Problem 1197: Minimum Knight Moves

Problem Information

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

Problem Description

In an

infinite

chess board with coordinates from

-infinity

to

+infinity

, you have a

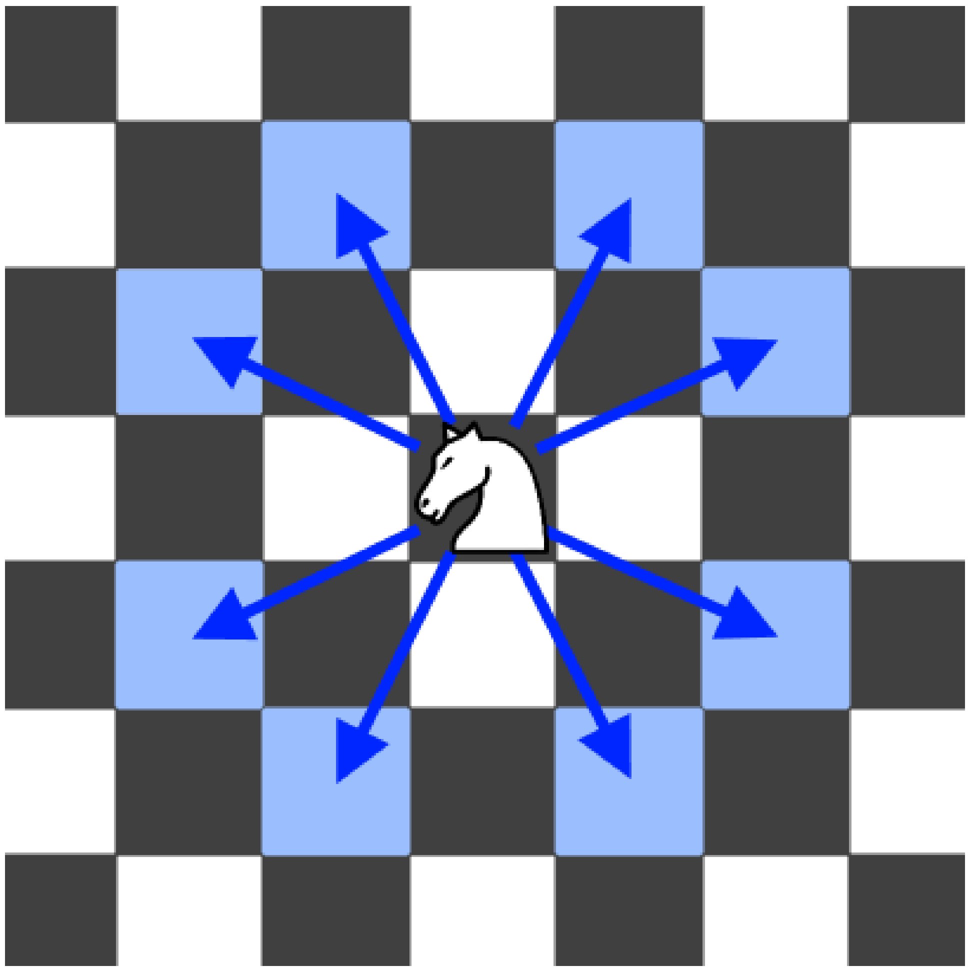
knight

at square

[0, 0]

.

A knight has 8 possible moves it can make, as illustrated below. Each move is two squares in a cardinal direction, then one square in an orthogonal direction.



Return

the minimum number of steps needed to move the knight to the square

[x, y]

. It is guaranteed the answer exists.

Example 1:

Input:

x = 2, y = 1

Output:

1

Explanation:

[0, 0] → [2, 1]

Example 2:

Input:

x = 5, y = 5

Output:

4

Explanation:

[0, 0] → [2, 1] → [4, 2] → [3, 4] → [5, 5]

Constraints:

-300 <= x, y <= 300

0 <= |x| + |y| <= 300

Code Snippets

C++:

```
class Solution {
public:
int minKnightMoves(int x, int y) {

}
};
```

Java:

```
class Solution {
public int minKnightMoves(int x, int y) {



}
}
```

Python3:

```
class Solution:
def minKnightMoves(self, x: int, y: int) -> int:
```

