Magical String

A magical string **S** consists of only '1' and '2' and obeys the following rules:

The string **S** is magical because concatenating the number of contiguous occurrences of characters '1' and '2' generates the string **S** itself.

The first few elements of string **S** is the following: **S** = "1221121221221121122……"

If we group the consecutive '1's and '2's in **S**, it will be:

1 22 11 2 1 22 1 22 11 2 11 22 ……

and the occurrences of '1's or '2's in each group are:

1 2 2 1 1 2 1 2 2 1 2 2 ……

You can see that the occurrence sequence above is the **S** itself.

Given an integer N as input, return the number of '1's in the first N number in the magical string **S**.

Note: N will not exceed 100,000.

Example 1:

```
Input: 6
Output: 3
Explanation: The first 6 elements of magical string S is "12211" and it contains th
ree 1's, so return 3.
```

Solution 1

Algorithm:

1. Create an `int` array `a` and initialize the first 3 elements with `1, 2, 2`.
2. Create two pointers `head` and `tail`. `head` points to the number which will be used to generate new numbers. `tail` points to the next empty position to put the new number. Then keep generating new numbers until `tail` >= `n`.
3. Need to create the array 1 element more than `n` to avoid overflow because the last round `head` might points to a number `2`.
4. A trick to flip number back and forth between `1` and `2`: `num = num ^ 3`

```
public class Solution {
    public int magicalString(int n) {
        if (n <= 0) return 0;
        if (n <= 3) return 1;

        int[] a = new int[n + 1];
        a[0] = 1; a[1] = 2; a[2] = 2;
        int head = 2, tail = 3, num = 1, result = 1;

        while (tail < n) {
            for (int i = 0; i < a[head]; i++) {
                a[tail] = num;
                if (num == 1 && tail < n) result++;
                tail++;
            }
            num = num ^ 3;
            head++;
        }

        return result;
    }
}
```

written by shawngao original link here

Solution 2

Just build enough of the string and then count.

```
int magicalString(int n) {
    string S = "122";
    int i = 2;
    while (S.size() < n)
        S += string(S[i++] - '0', S.back() ^ 3);
    return count(S.begin(), S.begin() + n, '1');
}
```

written by StefanPochmann original link here

Solution 3

```
public int magicalString(int n) {
        StringBuilder magic = new StringBuilder("12211212212211211122");
        int pt1 = 12, pt2 = magic.length(), count = 0; //initiate pointers
        //generate sequence directly
        while(magic.length() < n){
            if(magic.charAt(pt1) == '1'){
                if(magic.charAt(pt2-1) == '1') magic.append(2);
                else magic.append(1);
                pt2++;
            }else{ //==2
                if(magic.charAt(pt2-1) == '1') magic.append(22);
                else magic.append(11);
                pt2+=2;
            }
            pt1++;
        }
        for(int i=0;i<n;i++)
            if(magic.charAt(i)=='1') count++;
        return count;
    }
```

written by cgxy1995 original link here