Problem 2543: Check if Point Is Reachable

Problem Information

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

Problem Description

There exists an infinitely large grid. You are currently at point

(1, 1)

, and you need to reach the point

(targetX, targetY)

using a finite number of steps.

In one

step

, you can move from point

(x, y)

to any one of the following points:

(x, y - x)

(x - y, y)

(2 * x, y)

(x, 2 * y)

Given two integers

targetX

and

targetY

representing the X-coordinate and Y-coordinate of your final position, return

true

if you can reach the point from

(1, 1)

using some number of steps, and

false

otherwise

.

Example 1:

Input:

targetX = 6, targetY = 9

Output:

false

Explanation:

It is impossible to reach (6,9) from (1,1) using any sequence of moves, so false is returned.

Example 2:

Input:

targetX = 4, targetY = 7

Output:

true

Explanation:

You can follow the path (1,1) -> (1,2) -> (1,4) -> (1,8) -> (1,7) -> (2,7) -> (4,7).

Constraints:

1 <= targetX, targetY <= 10

9

Code Snippets

C++:

```
class Solution {
public:
bool isReachable(int targetX, int targetY) {

}
};
```

Java:

```
class Solution {
public boolean isReachable(int targetX, int targetY) {

}
}
```

Python3:

```
class Solution:
def isReachable(self, targetX: int, targetY: int) -> bool:
```

Python:

```
class Solution(object):
def isReachable(self, targetX, targetY):
"""
:type targetX: int
:type targetY: int
:rtype: bool
"""
```

JavaScript:

```
/**
* @param {number} targetX
* @param {number} targetY
* @return {boolean}
*/
var isReachable = function(targetX, targetY) {

};
```

TypeScript:

```
function isReachable(targetX: number, targetY: number): boolean {

};
```

C#:

```
public class Solution {
public bool IsReachable(int targetX, int targetY) {

}
}
```

C:

```
bool isReachable(int targetX, int targetY) {

}
```

Go:

```
func isReachable(targetX int, targetY int) bool {

}
```

Kotlin:

```
class Solution {
fun isReachable(targetX: Int, targetY: Int): Boolean {

}
}
```

Swift:

```
class Solution {
func isReachable(_ targetX: Int, _ targetY: Int) -> Bool {

}
}
```

Rust:

```
impl Solution {
pub fn is_reachable(target_x: i32, target_y: i32) -> bool {

}
}
```

Ruby:

```
# @param {Integer} target_x
# @param {Integer} target_y
# @return {Boolean}
def is_reachable(target_x, target_y)

end
```

PHP:

```
class Solution {

/**
* @param Integer $targetX
* @param Integer $targetY
* @return Boolean
*/
function isReachable($targetX, $targetY) {

}
}
```

Dart:

```
class Solution {
bool isReachable(int targetX, int targetY) {

}
}
```

Scala:

```
object Solution {
def isReachable(targetX: Int, targetY: Int): Boolean = {

}
}
```

Elixir:

```
defmodule Solution do
@spec is_reachable(target_x :: integer, target_y :: integer) :: boolean
def is_reachable(target_x, target_y) do

end
end
```

Erlang:

```
-spec is_reachable(TargetX :: integer(), TargetY :: integer()) -> boolean().
is_reachable(TargetX, TargetY) ->
```

```
.
```

Racket:

```
(define/contract (is-reachable targetX targetY)
(-> exact-integer? exact-integer? boolean?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Check if Point Is Reachable
* Difficulty: Hard
* 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:
bool isReachable(int targetX, int targetY) {

}
};
```

Java Solution:

```
/**
* Problem: Check if Point Is Reachable
* Difficulty: Hard
* 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 boolean isReachable(int targetX, int targetY) {

}
}
```

Python3 Solution:

```
"""
Problem: Check if Point Is Reachable
Difficulty: Hard
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 isReachable(self, targetX: int, targetY: int) -> bool:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def isReachable(self, targetX, targetY):
"""
:type targetX: int
:type targetY: int
:rtype: bool
"""
```

JavaScript Solution:

```
/**
* Problem: Check if Point Is Reachable
* Difficulty: Hard
* 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} targetX
* @param {number} targetY
* @return {boolean}
*/
var isReachable = function(targetX, targetY) {

};
```

TypeScript Solution:

```
/**
* Problem: Check if Point Is Reachable
* Difficulty: Hard
* 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 isReachable(targetX: number, targetY: number): boolean {

};
```

C# Solution:

```
/*
* Problem: Check if Point Is Reachable
* Difficulty: Hard
* 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 bool IsReachable(int targetX, int targetY) {

}
}
```

C Solution:

```
/*
* Problem: Check if Point Is Reachable
* Difficulty: Hard
* 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
*/

bool isReachable(int targetX, int targetY) {

}
```

Go Solution:

```
// Problem: Check if Point Is Reachable
// Difficulty: Hard
// 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 isReachable(targetX int, targetY int) bool {

}
```

Kotlin Solution:

```
class Solution {
fun isReachable(targetX: Int, targetY: Int): Boolean {
}
}
```

Swift Solution:

```
class Solution {
func isReachable(_ targetX: Int, _ targetY: Int) -> Bool {

}
}
```

Rust Solution:

```
// Problem: Check if Point Is Reachable
// Difficulty: Hard
// 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 is_reachable(target_x: i32, target_y: i32) -> bool {

}
}
```

Ruby Solution:

```
# @param {Integer} target_x
# @param {Integer} target_y
# @return {Boolean}
def is_reachable(target_x, target_y)

end
```

PHP Solution:

```
class Solution {
```

```
/**
* @param Integer $targetX
* @param Integer $targetY
* @return Boolean
*/
function isReachable($targetX, $targetY) {

}
}
```

Dart Solution:

```
class Solution {
bool isReachable(int targetX, int targetY) {

}
}
```

Scala Solution:

```
object Solution {
def isReachable(targetX: Int, targetY: Int): Boolean = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec is_reachable(target_x :: integer, target_y :: integer) :: boolean
def is_reachable(target_x, target_y) do

end
end
```

Erlang Solution:

```
-spec is_reachable(TargetX :: integer(), TargetY :: integer()) -> boolean().
is_reachable(TargetX, TargetY) ->
.
```

Racket Solution:

```
(define/contract (is-reachable targetX targetY)
(-> exact-integer? exact-integer? boolean?)
)
```