**Day 02-1929, , , , , .**

**10. Problem:**

**Solution:**

**My Solution:**

**Another Solution: 01**

**Another Solution: 02**

**Another Solution: 03**

**09. Problem:**

**Solution:**

**My Solution:**

**Another Solution: 01**

**Another Solution: 02**

**Another Solution: 03**

**08. Problem:**

**Solution:**

**My Solution:**

**Another Solution: 01**

**Another Solution: 02**

**Another Solution: 03**

**07. Problem:**

**Solution:**

**My Solution:**

**Another Solution: 01**

**Another Solution: 02**

**Another Solution: 03**

**06. Problem: 1929. Concatenation of Array**

Given an integer array nums of length n, you want to create an array ans of length 2n where ans[i] == nums[i] and ans[i + n] == nums[i] for 0 <= i < n (**0-indexed**).

Specifically, ans is the **concatenation** of two nums arrays.

Return *the array*ans.

**Example 1:**

**Input:** nums = [1,2,1]

**Output:** [1,2,1,1,2,1]

**Explanation:** The array ans is formed as follows:

- ans = [nums[0],nums[1],nums[2],nums[0],nums[1],nums[2]]

- ans = [1,2,1,1,2,1]

**Example 2:**

**Input:** nums = [1,3,2,1]

**Output:** [1,3,2,1,1,3,2,1]

**Explanation:** The array ans is formed as follows:

- ans = [nums[0],nums[1],nums[2],nums[3],nums[0],nums[1],nums[2],nums[3]]

- ans = [1,3,2,1,1,3,2,1]

**Constraints:**

n == nums.length

1 <= n <= 1000

1 <= nums[i] <= 1000

**Solution:**

**My Solution:**

public class O6\_ConcatenationOfArray\_1929 {  
 public static void main(String[] args) {  
 Solution\_1929 solution\_1929 = new Solution\_1929();  
 int[] arr = {1, 3, 2, 1};  
 System.*out*.println(Arrays.*toString*(solution\_1929.getConcatenation(arr)));  
 }  
}  
class Solution\_1929 {  
 public int[] getConcatenation(int[] nums) {  
 int[] newAnswer = new int[nums.length \* 2];  
 int givenArrayLength = nums.length;  
 for (int i = 0; i < nums.length; i++) {  
 newAnswer[i] = nums[i];  
 newAnswer[i + givenArrayLength] = nums[i];  
 }  
 return newAnswer;  
 }  
}  
  
// https://leetcode.com/problems/concatenation-of-array/  
  
*/\*\*  
 \* (int i = 0; i < nums.length; i++) --> must be i<nums.length or i<=nums.length-1  
 \*/*

**Another Solution: 01**

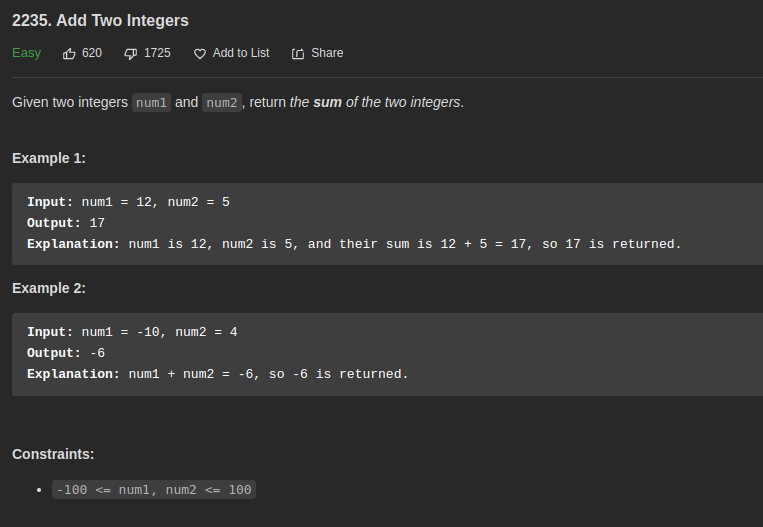
public int[] getConcatenation(int[] nums) {  
 int[] result = new int[nums.length \* 2];  
 for (int i = 0; i < nums.length; i++)  
 result[i + nums.length] = result[i] = nums[i];  
 return result;  
}

**Another Solution: 02**

public int[] getConcatenation(int[] nums) {  
 int[] ans = new int[nums.length \* 2];  
 for (int i = 0; i < nums.length; i++) {  
 ans[i] = nums[i];  
 ans[nums.length + i] = nums[i];  
 }  
 return ans;  
}

