Problem 1742: Maximum Number of Balls in a Box

Problem Information

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

Problem Description

You are working in a ball factory where you have

n

balls numbered from

lowLimit

up to

highLimit

inclusive

(i.e.,

n == highLimit - lowLimit + 1

), and an infinite number of boxes numbered from

1

to

infinity

.

Your job at this factory is to put each ball in the box with a number equal to the sum of digits of the ball's number. For example, the ball number

321

will be put in the box number

3 + 2 + 1 = 6

and the ball number

10

will be put in the box number

1 + 0 = 1

.

Given two integers

lowLimit

and

highLimit

, return

the number of balls in the box with the most balls.

Example 1:

Input:

lowLimit = 1, highLimit = 10

Output:

2

Explanation:

Box Number: 1 2 3 4 5 6 7 8 9 10 11 ... Ball Count: 2 1 1 1 1 1 1 1 1 0 0 ... Box 1 has the most number of balls with 2 balls.

Example 2:

Input:

lowLimit = 5, highLimit = 15

Output:

2

Explanation:

Box Number: 1 2 3 4 5 6 7 8 9 10 11 ... Ball Count: 1 1 1 1 2 2 1 1 1 0 0 ... Boxes 5 and 6 have the most number of balls with 2 balls in each.

Example 3:

Input:

lowLimit = 19, highLimit = 28

Output:

2

Explanation:

Box Number: 1 2 3 4 5 6 7 8 9 10 11 12 ... Ball Count: 0 1 1 1 1 1 1 1 1 2 0 0 ... Box 10 has the most number of balls with 2 balls.

Constraints:

1 <= lowLimit <= highLimit <= 10

5

Code Snippets

C++:

```
class Solution {
public:
int countBalls(int lowLimit, int highLimit) {

}
};
```

Java:

```
class Solution {
public int countBalls(int lowLimit, int highLimit) {

}
}
```

Python3:

```
class Solution:
def countBalls(self, lowLimit: int, highLimit: int) -> int:
```

Python:

```
class Solution(object):
def countBalls(self, lowLimit, highLimit):
"""
:type lowLimit: int
:type highLimit: int
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number} lowLimit
* @param {number} highLimit
* @return {number}
*/
var countBalls = function(lowLimit, highLimit) {

};
```

TypeScript:

```
function countBalls(lowLimit: number, highLimit: number): number {

};
```

C#:

```
public class Solution {
public int CountBalls(int lowLimit, int highLimit) {

}
}
```

C:

```
int countBalls(int lowLimit, int highLimit) {

}
```

Go:

```
func countBalls(lowLimit int, highLimit int) int {

}
```

Kotlin:

```
class Solution {
fun countBalls(lowLimit: Int, highLimit: Int): Int {

}
}
```

Swift:

```
class Solution {
func countBalls(_ lowLimit: Int, _ highLimit: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn count_balls(low_limit: i32, high_limit: i32) -> i32 {

}
}
```

Ruby:

```
# @param {Integer} low_limit
# @param {Integer} high_limit
# @return {Integer}
def count_balls(low_limit, high_limit)

end
```

PHP:

```
class Solution {

/**
* @param Integer $lowLimit
* @param Integer $highLimit
* @return Integer
*/
function countBalls($lowLimit, $highLimit) {

}
}
```

Dart:

```
class Solution {
int countBalls(int lowLimit, int highLimit) {



}
}
```

Scala:

```
object Solution {
def countBalls(lowLimit: Int, highLimit: Int): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec count_balls(low_limit :: integer, high_limit :: integer) :: integer
def count_balls(low_limit, high_limit) do

end
end
```

Erlang:

```
-spec count_balls(LowLimit :: integer(), HighLimit :: integer()) ->
integer().
count_balls(LowLimit, HighLimit) ->
.
```

