Problem 2566: Maximum Difference by Remapping a Digit

Problem Information

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

Problem Description

You are given an integer

num

. You know that Bob will sneakily

remap

one of the

10

possible digits (

0

to

9

) to another digit.

Return

the difference between the maximum and minimum values Bob can make by remapping

exactly

one

digit in

num

.

Notes:

When Bob remaps a digit

d1

to another digit

d2

, Bob replaces all occurrences of

d1

in

num

with

d2

.

Bob can remap a digit to itself, in which case

num

does not change.

Bob can remap different digits for obtaining minimum and maximum values respectively.

The resulting number after remapping can contain leading zeroes.

Example 1:

Input:

num = 11891

Output:

99009

Explanation:

To achieve the maximum value, Bob can remap the digit 1 to the digit 9 to yield 99899. To achieve the minimum value, Bob can remap the digit 1 to the digit 0, yielding 890. The difference between these two numbers is 99009.

Example 2:

Input:

num = 90

Output:

99

Explanation:

The maximum value that can be returned by the function is 99 (if 0 is replaced by 9) and the minimum value that can be returned by the function is 0 (if 9 is replaced by 0). Thus, we return 99.

Constraints:

1 <= num <= 10

8

Code Snippets

C++:

```
class Solution {
public:
int minMaxDifference(int num) {

}
};
```

Java:

```
class Solution {
public int minMaxDifference(int num) {

}
}
```

Python3:

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

Python:

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

JavaScript:

```
/**
* @param {number} num
```

```
* @return {number}
*/
var minMaxDifference = function(num) {

};
```

TypeScript:

```
function minMaxDifference(num: number): number {

};
```

C#:

```
public class Solution {
public int MinMaxDifference(int num) {

}
}
```

C:

```
int minMaxDifference(int num) {

}
```

Go:

```
func minMaxDifference(num int) int {

}
```

Kotlin:

```
class Solution {
fun minMaxDifference(num: Int): Int {

}
}
```

Swift:

```
class Solution {
func minMaxDifference(_ num: Int) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn min_max_difference(num: i32) -> i32 {

}
}
```

Ruby:

```
# @param {Integer} num
# @return {Integer}
def min_max_difference(num)

end
```

PHP:

```
class Solution {

/**
* @param Integer $num
* @return Integer
*/
function minMaxDifference($num) {

}
}
```

Dart:

```
class Solution {
int minMaxDifference(int num) {

}
}
```

Scala:

```
object Solution {
def minMaxDifference(num: Int): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec min_max_difference(num :: integer) :: integer
def min_max_difference(num) do

end
end
```

Erlang:

```
-spec min_max_difference(Num :: integer()) -> integer().
min_max_difference(Num) ->
.
```

Racket:

```
(define/contract (min-max-difference num)
(-> exact-integer? exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Maximum Difference by Remapping a Digit
* Difficulty: Easy
* 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 minMaxDifference(int num) {

}
};
```

Java Solution:

```
/**
* Problem: Maximum Difference by Remapping a Digit
* Difficulty: Easy
* 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 minMaxDifference(int num) {

}
}
```

Python3 Solution:

```
"""
Problem: Maximum Difference by Remapping a Digit
Difficulty: Easy
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 minMaxDifference(self, num: int) -> int:
# TODO: Implement optimized solution
```

```
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Maximum Difference by Remapping a Digit
* Difficulty: Easy
* 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} num
* @return {number}
*/
var minMaxDifference = function(num) {

};
```

TypeScript Solution:

```
/**
* Problem: Maximum Difference by Remapping a Digit
* Difficulty: Easy
* 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 minMaxDifference(num: number): number {

};
```

C# Solution:

```
/*
* Problem: Maximum Difference by Remapping a Digit
* Difficulty: Easy
* 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 MinMaxDifference(int num) {

}
}
```

C Solution:

```
/*
* Problem: Maximum Difference by Remapping a Digit
* Difficulty: Easy
* 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 minMaxDifference(int num) {

}
```

Go Solution:

```
// Problem: Maximum Difference by Remapping a Digit
// Difficulty: Easy
// 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 minMaxDifference(num int) int {

}
```

Kotlin Solution:

```
class Solution {
fun minMaxDifference(num: Int): Int {

}
}
```

Swift Solution:

```
class Solution {
func minMaxDifference(_ num: Int) -> Int {

}
}
```

Rust Solution:

```
// Problem: Maximum Difference by Remapping a Digit
// Difficulty: Easy
// 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_max_difference(num: i32) -> i32 {

}
```

```
}
```

Ruby Solution:

```
# @param {Integer} num
# @return {Integer}
def min_max_difference(num)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer $num
* @return Integer
*/
function minMaxDifference($num) {

}
}
```

Dart Solution:

```
class Solution {
int minMaxDifference(int num) {

}
}
```

Scala Solution:

```
object Solution {
def minMaxDifference(num: Int): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec min_max_difference(num :: integer) :: integer
def min_max_difference(num) do

end
end
```

Erlang Solution:

```
-spec min_max_difference(Num :: integer()) -> integer().
min_max_difference(Num) ->
.
```

Racket Solution:

```
(define/contract (min-max-difference num)
(-> exact-integer? exact-integer?)
)
```