**Day 01- 9, 1480, 1108, 1662, 2235.**

Problem: 2235



Solution:

package O1\_easy;

public class O1\_AddTwoIntegers\_2235 {

public static void main(String[] args) {

Solution s = new Solution();

System.out.println(s.sum(5,8));

}

}

class Solution {

public int sum(int num1, int num2) {

return num1+num2;

}

}

// https://leetcode.com/problems/add-two-integers/

Others solution:

class Solution {

public int sum(int num1, int num2) {

if(num2 == 0) return num1;

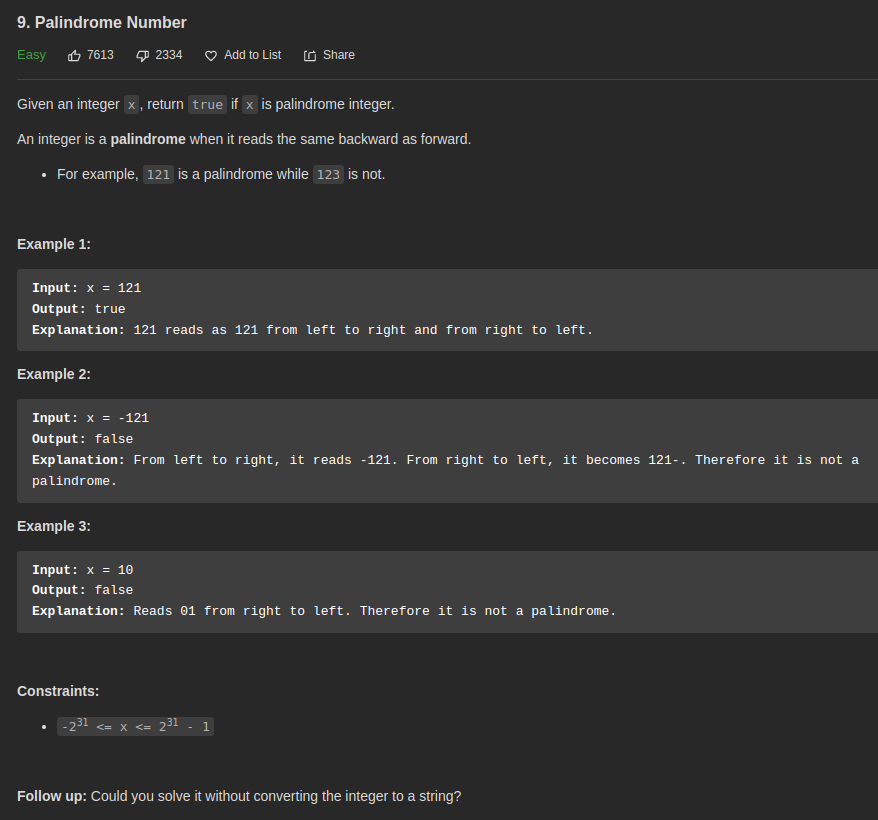
int temp = (num1 & num2) << 1;

return sum(num1 ^ num2,temp);

}

}

Problem: 09



Solution:

package O1\_easy;

public class O2\_Palindrome\_9 {

public static void main(String args[]){

Solution\_9 solution = new Solution\_9();

System.out.println(solution.isPalindrome(121));

System.out.println(solution.isPalindrome(123));

}

}

class Solution\_9 {

public boolean isPalindrome(int x) {

String s = String.valueOf(x);

StringBuffer sb = new StringBuffer(s);

String s1 = String.valueOf(sb.reverse());

if (s.equals(s1)){

return true;

}else {

return false;

}

}

}

// https://leetcode.com/problems/palindrome-number/

Another Solution:

public class Solution {

public boolean isPalindrome(int x) {

if(x < 0) return false;

int y = x;

int res = 0;

while(y != 0) {

res = res \* 10 + y % 10;

y /= 10;

}

return x == res;

}

}

Another Solution:

public boolean isPalindrome(int x) {

int p = x, q = 0;

while (p >= 1) {

q \*= 10;

q += p % 10;

