Problem 672: Bulb Switcher II

Problem Information

Difficulty: Medium
Acceptance Rate: 0.00%
Paid Only: No

Problem Description

There is a room with

n

bulbs labeled from

1

to

n

that all are turned on initially, and

four buttons

on the wall. Each of the four buttons has a different functionality where:

Button 1:

Flips the status of all the bulbs.

Button 2:

Flips the status of all the bulbs with even labels (i.e.,

2, 4, ...

).

Button 3:

Flips the status of all the bulbs with odd labels (i.e.,

1, 3, ...

).

Button 4:

Flips the status of all the bulbs with a label

$j = 3k + 1$

where

$k = 0, 1, 2, ...$

(i.e.,

1, 4, 7, 10, ...

).

You must make

exactly

presses

button presses in total. For each press, you may pick

any

of the four buttons to press.

Given the two integers

n

and

presses

, return

the number of

different possible statuses

after performing all

presses

button presses

.

Example 1:

Input:

n = 1, presses = 1

Output:

2

Explanation:

Status can be: - [off] by pressing button 1 - [on] by pressing button 2

Example 2:

Input:

n = 2, presses = 1

Output:

3

Explanation:

Status can be: - [off, off] by pressing button 1 - [on, off] by pressing button 2 - [off, on] by pressing button 3

Example 3:

Input:

n = 3, presses = 1

Output:

4

Explanation:

Status can be: - [off, off, off] by pressing button 1 - [off, on, off] by pressing button 2 - [on, off, on] by pressing button 3 - [off, on, on] by pressing button 4

Constraints:

1 <= n <= 1000

0 <= presses <= 1000

Code Snippets

C++:

```
class Solution {
public:
int flipLights(int n, int presses) {

}
};
```

Java:

```
class Solution {
public int flipLights(int n, int presses) {

}
}
```

Python3:

```
class Solution:
def flipLights(self, n: int, presses: int) -> int:
```

Python:

```
class Solution(object):
def flipLights(self, n, presses):
"""
:type n: int
:type presses: int
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number} n
* @param {number} presses
* @return {number}
*/
var flipLights = function(n, presses) {

};
```

TypeScript:

```
function flipLights(n: number, presses: number): number {

};
```

C#:

```
public class Solution {
public int FlipLights(int n, int presses) {

}
}
```

C:

```
int flipLights(int n, int presses) {

}
```

Go:

```
func flipLights(n int, presses int) int {

}
```

Kotlin:

```
class Solution {
fun flipLights(n: Int, presses: Int): Int {

}
}
```

Swift:

```
class Solution {
func flipLights(_ n: Int, _ presses: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn flip_lights(n: i32, presses: i32) -> i32 {

}
}
```

Ruby:

```
# @param {Integer} n
# @param {Integer} presses
# @return {Integer}
def flip_lights(n, presses)

end
```

PHP:

```
class Solution {

/**
* @param Integer $n
* @param Integer $presses
* @return Integer
*/
function flipLights($n, $presses) {

}
}
```

Dart:

```
class Solution {
int flipLights(int n, int presses) {

}
}
```

Scala:

```
object Solution {
def flipLights(n: Int, presses: Int): Int = {

}
```

```
}
```

Elixir:

```
defmodule Solution do
@spec flip_lights(n :: integer, presses :: integer) :: integer
def flip_lights(n, presses) do

end
end
```

Erlang:

```
-spec flip_lights(N :: integer(), Presses :: integer()) -> integer().
flip_lights(N, Presses) ->
.
```

Racket:

```
(define/contract (flip-lights n presses)
(-> exact-integer? exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Bulb Switcher II
* Difficulty: Medium
* Tags: math, search
*
* Approach: Optimized algorithm based on problem constraints
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public:
int flipLights(int n, int presses) {
```

```
}
};
```

Java Solution:

```
/**
* Problem: Bulb Switcher II
* Difficulty: Medium
* Tags: math, search
*
* Approach: Optimized algorithm based on problem constraints
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public int flipLights(int n, int presses) {

}
}
```

Python3 Solution:

```
"""
Problem: Bulb Switcher II
Difficulty: Medium
Tags: math, search

Approach: Optimized algorithm based on problem constraints
Time Complexity: O(n) to O(n^2) depending on approach
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:
def flipLights(self, n: int, presses: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def flipLights(self, n, presses):
"""
:type n: int
:type presses: int
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Bulb Switcher II
* Difficulty: Medium
* Tags: math, search
*
* Approach: Optimized algorithm based on problem constraints
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {number} n
* @param {number} presses
* @return {number}
*/
var flipLights = function(n, presses) {

};
```

TypeScript Solution:

```
/**
* Problem: Bulb Switcher II
* Difficulty: Medium
* Tags: math, search
*
* Approach: Optimized algorithm based on problem constraints
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

function flipLights(n: number, presses: number): number {
```

```
};
```

C# Solution:

```
/*
* Problem: Bulb Switcher II
* Difficulty: Medium
* Tags: math, search
*
* Approach: Optimized algorithm based on problem constraints
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public int FlipLights(int n, int presses) {

}
}
```

C Solution:

```
/*
* Problem: Bulb Switcher II
* Difficulty: Medium
* Tags: math, search
*
* Approach: Optimized algorithm based on problem constraints
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

int flipLights(int n, int presses) {

}
```

Go Solution:

```
// Problem: Bulb Switcher II
// Difficulty: Medium
```

```
// Tags: math, search
//
// Approach: Optimized algorithm based on problem constraints
// Time Complexity: O(n) to O(n^2) depending on approach
// Space Complexity: O(1) to O(n) depending on approach

func flipLights(n int, presses int) int {

}
```

Kotlin Solution:

```
class Solution {
fun flipLights(n: Int, presses: Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func flipLights(_ n: Int, _ presses: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: Bulb Switcher II
// Difficulty: Medium
// Tags: math, search
//
// Approach: Optimized algorithm based on problem constraints
// Time Complexity: O(n) to O(n^2) depending on approach
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn flip_lights(n: i32, presses: i32) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer} n
# @param {Integer} presses
# @return {Integer}
def flip_lights(n, presses)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $n
* @param Integer $presses
* @return Integer
*/
function flipLights($n, $presses) {

}
}
```

Dart Solution:

```
class Solution {
int flipLights(int n, int presses) {

}
}
```

Scala Solution:

```
object Solution {
def flipLights(n: Int, presses: Int): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec flip_lights(n :: integer, presses :: integer) :: integer
def flip_lights(n, presses) do

end
end
```

Erlang Solution:

```
-spec flip_lights(N :: integer(), Presses :: integer()) -> integer().
flip_lights(N, Presses) ->
.
```

Racket Solution:

```
(define/contract (flip-lights n presses)
(-> exact-integer? exact-integer? exact-integer?)
)
```