Maximum Subarray

Submission Detail

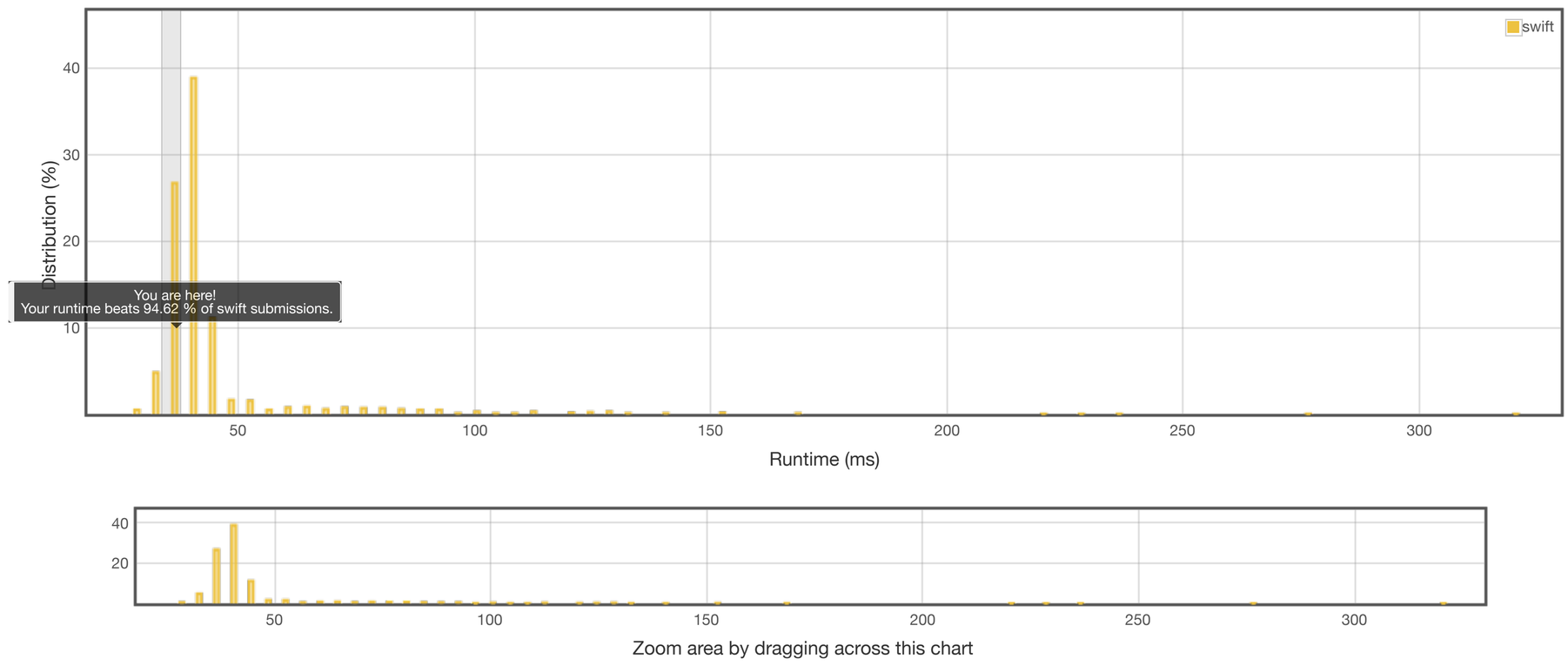
202 / 202 test cases passed.