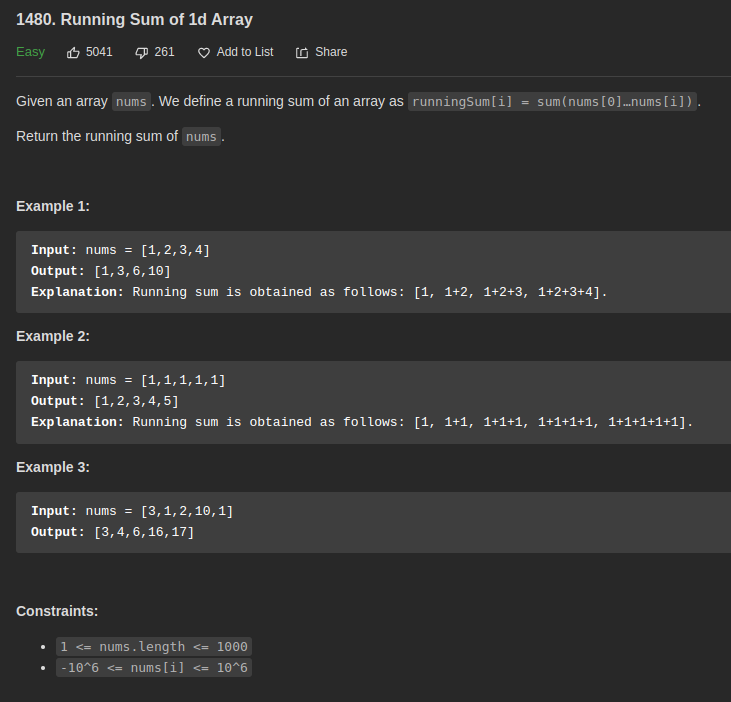
p /= 10;

}

return q == x;

}

Problem: 1480



Solution:

package O1\_easy;

import java.util.Arrays;

public class O3\_RunningSum1dArray\_1480 {

public static void main(String[] args) {

Solution\_1480 solution\_1480 = new Solution\_1480();

int[] nums = {1,1,1,1,1};

System.out.println(Arrays.toString(solution\_1480.runningSum(nums)));

int[] nums2 = {1,2,3,4};

System.out.println(Arrays.toString(solution\_1480.runningSum(nums2)));

}

}

class Solution\_1480 {

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

int[] result = new int[nums.length];

int sum = 0;

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

sum = sum + nums[i];

result[i] = sum;

}

return result;

}

}

/\*\*

\* {1, 2, 3, 4}

\*

\* Iteration: 1

\* sum = sum + num[i] | sum = 0 + 1 = 1

\* result[i] = sum | result[0] = 1

\*

\* Iteration: 2

\* sum = 1 + 2 = 3 , result[1] = 3

\*

\* Iteration: 3

\* sum = 3 + 3 = 6 , result[2] = 6

\*

\* Iteration: 4

\* sum = 3 + 6 = 10 , result[3] = 10

\*/

// https://leetcode.com/problems/running-sum-of-1d-array/

Another solution

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

// modify the input array, adding n[i] with n[i-1]

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

nums[i] += nums[i-1];

}

// return the modified array

return nums;

}

Another solution

class Solution {

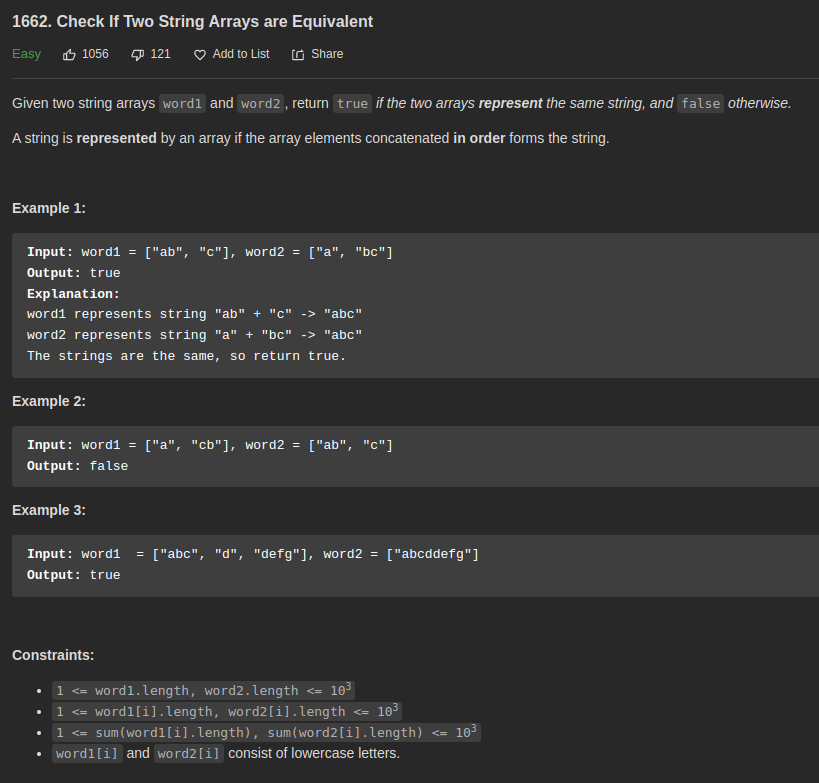
public int[] runningSum(int[] nums) {

return IntStream.range(0,nums.length).map(i->i==0?nums[i]:(nums[i]+=nums[i-1])).toArray();