Racket:

```
(define/contract (count-balls lowLimit highLimit)
(-> exact-integer? exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Maximum Number of Balls in a Box
* Difficulty: Easy
* Tags: math, hash
*
* Approach: Use hash map for O(1) lookups
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(n) for hash map
*/

class Solution {
public:
int countBalls(int lowLimit, int highLimit) {

}
};
```

Java Solution:

```
/**
* Problem: Maximum Number of Balls in a Box
* Difficulty: Easy
* Tags: math, hash
*
* Approach: Use hash map for O(1) lookups
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(n) for hash map
*/

class Solution {
public int countBalls(int lowLimit, int highLimit) {

}
}
```

Python3 Solution:

```
"""
Problem: Maximum Number of Balls in a Box
Difficulty: Easy
Tags: math, hash

Approach: Use hash map for O(1) lookups
```

```
Time Complexity: O(n) to O(n^2) depending on approach
Space Complexity: O(n) for hash map
"""

class Solution:
def countBalls(self, lowLimit: int, highLimit: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def countBalls(self, lowLimit, highLimit):
"""
:type lowLimit: int
:type highLimit: int
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Maximum Number of Balls in a Box
* Difficulty: Easy
* Tags: math, hash
*
* Approach: Use hash map for O(1) lookups
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(n) for hash map
*/

/**
* @param {number} lowLimit
* @param {number} highLimit
* @return {number}
*/
var countBalls = function(lowLimit, highLimit) {

};
```

TypeScript Solution:

```
/**
* Problem: Maximum Number of Balls in a Box
* Difficulty: Easy
* Tags: math, hash
*
* Approach: Use hash map for O(1) lookups
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(n) for hash map
*/

function countBalls(lowLimit: number, highLimit: number): number {

};
```

C# Solution:

```
/*
* Problem: Maximum Number of Balls in a Box
* Difficulty: Easy
* Tags: math, hash
*
* Approach: Use hash map for O(1) lookups
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(n) for hash map
*/

public class Solution {
public int CountBalls(int lowLimit, int highLimit) {

}
}
```

C Solution:

```
/*
* Problem: Maximum Number of Balls in a Box
* Difficulty: Easy
* Tags: math, hash
*
* Approach: Use hash map for O(1) lookups
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(n) for hash map
```

```
*/

int countBalls(int lowLimit, int highLimit) {

}
```

Go Solution:

```
// Problem: Maximum Number of Balls in a Box
// Difficulty: Easy
// Tags: math, hash
//
// Approach: Use hash map for O(1) lookups
// Time Complexity: O(n) to O(n^2) depending on approach
// Space Complexity: O(n) for hash map

func countBalls(lowLimit int, highLimit int) int {

}
```

Kotlin Solution:

```
class Solution {
fun countBalls(lowLimit: Int, highLimit: Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func countBalls(_ lowLimit: Int, _ highLimit: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: Maximum Number of Balls in a Box
// Difficulty: Easy
// Tags: math, hash
```

```
//
// Approach: Use hash map for O(1) lookups
// Time Complexity: O(n) to O(n^2) depending on approach
// Space Complexity: O(n) for hash map

impl Solution {
pub fn count_balls(low_limit: i32, high_limit: i32) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer} low_limit
# @param {Integer} high_limit
# @return {Integer}
def count_balls(low_limit, high_limit)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $lowLimit
* @param Integer $highLimit
* @return Integer
*/
function countBalls($lowLimit, $highLimit) {

}
}
```

Dart Solution:

```
class Solution {
int countBalls(int lowLimit, int highLimit) {

}
}
```

Scala Solution:

```
object Solution {
def countBalls(lowLimit: Int, highLimit: Int): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec count_balls(low_limit :: integer, high_limit :: integer) :: integer
def count_balls(low_limit, high_limit) do

end
end
```

Erlang Solution:

```
-spec count_balls(LowLimit :: integer(), HighLimit :: integer()) ->
integer().
count_balls(LowLimit, HighLimit) ->
.
```

Racket Solution:

```
(define/contract (count-balls lowLimit highLimit)
(-> exact-integer? exact-integer? exact-integer?)
)
```