Problem 1363: Largest Multiple of Three

Problem Information

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

Problem Description

Given an array of digits

digits

, return

the largest multiple of

three

that can be formed by concatenating some of the given digits in

any order

. If there is no answer return an empty string.

Since the answer may not fit in an integer data type, return the answer as a string. Note that the returning answer must not contain unnecessary leading zeros.

Example 1:

Input:

digits = [8,1,9]

Output:

"981"

Example 2:

Input:

digits = [8,6,7,1,0]

Output:

"8760"

Example 3:

Input:

digits = [1]

Output:

""

Constraints:

1 <= digits.length <= 10

4

0 <= digits[i] <= 9

Code Snippets

C++:

```
class Solution {
public:
string largestMultipleOfThree(vector<int>& digits) {
}
};
```

Java:

```
class Solution {
public String largestMultipleOfThree(int[] digits) {

}
}
```

Python3:

```
class Solution:
def largestMultipleOfThree(self, digits: List[int]) -> str:
```

Python:

```
class Solution(object):
def largestMultipleOfThree(self, digits):
"""
:type digits: List[int]
:rtype: str
"""
```

JavaScript:

```
/**
* @param {number[]} digits
* @return {string}
*/
var largestMultipleOfThree = function(digits) {

};
```

TypeScript:

```
function largestMultipleOfThree(digits: number[]): string {

};
```

C#:

```
}
```

```
public class Solution {
public string LargestMultipleOfThree(int[] digits) {

}
}
```

C:

```
char* largestMultipleOfThree(int* digits, int digitsSize) {

}
```

Go:

```
func largestMultipleOfThree(digits []int) string {

}
```

Kotlin:

```
class Solution {
fun largestMultipleOfThree(digits: IntArray): String {

}
}
```

Swift:

```
class Solution {
func largestMultipleOfThree(_ digits: [Int]) -> String {

}
}
```

Rust:

```
impl Solution {
pub fn largest_multiple_of_three(digits: Vec<i32>) -> String {

}
}
```

Ruby:

```
# @param {Integer[]} digits
# @return {String}
def largest_multiple_of_three(digits)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $digits
* @return String
*/
function largestMultipleOfThree($digits) {

}
}
```

Dart:

```
class Solution {
String largestMultipleOfThree(List<int> digits) {

}
}
```

Scala:

```
object Solution {
def largestMultipleOfThree(digits: Array[Int]): String = {

}
}
```

Elixir:

```
defmodule Solution do
@spec largest_multiple_of_three(digits :: [integer]) :: String.t
def largest_multiple_of_three(digits) do

end
end
```

Erlang:

```
-spec largest_multiple_of_three(Digits :: [integer()]) ->
unicode:unicode_binary().
largest_multiple_of_three(Digits) ->
.
```

Racket:

```
(define/contract (largest-multiple-of-three digits)
(-> (listof exact-integer?) string?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Largest Multiple of Three
* Difficulty: Hard
* Tags: array, string, dp, greedy, math, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public:
string largestMultipleOfThree(vector<int>& digits) {

}
};
```

Java Solution:

```
/**
* Problem: Largest Multiple of Three
* Difficulty: Hard
* Tags: array, string, dp, greedy, math, sort
*
```

```
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

class Solution {
public String largestMultipleOfThree(int[] digits) {

}
}
```

Python3 Solution:

```
"""
Problem: Largest Multiple of Three
Difficulty: Hard
Tags: array, string, dp, greedy, math, sort

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) or O(n * m) for DP table
"""

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

Python Solution:

```
class Solution(object):
def largestMultipleOfThree(self, digits):
"""
:type digits: List[int]
:rtype: str
"""
```

JavaScript Solution:

```
/**
* Problem: Largest Multiple of Three
* Difficulty: Hard
* Tags: array, string, dp, greedy, math, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

/**
* @param {number[]} digits
* @return {string}
*/
var largestMultipleOfThree = function(digits) {

};
```

TypeScript Solution:

```
/**
* Problem: Largest Multiple of Three
* Difficulty: Hard
* Tags: array, string, dp, greedy, math, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

function largestMultipleOfThree(digits: number[]): string {

};
```

C# Solution:

```
/*
* Problem: Largest Multiple of Three
* Difficulty: Hard
* Tags: array, string, dp, greedy, math, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

public class Solution {
public string LargestMultipleOfThree(int[] digits) {

}
}
```

C Solution:

```
/*
* Problem: Largest Multiple of Three
* Difficulty: Hard
* Tags: array, string, dp, greedy, math, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) or O(n * m) for DP table
*/

char* largestMultipleOfThree(int* digits, int digitsSize) {

}
```

Go Solution:

```
// Problem: Largest Multiple of Three
// Difficulty: Hard
// Tags: array, string, dp, greedy, math, sort
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

func largestMultipleOfThree(digits []int) string {

}
```

Kotlin Solution:

```
class Solution {
fun largestMultipleOfThree(digits: IntArray): String {

}
}
```

Swift Solution:

```
class Solution {
func largestMultipleOfThree(_ digits: [Int]) -> String {

}
}
```

Rust Solution:

```
// Problem: Largest Multiple of Three
// Difficulty: Hard
// Tags: array, string, dp, greedy, math, sort
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) or O(n * m) for DP table

impl Solution {
pub fn largest_multiple_of_three(digits: Vec<i32>) -> String {

}
}
```

Ruby Solution:

```
# @param {Integer[]} digits
# @return {String}
def largest_multiple_of_three(digits)

end
```

PHP Solution:

```
class Solution {
```

```
/**
* @param Integer[] $digits
* @return String
*/
function largestMultipleOfThree($digits) {

}
}
```

Dart Solution:

```
class Solution {
String largestMultipleOfThree(List<int> digits) {

}
}
```

Scala Solution:

```
object Solution {
def largestMultipleOfThree(digits: Array[Int]): String = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec largest_multiple_of_three(digits :: [integer]) :: String.t
def largest_multiple_of_three(digits) do

end
end
```

Erlang Solution:

```
-spec largest_multiple_of_three(Digits :: [integer()]) ->
unicode:unicode_binary().
largest_multiple_of_three(Digits) ->
.
```

Racket Solution:

```
(define/contract (largest-multiple-of-three digits)
(-> (listof exact-integer?) string?)
)
```