```
/*
格雷编码是一个二进制数字系统，在该系统中，两个连续的数值仅有一个位数的差异。

给定一个代表编码总位数的非负整数 n，打印其格雷编码序列。格雷编码序列必须以 0 开头。

示例 1:

输入: 2
输出: [0,1,3,2]
解释:
00 - 0
01 - 1
11 - 3
10 - 2

对于给定的 n，其格雷编码序列并不唯一。
例如，[0,2,3,1] 也是一个有效的格雷编码序列。

00 - 0
10 - 2
11 - 3
01 - 1

示例 2:

输入: 0
输出: [0]
解释: 我们定义格雷编码序列必须以 0 开头。
     给定编码总位数为 n 的格雷编码序列，其长度为 2n。当 n = 0 时，长度为 20 = 1。
     因此，当 n = 0 时，其格雷编码序列为 [0]。

来源：力扣（LeetCode）
链接：https://leetcode-cn.com/problems/gray-code
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:

- 探索**方法一(失败方法)**:从左到右从右到左依次迭代取反某一位,在$n = 2, 3$的时候碰巧通过,对于$n \geq 4$时,总会在生成$4(n-1)$个数后陷入死循环且无法再生成更多(只验证到了$n = 10$).

```
/*
n = 4, c = 12
[0,1,3,7,15,11,9,8,10,14,6,2,0,1,3,7]
n = 5, c = 16
[0,1,3,7,15,31,23,19,17,16,18,22,30,14,6,2,0,1,3,7,15,31,23,19,17,16,18,22,30,14,6,2]
n = 6, c = 20
[0,1,3,7,15,31,63,47,39,35,33,32,34,38,46,62,30,14,6,2,0,1,3,7,15,31,63,47,39,35,33,32,34,38,46,62,30,14,6,2,0,1,3,7,15,31,63,47,39,35,33,32,34,38,46,62,30,14,6,2,0,1,3,7]
n = 7, c = 24
```

```
[0,1,3,7,15,31,63,127,95,79,71,67,65,64,66,70,78,94,126,62,30,14,6,2,0,1,3,
7,15,31,63,127,95,79,71,67,65,64,66,70,78,94,126,62,30,14,6,2,0,1,3,7,15,31
,63,127,95,79,71,67,65,64,66,70,78,94,126,62,30,14,6,2,0,1,3,7,15,31,63,127
,95,79,71,67,65,64,66,70,78,94,126,62,30,14,6,2,0,1,3,7,15,31,63,127,95,79,
71,67,65,64,66,70,78,94,126,62,30,14,6,2,0,1,3,7,15,31,63,127]
n = 8, C = 28
[0,1,3,7,15,31,63,127,255,191,159,143,135,131,129,128,130,134,142,158,190,2
54,126,62,30,14,6,2,0,1,3,7,15,31,63,127,255,191,159,143,135,131,129,128,13
0,134,142,158,190,254,126,62,30,14,6,2,0,1,3,7,15,31,63,127,255,191,159,143
,135,131,129,128,130,134,142,158,190,254,126,62,30,14,6,2,0,1,3,7,15,31,63,
127,255,191,159,143,135,131,129,128,130,134,142,158,190,254,126,62,30,14,6,
2,0,1,3,7,15,31,63,127,255,191,159,143,135,131,129,128,130,134,142,158,190,
254,126,62,30,14,6,2,0,1,3,7,15,31,63,127,255,191,159,143,135,131,129,128,1
30,134,142,158,190,254,126,62,30,14,6,2,0,1,3,7,15,31,63,127,255,191,159,14
3,135,131,129,128,130,134,142,158,190,254,126,62,30,14,6,2,0,1,3,7,15,31,63
,127,255,191,159,143,135,131,129,128,130,134,142,158,190,254,126,62,30,14,6
,2,0,1,3,7,15,31,63,127,255,191,159,143,135,131,129,128,130,134,142,158,190
,254,126,62,30,14,6,2,0,1,3,7]
n = 9, C = 32
[0,1,3,7,15,31,63,127,255,511,383,319,287,271,263,259,257,256,258,262,270,2
86,318,382,510,254,126,62,30,14,6,2,0,1,3,7,15,31,63,127,255,511,383,319,28
7,271,263,259,257,256,258,262,270,286,318,382,510,254,126,62,30,14,6,2,0,1,
3,7,15,31,63,127,255,511,383,319,287,271,263,259,257,256,258,262,270,286,31
8,382,510,254,126,62,30,14,6,2,0,1,3,7,15,31,63,127,255,511,383,319,287,271
,263,259,257,256,258,262,270,286,318,382,510,254,126,62,30,14,6,2,0,1,3,7,1
5,31,63,127,255,511,383,319,287,271,263,259,257,256,258,262,270,286,318,382
,510,254,126,62,30,14,6,2,0,1,3,7,15,31,63,127,255,511,383,319,287,271,263,
259,257,256,258,262,270,286,318,382,510,254,126,62,30,14,6,2,0,1,3,7,15,31,
63,127,255,511,383,319,287,271,263,259,257,256,258,262,270,286,318,382,510,
254,126,62,30,14,6,2,0,1,3,7,15,31,63,127,255,511,383,319,287,271,263,259,2
57,256,258,262,270,286,318,382,510,254,126,62,30,14,6,2,0,1,3,7,15,31,63,12
7,255,511,383,319,287,271,263,259,257,256,258,262,270,286,318,382,510,254,1
26,62,30,14,6,2,0,1,3,7,1...
n = 10,C = 36
[0,1,3,7,15,31,63,127,255,511,1023,767,639,575,543,527,519,515,513,512,514,
518,526,542,574,638,766,1022,510,254,126,62,30,14,6,2,0,1,3,7,15,31,63,127,
255,511,1023,767,639,575,543,527,519,515,513,512,514,518,526,542,574,638,76
6,1022,510,254,126,62,30,14,6,2,0,1,3,7,15,31,63,127,255,511,1023,767,639,5
75,543,527,519,515,513,512,514,518,526,542,574,638,766,1022,510,254,126,62,
30,14,6,2,0,1,3,7,15,31,63,127,255,511,1023,767,639,575,543,527,519,515,513
,512,514,518,526,542,574,638,766,1022,510,254,126,62,30,14,6,2,0,1,3,7,15,3
1,63,127,255,511,1023,767,639,575,543,527,519,515,513,512,514,518,526,542,5
74,638,766,1022,510,254,126,62,30,14,6,2,0,1,3,7,15,31,63,127,255,511,1023,
767,639,575,543,527,519,515,513,512,514,518,526,542,574,638,766,1022,510,25
4,126,62,30,14,6,2,0,1,3,7,15,31,63,127,255,511,1023,767,639,575,543,527,51
9,515,513,512,514,518,526,542,574,638,766,1022,510,254,126,62,30,14,6,2,0,1
,3,7,15,31,63,127,255,511,1023,767,639,575,543,527,519,515,513,512,514,518,
526,542,574,638,766,1022,...

