Can Place Flowers

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.

Solution 1

We need to justify a greedy solution.

Call a plot ready if the very first flower is allowed to be planted there.
Consider the left-most ready plot x (if it exists). If x+1 is not ready, then we increase our answer strictly by planting at x, since x does not disturb any ready plots. If x+1 is ready, then planting at x instead of x+1 is atleast as good, since x disturbs only {x, x+1}, whereas x+1 disturbs {x, x+1, x+2}.

Now our implementation is trivial. For each plot from left to right, if we can plant a flower there, then do so. We can plant a flower if the left neighbor is 0 (or we are on the left edge), AND the right neighbor is 0 (or we are on the right edge).

```
def canPlaceFlowers(self, A, N):
    for i, x in enumerate(A):
        if (not x and (i == 0 or A[i-1] == 0)
                and (i == len(A)-1 or A[i+1] == 0)):
            N -= 1
            A[i] = 1
    return N <= 0
```

written by awice original link here

Solution 2

Greedily place a flower at every vacant spot encountered from left to right!

```
public class Solution {
    public boolean canPlaceFlowers(int[] flowerbed, int n) {
        int count = 0;
        for(int i = 0; i < flowerbed.length && count < n; i++) {
            if(flowerbed[i] == 0) {
      //get next and prev flower bed slot values. If i lies at the ends the next and
prev are considered as 0.
                int next = (i == flowerbed.length - 1) ? 0 : flowerbed[i + 1];
                int prev = (i == 0) ? 0 : flowerbed[i - 1];
                if(next == 0 && prev == 0) {
                    flowerbed[i] = 1;
                    count++;
                }
            }
        }

        return count == n;
    }
}
```

written by soumyadeep2007 original link here

Solution 3

C++

```
class Solution {
public:
    bool canPlaceFlowers(vector<int>& bed, int n) {
        for (int i = 0; i < bed.size(); i++) {
            if (!bed[i] && (i == 0 || !bed[i - 1]) && (i == bed.size() - 1 || !bed[i
+ 1])) {
                bed[i] = 1;
                n--;
            }
        }
        return n <= 0;
    }
};
```

Java

```
public class Solution {
    public boolean canPlaceFlowers(int[] bed, int n) {
        for (int i = 0; i < bed.length; i++) {
            if (bed[i] == 0 && (i == 0 || bed[i - 1] == 0) && (i == bed.length - 1
|| bed[i + 1] == 0)) {
                bed[i] = 1;
                n--;
            }
        }
        return n <= 0;
    }
}
```

written by alexander original link here