1-bit and 2-bit Characters

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 cha
racter 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 cha
racter is NOT one-bit character.
```

Note:

- `1 .`
- `bits[i]` is always `0` or `1`.

Solution 1

```
class Solution {
    public boolean isOneBitCharacter(int[] bits) {
        int n = bits.length, i = 0;
        while (i < n - 1) {
            if (bits[i] == 0) i++;
            else i += 2;
        }
        return i == n - 1;
    }
}
```

written by shawngao original link here

Solution 2

We don't need to traverse the whole array, just check the last part of it.

1. if there is only one symbol in array the answer is always true (as last element is 0)
2. if there are two 0s at the end again the answer is true no matter what the rest symbols are(...1100, ...1000,)
3. if there is 1 right before the last element(...10), the outcome depends on the count of sequential 1, i.e.
   a) if there is odd amount of 1(10, ...01110, etc) the answer is false as there is a single 1 without pair
   b) if it's even (110, ...011110, etc) the answer is true, as 0 at the end doesn't have anything to pair with

```
class Solution {
    public boolean isOneBitCharacter(int[] bits) {
        int ones = 0;
        for (int i = bits.length - 2; i >= 0 && bits[i] != 0 ; i--) { //starting fr
om one but last, as last one is always 0
            ones++;
        }
        if (ones % 2 > 0) return false;
        return true;
    }
}
```

written by Alafreulein original link here

Solution 3

...

```
    if not bits: return False
    n = len(bits)

    index = 0
    while index < n:
        if index == n-1 : return True
        if bits[index] == 1:
            index += 2
        else: index += 1
    return False
```

...

written by rsquare original link here