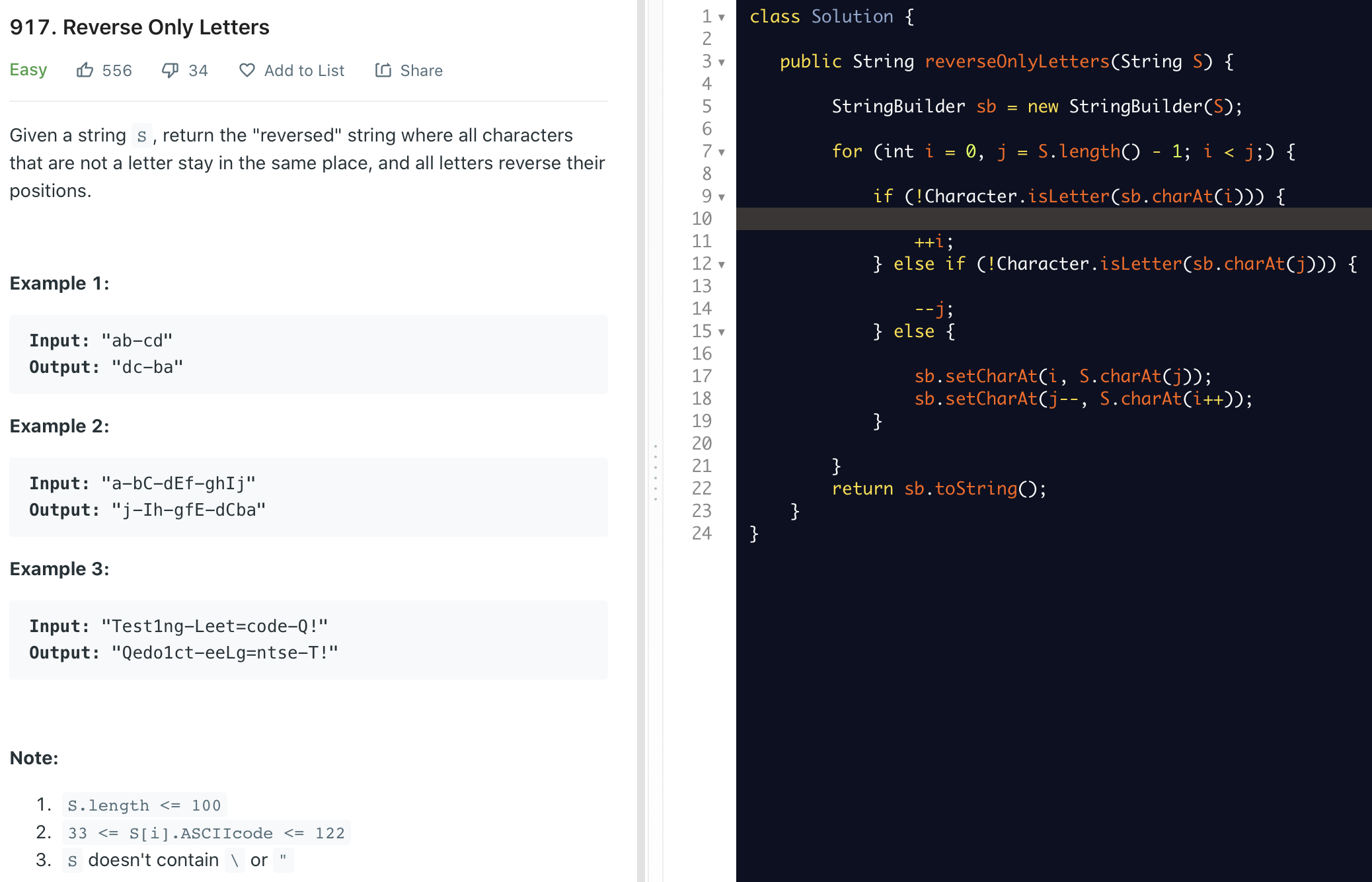


LeetCode 755









class Solution {

private static final int QUICKSORTSHOLD = 50;

private static final int MERGESORTSHOLD = 300;

public int[] sortArray(int[] nums) {

if (nums == null || nums.length < 2) return nums;

if (nums.length < QUICKSORTSHOLD) {

selectionSort(nums);

} else if (nums.length < MERGESORTSHOLD) {

quickSort(nums);

} else {

mergeSort(nums);

}

return nums;

}

// Selection Sort \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

private void selectionSort(int[] nums) {

for (int i = 0; i < nums.length; i++) {

int minIndex = i;

for (int j = i + 1; j < nums.length; j++) {

if (nums[j] < nums[minIndex]) {

minIndex = j;

}

}

exch(nums, i, minIndex);

}

}

// 常规位交换方法

private void exch(int[] nums, int i, int j) {

if (i == j) return;

nums[i] ^= nums[j];

nums[j] ^= nums[i];

nums[i] ^= nums[j];

}

// Quick Sort \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

private void quickSort(int[] nums) {

quickSort(nums, 0, nums.length - 1);

}

private void quickSort(int[] nums, int lo, int hi) {

if (lo >= hi) return;

int pivot = partition(nums, lo, hi);

quickSort(nums, lo, pivot - 1); quickSort(nums, pivot + 1, hi);

