Problem 1262: Greatest Sum Divisible by Three

Problem Information

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

Problem Description

Given an integer array

nums

, return

the

maximum possible sum

of elements of the array such that it is divisible by three

.

Example 1:

Input:

nums = [3,6,5,1,8]

Output:

18

Explanation:

Pick numbers 3, 6, 1 and 8 their sum is 18 (maximum sum divisible by 3).

Example 2:

Input:

nums = [4]

Output:

0

Explanation:

Since 4 is not divisible by 3, do not pick any number.

Example 3:

Input:

nums = [1,2,3,4,4]

Output:

12

Explanation:

Pick numbers 1, 3, 4 and 4 their sum is 12 (maximum sum divisible by 3).

Constraints:

1 <= nums.length <= 4 * 10

4

1 <= nums[i] <= 10

4

Code Snippets

C++:

```
class Solution {
public:
int maxSumDivThree(vector<int>& nums) {

}
};
```

Java:

```
class Solution {
public int maxSumDivThree(int[] nums) {

}
}
```

Python3:

```
class Solution:
def maxSumDivThree(self, nums: List[int]) -> int:
```

Python:

```
class Solution(object):
def maxSumDivThree(self, nums):
"""
:type nums: List[int]
:rtype: int
"""
```

JavaScript:

```
/**
* @param {number[]} nums
* @return {number}
*/
var maxSumDivThree = function(nums) {
};
```

TypeScript:

```
function maxSumDivThree(nums: number[]): number {

};
```

C#:

```
public class Solution {
public int MaxSumDivThree(int[] nums) {

}
}
```

C:

```
int maxSumDivThree(int* nums, int numsSize) {

}
```

Go:

```
func maxSumDivThree(nums []int) int {

}
```

Kotlin:

```
class Solution {
fun maxSumDivThree(nums: IntArray): Int {

}
}
```

Swift:

```
class Solution {
func maxSumDivThree(_ nums: [Int]) -> Int {

}
}
```

Rust:

```
impl Solution {
pub fn max_sum_div_three(nums: Vec<i32>) -> i32 {

}
}
```

Ruby:

```
# @param {Integer[]} nums
# @return {Integer}
def max_sum_div_three(nums)

end
```

PHP:

```
class Solution {

/**
* @param Integer[] $nums
* @return Integer
*/
function maxSumDivThree($nums) {

}
}
```

Dart:

```
class Solution {
int maxSumDivThree(List<int> nums) {

}
}
```

Scala:

```
object Solution {
def maxSumDivThree(nums: Array[Int]): Int = {

}
}
```

Elixir:

```
defmodule Solution do
@spec max_sum_div_three(nums :: [integer]) :: integer
def max_sum_div_three(nums) do

end
end
```

Erlang:

```
-spec max_sum_div_three(Nums :: [integer()]) -> integer().
max_sum_div_three(Nums) ->
.
```

Racket:

```
(define/contract (max-sum-div-three nums)
(-> (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Greatest Sum Divisible by Three
* Difficulty: Medium
* Tags: array, dp, greedy, 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:
int maxSumDivThree(vector<int>& nums) {

}
};
```

Java Solution:

```
/**
* Problem: Greatest Sum Divisible by Three
* Difficulty: Medium
* Tags: array, dp, greedy, 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 int maxSumDivThree(int[] nums) {

}
}
```

Python3 Solution:

```
"""
Problem: Greatest Sum Divisible by Three
Difficulty: Medium
Tags: array, dp, greedy, 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 maxSumDivThree(self, nums: List[int]) -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
class Solution(object):
def maxSumDivThree(self, nums):
"""
:type nums: List[int]
:rtype: int
"""
```

JavaScript Solution:

```
/**
* Problem: Greatest Sum Divisible by Three
* Difficulty: Medium
* Tags: array, dp, greedy, 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[]} nums
* @return {number}
*/
var maxSumDivThree = function(nums) {

};
```

TypeScript Solution:

```
/**
* Problem: Greatest Sum Divisible by Three
* Difficulty: Medium
* Tags: array, dp, greedy, 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 maxSumDivThree(nums: number[]): number {
};
```

C# Solution:

```
/*
* Problem: Greatest Sum Divisible by Three
* Difficulty: Medium
* Tags: array, dp, greedy, 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 int MaxSumDivThree(int[] nums) {

}
}
```

C Solution:

```
/*
* Problem: Greatest Sum Divisible by Three
* Difficulty: Medium
* Tags: array, dp, greedy, 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
*/

int maxSumDivThree(int* nums, int numsSize) {

}
```

Go Solution:

```
// Problem: Greatest Sum Divisible by Three
// Difficulty: Medium
// Tags: array, dp, greedy, 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 maxSumDivThree(nums []int) int {

}
```

Kotlin Solution:

```
class Solution {
fun maxSumDivThree(nums: IntArray): Int {

}
}
```

Swift Solution:

```
class Solution {
func maxSumDivThree(_ nums: [Int]) -> Int {

}
}
```

Rust Solution:

```
// Problem: Greatest Sum Divisible by Three
// Difficulty: Medium
// Tags: array, dp, greedy, 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 max_sum_div_three(nums: Vec<i32>) -> i32 {

}
}
```

Ruby Solution:

```
# @param {Integer[]} nums
# @return {Integer}
def max_sum_div_three(nums)

end
```

PHP Solution:

```
class Solution {

/**
* @param Integer[] $nums
* @return Integer
*/
function maxSumDivThree($nums) {

}
}
```

Dart Solution:

```
class Solution {
int maxSumDivThree(List<int> nums) {

}
}
```

Scala Solution:

```
object Solution {
def maxSumDivThree(nums: Array[Int]): Int = {

}
}
```

Elixir Solution:

```
defmodule Solution do
@spec max_sum_div_three(nums :: [integer]) :: integer
def max_sum_div_three(nums) do
```

```
end
end
```

Erlang Solution:

```
-spec max_sum_div_three(Nums :: [integer()]) -> integer().
max_sum_div_three(Nums) ->
.
```

Racket Solution:

```
(define/contract (max-sum-div-three nums)
(-> (listof exact-integer?) exact-integer?)
)
```