*/
```

- **方法二**:格雷编码的生成过程，好吧，我背的答案,$vi[i] = i \ \ xor \ \ (i/2)$.
- **方法三**:递归算法,致谢ygh578890840
    - 若已知$n-1$的格雷编码集合,在前面加一个0,然后把$n-1$的集合倒转在前面加一个1,就得到了n的格雷编码集合.

    - 翻译一下：把前(n-1)层的所有格雷码都按(n-1)位取反（代码中是用对称取值来实现取反），然后把第n位置为1，依次压入返回数组.
- **方法四**:**方法三**的内存优化版本,但好像没有空间上的提升.

方法一:C++_失败方法

```

class Solution
{
    public:
        vector<int> grayCode(int n)
        {
            vector<int> vi                          ;
            int         cur_v       =   0           ;
            int         i           =   0           ;
            int         sum_count   =   pow(2,n)    ;
            int         cur_count   =   0           ;
            int         cur_idx     =   0           ;
            int         direction   =   0           ;
            if(n >=0)
            {
                vi.push_back(cur_v) ;
                cur_count = 1       ;
                cur_idx   = 0     ;
                while(cur_count<sum_count)
                {
                    cur_v ^= (0x01 << cur_idx);
                    if(cur_idx == n-1)
                    {
                        direction = 0; /* left->right*/
                        cur_idx--;
                    }
                    else if(cur_idx == 0)
                    {
                        direction = 1;
                        cur_idx++;

                    }
                    else if(direction==0)
                    {
                        cur_idx--;
                    }
                    else
                    {
                        cur_idx++;
                    }

                    vi.push_back(cur_v) ;
                    cur_count++;
                }
            }
            return vi;
        }
};

/*
```

```
在 n >= 4,无法生成满足要求的序列.
*/

```

方法三:C++_递归算法

```
class Solution
{
    public:
        vector<int> grayCode(int n)
        {
            if(n == 0)
            {
                return {0};
            }
            vector<int> vi  =   grayCode(n-1)   ;
            int         i   =   0               ;
            /*n层共有 2^n 个格雷码, 前(n-1)层共有 2^(n-1)个格雷码*/
            for( i = pow(2,n-1)-1; i >= 0 ; i--)
            {
                vi.push_back(vi[i] | (0x01 << (n-1)));
            }
            return vi;
    }
};
/*
执行结果:
通过
显示详情
执行用时 :12 ms, 在所有 cpp 提交中击败了18.56% 的用户
内存消耗 :8.7 MB, 在所有 cpp 提交中击败了20.23%的用户
*/
```

方法四:C++_递归算法传引用优化

```
class Solution
{

    private:

        void helper(    vector<int>&    vi  ,
                        int             n

        )
        {
            if(n==0)
            {
                vi.push_back(0);
                return;
            }
            helper(vi,n-1);

            /*n层共有 2^n 个格雷码, 前(n-1)层共有 2^(n-1)个格雷码*/
            for(int i = pow(2,n-1)-1; i >= 0 ; i--)
```

```cpp
            {
                vi.push_back(vi[i] | (0x01 << (n-1)));
            }
        }


    public:
        vector<int> grayCode(int n)
        {


            vector<int> vi;

            if(n > 0)
            {
                helper(vi,n);
            }
            return vi;
    }
};
/*
执行结果:
通过
显示详情
执行用时 :8 ms, 在所有 cpp 提交中击败了65.15% 的用户
内存消耗 :8.8 MB, 在所有 cpp 提交中击败了10.45%的用户
*/
```

AlimyBreak
2019.12.20