Problem 1533: Find the Index of the Large Integer

Problem Information

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

Problem Description

We have an integer array

arr

, where all the integers in

arr

are equal except for one integer which is

larger

than the rest of the integers. You will not be given direct access to the array, instead, you will have an

API

ArrayReader

which have the following functions:

int compareSub(int l, int r, int x, int y)

: where

0 <= l, r, x, y < ArrayReader.length()

,

l <= r and

x <= y

. The function compares the sum of sub-array

arr[l..r]

with the sum of the sub-array

arr[x..y]

and returns:

1

if

arr[l]+arr[l+1]+...+arr[r] > arr[x]+arr[x+1]+...+arr[y]

.

0

if

arr[l]+arr[l+1]+...+arr[r] == arr[x]+arr[x+1]+...+arr[y]

.

-1

if

arr[l]+arr[l+1]+...+arr[r] < arr[x]+arr[x+1]+...+arr[y]

.

int length()

: Returns the size of the array.

You are allowed to call

compareSub()

20 times

at most. You can assume both functions work in

O(1)

time.

Return

the index of the array

arr

which has the largest integer

.

Example 1:

Input:

arr = [7,7,7,7,10,7,7,7]

Output:

4

Explanation:

The following calls to the API reader.compareSub(0, 0, 1, 1) // returns 0 this is a query comparing the sub-array (0, 0) with the sub array (1, 1), (i.e. compares arr[0] with arr[1]). Thus we know that arr[0] and arr[1] doesn't contain the largest element. reader.compareSub(2, 2, 3, 3) // returns 0, we can exclude arr[2] and arr[3]. reader.compareSub(4, 4, 5, 5) // returns 1, thus for sure arr[4] is the largest element in the array. Notice that we made only 3 calls, so the answer is valid.

Example 2:

Input:

nums = [6,6,12]

Output:

2

Constraints:

2 <= arr.length <= 5 * 10

5

1 <= arr[i] <= 100

All elements of

arr

are equal except for one element which is larger than all other elements.

Follow up:

What if there are two numbers in

arr

that are bigger than all other numbers?

What if there is one number that is bigger than other numbers and one number that is smaller than other numbers?

Code Snippets

C++:

```
/**
* // This is the ArrayReader's API interface.
* // You should not implement it, or speculate about its implementation
* class ArrayReader {
* public:
* // Compares the sum of arr[l..r] with the sum of arr[x..y]
* // return 1 if sum(arr[l..r]) > sum(arr[x..y])
* // return 0 if sum(arr[l..r]) == sum(arr[x..y])
* // return -1 if sum(arr[l..r]) < sum(arr[x..y])
* int compareSub(int l, int r, int x, int y);
*
* // Returns the length of the array
* int length();
* };
*/

class Solution {
public:
int getIndex(ArrayReader &reader) {

}
};
```

Java:

```
/**
* // This is ArrayReader's API interface.
* // You should not implement it, or speculate about its implementation
* interface ArrayReader {
* // Compares the sum of arr[l..r] with the sum of arr[x..y]
* // return 1 if sum(arr[l..r]) > sum(arr[x..y])
* // return 0 if sum(arr[l..r]) == sum(arr[x..y])
* // return -1 if sum(arr[l..r]) < sum(arr[x..y])
* public int compareSub(int l, int r, int x, int y) {}
*
* // Returns the length of the array
* public int length() {}
* }
*/

class Solution {
public int getIndex(ArrayReader reader) {

}
}
```

Python3:

```
# """
# This is ArrayReader's API interface.
# You should not implement it, or speculate about its implementation
# """
#class ArrayReader(object):
# # Compares the sum of arr[l..r] with the sum of arr[x..y]
# # return 1 if sum(arr[l..r]) > sum(arr[x..y])
# # return 0 if sum(arr[l..r]) == sum(arr[x..y])
# # return -1 if sum(arr[l..r]) < sum(arr[x..y])
# def compareSub(self, l: int, r: int, x: int, y: int) -> int:
#
# # Returns the length of the array
# def length(self) -> int:
#


class Solution:
def getIndex(self, reader: 'ArrayReader') -> int:
```

