Question: Longest Nice substring

class Solution {

public String longestNiceSubstring(String s) {

int n = s.length();

int max = 0;

String res = "";

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

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

StringBuilder sb = new StringBuilder();

for(int k = i; k <= j; k++){

sb.append(s.charAt(k));

}

if(checkIfNice(sb)){

if(j - i + 1 > max){

max = j - i + 1;

res = sb.toString();

}

}

}

}

return res;

}

public boolean checkIfNice(StringBuilder s){

HashSet<Character> set = new HashSet<>();

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

set.add(s.charAt(i));

}

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

if(s.charAt(i) >= 65 && s.charAt(i) <= 90){

if(!set.contains((char)(s.charAt(i) + 32))){

return false;

}

}

if(s.charAt(i) >= 97 && s.charAt(i) <= 122){

if(!set.contains((char)(s.charAt(i) - 32))){

return false;

}

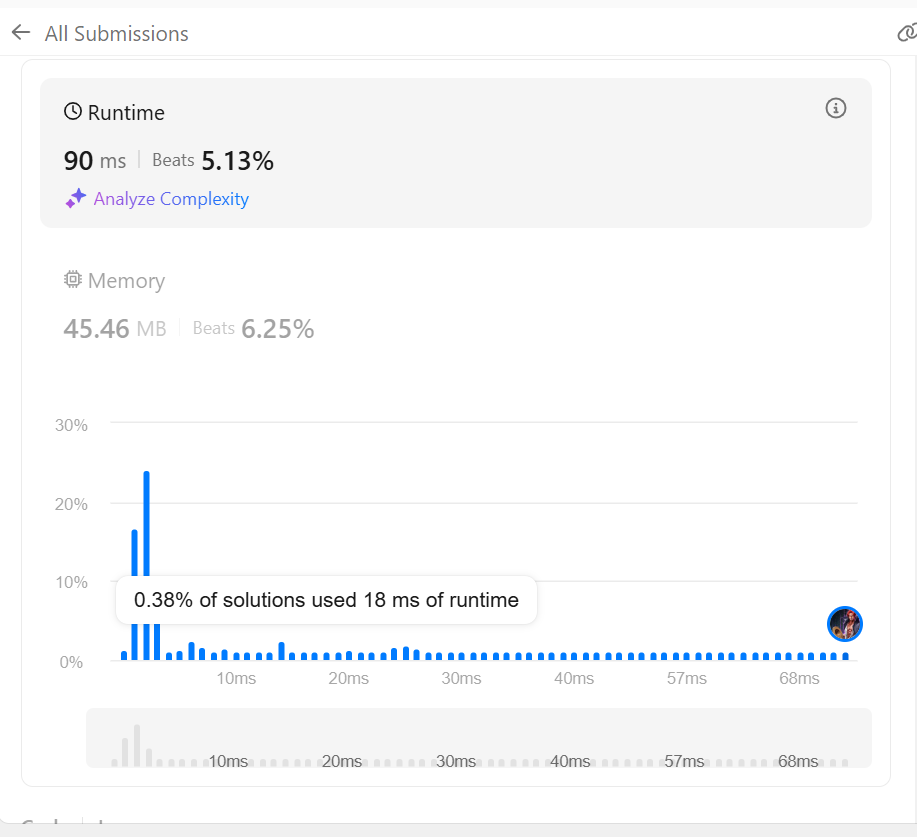
}

}

return true;

}

}



Question: Maximum subarray

class Solution {

public int maxSubArray(int[] nums) {

int res = 0;

int currSum = 0;

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

currSum += nums[i];

if (currSum < 0) {

currSum = 0;

} else {

res = Math.max(currSum, res);

}

}

if (res == 0) {

res = Integer.MIN\_VALUE;

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

res = Math.max(res, nums[i]);

