Problem 754: Reach a Number

Problem Information

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

Problem Description

You are standing at position

0

on an infinite number line. There is a destination at position

target

.

You can make some number of moves

numMoves

so that:

On each move, you can either go left or right.

During the

i

th

move (starting from

i == 1

to

i == numMoves

), you take

i

steps in the chosen direction.

Given the integer

target

, return

the

minimum

number of moves required (i.e., the minimum

numMoves

) to reach the destination

.

Example 1:

Input:

target = 2

Output:

3

Explanation:

On the 1

st

move, we step from 0 to 1 (1 step). On the 2

nd

move, we step from 1 to -1 (2 steps). On the 3

rd

move, we step from -1 to 2 (3 steps).

Example 2:

Input:

target = 3

Output:

2

Explanation:

On the 1

st

move, we step from 0 to 1 (1 step). On the 2

nd

move, we step from 1 to 3 (2 steps).

Constraints:

-10

9

<= target <= 10

9

target != 0

Code Snippets

C++:

```
class Solution {
public:
int reachNumber(int target) {

}
};
```

Java:

```
class Solution {
public int reachNumber(int target) {

}
}
```

Python3:

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

Python:

```
class Solution(object):
def reachNumber(self, target):
```

```
"""
:type target: int
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number} target
* @return {number}
*/
var reachNumber = function(target) {

};
```

TypeScript:

```
function reachNumber(target: number): number {

};
```

C#:

```
public class Solution {
public int ReachNumber(int target) {

}
}
```

C:

```
int reachNumber(int target) {

}
```

Go:

```
func reachNumber(target int) int {

}
```

Kotlin:

```
class Solution {
fun reachNumber(target: Int): Int {

}
}
```

Swift:

```
class Solution {
func reachNumber(_ target: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn reach_number(target: i32) -> i32 {

}
}
```

Ruby:

```
# @param {Integer} target
# @return {Integer}
def reach_number(target)

end
```

PHP:

```
class Solution {

/**
* @param Integer $target
* @return Integer
*/
function reachNumber($target) {

}
}
```

Dart:

```
class Solution {
int reachNumber(int target) {

}
}
```

Scala:

```
object Solution {
def reachNumber(target: Int): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec reach_number(target :: integer) :: integer
def reach_number(target) do

end
end
```

Erlang:

```
-spec reach_number(Target :: integer()) -> integer().
reach_number(Target) ->
.
```

Racket:

```
(define/contract (reach-number target)
(-> exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Reach a Number
* Difficulty: Medium
* Tags: math, search
*
* 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:
int reachNumber(int target) {

}
};
```

Java Solution:

```
/**
* Problem: Reach a Number
* Difficulty: Medium
* Tags: math, search
*
* 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 int reachNumber(int target) {

}
}
```

Python3 Solution:

```
"""
Problem: Reach a Number
Difficulty: Medium
Tags: math, search
```

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

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Reach a Number
* Difficulty: Medium
* Tags: math, search
*
* 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} target
* @return {number}
*/
var reachNumber = function(target) {

};
```

TypeScript Solution:

```
/**
* Problem: Reach a Number
* Difficulty: Medium
* Tags: math, search
*
* 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 reachNumber(target: number): number {

};
```

C# Solution:

```
/*
* Problem: Reach a Number
* Difficulty: Medium
* Tags: math, search
*
* 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 int ReachNumber(int target) {

}
}
```

C Solution:

```
/*
* Problem: Reach a Number
* Difficulty: Medium
* Tags: math, search
*
* 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
```

```
*/

int reachNumber(int target) {

}
```

Go Solution:

```
// Problem: Reach a Number
// Difficulty: Medium
// Tags: math, search
//
// 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 reachNumber(target int) int {

}
```

Kotlin Solution:

```
class Solution {
fun reachNumber(target: Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func reachNumber(_ target: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: Reach a Number
// Difficulty: Medium
// Tags: math, search
```

```
//
// 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 reach_number(target: i32) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer} target
# @return {Integer}
def reach_number(target)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $target
* @return Integer
*/
function reachNumber($target) {

}
}
```

Dart Solution:

```
class Solution {
int reachNumber(int target) {

}
}
```

Scala Solution:

```
object Solution {
def reachNumber(target: Int): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec reach_number(target :: integer) :: integer
def reach_number(target) do

end
end
```

Erlang Solution:

```
-spec reach_number(Target :: integer()) -> integer().
reach_number(Target) ->
.
```

Racket Solution:

```
(define/contract (reach-number target)
(-> exact-integer? exact-integer?)
)
```