

# Problem 1588: Sum of All Odd Length Subarrays

## Problem Information

Difficulty: [Easy](#)

Acceptance Rate: 0.00%

Paid Only: No

## Problem Description

Given an array of positive integers

arr

, return

the sum of all possible

odd-length subarrays

of

arr

.

A

subarray

is a contiguous subsequence of the array.

Example 1:

Input:

arr = [1,4,2,5,3]

Output:

58

Explanation:

The odd-length subarrays of arr and their sums are: [1] = 1 [4] = 4 [2] = 2 [5] = 5 [3] = 3 [1,4,2] = 7 [4,2,5] = 11 [2,5,3] = 10 [1,4,2,5,3] = 15 If we add all these together we get  $1 + 4 + 2 + 5 + 3 + 7 + 11 + 10 + 15 = 58$

Example 2:

Input:

arr = [1,2]

Output:

3

Explanation:

There are only 2 subarrays of odd length, [1] and [2]. Their sum is 3.

Example 3:

Input:

arr = [10,11,12]

Output:

66

Constraints:

$1 \leq \text{arr.length} \leq 100$

$1 \leq arr[i] \leq 1000$

Follow up:

Could you solve this problem in  $O(n)$  time complexity?

## Code Snippets

**C++:**

```
class Solution {  
public:  
    int sumOddLengthSubarrays(vector<int>& arr) {  
  
    }  
};
```

**Java:**

```
class Solution {  
public int sumOddLengthSubarrays(int[] arr) {  
  
}  
}
```

**Python3:**

```
class Solution:  
    def sumOddLengthSubarrays(self, arr: List[int]) -> int:
```

**Python:**

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

**JavaScript:**

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

### TypeScript:

```
function sumOddLengthSubarrays(arr: number[]): number {  
  
};
```

### C#:

```
public class Solution {  
    public int SumOddLengthSubarrays(int[] arr) {  
  
    }  
}
```

### C:

```
int sumOddLengthSubarrays(int* arr, int arrSize) {  
  
}
```

### Go:

```
func sumOddLengthSubarrays(arr []int) int {  
  
}
```

### Kotlin:

```
class Solution {  
    fun sumOddLengthSubarrays(arr: IntArray): Int {  
  
    }  
}
```

### Swift:

```
class Solution {  
func sumOddLengthSubarrays(_ arr: [Int]) -> Int {  
}  
}  
}
```

### Rust:

```
impl Solution {  
pub fn sum_odd_length_subarrays(arr: Vec<i32>) -> i32 {  
}  
}  
}
```

### Ruby:

```
# @param {Integer[]} arr  
# @return {Integer}  
def sum_odd_length_subarrays(arr)  
  
end
```

### PHP:

```
class Solution {  
  
/**  
 * @param Integer[] $arr  
 * @return Integer  
 */  
function sumOddLengthSubarrays($arr) {  
  
}  
}
```

### Dart:

```
class Solution {  
int sumOddLengthSubarrays(List<int> arr) {  
  
}  
}
```

### **Scala:**

```
object Solution {  
    def sumOddLengthSubarrays(arr: Array[Int]): Int = {  
  
    }  
}
```

### **Elixir:**

```
defmodule Solution do  
  @spec sum_odd_length_subarrays(arr :: [integer]) :: integer  
  def sum_odd_length_subarrays(arr) do  
  
  end  
end
```

### **Erlang:**

```
-spec sum_odd_length_subarrays(Arr :: [integer()]) -> integer().  
sum_odd_length_subarrays(Arr) ->  
.
```

### **Racket:**

```
(define/contract (sum-odd-length-subarrays arr)  
  (-> (listof exact-integer?) exact-integer?)  
)
```

## **Solutions**

### **C++ Solution:**

```
/*  
 * Problem: Sum of All Odd Length Subarrays  
 * Difficulty: Easy  
 * Tags: array, math  
 *  
 * Approach: Use two pointers or sliding window technique  
 * Time Complexity: O(n) or O(n log n)  
 * Space Complexity: O(1) to O(n) depending on approach  
 */
```

```
class Solution {  
public:  
    int sumOddLengthSubarrays(vector<int>& arr) {  
        }  
    };
```

### Java Solution:

```
/**  
 * Problem: Sum of All Odd Length Subarrays  
 * Difficulty: Easy  
 * Tags: array, math  
 *  
 * Approach: Use two pointers or sliding window technique  
 * Time Complexity: O(n) or O(n log n)  
 * Space Complexity: O(1) to O(n) depending on approach  
 */  
  
class Solution {  
public int sumOddLengthSubarrays(int[] arr) {  
    }  
}
```

