**Assignment- 10**

**Student Name: Avin Kumar UID: 22BCS10600 Section/Group: IOT-614 B Semester: 6th**

**Subject Name: Advanced Programming Subject Code:** **22CSP-351**

**Lab-2**

[**118. Pascal's Triangle**](https://leetcode.com/problems/pascals-triangle/):

**Code:**

class Solution {

  public List<List<Integer>> generate(int numRows) {

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

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

      Integer[] temp = new Integer[i + 1];

      Arrays.fill(temp, 1);

      ans.add(Arrays.asList(temp));

    }

    for (int i = 2; i < numRows; ++i)

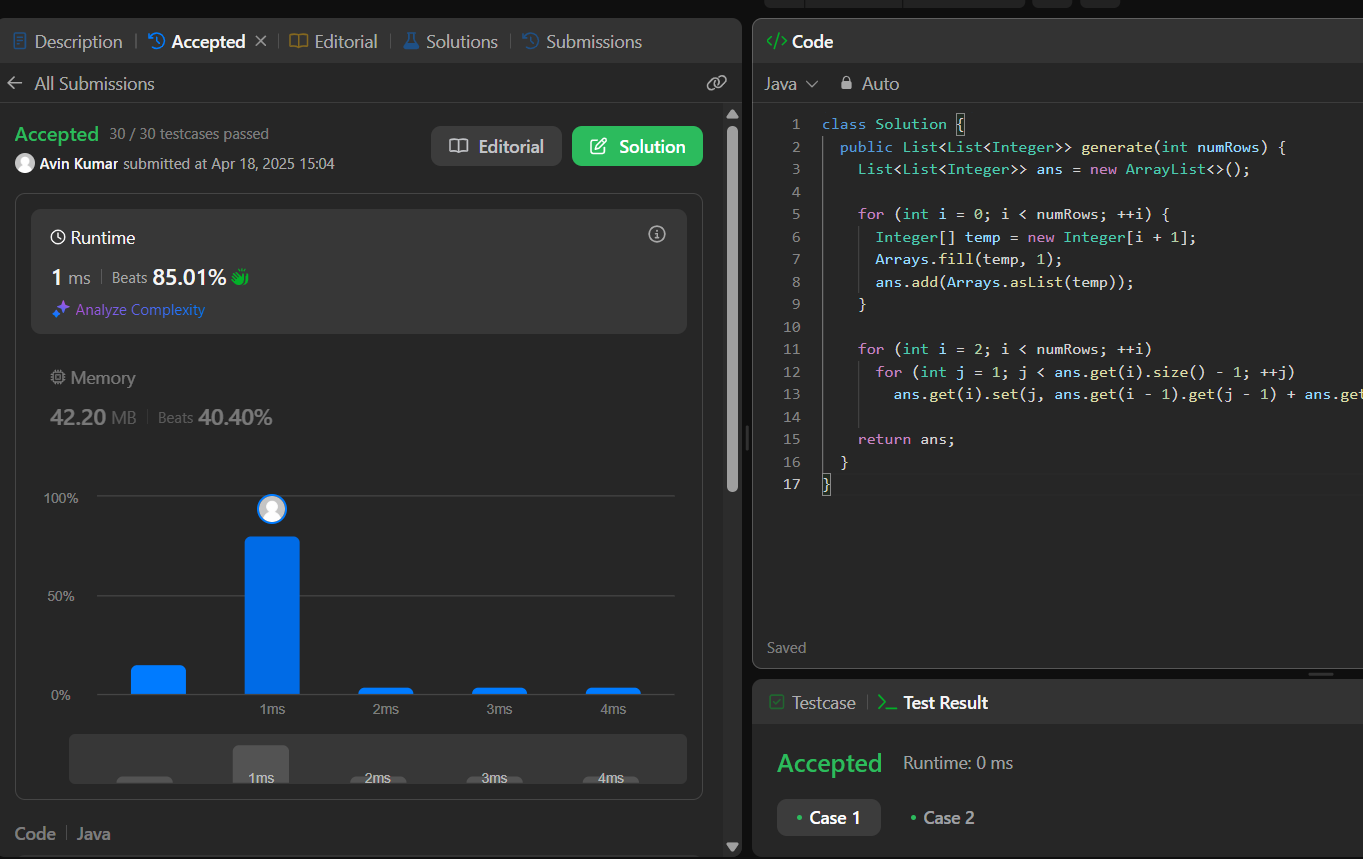
      for (int j = 1; j < ans.get(i).size() - 1; ++j)

        ans.get(i).set(j, ans.get(i - 1).get(j - 1) + ans.get(i - 1).get(j));

    return ans;

