二．题解

（一）思路

要找一个字符串的最长无重复子串，则可以通过枚举每个字符作为起始位置的最长无重复子串，来比较得出最长的那个子串。

但是，如果全部枚举的话，时间复杂度会很高，可以来找找规律。如果我们选择字符串中的第k个字符作为起始位置，且得到无重复的最长子串的结束位置为rk，则当我们选择第k+1个字符作为起始位置时，从k+1到rk的字符串定是不重复的，因此可以不断增大rk，知道出现右端出现重复字符。

（二）技术实现

使用双指针法，左指针指向子串的起始位置，右指针指向结束位置rk。两个指针构造了一个滑动窗口的假象，这是一个动态的窗口，窗口的大小会随着左右指针的移动而变化，左指针每次右移减小了窗口的大小，右指针的右移又增大了窗口的大小。

我们通过不断地右移左指针，每移一次代表着以下一个字符作为起始位置，右指针在保证不重复的情况下右移。如果右指针遇到了重复的字符，则停止移动右指针，计算子串的长度与目前的最大长度进行比较。

当左指针移动到了字符串的末尾时，所有的情况就都考虑到了，最大的长度即为答案。

（三）代码实现

class Solution {

public:

int lengthOfLongestSubstring(string s) {

//存储的是一个动态的字符串，随着左右指针的移动而变化

unordered\_set<char> occ;

int n = s.size();

int rk = -1, ans = 0;

//i代表着左指针

for (int i = 0; i < n; ++i) {

//左指针左移

if (i != 0) {

occ.erase(s[i - 1]);

}

//右指针右移

while (rk + 1 < n && !occ.count(s[rk + 1])) {

occ.insert(s[rk + 1]);

++rk;

}

//判断该次的最长字符串长度

ans = max(ans, rk - i + 1);

}

return ans;

}

};