1545. Find Kth Bit in Nth Binary String

Given two positive integers `n` and `k`, the binary string `Sn` is formed as follows:

- `S1 = "0"`
- `Si = Si - 1 + "1" + reverse(invert(Si - 1))` for `i > 1`

Where `+` denotes the concatenation operation, `reverse(x)` returns the reversed string `x`, and `invert(x)` inverts all the bits in `x` (`0` changes to `1` and `1` changes to `0`).

For example, the first four strings in the above sequence are:

- `S1 = "0"`
- `S2 = "0`**`1`**`1"`
- `S3 = "011`**`1`**`001"`
- `S4 = "0111001`**`1`**`0110001"`

Return *the* `kth` *bit in* `Sn`. It is guaranteed that `k` is valid for the given `n`.

Example 1:

```
Input: n = 3, k = 1
Output: "0"
Explanation: S3 is "0111001".
The 1st bit is "0".
```

Example 2:

```
Input: n = 4, k = 11
Output: "1"
Explanation: S4 is "011100110110001".
The 11th bit is "1".
```

Constraints:

- `1 <= n <= 20`
- `1 <= k <= 2n - 1`

1545. Find Kth Bit in Nth Binary String

```
/*
    Brute force
    Time complexity: O(2^n)
    Space complexity: O(2^n)
*/
class Solution {
    public:
        std::string invert(std::string s){
            int n=s.size();
            for(int i=0;i<n;++i){
                s[i]=s[i]=='0'?'1':'0';
            }
            return s;
        }

        std::string reverse(std::string s){
            std::reverse(s.begin(),s.end());
            return s;
        }

        char findKthBit(int n, int k){
            std::string s="0";
            for(int i=2;i<=n;++i){
                s+="1"+reverse(invert(s));
            }
            return s[k-1];
        }
};
```

1545. Find Kth Bit in Nth Binary String

```
/*
    Recursion
    Time complexity: O(n)
    Space complexity: O(n)
*/
class Solution{
    public:
        char findKthBit(int n, int k) {

            auto solve=[&](int n,int k,auto& self)->char{
                if(n==1) return '0';
                int nb_bits=(1<<n)-1;
                int mid=nb_bits/2+1;
                if(k==mid) return '1';
                if(k<mid) return self(n-1,k,self);
                return self(n-1,nb_bits-k+1,self)=='0'?'1':'0';
            };

            return solve(n,k,solve);
        }
};
```

1545. Find Kth Bit in Nth Binary String

```
/*
    Divide and conquer
    Time complexity: O(n)
    Space complexity: O(1)
*/
class Solution {
    public:
        char findKthBit(int n, int k){
            int nb_bits=(1<<n)-1;
            int nb_inversions=0;
            while(k>1){
                int mid=nb_bits/2;
                if(k==mid+1) return nb_inversions%2==0?'1':'0';
                if(k>mid){
                    k=nb_bits-k+1;
                    nb_inversions++;
                }
                nb_bits/=2;
            }
            return nb_inversions%2==0?'0':'1';
        }
};
```

1545. Find Kth Bit in Nth Binary String

```
/*
    Bit manipulation
    Time complexity: O(1)
    Space complexity: O(1)
*/
class Solution {
    public:
        char findKthBit(int n, int k) {
                // Find the group where k belong
                int rightmost_set_bit_pos=k&-k;

                // If the group is set => k-th bit is inverted
                // Otherwise => k-th bit remains same as original
                bool is_inverted=((k/rightmost_set_bit_pos)>>1&1)==1;

                // Determine if the original bit (before any inversion) would be 1
                // This is true if k is even (i.e., its least significant bit is 0)
                bool is_original_bit_one=(k&1)==0;

                if (is_inverted){
                    // If we're in the inverted part, we need to flip the bit
                    return is_original_bit_one?'0':'1';
                }

                // If we're not in the inverted part, return the original bit
                return is_original_bit_one?'1':'0';

        }
};
```