744 Find Smallest Letter Greater Than Target

2018年5月4日 14:13

Given a list of sorted characters letters containing only lowercase letters, and given a target letter target, find the smallest element in the list that is larger than the given target.
Letters also wrap around. For example, if the target is target = 'z' and letters = ['a', 'b'], the answer is 'a'.

Examples:

Input:
letters = ["c", "f", "j"]
target = "a"
Output: "c"

Input:
letters = ["c", "f", "j"]
target = "c"
Output: "f"

Input:
letters = ["c", "f", "j"]
target = "d"
Output: "f"

Input:
letters = ["c", "f", "j"]
target = "g"
Output: "j"

Input:
letters = ["c", "f", "j"]
target = "j"
Output: "c"

Input:
letters = ["c", "f", "j"]
target = "k"
Output: "c"

Note:

1. letters has a length in range [2, 10000].
2. letters consists of lowercase letters, and contains at least 2 unique letters.
3. target is a lowercase letter.

来自 <https://leetcode.com/problems/find-smallest-letter-greater-than-target/description/>

给定一个只包含小写字母的有序数组letters 和一个目标字母 target，寻找有序数组里面比目标字母大的最小字母。
数组里字母的顺序是循环的。举个例子，如果目标字母target = 'z' 并且有序数组为 letters = ['a', 'b']，则答案返回 'a'。

注:

1. letters长度范围在[2, 10000]区间内。
2. letters 仅由小写字母组成，最少包含两个不同的字母。
3. 目标字母target 是一个小写字母。

Solution for Python3:

```python
class Solution1:
    def nextGreatestLetter(self, letters, target):
        """
        :type letters: List[str]
        :type target: str
        :rtype: str
        """
        i = 0
        while i  < len(letters):
           if target < letters[i]:
               break
           else:
               i += 1
        if i < len(letters):
           return letters[i]
        return letters[0]

class Solution2:
    def nextGreatestLetter(self, letters, target):
        """
        :type letters: List[str]
        :type target: str
        :rtype: str
        """
        for c in letters:
            if c > target:
                return c
        return letters[0]

class Solution3:
    def nextGreatestLetter(self, letters, target):
        """
        :type letters: List[str]
        :type target: str
        :rtype: str
        """
        index = bisect.bisect(letters, target)
        return letters[index % len(letters)]
```

Solution for C++:

```cpp
class Solution1 {
public:
    char nextGreatestLetter(vector<char>& letters, char target) {
        int i = 0;
        while (i < letters.size()) {
            if (target < letters[i])
                break;
            else
                i++;
        }
        return i < letters.size() ? letters[i] : letters[0];
    }
};

class Solution2 {
public:
    char nextGreatestLetter(vector<char>& letters, char target) {
        for (char c : letters)
            if (c > target)
                return c;
        return letters[0];
    }
};

class Solution3 {
public:
    char nextGreatestLetter(vector<char>& letters, char target) {
        int index = upper_bound(letters.begin(), letters.begin() + letters.size(), target) - letters.begin();
        return letters[index % letters.size()];
    }
};
```