Python:

```
class Solution(object):
def minKnightMoves(self, x, y):
"""
:type x: int
:type y: int
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number} x
* @param {number} y
* @return {number}
*/
var minKnightMoves = function(x, y) {

};
```

TypeScript:

```
function minKnightMoves(x: number, y: number): number {

};
```

C#:

```
public class Solution {
public int MinKnightMoves(int x, int y) {

}
}
```

C:

```
int minKnightMoves(int x, int y) {

}
```

Go:

```
func minKnightMoves(x int, y int) int {

}
```

Kotlin:

```
class Solution {
fun minKnightMoves(x: Int, y: Int): Int {

}
}
```

Swift:

```
class Solution {
func minKnightMoves(_ x: Int, _ y: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn min_knight_moves(x: i32, y: i32) -> i32 {

}
}
```

Ruby:

```
# @param {Integer} x
# @param {Integer} y
# @return {Integer}
def min_knight_moves(x, y)

end
```

PHP:

```
class Solution {

/**
* @param Integer $x
* @param Integer $y
* @return Integer
*/
function minKnightMoves($x, $y) {

}
}
```

Dart:

```
class Solution {
int minKnightMoves(int x, int y) {

}
}
```

Scala:

```
object Solution {
def minKnightMoves(x: Int, y: Int): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec min_knight_moves(x :: integer, y :: integer) :: integer
def min_knight_moves(x, y) do

end
```

```
end
```

Erlang:

```
-spec min_knight_moves(X :: integer(), Y :: integer()) -> integer().
min_knight_moves(X, Y) ->
.
```

Racket:

```
(define/contract (min-knight-moves x y)
(-> exact-integer? exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Minimum Knight Moves
* Difficulty: Medium
* Tags: 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 minKnightMoves(int x, int y) {

}
};
```

Java Solution:

```
/**
* Problem: Minimum Knight Moves
* Difficulty: Medium
* Tags: 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 minKnightMoves(int x, int y) {

}
}
```

Python3 Solution:

```
"""
Problem: Minimum Knight Moves
Difficulty: Medium
Tags: 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 minKnightMoves(self, x: int, y: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def minKnightMoves(self, x, y):
"""
:type x: int
:type y: int
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Minimum Knight Moves
* Difficulty: Medium
* Tags: 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} x
* @param {number} y
* @return {number}
*/
var minKnightMoves = function(x, y) {

};
```

TypeScript Solution:

```
/**
* Problem: Minimum Knight Moves
* Difficulty: Medium
* Tags: 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 minKnightMoves(x: number, y: number): number {

};
```

C# Solution:

```
/*
* Problem: Minimum Knight Moves
* Difficulty: Medium
* Tags: 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 MinKnightMoves(int x, int y) {

}
}
```

C Solution:

```
/*
* Problem: Minimum Knight Moves
* Difficulty: Medium
* Tags: 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 minKnightMoves(int x, int y) {

}
```

Go Solution:

```
// Problem: Minimum Knight Moves
// Difficulty: Medium
// Tags: 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 minKnightMoves(x int, y int) int {

}
```

Kotlin Solution:

```
class Solution {
fun minKnightMoves(x: Int, y: Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func minKnightMoves(_ x: Int, _ y: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: Minimum Knight Moves
// Difficulty: Medium
// Tags: 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 min_knight_moves(x: i32, y: i32) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer} x
# @param {Integer} y
# @return {Integer}
def min_knight_moves(x, y)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $x
* @param Integer $y
* @return Integer
*/
function minKnightMoves($x, $y) {

}
}
```

Dart Solution:

```
class Solution {
int minKnightMoves(int x, int y) {

}
}
```

Scala Solution:

```
object Solution {
def minKnightMoves(x: Int, y: Int): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec min_knight_moves(x :: integer, y :: integer) :: integer
def min_knight_moves(x, y) do

end
end
```

Erlang Solution:

```
-spec min_knight_moves(X :: integer(), Y :: integer()) -> integer().
min_knight_moves(X, Y) ->
.
```

Racket Solution:

```
(define/contract (min-knight-moves x y)
(-> exact-integer? exact-integer? exact-integer?)
)
```