Perform String Shifts

Submission Detail

31 / 31 test cases passed.

Runtime: **8 ms**

Memory Usage: **21.4 MB**

Status: Accepted

Submitted: **0 minutes ago**

Accepted Solutions Runtime Distribution

Sorry. We do not have enough accepted submissions to show distribution chart.

Accepted Solutions Memory Distribution

Sorry. We do not have enough accepted submissions to show distribution chart.

Invite friends to challenge **Perform String Shifts**

12

Submitted Code: 0 minutes ago

Language: swift

Edit Code

```swift
class Solution {
    func reduceShifts(_ shift: [[Int]]) -> Int {
        return shift.reduce(0) { (prev, next) -> Int in
            return next[0] == 0 ? prev + next[1] : prev - next[1]
        }
    }

    func shiftLeft(_ s: String, _ positions: Int) -> String {
        return s.string(from: positions, upTo: s.count)! + s.string(from: 0, upTo: positions)!
    }

    func shiftRight(_ s: String, _ positions: Int) -> String {
        return s.string(from: s.count - positions, upTo: s.count)! + s.string(from: 0, upTo: s.count - positions)!
    }

    func stringShift(_ s: String, _ shift: [[Int]]) -> String {
        let shifts = reduceShifts(shift)
        //myPrint(shifts)
        let normalizedShifts = abs(shifts) % s.count
        //myPrint(normalizedShifts)

        if normalizedShifts == 0 {
            return s
        }

        if shifts > 0 {
            return shiftLeft(s,  normalizedShifts)
        } else {
            return shiftRight(s, normalizedShifts)
        }
    }
}

public extension String {
    func index(at i: Int) -> String.Index? {
        guard i < self.count && i >= 0 else {
            return nil
        }
        var index = self.startIndex
        formIndex(&index, offsetBy: i)
        return index
    }

    func string(at i: Int) -> String? {
        guard let index = index(at: i) else {
            return nil
        }

        return String(self[index])
    }

    subscript(i: Int) -> String? {
        return self.string(at: i)
    }

    func string(from i: Int, upTo j: Int) -> String? {
        guard let index1 = index(at: i) else {
            return nil
        }

        if let index2 = index(at: j) {
            return String(self[index1..<index2])
        } else if j == self.count {
            return String(self[index1...])
        } else {
            return nil
        }
    }

    subscript(r: Range<Int>) -> String? {
        return self.string(from: r.lowerBound, upTo: r.upperBound)
    }
}
```

Back to problem