  }

}



[**461. Hamming Distance**](https://leetcode.com/problems/hamming-distance/):

**Code:**

class Solution {

    public int hammingDistance(int x, int y) {

        int difference = 0;

        while(x>0&&y>0) {

            int last1 = x & 1;

            int last2 = y & 1;

            if ((last1 ^ last2) == 1) {

                difference++;

            }

            x=x>>1;

            y=y>>1;

        }

        while(x>0){

         if((x&1)==1){

             difference++;

         }

        x= x>>1;

        }

        while(y>0){

            if((y&1)==1){

                difference++;

            }

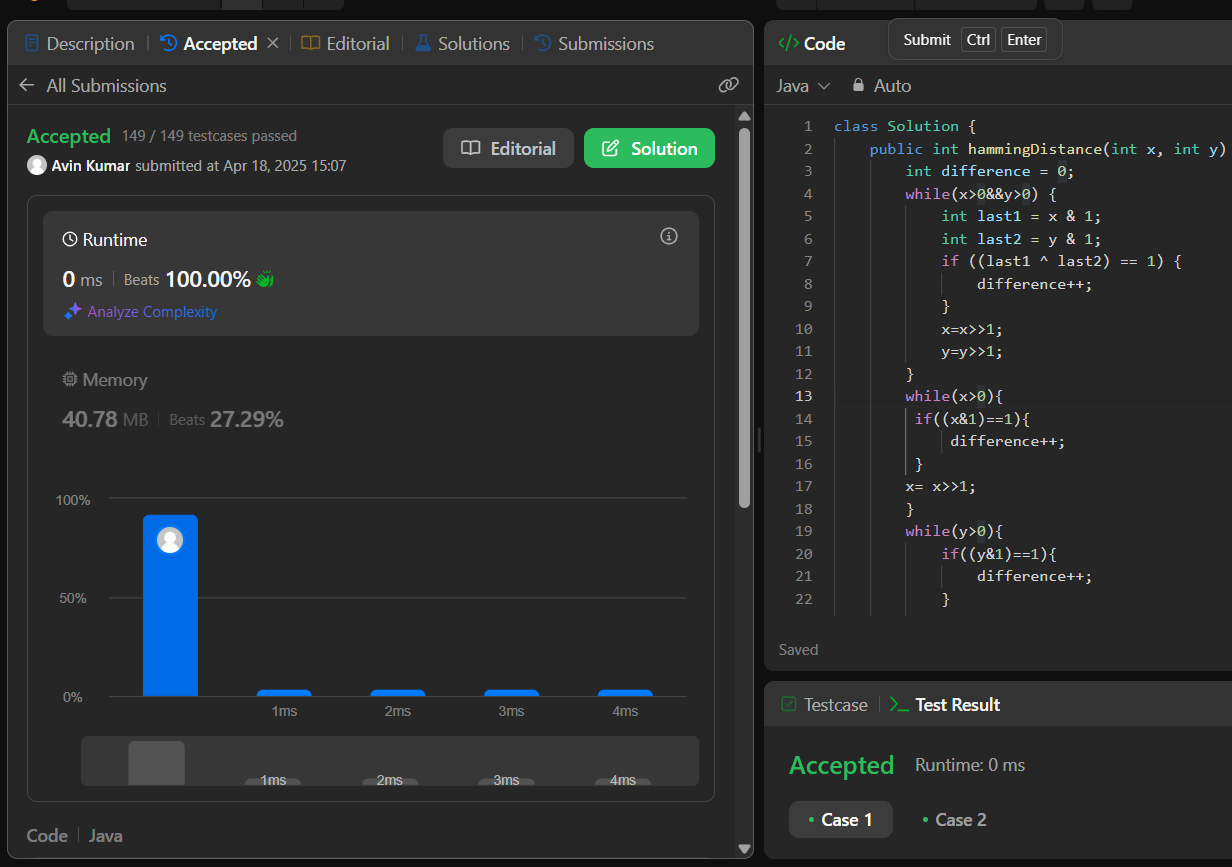
            y=y>>1;

