/\*

Given a string, find the length of the longest substring without repeating characters.

Examples:

Given "abcabcbb", the answer is "abc", which the length is 3.

Given "bbbbb", the answer is "b", with the length of 1.

Given "pwwkew", the answer is "wke", with the length of 3. Note that the answer must be a substring, "pwke" is a subsequence and not a substring.

way-1:大循环是每一位开始，小循环是往后找有没有重复的.

way-2:用map，遍历s每一位，

如果找到重复的，就统计map长度，然后把begin到重复项的s\_count删除

如果没有，就加入map

\*/

class Solution {

public:

int lengthOfLongestSubstring(string s)

{

//way-1

/\*

if(s=="")

return 0;

string m1;

int j=0;

int chang;

int max=1;

int im;

for(int is=0;is<s.length()-1;is++)

{

j++;

m1=s.substr(is,j-is);

chang=j-is;

while(m1.find(s[j])==-1 && j<s.length())

{

m1=m1+s[j];

chang++;

j++;

}

if(j<s.length())

{

im=m1.find(s[j]);

is=is+im;

}

if(chang>max)

max=chang;

}

return max;

\*/

//way-2

map<char,int> s\_count;

int maxlength=0;

int begin=0;

int i=0;

while(i<s.size())

{

map<char,int>::iterator it=s\_count.find(s[i]);

if(it==s\_count.end())

{

s\_count[s[i]]=i;

}

else

{

maxlength=max(maxlength,i-begin);

for(int j=begin;j<it->second;j++)

s\_count.erase(s[j]);

begin=it->second+1;

s\_count[s[i]]=i;

}

i++;

}

maxlength=max(maxlength,i-begin);

return maxlength;

}

};