Python:

```
# """
# This is ArrayReader's API interface.
# You should not implement it, or speculate about its implementation
# """
#class ArrayReader(object):
```

```
# # Compares the sum of arr[l..r] with the sum of arr[x..y]
# # return 1 if sum(arr[l..r]) > sum(arr[x..y])
# # return 0 if sum(arr[l..r]) == sum(arr[x..y])
# # return -1 if sum(arr[l..r]) < sum(arr[x..y])
# def compareSub(self, l, r, x, y):
# """
# :type l, r, x, y: int
# :rtype int
# """
#
# # Returns the length of the array
# def length(self):
# """
# :rtype int
# """

class Solution(object):
def getIndex(self, reader):
"""
:type reader: ArrayReader
:rtype: integer
"""
```

JavaScript:

```
/**
* // This is the ArrayReader's API interface.
* // You should not implement it, or speculate about its implementation
* function ArrayReader() {
* // Compares the sum of arr[l..r] with the sum of arr[x..y]
* // return 1 if sum(arr[l..r]) > sum(arr[x..y])
* // return 0 if sum(arr[l..r]) == sum(arr[x..y])
* // return -1 if sum(arr[l..r]) < sum(arr[x..y])
* @param {number} l, r, x, y
* @return {number}
* this.compareSub = function(l, r, x, y) {
* ...
* };
*
* // Returns the length of the array
* @return {number}
* this.length = function() {
```

```
* ...
* };
* };
*/

/**
* @param {ArrayReader} reader
* @return {number}
*/
var getIndex = function(reader) {

};
```

TypeScript:

```
/**
* // This is the ArrayReader's API interface.
* // You should not implement it, or speculate about its implementation
* class ArrayReader {
* // Compares the sum of arr[l..r] with the sum of arr[x..y]
* // return 1 if sum(arr[l..r]) > sum(arr[x..y])
* // return 0 if sum(arr[l..r]) == sum(arr[x..y])
* // return -1 if sum(arr[l..r]) < sum(arr[x..y])
* compareSub(l: number, r: number, x: number, y: number): number { };
*
* // Returns the length of the array
* length(): number { };
* };
*/

function getIndex(reader: ArrayReader): number {

};
```

C#:

```
/**
* // This is ArrayReader's API interface.
* // You should not implement it, or speculate about its implementation
* class ArrayReader {
* // Compares the sum of arr[l..r] with the sum of arr[x..y]
* // return 1 if sum(arr[l..r]) > sum(arr[x..y])
* // return 0 if sum(arr[l..r]) == sum(arr[x..y])
* // return -1 if sum(arr[l..r]) < sum(arr[x..y])
* public int CompareSub(int l, int r, int x, int y) {}
*
* // Returns the length of the array
* public int Length() {}
* }
*/

class Solution {
public int GetIndex(ArrayReader reader) {

}
}
```

C:

```
/**
* *********************************************************************
* // This is the ArrayReader's API interface.
* // You should not implement it, or speculate about its implementation
* *********************************************************************
*
* // Compares the sum of arr[l..r] with the sum of arr[x..y]
* // return 1 if sum(arr[l..r]) > sum(arr[x..y])
* // return 0 if sum(arr[l..r]) == sum(arr[x..y])
* // return -1 if sum(arr[l..r]) < sum(arr[x..y])
* int compareSub(ArrayReader *, int l, int r, int x, int y);
*
* // Returns the length of the array
* int length(ArrayReader *);
*/

int getIndex(ArrayReader* reader) {

}
```

Go:

```
/**
* // This is the ArrayReader's API interface.
* // You should not implement it, or speculate about its implementation
* type ArrayReader struct {
* }
* // Compares the sum of arr[l..r] with the sum of arr[x..y]
* // return 1 if sum(arr[l..r]) > sum(arr[x..y])
* // return 0 if sum(arr[l..r]) == sum(arr[x..y])
* // return -1 if sum(arr[l..r]) < sum(arr[x..y])
* func (this *ArrayReader) compareSub(l, r, x, y int) int {}
*
* // Returns the length of the array
* func (this *ArrayReader) length() int {}
*/

func getIndex(reader *ArrayReader) int {

}
```

