Problem 1186: Maximum Subarray Sum with One Deletion

Problem Information

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

Problem Description

Given an array of integers, return the maximum sum for a

non-empty

subarray (contiguous elements) with at most one element deletion. In other words, you want to choose a subarray and optionally delete one element from it so that there is still at least one element left and the sum of the remaining elements is maximum possible.

Note that the subarray needs to be

non-empty

after deleting one element.

Example 1:

Input:

arr = [1,-2,0,3]

Output:

4

Explanation:

Because we can choose [1, -2, 0, 3] and drop -2, thus the subarray [1, 0, 3] becomes the maximum value.

Example 2:

Input:

arr = [1,-2,-2,3]

Output:

3

Explanation:

We just choose [3] and it's the maximum sum.

Example 3:

Input:

arr = [-1,-1,-1,-1]

Output:

-1

Explanation:

The final subarray needs to be non-empty. You can't choose [-1] and delete -1 from it, then get an empty subarray to make the sum equals to 0.

Constraints:

1 <= arr.length <= 10

5

-10

4

<= arr[i] <= 10

4

Code Snippets

C++:

```
class Solution {
public:
int maximumSum(vector<int>& arr) {

}
};
```

Java:

```
class Solution {
public int maximumSum(int[] arr) {

}
}
```

Python3:

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

Python:

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

JavaScript:

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

};
```

TypeScript:

```
function maximumSum(arr: number[]): number {

};
```

C#:

```
public class Solution {
public int MaximumSum(int[] arr) {

}
}
```

C:

```
int maximumSum(int* arr, int arrSize) {

}
```

Go:

```
func maximumSum(arr []int) int {

}
```

Kotlin:

```
class Solution {
fun maximumSum(arr: IntArray): Int {

}
}
```

Swift:

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

}
}
```

Rust:

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

}
}
```

Ruby:

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

end
```

PHP:

```
class Solution {

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

}
}
```

Dart:

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

}
}
```

Scala:

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

}
}
```

Elixir:

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

end
end
```

Erlang:

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

Racket:

```
(define/contract (maximum-sum arr)
(-> (listof exact-integer?) exact-integer?)
)
```

Solutions

C++ Solution:

```
/*
* Problem: Maximum Subarray Sum with One Deletion
* Difficulty: Medium
* Tags: array, dp
*
* 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 maximumSum(vector<int>& arr) {

}
};
```

Java Solution:

```
/**
* Problem: Maximum Subarray Sum with One Deletion
* Difficulty: Medium
* Tags: array, dp
*
* 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 maximumSum(int[] arr) {

}
}
```

Python3 Solution:

```
"""
Problem: Maximum Subarray Sum with One Deletion
Difficulty: Medium
Tags: array, dp

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 maximumSum(self, arr: List[int]) -> int:
# TODO: Implement optimized solution
```

```
pass
```

Python Solution:

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

JavaScript Solution:

```
/**
* Problem: Maximum Subarray Sum with One Deletion
* Difficulty: Medium
* Tags: array, dp
*
* 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[]} arr
* @return {number}
*/
var maximumSum = function(arr) {

};
```

TypeScript Solution:

```
/**
* Problem: Maximum Subarray Sum with One Deletion
* Difficulty: Medium
* Tags: array, dp
*
* 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 maximumSum(arr: number[]): number {

};
```

C# Solution:

```
/*
* Problem: Maximum Subarray Sum with One Deletion
* Difficulty: Medium
* Tags: array, dp
*
* 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 MaximumSum(int[] arr) {

}
}
```

C Solution:

```
/*
* Problem: Maximum Subarray Sum with One Deletion
* Difficulty: Medium
* Tags: array, dp
*
* 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 maximumSum(int* arr, int arrSize) {

}
```

Go Solution:

```
// Problem: Maximum Subarray Sum with One Deletion
// Difficulty: Medium
// Tags: array, dp
//
// 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 maximumSum(arr []int) int {

}
```

Kotlin Solution:

```
class Solution {
fun maximumSum(arr: IntArray): Int {

}
}
```

Swift Solution:

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

}
}
```

Rust Solution:

```
// Problem: Maximum Subarray Sum with One Deletion
// Difficulty: Medium
// Tags: array, dp
//
// 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 maximum_sum(arr: Vec<i32>) -> i32 {

}
```

```
}
```

Ruby Solution:

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

end
```

PHP Solution:

```
class Solution {

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

}
}
```

Dart Solution:

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

}
}
```

Scala Solution:

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

}
}
```

Elixir Solution:

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

end
end
```

Erlang Solution:

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

Racket Solution:

```
(define/contract (maximum-sum arr)
(-> (listof exact-integer?) exact-integer?)
)
```