题目2：

给定一个字符串，找出不含有重复字符的最长子串的长度。

思路：

基础思路：外循环作为结果字符串的开始，第一次内循环作为字符串的结尾向后移动，最内层来对比从开始到结尾的前一个有没有和结尾相同的，如果没有相同的就把当前长度和之前的length相比并取更长的，否则就移动开始的位置。

葛林成思路：利用一个map数据结构，来标记每一个字符上一次出现的位置，初始化时全部置为0。遍历字符串：<把每一个字符对应的map值记为当前位。对每一个字符，如果对应map值不为0，说明之前出现过该字符，就先把最大长度与当前长度（当前位减去左指针的位）进行比较并取较大值，再把代表左指针的标记和该字符上一次出现的位置进行比较，并取靠右的。循环进行到最后一轮时，比较最大长度和最后一段无重复段的长度并取最大值。>本质思路类似于一个左指针和一个右指针，左指针指向上一个不重复的值，右指针不断右移直到结束。

存疑思路：依次对每个字符，在字符串中查找和它相同的，如果没找到就对下一个进行查找；如果找到了，就把字符串缩短到距离最远的两个相同字符之间，递归的查找更新后的字符串。

源码：

基础思路：

class Solution {

public:

int lengthOfLongestSubstring(string s) {

int length = 0;

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

int j = i + 1;

for (j = i+1; j < s.length(); j++) {

int k = i;

for (k = i; k < j; k++) {

if (s[k] == s[j])

break;

}

if (k != j) {

break;

}

}

length = (length >(j - i)) ? length : (j - i);

}

//cout << length << endl;

return length;

}

};

葛林成思路：

class Solution {

public:

int lengthOfLongestSubstring(string s) {

int length = 0;

map<char,int> pos;

//初始化pos

char tmp;

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

tmp = s[i];

pos[tmp] = 0;

}

//距离最近的上一个重复位(类似左指针）

int lasti = 0;

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

//遇到有重复

if (pos[s[i]] != 0) {

length = (length > (i-lasti)) ? length : (i - lasti);

lasti = (lasti > (pos[s[i]])) ? lasti : (pos[s[i]]);

}

//结束后处理最后一个不重复的情况!!!!!!!!!!!!!!!!!!这里很关键

if (i == (s.length() - 1)) {

length = (length > (s.length() - lasti)) ? length : (s.length() - lasti);

}

pos[s[i]] = i + 1;

}

return length;

}

};