Kotlin:

```
/**
* // This is ArrayReader's API interface.
* // You should not implement it, or speculate about its implementation
* interface ArrayReader {
* // Compares the sum of arr[l..r] with the sum of arr[x..y]
* // return 1 if sum(arr[l..r]) > sum(arr[x..y])
* // return 0 if sum(arr[l..r]) == sum(arr[x..y])
* // return -1 if sum(arr[l..r]) < sum(arr[x..y])
* fun compareSub(l: Int, r: Int, x: Int, y: Int): Int {}
*
* // Returns the length of the array
* fun length(): Int {}
* }
*/

class Solution {
fun getIndex(reader: ArrayReader): Int {

}
}
```

Swift:

```
/**
* // This is ArrayReader's API interface.
* // You should not implement it, or speculate about its implementation
* interface ArrayReader {
* // Compares the sum of arr[l..r] with the sum of arr[x..y]
* // return 1 if sum(arr[l..r]) > sum(arr[x..y])
* // return 0 if sum(arr[l..r]) == sum(arr[x..y])
* // return -1 if sum(arr[l..r]) < sum(arr[x..y])
* public func compareSub(_ l: Int, _ r: Int, _ x: Int, _ y: Int) -> Int {}
*
* // Returns the length of the array
* public func length() -> Int {}
* }
*/

class Solution {
func getIndex(_ reader: ArrayReader) -> Int {

}
}
```

Rust:

```
/**
* // This is the ArrayReader's API interface.
* // You should not implement it, or speculate about its implementation
* struct ArrayReader;
* impl Array Reader {
* pub fn compareSub(l: i32, r: i32, x: i32, y: i32) -> i32 {}
* // Compares the sum of arr[l..r] with the sum of arr[x..y]
* // return 1 if sum(arr[l..r]) > sum(arr[x..y])
* // return 0 if sum(arr[l..r]) == sum(arr[x..y])
* // return -1 if sum(arr[l..r]) < sum(arr[x..y])
* // Returns the length of the array
* }
*/

impl Solution {
pub fn get_index(reader: &ArrayReader) -> i32 {

}
}
```

Ruby:

```
# This is ArrayReader's API interface.
# You should not implement it, or speculate about its implementation
# class ArrayReader
# # Compares the sum of arr[l..r] with the sum of arr[x..y]
# # return 1 if sum(arr[l..r]) > sum(arr[x..y])
# # return 0 if sum(arr[l..r]) == sum(arr[x..y])
# # return -1 if sum(arr[l..r]) < sum(arr[x..y])
# def compare_sub(l, r, x, y):
#
# end
#
# # Returns the length of the array
# def length()
#
# end
# end

# @param {ArrayReader} reader
# @return {int}
def get_index(reader)

end
```

PHP:

```
/**
* // This is ArrayReader's API interface.
* // You should not implement it, or speculate about its implementation
* interface ArrayReader {
* // Compares the sum of arr[l..r] with the sum of arr[x..y]
* // return 1 if sum(arr[l..r]) > sum(arr[x..y])
* // return 0 if sum(arr[l..r]) == sum(arr[x..y])
* // return -1 if sum(arr[l..r]) < sum(arr[x..y])
* function compareSub($l, $r, $x, $y) {}
*
* // Returns the length of the array
* function length() {}
* }
*/
```

```
class Solution {
/**
* @param ArrayReader $reader
* @return Integer
*/
function getIndex($reader) {

}
}
```

