Q1. Equilibrium index of an array

class EquilibriumIndex {

int equilibrium(int arr[], int n)

{

int i, j;

int leftsum, rightsum;

for (i = 0; i < n; ++i) {

leftsum = 0;

for (j = 0; j < i; j++) leftsum += arr[j];

rightsum = 0;

for (j = i + 1; j < n; j++) rightsum += arr[j];

if (leftsum == rightsum) return i;

}

return -1;

}

public static void main(String[] args)

{

EquilibriumIndex equi = new EquilibriumIndex();

int arr[] = { -7, 1, 5, 2, -4, 3, 0 };

int arr\_size = arr.length;

System.out.println(equi.equilibrium(arr, arr\_size));

}

}

Q2. Special Index

class GFG {

static int cntIndexesToMakeBalance(int arr[], int n)

{

if (n == 1) return n;

if (n == 2) return 0;

int sumEven = 0, sumOdd = 0 , currOdd = 0 ;

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

if (i % 2 == 0) sumEven += arr[i];

else sumOdd += arr[i]; }

int currEven = arr[0] , res = 0 , newEvenSum = 0 , newOddSum = 0;

for (int i = 1; i < n - 1; i++){

if (i % 2 != 0){

currOdd += arr[i];

newEvenSum = currEven + sumOdd- currOdd;

newOddSum = currOdd + sumEven - currEven - arr[i]; }

else{

currEven += arr[i];

newOddSum = currOdd + sumEven- currEven;

newEvenSum = currEven + sumOdd - currOdd - arr[i];

}

if (newEvenSum == newOddSum) res++; }

if (sumOdd == sumEven - arr[0]) res++;

if (n % 2 == 1){

if (sumOdd == sumEven - arr[n - 1]) res++; }

else if (sumEven == sumOdd - arr[n - 1]) res++;

return res;}

public static void main (String[] args)

{

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

int n = arr.length;

System.out.println(cntIndexesToMakeBalance(arr, n));

}

}

Q3. Pick from both sides!

import java.util.Arrays;

import java.util.Scanner;

class GFG {

public static int maxPointCount(int arr[], int K, int size)

{

int curr\_points = 0 , max\_points = 0;

for (int i = 0; i < K; i++) curr\_points += arr[i];

max\_points = curr\_points;

int j = size - 1;

for (int i = K - 1; i >= 0; i--) {

curr\_points = curr\_points + arr[j] - arr[i];

max\_points = Math.max(curr\_points, max\_points);

j--;}

return max\_points;

}

public static void main(String args[])

{

int[] arr = { 8, 4, 4, 8, 12, 3, 2, 9 };

int K = 3;

int n = arr.length;

System.out.print(maxPointCount(arr, K, n));

}

}

Q4. Range Sum Query

import java.util.\*;

import java.lang.\*;

class GFG {

public static void preCompute(int arr[], int n, int pre[])

{ pre[0] = arr[0];

for (int i = 1; i < n; i++)

pre[i] = arr[i] + pre[i - 1];

}

public static int rangeSum(int i, int j, int pre[])

{ if (i == 0)

return pre[j];

return pre[j] - pre[i - 1];

}

public static void main(String[] args)

{

int arr[] = { 1, 2, 3, 4, 5 };

int n = arr.length;

int pre[] = new int[n];

preCompute(arr, n, pre);

System.out.println(rangeSum(1, 3, pre));

System.out.println(rangeSum(2, 4, pre));

}

}

Q5. Time to equality

import java.util.\*;

class GfG {

static int minCost(int A[], int n)

{

int cost = 0;

Arrays.sort(A);

int K = A[n / 2];

for (int i = 0; i < n; ++i)

cost += Math.abs(A[i] - K);

if (n % 2 == 0) {

int tempCost = 0;

K = A[(n / 2) - 1];

for (int i = 0; i < n; ++i)

tempCost += Math.abs(A[i] - K);

cost = Math.min(cost, tempCost);

}

return cost;

}

public static void main(String[] args)

{

int A[] = { 1, 6, 7, 10 };

int n = A.length;

System.out.println(minCost(A, n));

}

}

Q6. Product array puzzle

class ProductArray {

void productArray(int arr[], int n)

{

if (n == 1) {

System.out.print(0);

return;

}

int left[] = new int[n];

int right[] = new int[n];

int prod[] = new int[n];

int i, j;

left[0] = 1;

right[n - 1] = 1;

for (i = 1; i < n; i++)

left[i] = arr[i - 1] \* left[i - 1];

for (j = n - 2; j >= 0; j--)

right[j] = arr[j + 1] \* right[j + 1];

for (i = 0; i < n; i++)

prod[i] = left[i] \* right[i];

for (i = 0; i < n; i++)

System.out.print(prod[i] + " ");

return;

}

public static void main(String[] args)

{

ProductArray pa = new ProductArray();

int arr[] = { 10, 3, 5, 6, 2 };

int n = arr.length;

System.out.println("The product array is : ");

pa.productArray(arr, n);

}

}