717 1-bit and 2-bit Characters

2018年5月4日 9:47

We have two special characters. The first character can be represented by one bit 0. The second character can be represented by two bits (10 or 11).

Now given a string represented by several bits. Return whether the last character must be a one-bit character or not. The given string will always end with a zero.

Example 1:

Input:

bits = [1, 0, 0]

Output: True

Explanation:

The only way to decode it is two-bit character and one-bit character. So the last character is one-bit character.

Example 2:

Input:

bits = [1, 1, 1, 0]

Output: False

Explanation:

The only way to decode it is two-bit character and two-bit character. So the last character is NOT one-bit character.

Note:

- 1 <= len(bits) <= 1000.
- bits[i] is always 0 or 1.

来自 <https://leetcode.com/problems/1-bit-and-2-bit-characters/description/>

有两种特殊字符。第一种字符可以用一比特0来表示。第二种字符可以用两比特(10 或 11)来表示。

现给一个由若干比特组成的字符串。问最后一个字符是否必定为一个一比特字符。给定的字符串总是由0结束。

示例 1:

输入:

bits = [1, 0, 0]

输出: True

解释:

唯一的编码方式是一个两比特字符和一个一比特字符。所以最后一个字符是一比特字符。

示例 2:

输入:

bits = [1, 1, 1, 0]

输出: False

解释:

唯一的编码方式是两比特字符和两比特字符。所以最后一个字符不是一比特字符。

注意:

- 1 <= len(bits) <= 1000.
- bits[i] 总是0 或 1.

Solution for Python3:

```
1  class Solution1:
2      def isOneBitCharacter(self, bits):
3          """
```

```
        :type bits: List[int]
        :rtype: bool
        """
        i, n = 0, len(bits) - 1
        while i < n:
            if bits[i]:
                i += 2
            else:
                i += 1
        return i == n

class Solution2:
    def isOneBitCharacter(self, bits):
        """
        :type bits: List[int]
        :rtype: bool
        """
        i = 0
        while i < len(bits) - 1:
            i += bits[i] + 1
        return i == len(bits) - 1

# 倒数第二个0必定是数组第一个0或者某个字符后的一个0(0 or 10 or 110)
# 无论是上面哪一种情况，倒数第二个0及其之前的元素都对结果没影响
# 主要考虑倒数第二个0和最后一个0直接含有多少个1
# 只有偶数个1才能使数组最后一个0比定为1比特字符
class Solution3:
    def isOneBitCharacter(self, bits):
        """
        :type bits: List[int]
        :rtype: bool
        """
        P = bits.pop()
        while bits and bits.pop():
            P ^= 1 #奇数个1使得P结果为1，偶数个使P为0
        return P == 0
```

Solution for C++:

```
class Solution1 {
public:
    bool isOneBitCharacter(vector<int>& bits) {
        int i = 0, n = bits.size() - 1;
        while (i < n) {
```

```
            if (bits[i])
                i += 2;
            else
                i += 1;
        }
        return i == n;
    }
};

class Solution2 {
public:
    bool isOneBitCharacter(vector<int>& bits) {
        int i = 0, n = bits.size() - 1;
        while (i < n) {
            i += bits[i] + 1;
        }
        return i == n;
    }
};

class Solution3 {
public:
    bool isOneBitCharacter(vector<int>& bits) {
        int i = bits.size() - 2;
        while (i >= 0 && bits[i])
            i--;
        return (bits.size() - 1 - i) % 2;
    }
};
```