

## Bulb Switcher II

There is a room with  $n$  lights which are turned on initially and 4 buttons on the wall. After performing exactly  $m$  unknown operations towards buttons, you need to return how many different kinds of status of the  $n$  lights could be.

Suppose  $n$  lights are labeled as number  $[1, 2, 3 \dots, n]$ , function of these 4 buttons are given below:

1. Flip all the lights.
2. Flip lights with even numbers.
3. Flip lights with odd numbers.
4. Flip lights with  $(3k + 1)$  numbers,  $k = 0, 1, 2, \dots$

### Example 1:

**Input:**  $n = 1, m = 1$ .

**Output:** 2

**Explanation:** Status can be: [on], [off]

### Example 2:

**Input:**  $n = 2, m = 1$ .

**Output:** 3

**Explanation:** Status can be: [on, off], [off, on], [off, off]

### Example 3:

**Input:**  $n = 3, m = 1$ .

**Output:** 4

**Explanation:** Status can be: [off, on, off], [on, off, on], [off, off, off], [off, on, on].

**Note:**  $n$  and  $m$  both fit in range  $[0, 1000]$ .

## Solution 1

We only need to consider special cases which  $n \leq 2$  and  $m < 3$ . When  $n > 2$  and  $m \geq 3$ , the result is 8.

The four buttons:

1. Flip all the lights.
2. Flip lights with even numbers.
3. Flip lights with odd numbers.
4. Flip lights with  $(3k + 1)$  numbers,  $k = 0, 1, 2, \dots$

If we use button 1 and 2, it equals to use button 3.

Similarly...

$1 + 2 \rightarrow 3, 1 + 3 \rightarrow 2, 2 + 3 \rightarrow 1$

So, there are only 8 cases.

All\_on, 1, 2, 3, 4, 1+4, 2+4, 3+4

And we can get all the cases, when  $n > 2$  and  $m \geq 3$ .

```
class Solution {
    public int flipLights(int n, int m) {
        if(m==0) return 1;
        if(n==1) return 2;
        if(n==2&&m==1) return 3;
        if(n==2) return 4;
        if(m==1) return 4;
        if(m==2) return 7;
        if(m>=3) return 8;
        return 8;
    }
}
```

written by [woshifumingyuan](#) original link [here](#)

## Solution 2

When  $n \leq 1$ , the solution is trivial.

From the 4 types of operations,

1. Flip all the lights.
2. Flip lights with even numbers.
3. Flip lights with odd numbers.
4. Flip lights with  $(3k + 1)$  numbers,  $k = 0, 1, 2, \dots$

There are three important observations:

1. For any operation, only odd or even matters, i.e. 0 or 1. Two same operations equal no operation.
2. The first 3 operations can be reduced to 1 or 0 operation. For example, flip all + flip even = flip odd. So the result of the first 3 operations is the same as either 1 operation or original.
3. The solution for  $n > 3$  is the same as  $n = 3$ .  
For example, 1 0 0 ....., I use 0 and 1 to represent off and on.  
The state of 2nd digit indicates even flip; The state of 3rd digit indicates odd flip; And the state difference of 1st and 3rd digits indicates  $3k+1$  flip.

In summary, the question can be simplified as  $m \leq 3$ ,  $n \leq 3$ . I am sure you can figure out the rest easily.

```
class Solution {
public:
    int flipLights(int n, int m) {
        if (m == 0 || n == 0) return 1;
        if (n == 1) return 2;
        if (n == 2) return m == 1 ? 3 : 4;
        if (m == 1) return 4;
        return m == 2 ? 7 : 8;
    }
};
```

written by [zestypan](#) original link [here](#)

## Solution 3

Suppose we did  $f[0]$  of the first operation,  $f[1]$  of the second,  $f[2]$  of the third, and  $f[3]$  of the fourth, where  $\text{sum}(f) == m$ .

First, all these operations commute: doing operation A followed by operation B yields the same result as doing operation B followed by operation A. Also, doing operation A followed by operation A again is the same as doing nothing. So really, we only needed to know the residues  $\text{cand}[i] = f[i] \% 2$ . There are only 16 different possibilities for the residues in total, so we can try them all.

We'll loop  $\text{cand}$  through all 16 possibilities  $(0, 0, 0, 0), (0, 0, 0, 1), \dots, (1, 1, 1, 1)$ . A necessary and sufficient condition for  $\text{cand}$  to be valid is that  $\text{sum}(\text{cand}) \% 2 == m \% 2$  and  $\text{sum}(\text{cand}) \leq m$ , as only when these conditions are satisfied can we find some  $f$  with  $\text{sum}(f) == m$  and  $\text{cand}[i] = f[i] \% 2$ .

Also, as the sequence of lights definitely repeats every 6 lights, we could replace  $n$  with  $\min(n, 6)$ . Actually, we could replace it with  $\min(n, 3)$ , as those lights are representative: that is, knowing the first 3 lights is enough to reconstruct what the next 3 lights will be. If the first 3 lights are X, Y, Z, then with a little effort we can prove the next 3 lights will be  $(X^Y^Z), Z, Y$ .

```
def flipLights(self, n, m):
    seen = set()
    for cand in itertools.product((0, 1), repeat = 4):
        if sum(cand) % 2 == m % 2 and sum(cand) <= m:
            A = []
            for i in xrange(min(n, 3)):
                light = 1
                light ^= cand[0]
                light ^= cand[1] and i % 2
                light ^= cand[2] and i % 2 == 0
                light ^= cand[3] and i % 3 == 0
            A.append(light)
            seen.add(tuple(A))

    return len(seen)
```

written by [awice](#) original link [here](#)

From [Leetcode](#).