Problem 1533: Find the Index of the Large Integer

Problem Information

Difficulty: Medium
Acceptance Rate: 56.65%
Paid Only: Yes
Tags: Array, Binary Search, Interactive

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 * 105` * `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:
```