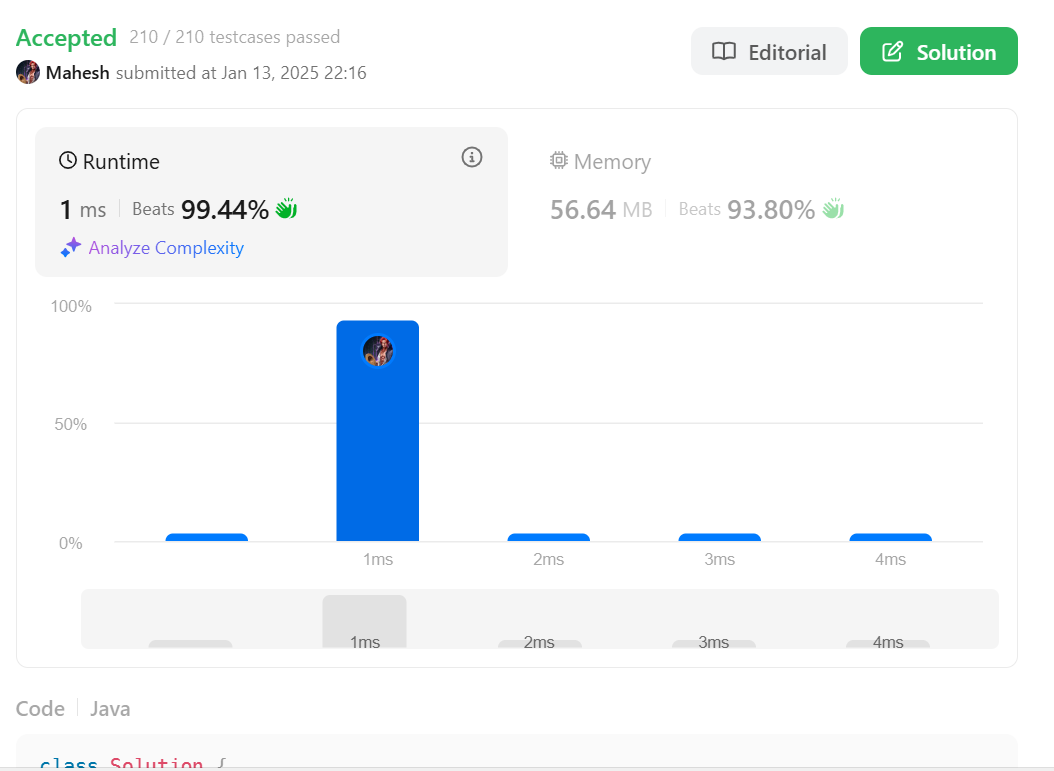
}

}

return res;

}

}



Question : Search a 2D matrix II  
public class Solution {

public boolean searchMatrix(int[][] matrix, int target) {

if(matrix == null || matrix.length < 1 || matrix[0].length <1) {

return false;

}

// Starting from top right corner

int col = matrix[0].length-1;

int row = 0;

while(col >= 0 && row <= matrix.length-1) {

if(target == matrix[row][col]) {

return true;

} else if(target < matrix[row][col]) {

col--;

} else if(target > matrix[row][col]) {

row++;

