Problem 2259: Remove Digit From Number to Maximize Result

Problem Information

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

Problem Description

You are given a string

number

representing a

positive integer

and a character

digit

.

Return

the resulting string after removing

exactly one occurrence

of

digit

from

number

such that the value of the resulting string in

decimal

form is

maximized

. The test cases are generated such that

digit

occurs at least once in

number

.

Example 1:

Input:

number = "123", digit = "3"

Output:

"12"

Explanation:

There is only one '3' in "123". After removing '3', the result is "12".

Example 2:

Input:

number = "1231", digit = "1"

Output:

"231"

Explanation:

We can remove the first '1' to get "231" or remove the second '1' to get "123". Since 231 > 123, we return "231".

Example 3:

Input:

number = "551", digit = "5"

Output:

"51"

Explanation:

We can remove either the first or second '5' from "551". Both result in the string "51".

Constraints:

2 <= number.length <= 100

number

consists of digits from

'1'

to

'9'

.

digit

is a digit from

'1'

to

'9'

.

digit

occurs at least once in

number

.

Code Snippets

C++:

```
class Solution {
public:
string removeDigit(string number, char digit) {

}
};
```

Java:

```
class Solution {
public String removeDigit(String number, char digit) {

}
```

```
}
```

Python3:

```
class Solution:
def removeDigit(self, number: str, digit: str) -> str:
```

Python:

```
class Solution(object):
def removeDigit(self, number, digit):
"""
:type number: str
:type digit: str
:rtype: str
"""
```

JavaScript:

```
/**
* @param {string} number
* @param {character} digit
* @return {string}
*/
var removeDigit = function(number, digit) {

};
```

TypeScript:

```
function removeDigit(number: string, digit: string): string {

};
```

C#:

```
public class Solution {
public string RemoveDigit(string number, char digit) {

}
}
```

C:

```
char* removeDigit(char* number, char digit) {

}
```

Go:

```
func removeDigit(number string, digit byte) string {

}
```

Kotlin:

```
class Solution {
fun removeDigit(number: String, digit: Char): String {

}
}
```

Swift:

```
class Solution {
func removeDigit(_ number: String, _ digit: Character) -> String {

}
}
```

Rust:

```
impl Solution {
pub fn remove_digit(number: String, digit: char) -> String {

}
}
```

Ruby:

```
# @param {String} number
# @param {Character} digit
# @return {String}
def remove_digit(number, digit)
```

```
end
```

PHP:

```
class Solution {

/**
* @param String $number
* @param String $digit
* @return String
*/
function removeDigit($number, $digit) {

}
}
```

Dart:

```
class Solution {
String removeDigit(String number, String digit) {

}
}
```

Scala:

```
object Solution {
def removeDigit(number: String, digit: Char): String = {

}
}
```

Elixir:

```
defmodule Solution do
@spec remove_digit(number :: String.t, digit :: char) :: String.t
def remove_digit(number, digit) do

end
end
```

Erlang:

```
-spec remove_digit(Number :: unicode:unicode_binary(), Digit :: char()) ->
unicode:unicode_binary().
remove_digit(Number, Digit) ->
.
```

Racket:

```
(define/contract (remove-digit number digit)
(-> string? char? string?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Remove Digit From Number to Maximize Result
* Difficulty: Easy
* Tags: string, greedy
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public:
string removeDigit(string number, char digit) {

}
};
```

Java Solution:

```
/**
* Problem: Remove Digit From Number to Maximize Result
* Difficulty: Easy
* Tags: string, greedy
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
```

```
* Space Complexity: O(1) to O(n) depending on approach
*/

class Solution {
public String removeDigit(String number, char digit) {

}
}
```

Python3 Solution:

```
"""
Problem: Remove Digit From Number to Maximize Result
Difficulty: Easy
Tags: string, greedy

Approach: String manipulation with hash map or two pointers
Time Complexity: O(n) or O(n log n)
Space Complexity: O(1) to O(n) depending on approach
"""

class Solution:
def removeDigit(self, number: str, digit: str) -> str:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def removeDigit(self, number, digit):
"""
:type number: str
:type digit: str
:rtype: str
"""
```

JavaScript Solution:

```
/**
* Problem: Remove Digit From Number to Maximize Result
* Difficulty: Easy
```

```
* Tags: string, greedy
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

/**
* @param {string} number
* @param {character} digit
* @return {string}
*/
var removeDigit = function(number, digit) {

};
```

TypeScript Solution:

```
/**
* Problem: Remove Digit From Number to Maximize Result
* Difficulty: Easy
* Tags: string, greedy
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

function removeDigit(number: string, digit: string): string {

};
```

C# Solution:

```
/*
* Problem: Remove Digit From Number to Maximize Result
* Difficulty: Easy
* Tags: string, greedy
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
```

```
* Space Complexity: O(1) to O(n) depending on approach
*/

public class Solution {
public string RemoveDigit(string number, char digit) {

}
}
```

C Solution:

```
/*
* Problem: Remove Digit From Number to Maximize Result
* Difficulty: Easy
* Tags: string, greedy
*
* Approach: String manipulation with hash map or two pointers
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(1) to O(n) depending on approach
*/

char* removeDigit(char* number, char digit) {

}
```

Go Solution:

```
// Problem: Remove Digit From Number to Maximize Result
// Difficulty: Easy
// Tags: string, greedy
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func removeDigit(number string, digit byte) string {

}
```

Kotlin Solution:

```
class Solution {
fun removeDigit(number: String, digit: Char): String {

}
}
```

Swift Solution:

```
class Solution {
func removeDigit(_ number: String, _ digit: Character) -> String {

}
}
```

Rust Solution:

```
// Problem: Remove Digit From Number to Maximize Result
// Difficulty: Easy
// Tags: string, greedy
//
// Approach: String manipulation with hash map or two pointers
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
pub fn remove_digit(number: String, digit: char) -> String {

}
}
```

Ruby Solution:

```
# @param {String} number
# @param {Character} digit
# @return {String}
def remove_digit(number, digit)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $number
* @param String $digit
* @return String
*/
function removeDigit($number, $digit) {

}
}
```

Dart Solution:

```
class Solution {
String removeDigit(String number, String digit) {

}
}
```

Scala Solution:

```
object Solution {
def removeDigit(number: String, digit: Char): String = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec remove_digit(number :: String.t, digit :: char) :: String.t
def remove_digit(number, digit) do

end
end
```

Erlang Solution:

```
-spec remove_digit(Number :: unicode:unicode_binary(), Digit :: char()) ->
unicode:unicode_binary().
remove_digit(Number, Digit) ->
```

```
.
```

Racket Solution:

```
(define/contract (remove-digit number digit)
(-> string? char? string?)
)
```