

Problem 2007: Find Original Array From Doubled Array

Problem Information

Difficulty: Medium

Acceptance Rate: 0.00%

Paid Only: No

Problem Description

An integer array

original

is transformed into a

doubled

array

changed

by appending

twice the value

of every element in

original

, and then randomly

shuffling

the resulting array.

Given an array

changed

, return

original

if

changed

is a

doubled

array. If

changed

is not a

doubled

array, return an empty array. The elements in

original

may be returned in

any

order

.

Example 1:

Input:

changed = [1,3,4,2,6,8]

Output:

[1,3,4]

Explanation:

One possible original array could be [1,3,4]: - Twice the value of 1 is $1 * 2 = 2$. - Twice the value of 3 is $3 * 2 = 6$. - Twice the value of 4 is $4 * 2 = 8$. Other original arrays could be [4,3,1] or [3,1,4].

Example 2:

Input:

changed = [6,3,0,1]

Output:

[]

Explanation:

changed is not a doubled array.

Example 3:

Input:

changed = [1]

Output:

[]

Explanation:

changed is not a doubled array.

Constraints:

$1 \leq \text{changed.length} \leq 10$

5

$0 \leq \text{changed}[i] \leq 10$

5

Code Snippets

C++:

```
class Solution {  
public:  
    vector<int> findOriginalArray(vector<int>& changed) {  
  
    }  
};
```

Java:

```
class Solution {  
public int[] findOriginalArray(int[] changed) {  
  
}  
}
```

Python3:

```
class Solution:  
    def findOriginalArray(self, changed: List[int]) -> List[int]:
```

Python:

```
class Solution(object):  
    def findOriginalArray(self, changed):
```

```
"""
:type changed: List[int]
:rtype: List[int]
"""
```

JavaScript:

```
/**
 * @param {number[]} changed
 * @return {number[]}
 */
var findOriginalArray = function(changed) {

};
```

TypeScript:

```
function findOriginalArray(changed: number[]): number[] {
};

}
```

C#:

```
public class Solution {
public int[] FindOriginalArray(int[] changed) {

}
}
```

C:

```
/*
* Note: The returned array must be malloced, assume caller calls free().
*/
int* findOriginalArray(int* changed, int changedSize, int* returnSize) {

}
```

Go:

```
func findOriginalArray(changed []int) []int {
```

```
}
```

Kotlin:

```
class Solution {  
    fun findOriginalArray(changed: IntArray): IntArray {  
          
        }  
    }
```

Swift:

```
class Solution {  
    func findOriginalArray(_ changed: [Int]) -> [Int] {  
          
        }  
    }
```

Rust:

```
impl Solution {  
    pub fn find_original_array(changed: Vec<i32>) -> Vec<i32> {  
          
        }  
    }
```

Ruby:

```
# @param {Integer[]} changed  
# @return {Integer[]}  
def find_original_array(changed)  
  
end
```

PHP:

```
class Solution {  
  
    /**  
     * @param Integer[] $changed  
     * @return Integer[]  
     */
```

```
function findOriginalArray($changed) {  
}  
}  
}
```

Dart:

```
class Solution {  
List<int> findOriginalArray(List<int> changed) {  
}  
}  
}
```

Scala:

```
object Solution {  
def findOriginalArray(changed: Array[Int]): Array[Int] = {  
}  
}  
}
```

Elixir:

```
defmodule Solution do  
@spec find_original_array(changed :: [integer]) :: [integer]  
def find_original_array(changed) do  
  
end  
end
```

Erlang:

```
-spec find_original_array(Changed :: [integer()]) -> [integer()].  
find_original_array(Changed) ->  
.
```

Racket:

```
(define/contract (find-original-array changed)  
(-> (listof exact-integer?) (listof exact-integer?))  
)
```

Solutions

C++ Solution:

```
/*
 * Problem: Find Original Array From Doubled Array
 * Difficulty: Medium
 * Tags: array, greedy, hash, sort
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(n) for hash map
 */

class Solution {
public:
vector<int> findOriginalArray(vector<int>& changed) {

}
```

Java Solution:

```
/**
 * Problem: Find Original Array From Doubled Array
 * Difficulty: Medium
 * Tags: array, greedy, hash, sort
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(n) for hash map
 */

class Solution {
public int[] findOriginalArray(int[] changed) {

}
```