        }

      return difference;

    }

}



[**621. Task Scheduler**](https://leetcode.com/problems/task-scheduler/):

**Code:**

class Solution {

public int leastInterval(char[] tasks, int n) {

int[] freq = new int[26];

for(char task : tasks){

freq[task-'A']++;

}

Arrays.sort(freq);

int max = freq[25]-1;

int idles = max \* n;

for(int i=freq.length-2;i>=0 && freq[i]!=0;i--){

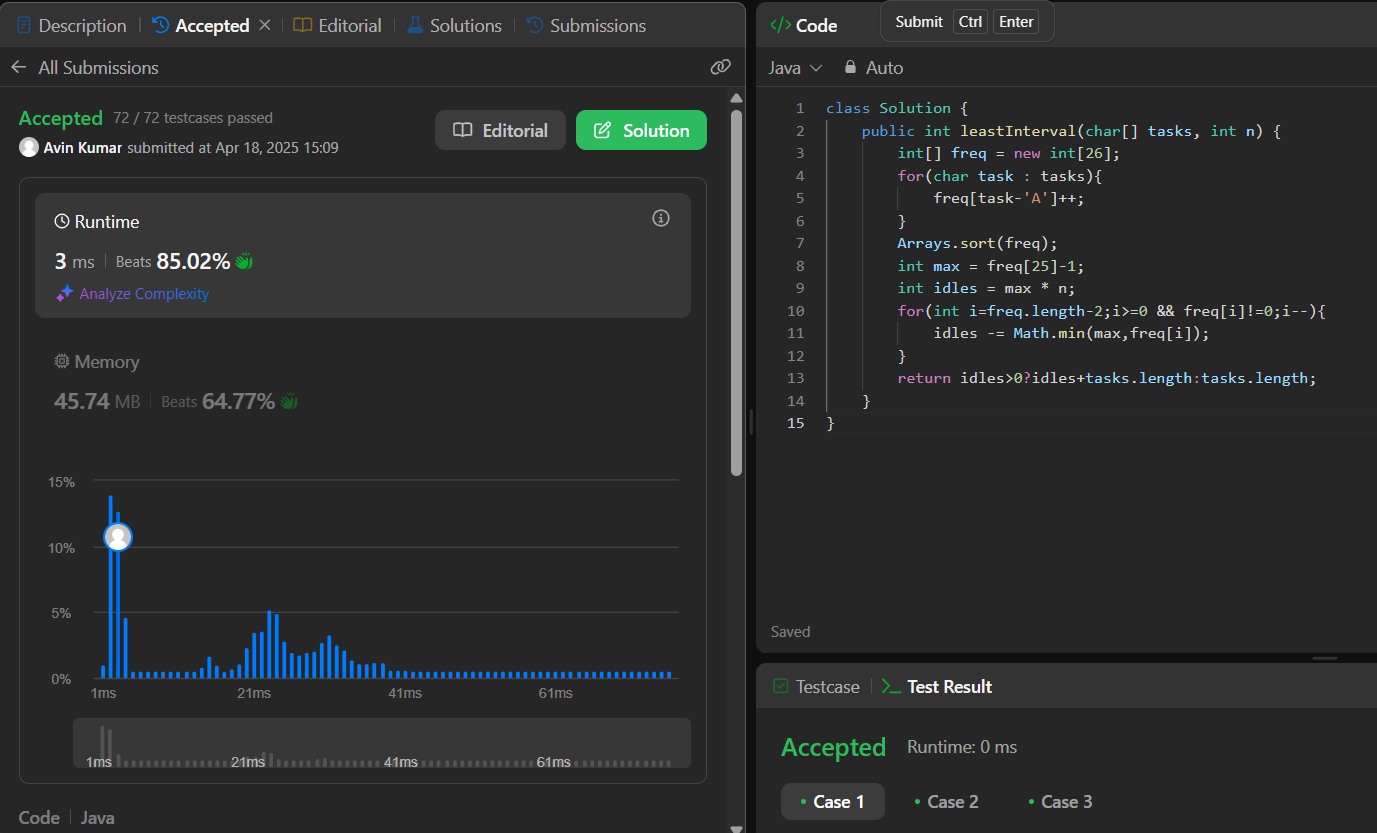
idles -= Math.min(max,freq[i]);

