# 题目

给定一个字符串，找到它的第一个不重复的字符，并返回它的索引。如果不存在，则返回 -1。

**示例：**

s = "leetcode"

返回 0

s = "loveleetcode"

返回 2

**提示：**你可以假定该字符串只包含小写字母。

# 分析

## 方法一：哈希表

class Solution {

public:

int firstUniqChar(string s) {

int hash[26] = {0}; // 预定义所有字母

for(auto ch : s)

{

hash[ch-'a']++;

}

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

{

if(hash[s[i]-'a'] == 1)

return i;

}

return -1;

}

};

复杂度分析：

时间复杂度：O(n)，其中n是字符串s的长度。我们需要进行两次遍历。

空间复杂度：O(|Σ|)，其中Σ是字符集，在本题中s只包含小写字母，因此|Σ|≤26。我们需要O(|Σ|)的空间存储哈希映射。

## 方法二：队列

**思路：**

我们也可以借助队列找到第一个不重复的字符。队列具有「先进先出」的性质，因此很适合用来找出第一个满足某个条件的元素。

具体地，我们使用与方法二相同的哈希映射，并且使用一个额外的队列，按照顺序存储每一个字符以及它们第一次出现的位置。当我们对字符串进行遍历时，设当前遍历到的字符为c，如果c不在哈希映射中，我们就将c与它的索引作为一个二元组放入队尾，否则我们就需要检查队列中的元素是否都满足「只出现一次」的要求，即我们不断地根据哈希映射中存储的值（是否为−1）选择弹出队首的元素，直到队首元素「真的」只出现了一次或者队列为空。

在遍历完成后，如果队列为空，说明没有不重复的字符，返回−1，否则队首的元素即为第一个不重复的字符以及其索引的二元组。

代码：

class Solution {

public:

int firstUniqChar(string s) {

unordered\_map<char, int> position;

queue<pair<char, int>> q;

int n = s.size();

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

if (!position.count(s[i])) {

position[s[i]] = i;

q.emplace(s[i], i);

}

else {

position[s[i]] = -1;

while (!q.empty() && position[q.front().first] == -1) {

q.pop();

}

}

}

return q.empty() ? -1 : q.front().second;

}

};

复杂度分析

时间复杂度：O(|Σ|)，其中n是字符串s的长度。遍历字符串的时间复杂度为O(n)，而在遍历的过程中我们还维护了一个队列，由于每一个字符最多只会被放入和弹出队列最多各一次，因此维护队列的总时间复杂度为O(|Σ|)，由于s 包含的字符种类数一定小于s的长度，因此O(|Σ|)在渐进意义下小于O(n)，可以忽略。

空间复杂度：O(|Σ|)，其中Σ是字符集，在本题中s只包含小写字母，因此|Σ|≤26。我们需要O(|Σ|)的空间存储哈希映射以及队列。