**Longest K unique characters substring :-**

Medium Accuracy: 34.65% Submissions: 100K+ Points: 4

Given a string you need to print the size of the longest possible substring that has exactly **K unique** characters. If there is no possible substring then print -1.

**Example 1:**

**Input:**

S = "aabacbebebe", K = 3

**Output:**   
7

**Explanation**:   
"cbebebe" is the longest substring with 3 distinct characters.

**Example 2:**

**Input**:

S = "aaaa", K = 2

**Output:** -1

**Explanation**:   
There's no substring with 2 distinct characters.

**Your Task:**  
You don't need to read input or print anything. Your task is to complete the function **longestKSubstr()**which takes the string S and an integer K as input and returns the length of the longest substring with exactly K distinct characters. If there is no substring with exactly K distinct characters then return -1.

**Expected Time Complexity:**O(|S|).  
**Expected Auxiliary Space:**O(|S|).

**Constraints:**  
1 ≤ |S| ≤ 105  
1 ≤ K ≤ 26  
All characters are lowercase latin characters.

**Code :-**

//{ Driver Code Starts

//Initial template for C++

#include <bits/stdc++.h>

using namespace std;

// } Driver Code Ends

//User function template for C++

class Solution{

public:

int longestKSubstr(string s, int k){

unordered\_map<char,int> mp;

int n=s.size(), ans=-1, low=0, high=0;

while(high < n){

++mp[s[high]];

// added to map and map size > k

if(mp.size() > k){

// answer calculation

ans = max(ans, high-low);

char remove;

while(true){

remove = s[low];

--mp[remove];

++low;

if(mp[remove]==0)

break;

}

mp.erase(remove);

}

++high;

}

if(mp.size() == k){

ans = max(ans, high-low);

}

return ans;

}

};

//{ Driver Code Starts.

int main() {

int t;

cin >> t;

while (t--) {

string s;

cin >> s;

int k;

cin >> k;

Solution ob;

cout << ob.longestKSubstr(s, k) << endl;

}

}

// } Driver Code Ends

**T.C :- O(N)**

**S.C :- O(k), k = no. of unique characters**