Python3 Solution:

```

"""
Problem: Find Original Array From Doubled Array
Difficulty: Medium
Tags: array, greedy, hash, sort

Approach: Use two pointers or sliding window technique
Time Complexity: O(n) or O(n log n)
Space Complexity: O(n) for hash map
"""

```

```

class Solution:

def findOriginalArray(self, changed: List[int]) -> List[int]:
    # TODO: Implement optimized solution
    pass

```

Python Solution:

```

class Solution(object):

def findOriginalArray(self, changed):
    """
    :type changed: List[int]
    :rtype: List[int]
    """

```

JavaScript Solution:

```

/**
 * Problem: Find Original Array From Doubled Array
 * Difficulty: Medium
 * Tags: array, greedy, hash, sort
 *
 * Approach: Use two pointers or sliding window technique
 * Time Complexity: O(n) or O(n log n)
 * Space Complexity: O(n) for hash map
 */

var findOriginalArray = function(changed) {

```

```
};
```

TypeScript Solution:

```
/**  
 * Problem: Find Original Array From Doubled Array  
 * Difficulty: Medium  
 * Tags: array, greedy, hash, sort  
 *  
 * Approach: Use two pointers or sliding window technique  
 * Time Complexity: O(n) or O(n log n)  
 * Space Complexity: O(n) for hash map  
 */  
  
function findOriginalArray(changed: number[]): number[] {  
  
};
```

C# Solution:

```
/*  
 * Problem: Find Original Array From Doubled Array  
 * Difficulty: Medium  
 * Tags: array, greedy, hash, sort  
 *  
 * Approach: Use two pointers or sliding window technique  
 * Time Complexity: O(n) or O(n log n)  
 * Space Complexity: O(n) for hash map  
 */  
  
public class Solution {  
    public int[] FindOriginalArray(int[] changed) {  
  
    }  
}
```

C Solution:

```
/*  
 * Problem: Find Original Array From Doubled Array  
 * Difficulty: Medium
```

```

* Tags: array, greedy, hash, sort
*
* Approach: Use two pointers or sliding window technique
* Time Complexity: O(n) or O(n log n)
* Space Complexity: O(n) for hash map
*/
/***
* Note: The returned array must be malloced, assume caller calls free().
*/
int* findOriginalArray(int* changed, int changedSize, int* returnSize) {

}

```

Go Solution:

```

// Problem: Find Original Array From Doubled Array
// Difficulty: Medium
// Tags: array, greedy, hash, sort
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

func findOriginalArray(changed []int) []int {
}

```

Kotlin Solution:

```

class Solution {
    fun findOriginalArray(changed: IntArray): IntArray {
        }
    }
}

```

Swift Solution:

```

class Solution {
    func findOriginalArray(_ changed: [Int]) -> [Int] {
}

```

```
}
```

```
}
```

Rust Solution:

```
// Problem: Find Original Array From Doubled Array
// Difficulty: Medium
// Tags: array, greedy, hash, sort
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(n) for hash map

impl Solution {
    pub fn find_original_array(changed: Vec<i32>) -> Vec<i32> {
        }

    }
}
```

Ruby Solution:

```
# @param {Integer[]} changed
# @return {Integer[]}
def find_original_array(changed)

end
```

PHP Solution:

```
class Solution {

    /**
     * @param Integer[] $changed
     * @return Integer[]
     */
    function findOriginalArray($changed) {

    }
}
```

Dart Solution:

```
class Solution {  
    List<int> findOriginalArray(List<int> changed) {  
          
    }  
}
```

Scala Solution:

```
object Solution {  
    def findOriginalArray(changed: Array[Int]): Array[Int] = {  
          
    }  
}
```

Elixir Solution:

```
defmodule Solution do  
    @spec find_original_array(changed :: [integer]) :: [integer]  
    def find_original_array(changed) do  
  
    end  
end
```

Erlang Solution:

```
-spec find_original_array(Changed :: [integer()]) -> [integer()].  
find_original_array(Changed) ->  
.
```

Racket Solution:

```
(define/contract (find-original-array changed)  
  (-> (listof exact-integer?) (listof exact-integer?))  
)
```