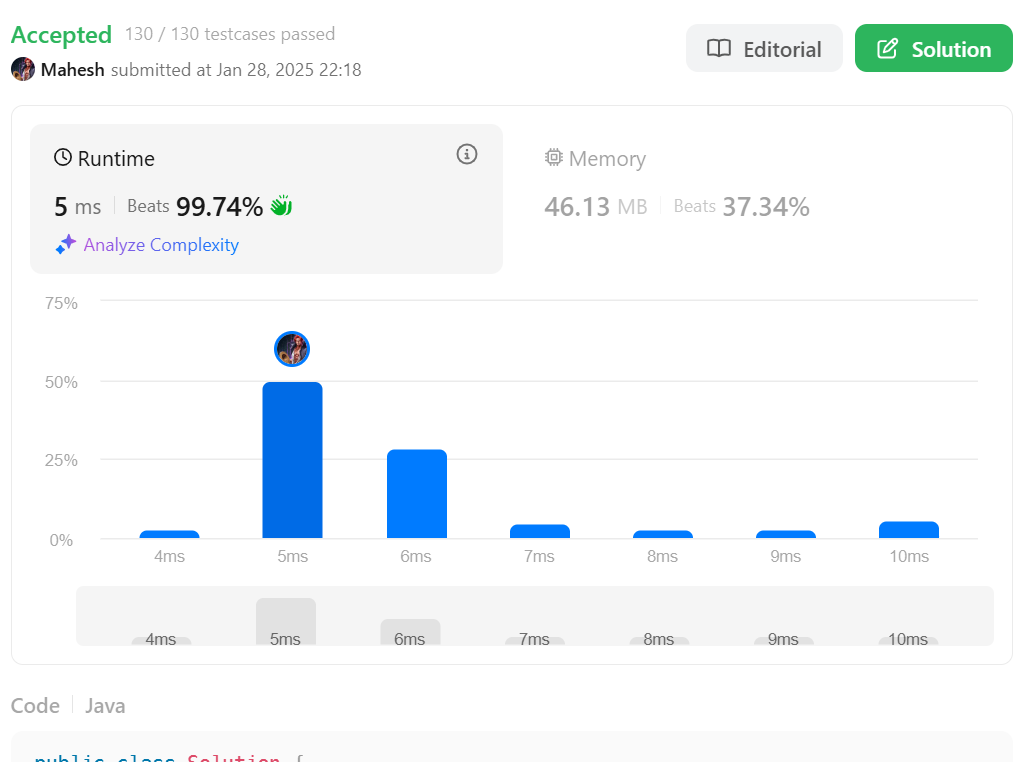
}

}

return false;

}

}



Question: Merge Sorted array

class Solution {

public void merge(int[] nums1, int m, int[] nums2, int n) {

int end = nums1.length - 1;

int i = m - 1;

int j = n - 1;

while (end >= 0 && i >= 0 && j >= 0) {

if (nums1[i] <= nums2[j]) {

nums1[end] = nums2[j];

j--;

end--;

} else if (nums1[i] > nums2[j]) {

nums1[end] = nums1[i];

i--;

end--;

}

}

while (j >= 0) {

nums1[end] = nums2[j];

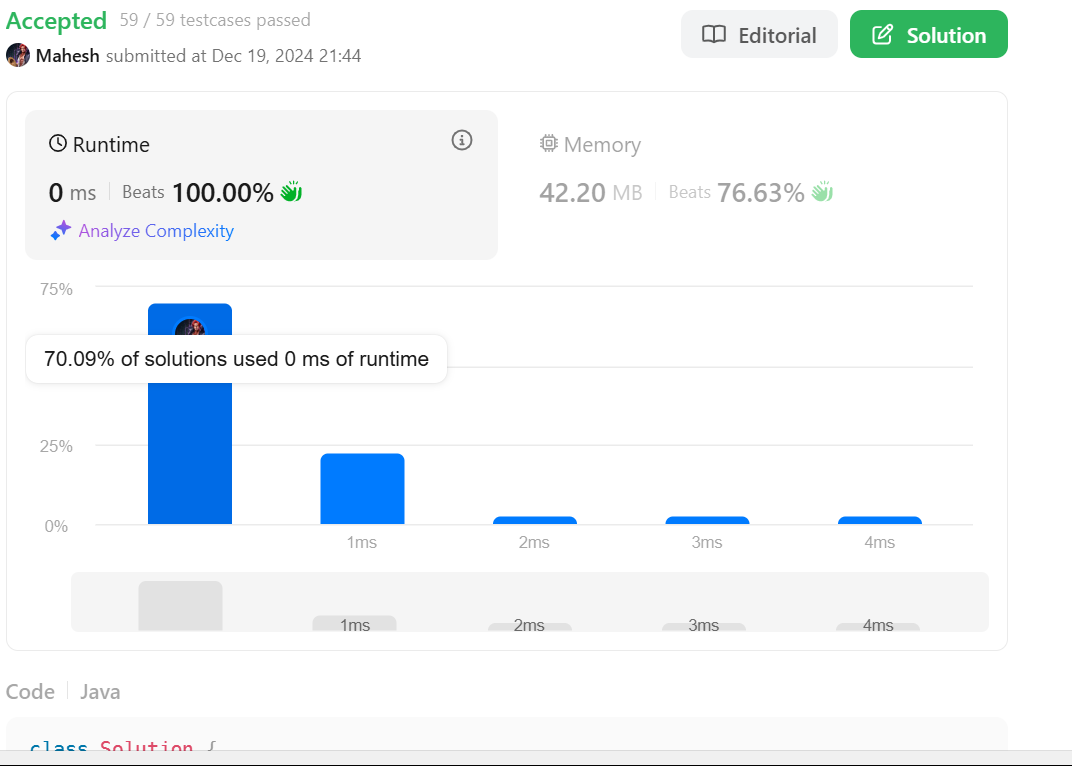
j--;

end--;

