Problem 2769: Find the Maximum Achievable Number

Problem Information

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

Problem Description

Given two integers,

num

and

t

. A

number

x

is

achievable

if it can become equal to

num

after applying the following operation

at most

t

times:

Increase or decrease

x

by

1

, and

simultaneously

increase or decrease

num

by

1

.

Return the

maximum

possible value of

x

.

Example 1:

Input:

num = 4, t = 1

Output:

6

Explanation:

Apply the following operation once to make the maximum achievable number equal to

num

:

Decrease the maximum achievable number by 1, and increase

num

by 1.

Example 2:

Input:

num = 3, t = 2

Output:

7

Explanation:

Apply the following operation twice to make the maximum achievable number equal to

num

:

Decrease the maximum achievable number by 1, and increase

num

by 1.

Constraints:

1 <= num, t <= 50

Code Snippets

C++:

```
class Solution {
public:
int theMaximumAchievableX(int num, int t) {

}
};
```

Java:

```
class Solution {
public int theMaximumAchievableX(int num, int t) {

}
}
```

Python3:

```
class Solution:
def theMaximumAchievableX(self, num: int, t: int) -> int:
```

Python:

```
class Solution(object):
def theMaximumAchievableX(self, num, t):
"""
:type num: int
```

```
:type t: int
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number} num
* @param {number} t
* @return {number}
*/
var theMaximumAchievableX = function(num, t) {

};
```

TypeScript:

```
function theMaximumAchievableX(num: number, t: number): number {

};
```

C#:

```
public class Solution {
public int TheMaximumAchievableX(int num, int t) {

}
}
```

C:

```
int theMaximumAchievableX(int num, int t) {

}
```

Go:

```
func theMaximumAchievableX(num int, t int) int {

}
```

Kotlin:

```
class Solution {
fun theMaximumAchievableX(num: Int, t: Int): Int {

}
}
```

Swift:

```
class Solution {
func theMaximumAchievableX(_ num: Int, _ t: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn the_maximum_achievable_x(num: i32, t: i32) -> i32 {

}
}
```

Ruby:

```
# @param {Integer} num
# @param {Integer} t
# @return {Integer}
def the_maximum_achievable_x(num, t)

end
```

PHP:

```
class Solution {

/**
* @param Integer $num
* @param Integer $t
* @return Integer
*/
function theMaximumAchievableX($num, $t) {

}

}
```

Dart:

```
class Solution {
int theMaximumAchievableX(int num, int t) {

}
}
```

Scala:

```
object Solution {
def theMaximumAchievableX(num: Int, t: Int): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec the_maximum_achievable_x(num :: integer, t :: integer) :: integer
def the_maximum_achievable_x(num, t) do

end
end
```

Erlang:

```
-spec the_maximum_achievable_x(Num :: integer(), T :: integer()) ->
integer().
the_maximum_achievable_x(Num, T) ->
.
```

Racket:

```
(define/contract (the-maximum-achievable-x num t)
(-> exact-integer? exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Find the Maximum Achievable Number
* Difficulty: Easy
* 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:
int theMaximumAchievableX(int num, int t) {

}
};
```

Java Solution:

```
/**
* Problem: Find the Maximum Achievable Number
* Difficulty: Easy
* 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 int theMaximumAchievableX(int num, int t) {

}
}
```

Python3 Solution:

```
"""
Problem: Find the Maximum Achievable Number
```

```
Difficulty: Easy
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 theMaximumAchievableX(self, num: int, t: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def theMaximumAchievableX(self, num, t):
"""
:type num: int
:type t: int
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Find the Maximum Achievable Number
* Difficulty: Easy
* 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} num
* @param {number} t
* @return {number}
*/
var theMaximumAchievableX = function(num, t) {
};
```

TypeScript Solution:

```
/**
* Problem: Find the Maximum Achievable Number
* Difficulty: Easy
* 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 theMaximumAchievableX(num: number, t: number): number {

};
```

C# Solution:

```
/*
* Problem: Find the Maximum Achievable Number
* Difficulty: Easy
* 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 int TheMaximumAchievableX(int num, int t) {

}
}
```

C Solution:

```
/*
* Problem: Find the Maximum Achievable Number
* Difficulty: Easy
* 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
*/

int theMaximumAchievableX(int num, int t) {

}
```

Go Solution:

```
// Problem: Find the Maximum Achievable Number
// Difficulty: Easy
// 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 theMaximumAchievableX(num int, t int) int {

}
```

Kotlin Solution:

```
class Solution {
fun theMaximumAchievableX(num: Int, t: Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func theMaximumAchievableX(_ num: Int, _ t: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: Find the Maximum Achievable Number
// Difficulty: Easy
// 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 the_maximum_achievable_x(num: i32, t: i32) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer} num
# @param {Integer} t
# @return {Integer}
def the_maximum_achievable_x(num, t)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $num
* @param Integer $t
* @return Integer
*/
function theMaximumAchievableX($num, $t) {

}
}
```

Dart Solution:

```
class Solution {
int theMaximumAchievableX(int num, int t) {

}
}
```

Scala Solution:

```
object Solution {
def theMaximumAchievableX(num: Int, t: Int): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec the_maximum_achievable_x(num :: integer, t :: integer) :: integer
def the_maximum_achievable_x(num, t) do

end
end
```

Erlang Solution:

```
-spec the_maximum_achievable_x(Num :: integer(), T :: integer()) ->
integer().
the_maximum_achievable_x(Num, T) ->
.
```

Racket Solution:

```
(define/contract (the-maximum-achievable-x num t)
(-> exact-integer? exact-integer? exact-integer?)
)
```