剑指50 第一个只出现一次的字符

哈希表 和 有序哈希表 两种解法 【挺有用的！！！】

哈希表：注意这种hash用法：
用<char, bool>这种key，value的map
1.如果第一次遇到这个char，就添加到map中value为true。
2.如果第n次遇到这个char，就把value改为false。
3.没有遇到，自然map中不存在。
4.最后，需要涉及到顺序的，那就直接再遍历s就行了，我居然没想到！！

```
class Solution {
public:
    char firstUniqChar(string s) {
        unordered_map<char, bool> dic;
        for(char c : s)
            dic[c] = dic.find(c) == dic.end();
        for(char c : s)
            if(dic[c]) return c;
        return ' ';
    }
};
```

有序hash表：
方法二减少了第二轮遍历的循环次数。当字符串很长（重复字符很多）时，方法二只需遍历长度26，时间效率更高。
由于 C++ 未提供自带的链式哈希表，因此借助一个 vector 按序存储哈希表 dic 中的 key ，第二轮遍历此 vector 即可。
空间上则是仅多了一个26长度的vector数组。

```
class Solution {
public:
    char firstUniqChar(string s) {
        vector<char> keys;
        unordered_map<char, bool> dic;
        for(char c : s) {
            if(dic.find(c) == dic.end())
                keys.push_back(c);
            dic[c] = dic.find(c) == dic.end();
        }
        for(char c : keys) {
            if(dic[c]) return c;
        }
        return ' ';
    }
};

```

但是！！最秀的还是这个b玩意儿!!!

arr[c - 'a'] 能实现只针对字母表的有序**hash**表，里面值代表出现几次。

```cpp
class Solution {
public:
    char firstUniqChar(string s) {
        int arr[26] = {0};
        for (char ch : s) {
            arr[ch - 'a']++;
        }
        for (char c : s) {
            if (arr[c - 'a'] == 1) {
                return c;
            }
        }
        return ' ';
    }
};
```