### Python3 Solution:

```
"""  
Problem: Sum of All Odd Length Subarrays  
Difficulty: Easy  
Tags: array, math  
  
Approach: Use two pointers or sliding window technique  
Time Complexity: O(n) or O(n log n)  
Space Complexity: O(1) to O(n) depending on approach  
"""  
  
class Solution:  
    def sumOddLengthSubarrays(self, arr: List[int]) -> int:  
        # TODO: Implement optimized solution
```

```
pass
```

### Python Solution:

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

### JavaScript Solution:

```
/**  
 * Problem: Sum of All Odd Length Subarrays  
 * Difficulty: Easy  
 * Tags: array, math  
 *  
 * Approach: Use two pointers or sliding window technique  
 * Time Complexity: O(n) or O(n log n)  
 * Space Complexity: O(1) to O(n) depending on approach  
 */  
  
/**  
 * @param {number[]} arr  
 * @return {number}  
 */  
var sumOddLengthSubarrays = function(arr) {  
  
};
```

### TypeScript Solution:

```
/**  
 * Problem: Sum of All Odd Length Subarrays  
 * Difficulty: Easy  
 * Tags: array, math  
 *  
 * Approach: Use two pointers or sliding window technique  
 * Time Complexity: O(n) or O(n log n)  
 * Space Complexity: O(1) to O(n) depending on approach
```

```
*/\n\nfunction sumOddLengthSubarrays(arr: number[]): number {\n}\n\n};
```

### C# Solution:

```
/*\n * Problem: Sum of All Odd Length Subarrays\n * Difficulty: Easy\n * Tags: array, math\n *\n * Approach: Use two pointers or sliding window technique\n * Time Complexity: O(n) or O(n log n)\n * Space Complexity: O(1) to O(n) depending on approach\n */\n\npublic class Solution {\n    public int SumOddLengthSubarrays(int[] arr) {\n\n    }\n}
```

### C Solution:

```
/*\n * Problem: Sum of All Odd Length Subarrays\n * Difficulty: Easy\n * Tags: array, math\n *\n * Approach: Use two pointers or sliding window technique\n * Time Complexity: O(n) or O(n log n)\n * Space Complexity: O(1) to O(n) depending on approach\n */\n\nint sumOddLengthSubarrays(int* arr, int arrSize) {\n\n}
```

### Go Solution:

```

// Problem: Sum of All Odd Length Subarrays
// Difficulty: Easy
// Tags: array, math
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

func sumOddLengthSubarrays(arr []int) int {

}

```

### Kotlin Solution:

```

class Solution {
    fun sumOddLengthSubarrays(arr: IntArray): Int {
        }

    }

```

### Swift Solution:

```

class Solution {
    func sumOddLengthSubarrays(_ arr: [Int]) -> Int {
        }

    }

```

### Rust Solution:

```

// Problem: Sum of All Odd Length Subarrays
// Difficulty: Easy
// Tags: array, math
//
// Approach: Use two pointers or sliding window technique
// Time Complexity: O(n) or O(n log n)
// Space Complexity: O(1) to O(n) depending on approach

impl Solution {
    pub fn sum_odd_length_subarrays(arr: Vec<i32>) -> i32 {
        }
}

```

```
}
```

### Ruby Solution:

```
# @param {Integer[]} arr
# @return {Integer}
def sum_odd_length_subarrays(arr)

end
```

### PHP Solution:

```
class Solution {

    /**
     * @param Integer[] $arr
     * @return Integer
     */
    function sumOddLengthSubarrays($arr) {

    }
}
```

### Dart Solution:

```
class Solution {
int sumOddLengthSubarrays(List<int> arr) {

}
```

### Scala Solution:

```
object Solution {
def sumOddLengthSubarrays(arr: Array[Int]): Int = {

}
```

### Elixir Solution:

```
defmodule Solution do
@spec sum_odd_length_subarrays(arr :: [integer]) :: integer
def sum_odd_length_subarrays(arr) do

end
end
```

### Erlang Solution:

```
-spec sum_odd_length_subarrays(Arr :: [integer()]) -> integer().
sum_odd_length_subarrays(Arr) ->
.
```

### Racket Solution:

```
(define/contract (sum-odd-length-subarrays arr)
(-> (listof exact-integer?) exact-integer?))
```