```
/*
给定一个正整数，检查他是否为交替位二进制数：换句话说，就是他的二进制数相邻的两个位数永不相等。

示例 1:

输入: 5
输出: True
解释:
5的二进制数是: 101

示例 2:

输入: 7
输出: False
解释:
7的二进制数是: 111

示例 3:

输入: 11
输出: False
解释:
11的二进制数是: 1011

 示例 4:

输入: 10
输出: True
解释:
10的二进制数是: 1010

来源：力扣（LeetCode）
链接：https://leetcode-cn.com/problems/binary-number-with-alternating-bits
著作权归领扣网络所有。商业转载请联系官方授权，非商业转载请注明出处。
*/
```

分析:

- 方法一:联系异或操作的特性,可以判断n^(n>>1)的有效各位是否为1即可
- 方法二:在int范围内,交替位二进制数是可以被穷举完的

方法一: C_异或特性

```
bool hasAlternatingBits(int n)
{
    unsigned int temp = (n^(n>>1));
    if(temp !=2147483647 )
```

```
    {
        return (temp&(temp+1))==0;
    }
    else
    {
        // 对应特殊值(n==1431655765)
        return true;
    }
}
/*
执行结果:
通过
显示详情
执行用时 :4 ms, 在所有 C 提交中击败了69.14% 的用户
内存消耗 :6.7 MB, 在所有 C 提交中击败了81.13%的用户
*/
```

方法二:C_穷举法

```
bool hasAlternatingBits(int n)
{
    switch(n)
    {
        case 0x00000001:
        case 0x00000005:
        case 0x00000015:
        case 0x00000055:
        case 0x00000155:
        case 0x00000555:
        case 0x00001555:
        case 0x00005555:
        case 0x00015555:
        case 0x00055555:
        case 0x00155555:
        case 0x00555555:
        case 0x01555555:
        case 0x05555555:
        case 0x15555555:
        case 0x55555555:
        case 0x00000002:
        case 0x0000000a:
        case 0x0000002a:
        case 0x000000aa:
        case 0x000002aa:
        case 0x00000aaa:
        case 0x00002aaa:
        case 0x0000aaaa:
        case 0x0002aaaa:
        case 0x000aaaaa:
        case 0x002aaaaa:
        case 0x00aaaaaa:
```

```
        case 0x02aaaaaa:
        case 0x0aaaaaaa:
        case 0x2aaaaaaa:
        case 0xaaaaaaaa:
            return true;
            break;
        default:
            return false;
    }
}

/*
执行结果:
通过
显示详情
执行用时 :4 ms, 在所有 C 提交中击败了69.14% 的用户
内存消耗 :6.6 MB, 在所有 C 提交中击败了92.45%的用户
*/
```

AlimyBreak
2019.08.12