交替位二进制数

给定一个正整数，检查它的二进制表示是否总是 0、1 交替出现：换句话说，就是二进制表示中相邻两位的数字永不相同。

示例 1：

输入：n = 5

输出：true

解释：5 的二进制表示是：101

示例 2：

输入：n = 7

输出：false

解释：7 的二进制表示是：111.

示例 3：

输入：n = 11

输出：false

解释：11 的二进制表示是：1011.

示例 4：

输入：n = 10

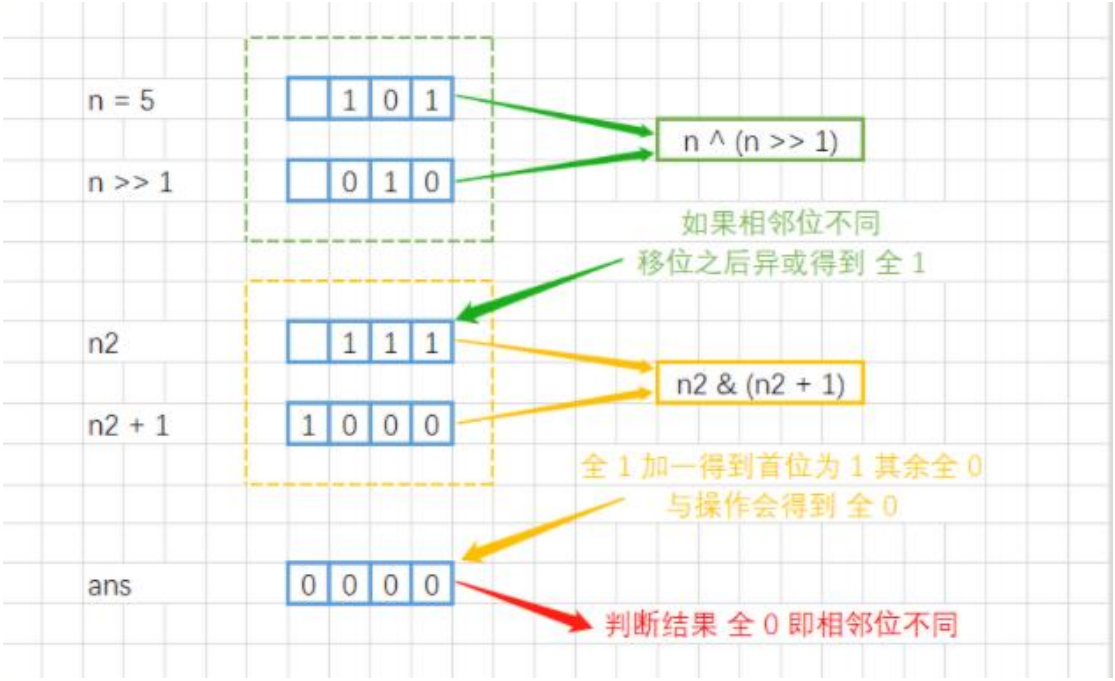
输出：true

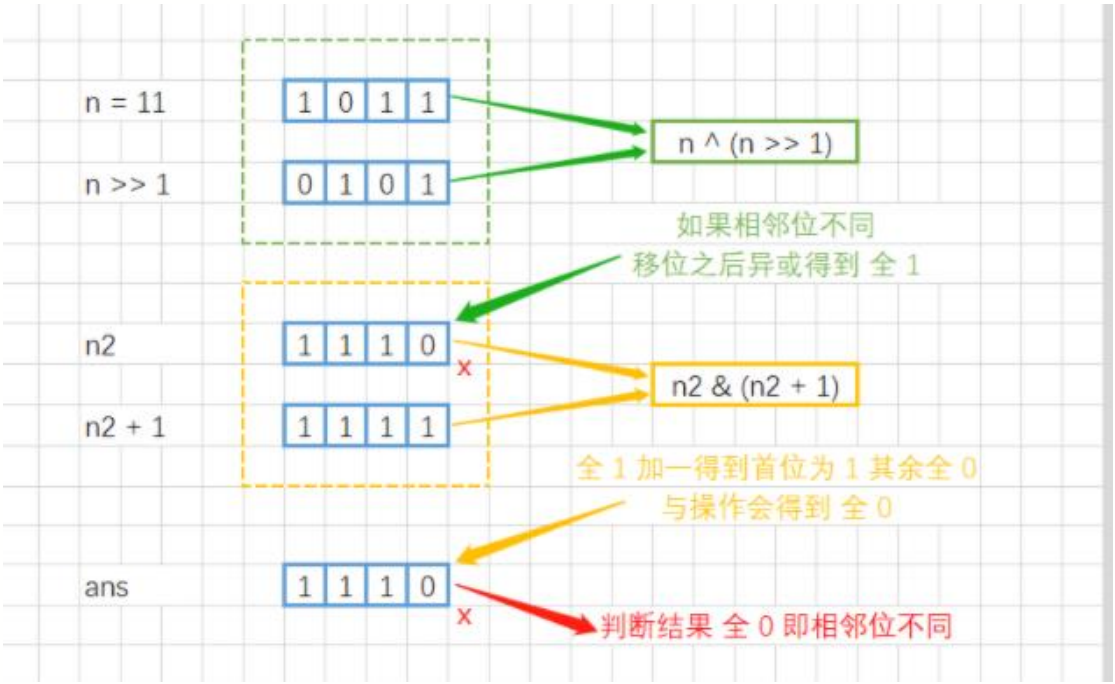
解释：10 的二进制表示是：1010.

示例 5：

输入：n = 3

输出：false






```
class Solution {
public:
    bool hasAlternatingBits(int n) {
        /*
        原理就是凡是符合题目中的交替位二进制，那必然错位异或的话全是 1，所以将错位异或的值+1 后就会生成只有 1 个二进制 1 的值，再用 n ^ (n-1) 进行检查消除一个二进制 1 后是否为 0 即可
        */
```

```
        /*
        if((res&0)==0)
            return true;
        else
            return false;
        */
        //此解法下测试发现 对于 7 无法正解 0111^1110=1001  &0000=0000  但并非交
替
        //构造错位值,错位异或
        n = (n ^ (n >> 1));
        return (n & ((long)n + 1)) == 0;
    }
};
```