer array as a hashset, in order to save extra space

\* the basic algorithm is the same

\* \*/

public int lengthOfLongestSubstring2(String s) {

int[] set = new int[256];

Arrays.fill(set, -1);

int res = 0, left = -1;

for (int i = 0; i < s.length(); ++i) {

left = Math.max(left, set[s.charAt(i)]);

set[s.charAt(i)] = i;

res = Math.max(res, i - left);

}

return res;

}