Tony Zhong, S3C7

#has23

public boolean has23(int[] nums) {

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

if (nums[i] == 2 || nums[i] == 3)

return true;

}

return false;

}

#no23

public boolean no23(int[] nums) {

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

if (nums[i] == 2 || nums[i] == 3)

return false;

}

return true;

}

#makelast

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

int[] result = new int[2 \* nums.length];

result[result.length - 1] = nums[nums.length - 1];

return result;

}

#double23

public boolean double23(int[] nums) {

int twoCount = 0;

int threeCount = 0;

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

if (nums[i] == 2)

twoCount++;

if (nums[i] == 3)

threeCount++;

}

return twoCount == 2 || threeCount == 2;

}

#fix23

public int[] fix23(int[] nums) {

for (int i = 0; i < nums.length - 1; i++) {

if (nums[i] == 2 && nums[i + 1] == 3)

nums[i + 1] = 0;

}

return nums;

}

#start1

public int start1(int[] a, int[] b) {

int count = 0;

if (a.length > 0 && a[0] == 1)

count++;

if (b.length > 0 && b[0] == 1)

count++;

return count;

}

#biggertwo

public int[] biggerTwo(int[] a, int[] b) {

if (a[0] + a[1] < b[0] + b[1])

return b;

return a;

}

#makemiddle

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

int[] result = { nums[nums.length / 2 - 1], nums[nums.length / 2] };

return result;

}

#plustwo

public int[] plusTwo(int[] a, int[] b) {

int[] result = { a[0], a[1], b[0], b[1] };

return result;

}

#swapEnds

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

int first = nums[0];

int last = nums[nums.length - 1];

nums[0] = last;

nums[nums.length - 1] = first;

return nums;

}

#midthree

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

int mid = nums.length / 2;

int[] result = { nums[mid - 1], nums[mid], nums[mid + 1] };

return result;

}

#maxTriple

public int maxTriple(int[] nums) {

int mid = nums.length / 2;

int end = nums.length - 1;

int max = nums[0];

if (nums[mid] > max) max = nums[mid];

if (nums[end] > max) max = nums[end];

return max;

}

#frontPiece

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

if (nums.length <= 1) return nums;

int[] result = { nums[0], nums[1] };

return result;

}

#unlucky1

public boolean unlucky1(int[] nums) {

int len = nums.length;

if (len <= 1) return false;

for (int i = 0; i <= 1; i++) {

if (nums[i] == 1 && nums[i + 1] == 3)

return true;

if (len < 3) break;

}

return nums[len - 2] == 1 && nums[len - 1] == 3;

}

#make2

public int[] make2(int[] a, int[] b) {

int[] res = new int[2];

if (a.length == 0) {

res[0] = b[0];

res[1] = b[1];

} else if (a.length == 1) {

res[0] = a[0];

res[1] = b[0];

} else {

res[0] = a[0];

res[1] = a[1];

}

return res;

}

#front11

public int[] front11(int[] a, int[] b) {

int[] one = new int[1];

int[] two = new int[2];

if (a.length == 0 && b.length == 0) {

return a;

}

if (a.length >= 1 && b.length == 0) {

one[0] = a[0];

return one;

}

if (a.length >= 1 && b.length >= 1) {

two[0] = a[0];

two[1] = b[0];

return two;

}

if (a.length == 0 && b.length >= 1) {

one[0] = b[0];

return one;

}

return a;

}