}

private int partition(int[] nums, int lo, int hi) {

int q = lo + (int) (Math.random() \* (hi - lo + 1));

exch(nums, lo, q);

int index = lo + 1;

for (int i = lo + 1; i <= hi; i++) {

if (nums[i] < nums[lo]) {

exch(nums, i, index++);

}

}

exch(nums, lo, --index);

return index;

}

// Merge Sort \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

private void mergeSort(int[] nums) { mergeSort(nums, 0, nums.length - 1); }

private void mergeSort(int[] nums, int lo, int hi) {

if (lo >= hi) return;

int mid = (lo + hi) >>> 1; // 其实 >> 就行了。。。。

mergeSort(nums, lo, mid); mergeSort(nums, mid + 1, hi);

merge(nums, lo, mid, mid + 1, hi);

}

private void merge(int[] nums, int preLo, int preHi, int endLo, int endHi) {

if (preLo == endHi) return;

int lo = preLo; int hi = endHi;

int[] newArr = new int[preHi - preLo + 1 + endHi - endLo + 1];

int index = 0;

while (preLo <= preHi && endLo <= endHi) {

newArr[index++] = (nums[preLo] < nums[endLo]) ? nums[preLo++] : nums[endLo++];

}

while (preLo <= preHi) { newArr[index++] = nums[preLo++]; }

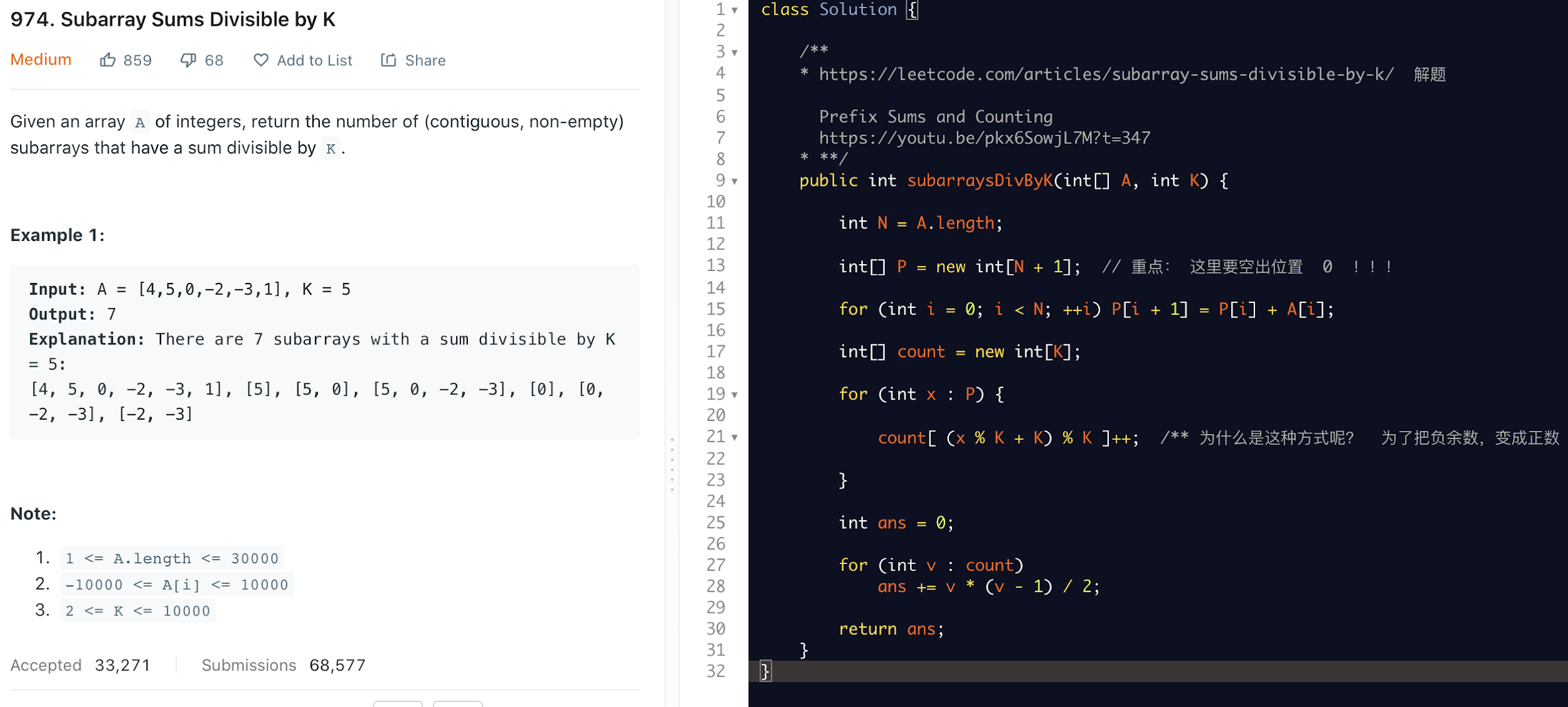
while (endLo <= endHi) { newArr[index++] = nums[endLo++]; }

index = 0;

while (lo <= endHi) { nums[lo++] = newArr[index++]; }

}

}



71



