**Arrays Solution**

A screenshot of a video link

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

1. Build Array from permutation

Given a **zero-based permutation** nums (**0-indexed**), build an array ans of the **same length** where ans[i] = nums[nums[i]] for each 0 <= i < nums.length and return it.

**Input:** nums = [0,2,1,5,3,4]

**Output:** [0,1,2,4,5,3]

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

ans = [nums[nums[0]], nums[nums[1]], nums[nums[2]], nums[nums[3]], nums[nums[4]], nums[nums[5]]]

= [nums[0], nums[2], nums[1], nums[5], nums[3], nums[4]]

= [0,1,2,4,5,3]

class Solution {

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

        int len=nums.length;

        int [] ans=new int[len];

        for(int i=0;i<len;i++){

            ans[i]=nums[nums[i]];

        }

        return ans;

    }

}

1. Concatenation of Arrays

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

**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]

class Solution {

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

        int len = nums.length;

        int[] ans = new int[2 \* len];

        for (int i = 0; i < len; i++) {

            ans[i] = nums[i]; // Copy first half

            ans[i + len] = nums[i]; // Copy second half

        }

        return ans;

    }

}

1. Running sum of 1D Array

**Input:** nums = [1,2,3,4]

**Output:** [1,3,6,10]

**Explanation:** Running sum is obtained as follows: [1, 1+2, 1+2+3, 1+2+3+4].

class Solution {

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

        int len = nums.length;

        int[] ans = new int[len];

        ans[0] = nums[0];

        for (int i = 1; i < len; i++) {

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

        }

        return ans;

    }

}

1. Richest Customer Wealth

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

**Output:** 6

**Explanation:**

1st customer has wealth = 1 + 2 + 3 = 6

2nd customer has wealth = 3 + 2 + 1 = 6

Both customers are considered the richest with a wealth of 6 each, so return 6

class Solution {

    public int maximumWealth(int[][] accounts) {

        int len=accounts.length;

        int max=1;

        for(int i=0;i<len;i++){

            int sum=0;

            for(int j=0;j<accounts[i].length;j++){

                sum+=accounts[i][j];

            }

            if(sum>max){

                max=sum;

            }

        }

        return max;

    }

}

1. Shuffle the array

**Input:** nums = [2,5,1,3,4,7], n = 3

**Output:** [2,3,5,4,1,7]

**Explanation:** Since x1=2, x2=5, x3=1, y1=3, y2=4, y3=7 then the answer is [2,3,5,4,1,7].

class Solution {

    public int[] shuffle(int[] nums, int n) {

        int[] ans = new int[n << 1];

        for (int i = 0, j = 0; i < n; ++i) {

            ans[j++] = nums[i];

            ans[j++] = nums[i + n];

        }

        return ans;

    }

}

1. Kids with greater number of candies

**Input:** candies = [2,3,5,1,3], extraCandies = 3

**Output:** [true,true,true,false,true]

**Explanation:** If you give all extraCandies to:

- Kid 1, they will have 2 + 3 = 5 candies, which is the greatest among the kids.

- Kid 2, they will have 3 + 3 = 6 candies, which is the greatest among the kids.

- Kid 3, they will have 5 + 3 = 8 candies, which is the greatest among the kids.

- Kid 4, they will have 1 + 3 = 4 candies, which is not the greatest among the kids.

- Kid 5, they will have 3 + 3 = 6 candies, which is the greatest among the kids.

class Solution {

    public List<Boolean> kidsWithCandies(int[] candies, int extraCandies) {

        List<Boolean> result = new ArrayList<>();

        int max = 0;

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

            if (candies[i] > max) {

                max = candies[i];

            }

        }

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

            if (candies[i] + extraCandies >= max) {

                result.add(true);

            }

            else {

                result.add(false);

            }

        }

        return result;

    }

}

1. Number of good pair

Given an array of integers nums, return the number of ***good pairs***.

A pair (i, j) is called good if nums[i] == nums[j] and i < j.

**Example 1:**

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

**Output:** 4

**Explanation:** There are 4 good pairs (0,3), (0,4), (3,4), (2,5) 0-indexed.

class Solution {

    public int numIdenticalPairs(int[] nums) {

        int count=0;

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

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

                if(nums[i]==nums[j] && i<j){

                    count++;

                }

            }

        }

        return count;

    }

}

1. How many numbers are smaller than current number

Given the array nums, for each nums[i] find out how many numbers in the array are smaller than it. That is, for each nums[i] you have to count the number of valid j's such that j != i **and** nums[j] < nums[i].

Return the answer in an array.

class Solution {

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

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

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

            int count=0;

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

                if(j!=i && nums[j]<nums[i]){

                    count++;

                }

            }

            ans[i]=count;

        }

        return ans;

    }

}

1. Create target in given order
2. **Input:** nums = [0,1,2,3,4], index = [0,1,2,2,1]
3. **Output:** [0,4,1,3,2]
4. **Explanation:**
5. nums index target
6. 0 0 [0]
7. 1 1 [0,1]
8. 2 2 [0,1,2]
9. 3 2 [0,1,3,2]
10. 4 1 [0,4,1,3,2]

class Solution {

    public int[] createTargetArray(int[] nums, int[] index) {

        int n = nums.length;

        ArrayList<Integer>List = new ArrayList<>(n);

        for (int i=0; i<n ; i++){

            List.add(index[i],nums[i]);

        }

        for (int i=0; i<n; i++){

            nums[i] = List.get(i);

        }

        return nums;

    }

}

10. The sentence is pangram

**Input:** sentence = "thequickbrownfoxjumpsoverthelazydog"

**Output:** true

**Explanation:** sentence contains at least one of every letter of the English alphabet.

class Solution {

    public boolean checkIfPangram(String sentence) {

        if(sentence.length()<26){

            return false;

        }

        boolean[] present = new boolean[26];

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

            int letter = sentence.charAt(i) -'a';

            present[letter] = true;

        }

        for(int i=0;i<26;i++){

            if(!present[i]){

                return false;

            }

        }

        return true;

    }

    }