Problem 1739: Building Boxes

Problem Information

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

Problem Description

You have a cubic storeroom where the width, length, and height of the room are all equal to

n

units. You are asked to place

n

boxes in this room where each box is a cube of unit side length. There are however some rules to placing the boxes:

You can place the boxes anywhere on the floor.

If box

x

is placed on top of the box

y

, then each side of the four vertical sides of the box

y

must

either be adjacent to another box or to a wall.

Given an integer

n

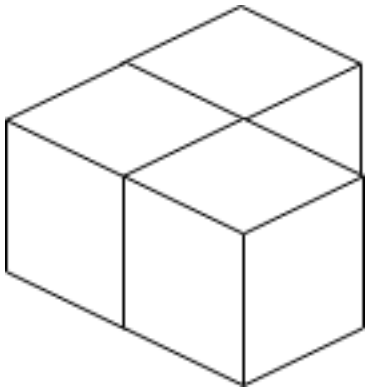
, return

the

minimum

possible number of boxes touching the floor.

Example 1:



Input:

n = 3

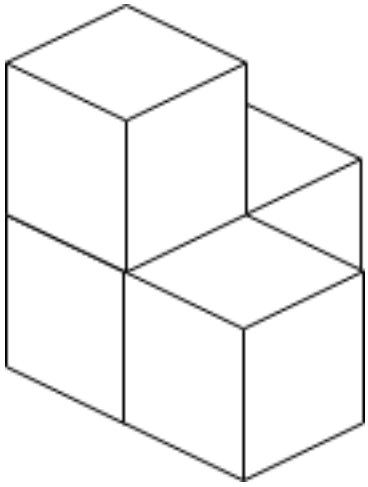
Output:

3

Explanation:

The figure above is for the placement of the three boxes. These boxes are placed in the corner of the room, where the corner is on the left side.

Example 2:



Input:

n = 4

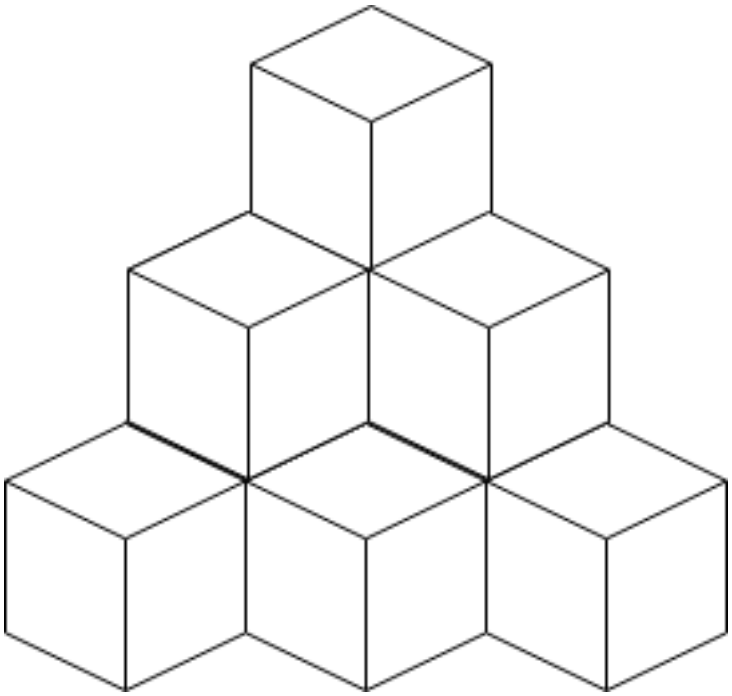
Output:

3

Explanation:

The figure above is for the placement of the four boxes. These boxes are placed in the corner of the room, where the corner is on the left side.

Example 3:



Input:

n = 10

Output:

6

Explanation:

The figure above is for the placement of the ten boxes. These boxes are placed in the corner of the room, where the corner is on the back side.