Scala:

```
/**
* // This is ArrayReader's API interface.
* // You should not implement it, or speculate about its implementation
* interface ArrayReader {
* // Compares the sum of arr[l..r] with the sum of arr[x..y]
* // return 1 if sum(arr[l..r]) > sum(arr[x..y])
* // return 0 if sum(arr[l..r]) == sum(arr[x..y])
* // return -1 if sum(arr[l..r]) < sum(arr[x..y])
* def compareSub(l: Int, r: Int, x: Int, y: Int): Int {}
*
* // Returns the length of the array
* def length(): Int {}
* }
*/

object Solution {
def getIndex(reader: ArrayReader): Int = {

}
}
```

Solutions

C++ Solution:

```
/*
* Problem: Find the Index of the Large Integer
* Difficulty: Medium
* Tags: array, search
*
* 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
*/

/**
* // This is the ArrayReader's API interface.
* // You should not implement it, or speculate about its implementation
* class ArrayReader {
* public:
* // Compares the sum of arr[l..r] with the sum of arr[x..y]
* // return 1 if sum(arr[l..r]) > sum(arr[x..y])
* // return 0 if sum(arr[l..r]) == sum(arr[x..y])
* // return -1 if sum(arr[l..r]) < sum(arr[x..y])
* int compareSub(int l, int r, int x, int y);
*
* // Returns the length of the array
* int length();
* };
*/

class Solution {
public:
int getIndex(ArrayReader &reader) {

}
};
```

Java Solution:

```
/**
* Problem: Find the Index of the Large Integer
* Difficulty: Medium
* Tags: array, search
*
* 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
*/
```

```
/**
* // This is ArrayReader's API interface.
* // You should not implement it, or speculate about its implementation
* interface ArrayReader {
* // Compares the sum of arr[l..r] with the sum of arr[x..y]
* // return 1 if sum(arr[l..r]) > sum(arr[x..y])
* // return 0 if sum(arr[l..r]) == sum(arr[x..y])
* // return -1 if sum(arr[l..r]) < sum(arr[x..y])
* public int compareSub(int l, int r, int x, int y) {
// TODO: Implement optimized solution
return 0;
}
*
* // Returns the length of the array
* public int length() {
// TODO: Implement optimized solution
return 0;
}
* }
*/

class Solution {
public int getIndex(ArrayReader reader) {

}
}
```

Python3 Solution:

```
"""
Problem: Find the Index of the Large Integer
Difficulty: Medium
Tags: array, search

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
"""

# """
# This is ArrayReader's API interface.
# You should not implement it, or speculate about its implementation
# """
#class ArrayReader(object):
# # Compares the sum of arr[l..r] with the sum of arr[x..y]
# # return 1 if sum(arr[l..r]) > sum(arr[x..y])
# # return 0 if sum(arr[l..r]) == sum(arr[x..y])
# # return -1 if sum(arr[l..r]) < sum(arr[x..y])
# def compareSub(self, l: int, r: int, x: int, y: int) -> int:
#
# # Returns the length of the array
# def length(self) -> int:
#



class Solution:
def getIndex(self, reader: 'ArrayReader') -> int:
# TODO: Implement optimized solution
pass
```

Python Solution:

```
# """
# This is ArrayReader's API interface.
# You should not implement it, or speculate about its implementation
# """
#class ArrayReader(object):
# # Compares the sum of arr[l..r] with the sum of arr[x..y]
# # return 1 if sum(arr[l..r]) > sum(arr[x..y])
# # return 0 if sum(arr[l..r]) == sum(arr[x..y])
# # return -1 if sum(arr[l..r]) < sum(arr[x..y])
# def compareSub(self, l, r, x, y):
# """
# :type l, r, x, y: int
# :rtype int
# """
#
# # Returns the length of the array
# def length(self):
# """
# :rtype int
# """

class Solution(object):
def getIndex(self, reader):
"""
:type reader: ArrayReader
:rtype: integer
"""
```

