Problem 2139: Minimum Moves to Reach Target Score

Problem Information

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

Problem Description

You are playing a game with integers. You start with the integer

1

and you want to reach the integer

target

.

In one move, you can either:

Increment

the current integer by one (i.e.,

x = x + 1

).

Double

the current integer (i.e.,

x = 2 * x

).

You can use the

increment

operation

any

number of times, however, you can only use the

double

operation

at most

maxDoubles

times.

Given the two integers

target

and

maxDoubles

, return

the minimum number of moves needed to reach

target

starting with

1

.

Example 1:

Input:

target = 5, maxDoubles = 0

Output:

4

Explanation:

Keep incrementing by 1 until you reach target.

Example 2:

Input:

target = 19, maxDoubles = 2

Output:

7

Explanation:

Initially, x = 1 Increment 3 times so x = 4 Double once so x = 8 Increment once so x = 9 Double again so x = 18 Increment once so x = 19

Example 3:

Input:

target = 10, maxDoubles = 4

Output:

4

Explanation:

Initially, x = 1 Increment once so x = 2 Double once so x = 4 Increment once so x = 5 Double again so x = 10

Constraints:

1 <= target <= 10

9

0 <= maxDoubles <= 100

Code Snippets

C++:

```
class Solution {
public:
int minMoves(int target, int maxDoubles) {

}
};
```

Java:

```
class Solution {
public int minMoves(int target, int maxDoubles) {

}
}
```

Python3:

```
class Solution:
def minMoves(self, target: int, maxDoubles: int) -> int:
```

Python:

```
class Solution(object):
def minMoves(self, target, maxDoubles):
"""
:type target: int
:type maxDoubles: int
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number} target
* @param {number} maxDoubles
* @return {number}
*/
var minMoves = function(target, maxDoubles) {

};
```

TypeScript:

```
function minMoves(target: number, maxDoubles: number): number {

};
```

C#:

```
public class Solution {
public int MinMoves(int target, int maxDoubles) {

}
}
```

C:

```
int minMoves(int target, int maxDoubles) {

}
```

Go:

```
func minMoves(target int, maxDoubles int) int {

}
```

Kotlin:

```
class Solution {
fun minMoves(target: Int, maxDoubles: Int): Int {

}
}
```

Swift:

```
class Solution {
func minMoves(_ target: Int, _ maxDoubles: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn min_moves(target: i32, max_doubles: i32) -> i32 {

}
}
```

Ruby:

```
# @param {Integer} target
# @param {Integer} max_doubles
# @return {Integer}
def min_moves(target, max_doubles)

end
```

PHP:

```
class Solution {

/**
* @param Integer $target
* @param Integer $maxDoubles
* @return Integer
*/
function minMoves($target, $maxDoubles) {

}
}
```

Dart:

```
class Solution {
int minMoves(int target, int maxDoubles) {

}
}
```

Scala:

```
object Solution {
def minMoves(target: Int, maxDoubles: Int): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec min_moves(target :: integer, max_doubles :: integer) :: integer
def min_moves(target, max_doubles) do

end
end
```

Erlang:

```
-spec min_moves(Target :: integer(), MaxDoubles :: integer()) -> integer().
min_moves(Target, MaxDoubles) ->
.
```

Racket:

```
(define/contract (min-moves target maxDoubles)
(-> exact-integer? exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Minimum Moves to Reach Target Score
* Difficulty: Medium
* Tags: greedy, math
*
* 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 minMoves(int target, int maxDoubles) {

}
};
```

Java Solution:

```
/**
* Problem: Minimum Moves to Reach Target Score
* Difficulty: Medium
* Tags: greedy, math
*
* 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 minMoves(int target, int maxDoubles) {

}
}
```

Python3 Solution:

```
"""
Problem: Minimum Moves to Reach Target Score
Difficulty: Medium
Tags: greedy, math

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 minMoves(self, target: int, maxDoubles: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def minMoves(self, target, maxDoubles):
"""
:type target: int
:type maxDoubles: int
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Minimum Moves to Reach Target Score
* Difficulty: Medium
* Tags: greedy, math
*
* 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} target
* @param {number} maxDoubles
* @return {number}
*/
var minMoves = function(target, maxDoubles) {

};
```

TypeScript Solution:

```
/**
* Problem: Minimum Moves to Reach Target Score
* Difficulty: Medium
* Tags: greedy, math
*
* 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 minMoves(target: number, maxDoubles: number): number {

};
```

C# Solution:

```
/*
* Problem: Minimum Moves to Reach Target Score
* Difficulty: Medium
* Tags: greedy, math
*
* 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 MinMoves(int target, int maxDoubles) {

}

}
```

C Solution:

```
/*
* Problem: Minimum Moves to Reach Target Score
* Difficulty: Medium
* Tags: greedy, math
*
* 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 minMoves(int target, int maxDoubles) {

}
```

Go Solution:

```
// Problem: Minimum Moves to Reach Target Score
// Difficulty: Medium
// Tags: greedy, math
//
// 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 minMoves(target int, maxDoubles int) int {

}
```

Kotlin Solution:

```
class Solution {
fun minMoves(target: Int, maxDoubles: Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func minMoves(_ target: Int, _ maxDoubles: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: Minimum Moves to Reach Target Score
// Difficulty: Medium
// Tags: greedy, math
//
// 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 min_moves(target: i32, max_doubles: i32) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer} target
# @param {Integer} max_doubles
# @return {Integer}
def min_moves(target, max_doubles)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $target
* @param Integer $maxDoubles
* @return Integer
*/
function minMoves($target, $maxDoubles) {

}
}
```

Dart Solution:

```
class Solution {
int minMoves(int target, int maxDoubles) {

}
}
```

Scala Solution:

```
object Solution {
def minMoves(target: Int, maxDoubles: Int): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec min_moves(target :: integer, max_doubles :: integer) :: integer
def min_moves(target, max_doubles) do

end
end
```

Erlang Solution:

```
-spec min_moves(Target :: integer(), MaxDoubles :: integer()) -> integer().
min_moves(Target, MaxDoubles) ->
.
```

Racket Solution:

```
(define/contract (min-moves target maxDoubles)
(-> exact-integer? exact-integer? exact-integer?)
)
```