Constraints:

1 <= n <= 10

9

Code Snippets

C++:

```
class Solution {
public:
int minimumBoxes(int n) {

}
};
```

Java:

```
class Solution {
public int minimumBoxes(int n) {

}
}
```

Python3:

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

Python:

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

JavaScript:

```
/**
* @param {number} n
* @return {number}
*/
var minimumBoxes = function(n) {

};
```

TypeScript:

```
function minimumBoxes(n: number): number {
```

```
};
```

C#:

```
public class Solution {
public int MinimumBoxes(int n) {

}
}
```

C:

```
int minimumBoxes(int n) {

}
```

Go:

```
func minimumBoxes(n int) int {

}
```

Kotlin:

```
class Solution {
fun minimumBoxes(n: Int): Int {

}
}
```

Swift:

```
class Solution {
func minimumBoxes(_ n: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn minimum_boxes(n: i32) -> i32 {
```

```
}
}
```

Ruby:

```
# @param {Integer} n
# @return {Integer}
def minimum_boxes(n)

end
```

PHP:

```
class Solution {

/**
* @param Integer $n
* @return Integer
*/
function minimumBoxes($n) {

}
}
```

Dart:

```
class Solution {
int minimumBoxes(int n) {

}
}
```

Scala:

```
object Solution {
def minimumBoxes(n: Int): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec minimum_boxes(n :: integer) :: integer
def minimum_boxes(n) do

end
end
```

Erlang:

```
-spec minimum_boxes(N :: integer()) -> integer().
minimum_boxes(N) ->
.
```

Racket:

```
(define/contract (minimum-boxes n)
(-> exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Building Boxes
* Difficulty: Hard
* Tags: greedy, math, search
*
* Approach: Greedy algorithm with local optimal choices
* 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 minimumBoxes(int n) {

}
};
```

Java Solution:

```
/**
* Problem: Building Boxes
* Difficulty: Hard
* Tags: greedy, math, search
*
* Approach: Greedy algorithm with local optimal choices
* 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 minimumBoxes(int n) {

}
}
```

Python3 Solution:

```
"""
Problem: Building Boxes
Difficulty: Hard
Tags: greedy, math, search

Approach: Greedy algorithm with local optimal choices
Time Complexity: O(n) to O(n^2) depending on approach
Space Complexity: O(1) to O(n) depending on approach
"""

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Building Boxes
* Difficulty: Hard
* Tags: greedy, math, search
*
* Approach: Greedy algorithm with local optimal choices
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {number} n
* @return {number}
*/
var minimumBoxes = function(n) {

};
```

TypeScript Solution:

```
/**
* Problem: Building Boxes
* Difficulty: Hard
* Tags: greedy, math, search
*
* Approach: Greedy algorithm with local optimal choices
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

function minimumBoxes(n: number): number {

};
```

C# Solution:

```
/*
* Problem: Building Boxes
* Difficulty: Hard
* Tags: greedy, math, search
*
```

```
* Approach: Greedy algorithm with local optimal choices
* 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 MinimumBoxes(int n) {

}
}
```

C Solution:

```
/*
* Problem: Building Boxes
* Difficulty: Hard
* Tags: greedy, math, search
*
* Approach: Greedy algorithm with local optimal choices
* Time Complexity: O(n) to O(n^2) depending on approach
* Space Complexity: O(1) to O(n) depending on approach
*/

int minimumBoxes(int n) {

}
```

Go Solution:

```
// Problem: Building Boxes
// Difficulty: Hard
// Tags: greedy, math, search
//
// Approach: Greedy algorithm with local optimal choices
// Time Complexity: O(n) to O(n^2) depending on approach
// Space Complexity: O(1) to O(n) depending on approach

func minimumBoxes(n int) int {

}
```

Kotlin Solution:

```
class Solution {
fun minimumBoxes(n: Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func minimumBoxes(_ n: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: Building Boxes
// Difficulty: Hard
// Tags: greedy, math, search
//
// Approach: Greedy algorithm with local optimal choices
// 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 minimum_boxes(n: i32) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer} n
# @return {Integer}
def minimum_boxes(n)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $n
* @return Integer
*/
function minimumBoxes($n) {

}
}
```

Dart Solution:

```
class Solution {
int minimumBoxes(int n) {

}
}
```

Scala Solution:

```
object Solution {
def minimumBoxes(n: Int): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec minimum_boxes(n :: integer) :: integer
def minimum_boxes(n) do

end
end
```

Erlang Solution:

```
-spec minimum_boxes(N :: integer()) -> integer().
minimum_boxes(N) ->
.
```

Racket Solution:

```
(define/contract (minimum-boxes n)
(-> exact-integer? exact-integer?)
)
```