

Problem 2929: Distribute Candies Among Children II

Problem Information

Difficulty: **Medium**

Acceptance Rate: 0.00%

Paid Only: No

Problem Description

You are given two positive integers

n

and

limit

.

Return

the

total number

of ways to distribute

n

candies among

3

children such that no child gets more than

limit

candies.

Example 1:

Input:

$n = 5$, limit = 2

Output:

3

Explanation:

There are 3 ways to distribute 5 candies such that no child gets more than 2 candies: (1, 2, 2), (2, 1, 2) and (2, 2, 1).

Example 2:

Input:

$n = 3$, limit = 3

Output:

10

Explanation:

There are 10 ways to distribute 3 candies such that no child gets more than 3 candies: (0, 0, 3), (0, 1, 2), (0, 2, 1), (0, 3, 0), (1, 0, 2), (1, 1, 1), (1, 2, 0), (2, 0, 1), (2, 1, 0) and (3, 0, 0).

Constraints:

$1 \leq n \leq 10$

6

1 <= limit <= 10

6

Code Snippets

C++:

```
class Solution {  
public:  
    long long distributeCandies(int n, int limit) {  
  
    }  
};
```

Java:

```
class Solution {  
    public long distributeCandies(int n, int limit) {  
  
    }  
}
```

Python3:

```
class Solution:  
    def distributeCandies(self, n: int, limit: int) -> int:
```

Python:

```
class Solution(object):  
    def distributeCandies(self, n, limit):  
        """  
        :type n: int  
        :type limit: int  
        :rtype: int  
        """
```

JavaScript:

```

/**
 * @param {number} n
 * @param {number} limit
 * @return {number}
 */
var distributeCandies = function(n, limit) {

};

```

TypeScript:

```

function distributeCandies(n: number, limit: number): number {

};

```

C#:

```

public class Solution {
    public long DistributeCandies(int n, int limit) {

    }
}

```

C:

```

long long distributeCandies(int n, int limit) {

}

```

Go:

```

func distributeCandies(n int, limit int) int64 {

}

```

Kotlin:

```

class Solution {
    fun distributeCandies(n: Int, limit: Int): Long {

    }
}

```

Swift:

```
class Solution {  
    func distributeCandies(_ n: Int, _ limit: Int) -> Int {  
  
    }  
}
```

Rust:

```
impl Solution {  
    pub fn distribute_candies(n: i32, limit: i32) -> i64 {  
  
    }  
}
```

Ruby:

```
# @param {Integer} n  
# @param {Integer} limit  
# @return {Integer}  
def distribute_candies(n, limit)  
  
end
```

PHP:

```
class Solution {  
  
    /**  
     * @param Integer $n  
     * @param Integer $limit  
     * @return Integer  
     */  
    function distributeCandies($n, $limit) {  
  
    }  
}
```

Dart:

```
class Solution {  
    int distributeCandies(int n, int limit) {
```

```
}  
}
```

Scala:

```
object Solution {  
  def distributeCandies(n: Int, limit: Int): Long = {  
  
  }  
}
```

Elixir:

```
defmodule Solution do  
  @spec distribute_candies(n :: integer, limit :: integer) :: integer  
  def distribute_candies(n, limit) do  
  
  end  
end
```

Erlang:

```
-spec distribute_candies(N :: integer(), Limit :: integer()) -> integer().  
distribute_candies(N, Limit) ->  
.
```

Racket:

```
(define/contract (distribute-candies n limit)  
  (-> exact-integer? exact-integer? exact-integer?)  
)
```

Solutions

C++ Solution:

```
/*  
 * Problem: Distribute Candies Among Children II  
 * Difficulty: Medium
```

```

* Tags: math
*
* 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:
    long long distributeCandies(int n, int limit) {

    }
};

```

Java Solution:

```

/**
 * Problem: Distribute Candies Among Children II
 * Difficulty: Medium
 * Tags: math
 *
 * 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 long distributeCandies(int n, int limit) {

    }
}

```

Python3 Solution:

```

"""
Problem: Distribute Candies Among Children II
Difficulty: Medium
Tags: math

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 distributeCandies(self, n: int, limit: int) -> int:
        # TODO: Implement optimized solution
        pass

```

Python Solution:

```

class Solution(object):
    def distributeCandies(self, n, limit):
        """
        :type n: int
        :type limit: int
        :rtype: int
        """

```

JavaScript Solution:

```

/**
 * Problem: Distribute Candies Among Children II
 * Difficulty: Medium
 * Tags: math
 *
 * 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} limit
 * @return {number}
 */
var distributeCandies = function(n, limit) {

};

```

TypeScript Solution:


```

/**
 * Problem: Distribute Candies Among Children II
 * Difficulty: Medium
 * Tags: math
 *
 * 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 distributeCandies(n: number, limit: number): number {

};

```

C# Solution:

```

/*
 * Problem: Distribute Candies Among Children II
 * Difficulty: Medium
 * Tags: math
 *
 * 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 long DistributeCandies(int n, int limit) {

    }
}

```

C Solution:

```

/*
 * Problem: Distribute Candies Among Children II
 * Difficulty: Medium
 * Tags: math
 *
 * 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

```

```

*/

long long distributeCandies(int n, int limit) {

}

```

Go Solution:

```

// Problem: Distribute Candies Among Children II
// Difficulty: Medium
// Tags: math
//
// 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 distributeCandies(n int, limit int) int64 {

}

```

Kotlin Solution:

```

class Solution {
    fun distributeCandies(n: Int, limit: Int): Long {

    }
}

```

Swift Solution:

```

class Solution {
    func distributeCandies(_ n: Int, _ limit: Int) -> Int {

    }
}

```

Rust Solution:

```

// Problem: Distribute Candies Among Children II
// Difficulty: Medium
// Tags: math

```

```
//
// 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 distribute_candies(n: i32, limit: i32) -> i64 {

    }
}
```

Ruby Solution:

```
# @param {Integer} n
# @param {Integer} limit
# @return {Integer}
def distribute_candies(n, limit)

end
```

PHP Solution:

```
class Solution {

    /**
     * @param Integer $n
     * @param Integer $limit
     * @return Integer
     */
    function distributeCandies($n, $limit) {

    }

}
```

Dart Solution:

```
class Solution {
    int distributeCandies(int n, int limit) {

    }
}
```

Scala Solution:

```
object Solution {  
  def distributeCandies(n: Int, limit: Int): Long = {  
  
  }  
}
```

Elixir Solution:

```
defmodule Solution do  
  @spec distribute_candies(n :: integer, limit :: integer) :: integer  
  def distribute_candies(n, limit) do  
  
  end  
end
```

Erlang Solution:

```
-spec distribute_candies(N :: integer(), Limit :: integer()) -> integer().  
distribute_candies(N, Limit) ->  
.
```

Racket Solution:

```
(define/contract (distribute-candies n limit)  
  (-> exact-integer? exact-integer? exact-integer?)  
)
```