Runtime: **36 ms**
Memory Usage: **21.4 MB**

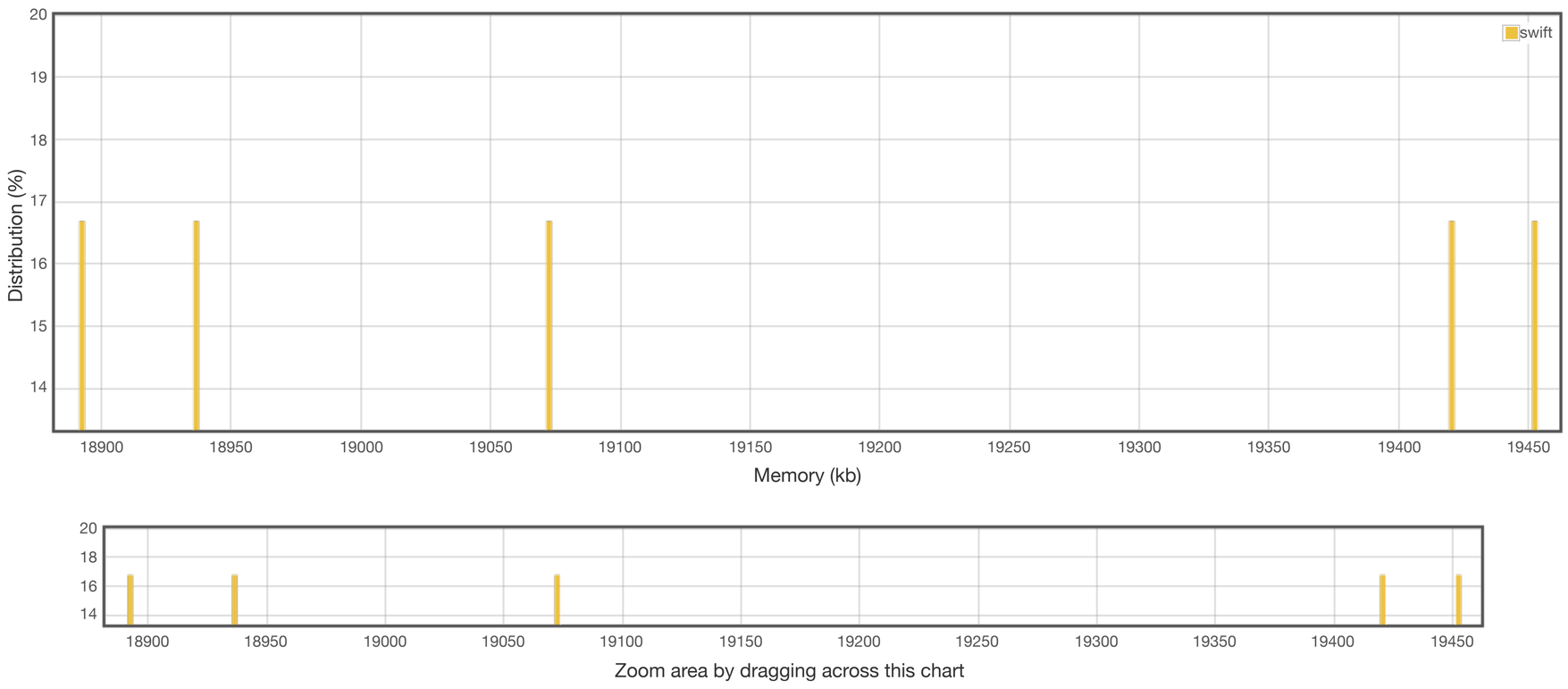
Status: Accepted

Submitted: **0 minutes ago**

Accepted Solutions Runtime Distribution



Accepted Solutions Memory Distribution



Invite friends to challenge **Maximum Subarray**

28

Submitted Code: 0 minutes ago

Language: swift

Edit Code

```swift
func myPrint(_ arg: Any) {
    //print(arg)
}

class Solution {
    func maxSubArray(_ nums: [Int]) -> Int {
        assert(nums.count > 0)

        //return maxSubArrayA(nums)
        return maxSubArrayB(nums)
    }

    // Time: O(n^2)
    // Space: O(1)
    private func maxSubArrayA(_ nums: [Int]) -> Int {
        var maxSum = nums[0]
        myPrint(maxSum)
        var i = 0
        while i < nums.count {
            var j = i + 1
            var newSum = nums[i]
            if newSum > maxSum {
                maxSum = newSum
                myPrint("update max to \(maxSum)")
            }
            myPrint("i: \(i), j: \(j), newSum: \(newSum)")
            while j < nums.count {
                newSum += nums[j]
                myPrint("i: \(i), j: \(j), newSum: \(newSum)")
                if newSum > maxSum {
                    maxSum = newSum
                    myPrint("update max to \(maxSum)")
                }
                j += 1
            }
            myPrint("")
            i += 1
        }
        return maxSum
    }

    // Time: O(n)
    // Space: O(1)
    private func maxSubArrayB(_ nums: [Int]) -> Int {
        var maxSum = nums[0]
        var curSum = nums[0]
        var i = 1
        while i < nums.count {
            curSum = max(nums[i], curSum + nums[i])
            if curSum > maxSum {
                maxSum = curSum
            }
            i += 1
        }

        return maxSum
    }
}
```

Back to problem

Copyright © 2020 LeetCode