}

}

Problem: 1662

Solution:

package O1\_easy;

public class O4\_StringEquivalent\_1662 {

public static void main(String[] args) {

Solution\_1662 solution\_1662 = new Solution\_1662();

String[] word1 = {"a", "cb"};

String[] word2 = {"ab", "c"};

System.out.println(solution\_1662.arrayStringsAreEqual(word1, word2));

}

}

class Solution\_1662 {

public boolean arrayStringsAreEqual(String[] word1, String[] word2) {

String r1 = "";

for (int i = 0; i<word1.length ; i++){

r1 += word1[i];

}

System.out.println(r1);

String r2 = "";

for (int i = 0; i<word2.length ; i++){

r2 += word2[i];

}

System.out.println(r2);

if (r1.equals(r2)){

return true;

}else {

return false;

}

}

}

// https://leetcode.com/problems/check-if-two-string-arrays-are-equivalent/

Another Solution:

class Solution {

public boolean arrayStringsAreEqual(String[] word1, String[] word2) {

return String.join("", word1).equals(String.join("", word2));

}

}

class Solution {

public boolean arrayStringsAreEqual(String[] word1, String[] word2) {

StringBuilder sb1 = new StringBuilder(), sb2 = new StringBuilder();

for(String word: word1)

sb1.append(word);

for(String word: word2)

sb2.append(word);

return sb1.toString().equals(sb2.toString());

}

}

Another Solution:

class Solution {

public boolean arrayStringsAreEqual(String[] word1, String[] word2) {

int idx1 = 0, idx2 = 0, arrIdx1 = 0, arrIdx2 = 0;

while (arrIdx1 < word1.length && arrIdx2 < word2.length) {

if (word1[arrIdx1].charAt(idx1) != word2[arrIdx2].charAt(idx2)) return false;

if (idx1 == word1[arrIdx1].length() - 1) {

idx1 = 0;

arrIdx1++;

} else idx1++;

if (idx2 == word2[arrIdx2].length() - 1) {

idx2 = 0;

arrIdx2++;

} else idx2++;

}

return arrIdx1 == word1.length && arrIdx2 == word2.length;

}

}

Same but shorter:

class Solution {

public boolean arrayStringsAreEqual(String[] word1, String[] word2) {

int idx1 = 0, idx2 = 0, arrIdx1 = 0, arrIdx2 = 0;

while (arrIdx1 < word1.length && arrIdx2 < word2.length) {

if (word1[arrIdx1].charAt(idx1) != word2[arrIdx2].charAt(idx2)) return false;

idx1 = (++idx1) % word1[arrIdx1].length();

idx2 = (++idx2) % word2[arrIdx2].length();

if (idx1 == 0) arrIdx1++;

if (idx2 == 0) arrIdx2++;

}

return arrIdx1 == word1.length && arrIdx2 == word2.length;

}

}

Problem: 1108



Solution:

package O1\_easy;

public class O5\_DefiningIpAddress\_1108 {

public static void main(String[] args) {

Solution\_1108 solution\_1108 = new Solution\_1108();

System.out.println(solution\_1108.defangIPaddr("1.1.1.1"));

}

}

class Solution\_1108 {

public String defangIPaddr(String address) {

return address.replace(".", "[.]");

}

}

Another solution:

class Solution {

public String defangIPaddr(String address) {

StringBuilder str = new StringBuilder();

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

if (address.charAt(i) == '.'){

str.append("[.]");

} else {

str.append(address.charAt(i));

}

}

return str.toString();

}

}

Another Solution:

public String defangIPaddr(String address) {

return address.replaceAll("\\.", "[.]");

}

**Day 00-1929, , , , , .**

**5. Problem:**

**Solution:**

**My Solution:**

**Another Solution: 01**

**Another Solution: 02**

**Another Solution: 03**

**04. Problem:**

**Solution:**

**My Solution:**

**Another Solution: 01**

**Another Solution: 02**

**Another Solution: 03**

**03. Problem:**

**Solution:**

**My Solution:**

**Another Solution: 01**

**Another Solution: 02**

**Another Solution: 03**

**02. Problem:**

**Solution:**

**My Solution:**

**Another Solution: 01**

**Another Solution: 02**

**Another Solution: 03**

**01. Problem:**

**Solution:**

**My Solution:**

**Another Solution: 01**

**Another Solution: 02**

**Another Solution: 03**

