K-th Symbol in Grammar

On the first row, we write a `0`. Now in every subsequent row, we look at the previous row and replace each occurrence of `0` with `01`, and each occurrence of `1` with `10`.

Given row `N` and index `K`, return the `K`-th indexed symbol in row `N`. (The values of `K` are 1-indexed.) (1 indexed).

```
Examples:
Input: N = 1, K = 1
Output: 0

Input: N = 2, K = 1
Output: 0

Input: N = 2, K = 2
Output: 1

Input: N = 4, K = 5
Output: 1

Explanation:
row 1: 0
row 2: 01
row 3: 0110
row 4: 01101001
```

Note:

1. `N` will be an integer in the range `[1, 30]`.
2. `K` will be an integer in the range `[1, 2^(N-1)]`.

Solution 1

The whole structure can be viewed a binary tree, when a node is 0, their two children nodes are 0 and 1, similarly, when a node is 1, two children nodes are 1 and 0. We can know whether the position of K is a left node or a right node by dividing 2. If K is even, current node is right child, and its parent is the (K/2)th node in previous row; else if K is odd, current node is left child and its parent is the ((K+1)/2)th node in previous row.
The value of current node depends on its parent node, without knowing its parent node value, we still cannot determine current node value. That's why we need recursion, we keep going previous row to find the parent node until reach the first row. Then all the parent node value will be determined after the recursion function returns.

```
class Solution {
public:
    int kthGrammar(int N, int K) {
 if (N == 1) return 0;
 if (K % 2 == 0) return (kthGrammar(N - 1, K / 2) == 0) ? 1 : 0;
 else return (kthGrammar(N - 1, (K + 1) / 2) == 0) ? 0 : 1;
    }
};
```

written by grandyang original link here

Solution 2

If there are odd number of 1s in K-1's binary representation, it is 1, otherwise 0.

```
class Solution(object):
    def kthGrammar(self, N, K):
        """
        :type N: int
        :type K: int
        :rtype: int
        """
        return bin(K-1).count('1')%2
```

written by cks original link here

Solution 3

```
public int kthGrammar(int N, int K) {
    return Integer.bitCount(K-1) & 1;
}
```

written by tyuan73 original link here