Problem 1946: Largest Number After Mutating Substring

Problem Information

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

Problem Description

You are given a string

num

, which represents a large integer. You are also given a

0-indexed

integer array

change

of length

10

that maps each digit

0-9

to another digit. More formally, digit

d

maps to digit

change[d]

.

You may

choose

to

mutate a single substring

of

num

. To mutate a substring, replace each digit

num[i]

with the digit it maps to in

change

(i.e. replace

num[i]

with

change[num[i]]

).

Return

a string representing the

largest

possible integer after

mutating

(or choosing not to) a

single substring

of

num

.

A

substring

is a contiguous sequence of characters within the string.

Example 1:

Input:

num = "

1

32", change = [9,8,5,0,3,6,4,2,6,8]

Output:

"

8

32"

Explanation:

Replace the substring "1": - 1 maps to change[1] = 8. Thus, "

1

32" becomes "

8

32". "832" is the largest number that can be created, so return it.

Example 2:

Input:

num = "

021

", change = [9,4,3,5,7,2,1,9,0,6]

Output:

"

934

"

Explanation:

Replace the substring "021": - 0 maps to change[0] = 9. - 2 maps to change[2] = 3. - 1 maps to change[1] = 4. Thus, "

021

" becomes "

934

". "934" is the largest number that can be created, so return it.

Example 3:

Input:

num = "5", change = [1,4,7,5,3,2,5,6,9,4]

Output:

"5"

Explanation:

"5" is already the largest number that can be created, so return it.

Constraints:

1 <= num.length <= 10

5

num

consists of only digits

0-9

.

change.length == 10

0 <= change[d] <= 9

Code Snippets

C++:

```
class Solution {
public:
string maximumNumber(string num, vector<int>& change) {

}
};
```

Java:

```
class Solution {
public String maximumNumber(String num, int[] change) {

}
}
```

Python3:

```
class Solution:
def maximumNumber(self, num: str, change: List[int]) -> str:
```

Python:

```
class Solution(object):
def maximumNumber(self, num, change):
"""
:type num: str
:type change: List[int]
:rtype: str
"""
```

JavaScript:

```
/**
* @param {string} num
* @param {number[]} change
* @return {string}
*/
var maximumNumber = function(num, change) {

};
```

TypeScript:

```
function maximumNumber(num: string, change: number[]): string {

};
```

C#:

```
public class Solution {
public string MaximumNumber(string num, int[] change) {

}
}
```

C:

```
char* maximumNumber(char* num, int* change, int changeSize) {

}
```

Go:

```
func maximumNumber(num string, change []int) string {

}
```

Kotlin:

```
class Solution {
fun maximumNumber(num: String, change: IntArray): String {

}
}
```

Swift:

```
class Solution {
func maximumNumber(_ num: String, _ change: [Int]) -> String {

}
}
```

Rust:

```
impl Solution {
pub fn maximum_number(num: String, change: Vec<i32>) -> String {

}
}
```

Ruby:

```
# @param {String} num
# @param {Integer[]} change
# @return {String}
def maximum_number(num, change)

end
```

PHP:

```
class Solution {

/**
* @param String $num
* @param Integer[] $change
* @return String
*/
function maximumNumber($num, $change) {

}
}
```

Dart:

```
class Solution {
String maximumNumber(String num, List<int> change) {

}
}
```

Scala:

```
object Solution {
def maximumNumber(num: String, change: Array[Int]): String = {

}
```

```
}
```

Elixir:

```
defmodule Solution do
@spec maximum_number(num :: String.t, change :: [integer]) :: String.t
def maximum_number(num, change) do

end
end
```

Erlang:

```
-spec maximum_number(Num :: unicode:unicode_binary(), Change :: [integer()])
-> unicode:unicode_binary().
maximum_number(Num, Change) ->
.
```

Racket:

```
(define/contract (maximum-number num change)
(-> string? (listof exact-integer?) string?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Largest Number After Mutating Substring
* Difficulty: Medium
* Tags: array, string, tree, greedy
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

class Solution {
public:
```

```
string maximumNumber(string num, vector<int>& change) {

}
};
```

Java Solution:

```
/**
* Problem: Largest Number After Mutating Substring
* Difficulty: Medium
* Tags: array, string, tree, greedy
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

class Solution {
public String maximumNumber(String num, int[] change) {

}
}
```

Python3 Solution:

```
"""
Problem: Largest Number After Mutating Substring
Difficulty: Medium
Tags: array, string, tree, greedy

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(h) for recursion stack where h is height
"""

class Solution:
def maximumNumber(self, num: str, change: List[int]) -> str:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def maximumNumber(self, num, change):
"""
:type num: str
:type change: List[int]
:rtype: str
"""
```

JavaScript Solution:

```
/**
* Problem: Largest Number After Mutating Substring
* Difficulty: Medium
* Tags: array, string, tree, greedy
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

/**
* @param {string} num
* @param {number[]} change
* @return {string}
*/
var maximumNumber = function(num, change) {

};
```

TypeScript Solution:

```
/**
* Problem: Largest Number After Mutating Substring
* Difficulty: Medium
* Tags: array, string, tree, greedy
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

function maximumNumber(num: string, change: number[]): string {
```

```
};
```

C# Solution:

```
/*
* Problem: Largest Number After Mutating Substring
* Difficulty: Medium
* Tags: array, string, tree, greedy
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

public class Solution {
public string MaximumNumber(string num, int[] change) {

}
}
```

C Solution:

```
/*
* Problem: Largest Number After Mutating Substring
* Difficulty: Medium
* Tags: array, string, tree, greedy
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(h) for recursion stack where h is height
*/

char* maximumNumber(char* num, int* change, int changeSize) {

}
```

Go Solution:

```
// Problem: Largest Number After Mutating Substring
// Difficulty: Medium
```

```
// Tags: array, string, tree, greedy
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

func maximumNumber(num string, change []int) string {

}
```

Kotlin Solution:

```
class Solution {
fun maximumNumber(num: String, change: IntArray): String {

}
}
```

Swift Solution:

```
class Solution {
func maximumNumber(_ num: String, _ change: [Int]) -> String {

}
}
```

Rust Solution:

```
// Problem: Largest Number After Mutating Substring
// Difficulty: Medium
// Tags: array, string, tree, greedy
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(h) for recursion stack where h is height

impl Solution {
pub fn maximum_number(num: String, change: Vec<i32>) -> String {

}
}
```

Ruby Solution:

```
# @param {String} num
# @param {Integer[]} change
# @return {String}
def maximum_number(num, change)

end
```

PHP Solution:

```
class Solution {

/**
* @param String $num
* @param Integer[] $change
* @return String
*/
function maximumNumber($num, $change) {

}
}
```

Dart Solution:

```
class Solution {
String maximumNumber(String num, List<int> change) {

}
}
```

Scala Solution:

```
object Solution {
def maximumNumber(num: String, change: Array[Int]): String = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec maximum_number(num :: String.t, change :: [integer]) :: String.t
def maximum_number(num, change) do

end
end
```

Erlang Solution:

```
-spec maximum_number(Num :: unicode:unicode_binary(), Change :: [integer()])
-> unicode:unicode_binary().
maximum_number(Num, Change) ->
.
```

Racket Solution:

```
(define/contract (maximum-number num change)
(-> string? (listof exact-integer?) string?)
)
```