JavaScript Solution:

```
/**
* Problem: Find the Index of the Large Integer
* Difficulty: Medium
* Tags: array, search
*
* 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
*/

/**
* // This is the ArrayReader's API interface.
* // You should not implement it, or speculate about its implementation
* function ArrayReader() {
* // Compares the sum of arr[l..r] with the sum of arr[x..y]
* // return 1 if sum(arr[l..r]) > sum(arr[x..y])
* // return 0 if sum(arr[l..r]) == sum(arr[x..y])
* // return -1 if sum(arr[l..r]) < sum(arr[x..y])
* @param {number} l, r, x, y
* @return {number}
* this.compareSub = function(l, r, x, y) {
* ...
* };
*
* // Returns the length of the array
* @return {number}
* this.length = function() {
* ...
* };
* };
```

```
*/

/**
* @param {ArrayReader} reader
* @return {number}
*/
var getIndex = function(reader) {

};
```

TypeScript Solution:

```
/**
* Problem: Find the Index of the Large Integer
* Difficulty: Medium
* Tags: array, search
*
* 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
*/

/**
* // This is the ArrayReader's API interface.
* // You should not implement it, or speculate about its implementation
* class ArrayReader {
* // Compares the sum of arr[l..r] with the sum of arr[x..y]
* // return 1 if sum(arr[l..r]) > sum(arr[x..y])
* // return 0 if sum(arr[l..r]) == sum(arr[x..y])
* // return -1 if sum(arr[l..r]) < sum(arr[x..y])
* compareSub(l: number, r: number, x: number, y: number): number { };
*
* // Returns the length of the array
* length(): number { };
* };
*/

function getIndex(reader: ArrayReader): number {

};
```

C# Solution:

```
/*
* Problem: Find the Index of the Large Integer
* Difficulty: Medium
* Tags: array, search
*
* 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
*/

/**
* // This is ArrayReader's API interface.
* // You should not implement it, or speculate about its implementation
* class ArrayReader {
* // Compares the sum of arr[l..r] with the sum of arr[x..y]
* // return 1 if sum(arr[l..r]) > sum(arr[x..y])
* // return 0 if sum(arr[l..r]) == sum(arr[x..y])
* // return -1 if sum(arr[l..r]) < sum(arr[x..y])
* public int CompareSub(int l, int r, int x, int y) {}
*
* // Returns the length of the array
* public int Length() {}
* }
*/

class Solution {
public int GetIndex(ArrayReader reader) {

}
}
```

C Solution:

```
/*
* Problem: Find the Index of the Large Integer
* Difficulty: Medium
* Tags: array, search
*
* 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
```

```
*/

/**
* ********************************************************************
* // This is the ArrayReader's API interface.
* // You should not implement it, or speculate about its implementation
* ********************************************************************
*
* // Compares the sum of arr[l..r] with the sum of arr[x..y]
* // return 1 if sum(arr[l..r]) > sum(arr[x..y])
* // return 0 if sum(arr[l..r]) == sum(arr[x..y])
* // return -1 if sum(arr[l..r]) < sum(arr[x..y])
* int compareSub(ArrayReader *, int l, int r, int x, int y);
*
* // Returns the length of the array
* int length(ArrayReader *);
*/

int getIndex(ArrayReader* reader) {

}
```

Go Solution:

```
// Problem: Find the Index of the Large Integer
// Difficulty: Medium
// Tags: array, search
//
// 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

/**
* // This is the ArrayReader's API interface.
* // You should not implement it, or speculate about its implementation
* type ArrayReader struct {
* }
* // Compares the sum of arr[l..r] with the sum of arr[x..y]
* // return 1 if sum(arr[l..r]) > sum(arr[x..y])
* // return 0 if sum(arr[l..r]) == sum(arr[x..y])
* // return -1 if sum(arr[l..r]) < sum(arr[x..y])
* func (this *ArrayReader) compareSub(l, r, x, y int) int {}
*
* // Returns the length of the array
* func (this *ArrayReader) length() int {}
*/

func getIndex(reader *ArrayReader) int {

}
```

