605 Can Place Flowers

2018年4月16日 19:14

Suppose you have a long flowerbed in which some of the plots are planted and some are not. However, flowers cannot be planted in adjacent plots - they would compete for water and both would die.

Given a flowerbed (represented as an array containing 0 and 1, where 0 means empty and 1 means not empty), and a number **n**, return if **n** new flowers can be planted in it without violating the no-adjacent-flowers rule.

Example 1:

Input: flowerbed = [1,0,0,0,1], n = 1

Output: True

Example 2:

Input: flowerbed = [1,0,0,0,1], n = 2

Output: False

Note:

1. The input array won't violate no-adjacent-flowers rule.
2. The input array size is in the range of [1, 20000].
3. **n** is a non-negative integer which won't exceed the input array size.

来自 <https://leetcode.com/problems/can-place-flowers/description/>

假设你有一个很长的花坛，一部分地块种植了花，另一部分却没有。可是，花卉不能种植在相邻的地块上，它们会争夺水源，两者都会死去。

给定一个花坛（表示为一个数组包含0和1，其中0表示没种植花，1表示种植了花），和一个数 **n** 。能否在不打破种植规则的情况下种入 **n** 朵花？能则返回True，不能则返回False。

示例 1:

输入: flowerbed = [1,0,0,0,1], n = 1

输出: True

示例 2:

输入: flowerbed = [1,0,0,0,1], n = 2

输出: False

注意:

1. 数组内已种好的花不会违反种植规则。
2. 输入的数组长度范围为 [1, 20000]。
3. **n** 是非负整数，且不会超过输入数组的大小。

Solution for Python3:

```
1 class Solution1:
2     def canPlaceFlowers(self, flowerbed, n):
3         """
```

```
        :type flowerbed: List[int]
        :type n: int
        :rtype: bool
        """
        pre = -2
        for i in range(len(flowerbed)):
            if flowerbed[i]:
                if i - pre >= 2:

                    n -= (i - pre - 2) // 2
                pre = i
        if len(flowerbed) - 1 > pre :
            n -= (len(flowerbed) - 1 - pre) // 2
        return True if n <= 0 else False

class Solution2:
    def canPlaceFlowers(self, flowerbed, n):
        """
        :type flowerbed: List[int]
        :type n: int
        :rtype: bool
        """
        i, cnt = 0, 0
        while cnt < n and i < len(flowerbed):
            if flowerbed[i] == 0:
                prev = flowerbed[i-1] if i > 0 else 0
                nextv = flowerbed[i+1] if i
< len(flowerbed)-1 else 0
                if prev == nextv == 0:
                    cnt += 1
                    flowerbed[i] = 1
            i += 1
        return cnt == n
```

Solution for C++:

```
class Solution {
public:
    bool canPlaceFlowers(vector<int>& flowerbed, int n)
{
```

```
        int cnt = 0, next, prev;
        for (int i = 0; i < flowerbed.size() && cnt
< n; i++) {
            if (flowerbed[i] == 0) {
                next = (i == flowerbed.size() - 1) ?
0 : flowerbed[i + 1];
                prev = (i == 0) ? 0 : flowerbed[i - 1];
                if (next == 0 && prev == 0) {
                    flowerbed[i] = 1;
                    cnt++;
                }
            }
        }
        return cnt == n;
    }
};
```