Problem 1432: Max Difference You Can Get From Changing an Integer

Problem Information

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

Problem Description

You are given an integer

num

. You will apply the following steps to

num

two

separate times:

Pick a digit

x (0 <= x <= 9)

.

Pick another digit

y (0 <= y <= 9)

. Note

y

can be equal to

x

.

Replace all the occurrences of

x

in the decimal representation of

num

by

y

.

Let

a

and

b

be the two results from applying the operation to

num

independently

.

Return

the max difference

between

a

and

b

.

Note that neither

a

nor

b

may have any leading zeros, and

must not

be 0.

Example 1:

Input:

num = 555

Output:

888

Explanation:

The first time pick x = 5 and y = 9 and store the new integer in a. The second time pick x = 5 and y = 1 and store the new integer in b. We have now a = 999 and b = 111 and max difference = 888

Example 2:

Input:

num = 9

Output:

8

Explanation:

The first time pick x = 9 and y = 9 and store the new integer in a. The second time pick x = 9 and y = 1 and store the new integer in b. We have now a = 9 and b = 1 and max difference = 8

Constraints:

1 <= num <= 10

8

Code Snippets

C++:

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

}
};
```

Java:

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

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

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

};
```

C#:

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

}
}
```

C:

```
int maxDiff(int num) {

}
```

Go:

```
func maxDiff(num int) int {

}
```

Kotlin:

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

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

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

Solutions

C++ Solution:

```
/*
* Problem: Max Difference You Can Get From Changing an Integer
* Difficulty: Medium
* 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 maxDiff(int num) {

}
};
```

Java Solution:

```
/**
* Problem: Max Difference You Can Get From Changing an Integer
* Difficulty: Medium
* 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 maxDiff(int num) {
}
}
```

Python3 Solution:

```
"""
Problem: Max Difference You Can Get From Changing an Integer
Difficulty: Medium
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 maxDiff(self, num: int) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Max Difference You Can Get From Changing an Integer
* Difficulty: Medium
* 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 maxDiff = function(num) {

};
```

TypeScript Solution:

```
/**
* Problem: Max Difference You Can Get From Changing an Integer
* Difficulty: Medium
* 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 maxDiff(num: number): number {

};
```

C# Solution:

```
/*
* Problem: Max Difference You Can Get From Changing an Integer
* Difficulty: Medium
* 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 MaxDiff(int num) {

}
}
```

C Solution:

```
/*
* Problem: Max Difference You Can Get From Changing an Integer
* Difficulty: Medium
* 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 maxDiff(int num) {

}
```

Go Solution:

```
// Problem: Max Difference You Can Get From Changing an Integer
// Difficulty: Medium
// 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 maxDiff(num int) int {

}
```

Kotlin Solution:

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

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Max Difference You Can Get From Changing an Integer
// Difficulty: Medium
// 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 max_diff(num: i32) -> i32 {

}
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

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