}

}

}



Question : The Skyline problem

class TopNode {

int x;

int h;

TopNode next;

TopNode() {

}

TopNode(int x, int h) {

this.x = x;

this.h = h;

}

void insert(TopNode n) {

n.next = next;

next = n;

}

}

class Solution {

static final int LEFT=0, RIGHT=1, HEIGHT=2;

public List<List<Integer>> getSkyline(int[][] buildings) {

TopNode head = new TopNode(0,0);

head.insert(new TopNode(Integer.MAX\_VALUE, 0));

TopNode start = head;

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

int[] b = buildings[i];

int bL = buildings[i][LEFT];

int bR = buildings[i][RIGHT];

int bH = buildings[i][HEIGHT];

//System.out.println(Arrays.toString(buildings[i]));

while (bL >= start.next.x) { start = start.next; }

//System.out.println(start.toString());

for (TopNode t = start ; bR > t.x; t = t.next) {

//System.out.println(head.toString());

if (bH <= t.h) {

continue;

}

TopNode stop = t;

while (stop.next != null && stop.next.x < bR && stop.next.h <= bH ) {

stop = stop.next;

}

if (bL <= t.x) {

if (bR >= stop.next.x) {

t.next = stop.next;

t.h = bH;

}

else if (t == stop) {

t.insert(new TopNode(bR,t.h));

t.h = bH;

break;

}

else {

stop.x = bR;

t.h = bH;

t.next = stop;

break;

}

}

else {

if (bR >= stop.next.x) {

if (t == stop) {

t.insert(new TopNode(bL, bH));

}

else {

t.next = stop;

stop.x = bL;

stop.h = bH;

}

break;

}

else if (t == stop) {

t.insert(new TopNode(bL,bH));

t.next.insert(new TopNode(bR,t.h));

break;

}

else {

t.next = stop;

t.insert(new TopNode(bL,bH));

stop.x = bR;

break;

}

}

t = stop;

}

}

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

if (head.h == 0)

head = head.next;

while (head != null) {

int height = head.h;

skyline.add(List.of(head.x, height));

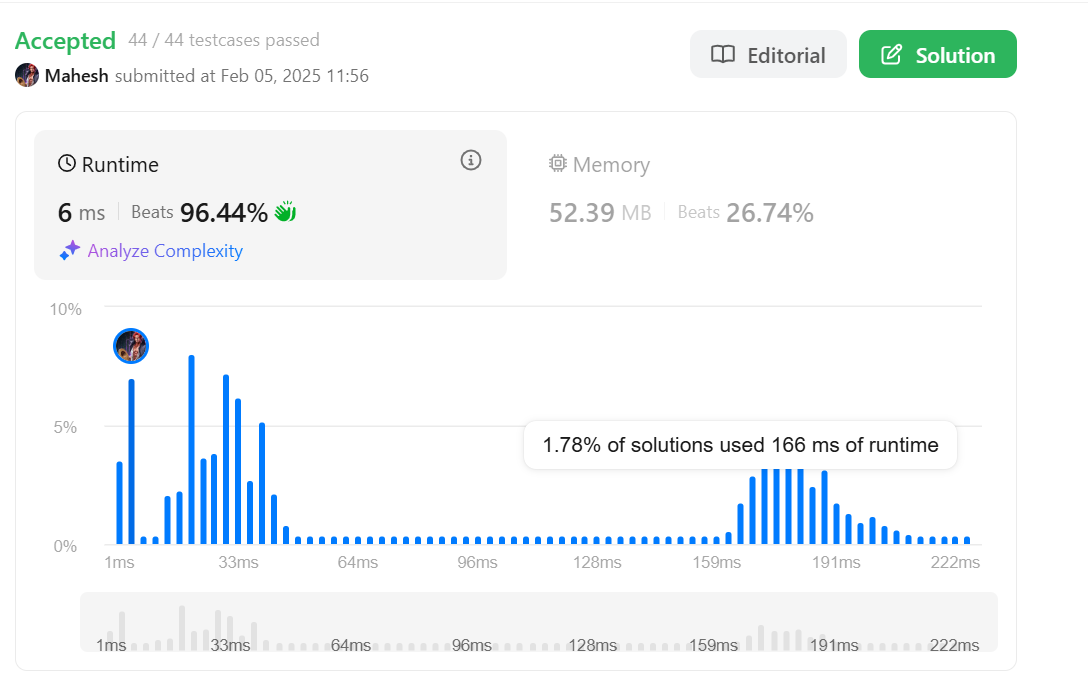
while ( (head = head.next) != null && head.h == height) {}