Kotlin Solution:

```
/**
* // This is ArrayReader's API interface.
* // You should not implement it, or speculate about its implementation
* interface ArrayReader {
* // Compares the sum of arr[l..r] with the sum of arr[x..y]
* // return 1 if sum(arr[l..r]) > sum(arr[x..y])
* // return 0 if sum(arr[l..r]) == sum(arr[x..y])
* // return -1 if sum(arr[l..r]) < sum(arr[x..y])
* fun compareSub(l: Int, r: Int, x: Int, y: Int): Int {}
*
* // Returns the length of the array
* fun length(): Int {}
* }
*/

class Solution {
fun getIndex(reader: ArrayReader): Int {

}
}
```

Swift Solution:

```
/**
* // This is ArrayReader's API interface.
* // You should not implement it, or speculate about its implementation
* interface ArrayReader {
* // Compares the sum of arr[l..r] with the sum of arr[x..y]
* // return 1 if sum(arr[l..r]) > sum(arr[x..y])
* // return 0 if sum(arr[l..r]) == sum(arr[x..y])
* // return -1 if sum(arr[l..r]) < sum(arr[x..y])
* public func compareSub(_ l: Int, _ r: Int, _ x: Int, _ y: Int) -> Int {}
*
* // Returns the length of the array
* public func length() -> Int {}
* }
*/

class Solution {
func getIndex(_ reader: ArrayReader) -> Int {

}
}
```

Rust Solution:

```
// Problem: Find the Index of the Large Integer
// Difficulty: Medium
// Tags: array, search
//
// 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

/**
* // This is the ArrayReader's API interface.
* // You should not implement it, or speculate about its implementation
* struct ArrayReader;
* impl Array Reader {
* pub fn compareSub(l: i32, r: i32, x: i32, y: i32) -> i32 {}
* // Compares the sum of arr[l..r] with the sum of arr[x..y]
* // return 1 if sum(arr[l..r]) > sum(arr[x..y])
* // return 0 if sum(arr[l..r]) == sum(arr[x..y])
* // return -1 if sum(arr[l..r]) < sum(arr[x..y])
* // Returns the length of the array
* }
*/

impl Solution {
pub fn get_index(reader: &ArrayReader) -> i32 {
}
}
```

Ruby Solution:

```
# This is ArrayReader's API interface.
# You should not implement it, or speculate about its implementation
# class ArrayReader
# # Compares the sum of arr[l..r] with the sum of arr[x..y]
# # return 1 if sum(arr[l..r]) > sum(arr[x..y])
# # return 0 if sum(arr[l..r]) == sum(arr[x..y])
# # return -1 if sum(arr[l..r]) < sum(arr[x..y])
# def compare_sub(l, r, x, y):
#
# end
#
# # Returns the length of the array
# def length()
#
# end
# end

# @param {ArrayReader} reader
# @return {int}
def get_index(reader)

end
```

PHP Solution:

```
/**
* // This is ArrayReader's API interface.
* // You should not implement it, or speculate about its implementation
* interface ArrayReader {
* // Compares the sum of arr[l..r] with the sum of arr[x..y]
* // return 1 if sum(arr[l..r]) > sum(arr[x..y])
* // return 0 if sum(arr[l..r]) == sum(arr[x..y])
* // return -1 if sum(arr[l..r]) < sum(arr[x..y])
* function compareSub($l, $r, $x, $y) {}
*
* // Returns the length of the array
* function length() {}
* }
*/

class Solution {
/**
* @param ArrayReader $reader
* @return Integer
*/
function getIndex($reader) {

}
}
```

Scala Solution:

```
/**
* // This is ArrayReader's API interface.
* // You should not implement it, or speculate about its implementation
* interface ArrayReader {
* // Compares the sum of arr[l..r] with the sum of arr[x..y]
* // return 1 if sum(arr[l..r]) > sum(arr[x..y])
* // return 0 if sum(arr[l..r]) == sum(arr[x..y])
* // return -1 if sum(arr[l..r]) < sum(arr[x..y])
* def compareSub(l: Int, r: Int, x: Int, y: Int): Int {}
*
* // Returns the length of the array
* def length(): Int {}
* }
*/

object Solution {
def getIndex(reader: ArrayReader): Int = {

}
}
```