Problem 1318: Minimum Flips to Make a OR b Equal to c

Problem Information

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

Problem Description

Given 3 positives numbers

a

,

b

and

c

. Return the minimum flips required in some bits of

a

and

b

to make (

a

OR

b

==

c

). (bitwise OR operation).

Flip operation consists of change

any

single bit 1 to 0 or change the bit 0 to 1 in their binary representation.

Example 1:

Input:

a = 2, b = 6, c = 5

Output:

3

Explanation:

After flips a = 1 , b = 4 , c = 5 such that (

a

OR

b

==

c

)

Example 2:

Input:

a = 4, b = 2, c = 7

Output:

1

Example 3:

Input:

a = 1, b = 2, c = 3

Output:

0

Constraints:

1 <= a <= 10^9

1 <= b <= 10^9

1 <= c <= 10^9

Code Snippets

C++:

```
class Solution {
public:
int minFlips(int a, int b, int c) {

}
};
```

Java:

```
class Solution {
public int minFlips(int a, int b, int c) {

}
}
```

Python3:

```
class Solution:
def minFlips(self, a: int, b: int, c: int) -> int:
```

Python:

```
class Solution(object):
def minFlips(self, a, b, c):
"""
:type a: int
:type b: int
:type c: int
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number} a
* @param {number} b
* @param {number} c
* @return {number}
*/
var minFlips = function(a, b, c) {

};
```

TypeScript:

```
function minFlips(a: number, b: number, c: number): number {

};
```

C#:

```
public class Solution {
public int MinFlips(int a, int b, int c) {

}
}
```

C:

```


int minFlips(int a, int b, int c){

}
```

Go:

```
func minFlips(a int, b int, c int) int {

}
```

Kotlin:

```
class Solution {
fun minFlips(a: Int, b: Int, c: Int): Int {

}
}
```

Swift:

```
class Solution {
func minFlips(_ a: Int, _ b: Int, _ c: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn min_flips(a: i32, b: i32, c: i32) -> i32 {

}
}
```

Ruby:

```
# @param {Integer} a
# @param {Integer} b
# @param {Integer} c
# @return {Integer}
def min_flips(a, b, c)

end
```

PHP:

```
class Solution {

/**
* @param Integer $a
* @param Integer $b
* @param Integer $c
* @return Integer
*/
function minFlips($a, $b, $c) {

}
}
```

Scala:

```
object Solution {
def minFlips(a: Int, b: Int, c: Int): Int = {

}
}
```

Solutions

C++ Solution:

```
/*
* Problem: Minimum Flips to Make a OR b Equal to c
* Difficulty: Medium
* Tags: general
*
* 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 minFlips(int a, int b, int c) {

}
};
```

Java Solution:

```
/**
* Problem: Minimum Flips to Make a OR b Equal to c
* Difficulty: Medium
* Tags: general
*
* 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 minFlips(int a, int b, int c) {

}
}
```

Python3 Solution:

```
"""
Problem: Minimum Flips to Make a OR b Equal to c
```

```
Difficulty: Medium
Tags: general

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 minFlips(self, a: int, b: int, c: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def minFlips(self, a, b, c):
"""
:type a: int
:type b: int
:type c: int
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Minimum Flips to Make a OR b Equal to c
* Difficulty: Medium
* Tags: general
*
* 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} a
* @param {number} b
* @param {number} c
* @return {number}
```

```
*/
var minFlips = function(a, b, c) {

};
```

TypeScript Solution:

```
/**
* Problem: Minimum Flips to Make a OR b Equal to c
* Difficulty: Medium
* Tags: general
*
* 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 minFlips(a: number, b: number, c: number): number {

};
```

C# Solution:

```
/*
* Problem: Minimum Flips to Make a OR b Equal to c
* Difficulty: Medium
* Tags: general
*
* 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 MinFlips(int a, int b, int c) {

}
}
```

C Solution:

```
/*
* Problem: Minimum Flips to Make a OR b Equal to c
* Difficulty: Medium
* Tags: general
*
* 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 minFlips(int a, int b, int c){

}
```

Go Solution:

```
// Problem: Minimum Flips to Make a OR b Equal to c
// Difficulty: Medium
// Tags: general
//
// 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 minFlips(a int, b int, c int) int {

}
```

Kotlin Solution:

```
class Solution {
fun minFlips(a: Int, b: Int, c: Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func minFlips(_ a: Int, _ b: Int, _ c: Int) -> Int {
```

```
}
}
```

Rust Solution:

```
// Problem: Minimum Flips to Make a OR b Equal to c
// Difficulty: Medium
// Tags: general
//
// 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 min_flips(a: i32, b: i32, c: i32) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer} a
# @param {Integer} b
# @param {Integer} c
# @return {Integer}
def min_flips(a, b, c)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $a
* @param Integer $b
* @param Integer $c
* @return Integer
*/
function minFlips($a, $b, $c) {
```

```
}
}
```

Scala Solution:

```
object Solution {
def minFlips(a: Int, b: Int, c: Int): Int = {

}
}
```