}

return skyline;

}

}



Question: Reverse Bits

public class Solution {

// you need treat n as an unsigned value

public int reverseBits(int n) {

int result=0;

for(int i=0;i<32;i++){

result<<=1;

result|=(n&1);

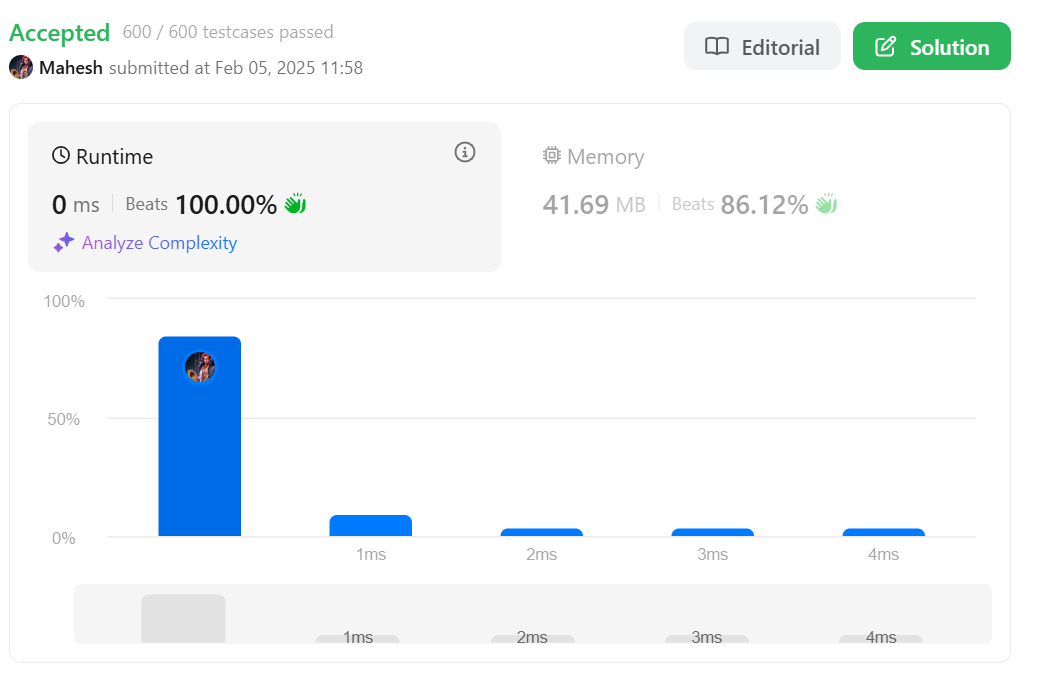
n>>>=1;

}

return result;

}

}

  
  
Question: Beautiful Array

class Solution {

public int[] beautifulArray(int n) {

List<Integer> ordered = IntStream.rangeClosed(1, n).boxed().collect(Collectors.toList());

return arrangeBeautifully(ordered).stream().mapToInt(it -> it).toArray();

}

private List<Integer> arrangeBeautifully(List<Integer> numbers) {

// two numbers or less are already arranged

if (numbers.size() <= 2) {

return numbers;

}

// split into two lists by taking every other number. all odds indexes go left, all even indexes go right

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

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

for (int i = 0; i < numbers.size(); ++i) {

if (i % 2 == 0) {

left.add(numbers.get(i));

} else {

right.add(numbers.get(i));

}

}

// arrange both halves recursively

List<Integer> arrangedLeft = arrangeBeautifully(left);

List<Integer> arrangedRight = arrangeBeautifully(right);

// combine arranged halves into one list

arrangedLeft.addAll(arrangedRight);

return arrangedLeft;

}

}