}

return idles>0?idles+tasks.length:tasks.length;

}

}



[**191. Number of 1 Bits**](https://leetcode.com/problems/number-of-1-bits/):

**Code:**

class Solution {

public int hammingWeight(int n) {

String x = Integer.toBinaryString(n);

int count =0;

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

if(x.charAt(i)=='1'){

count++;

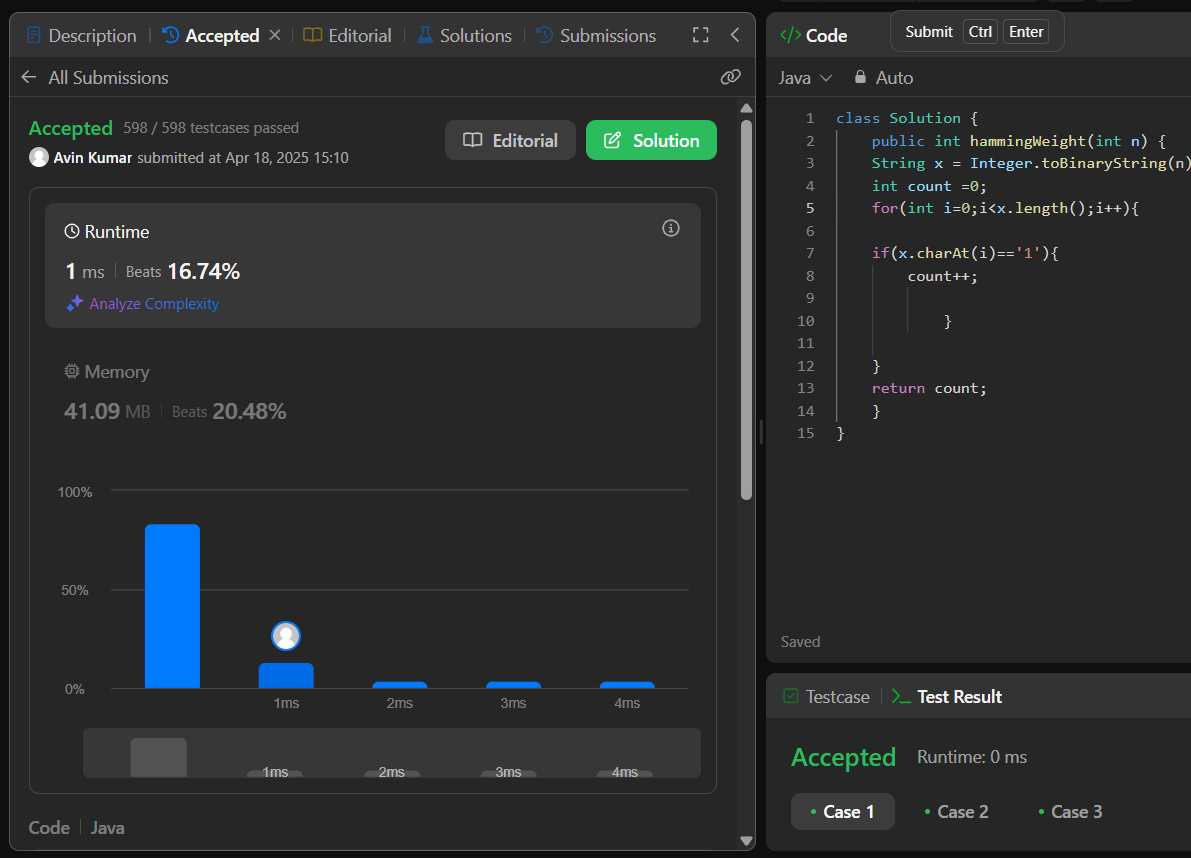
}

}

return count;

}

}



[**20. Valid Parentheses**](https://leetcode.com/problems/valid-parentheses/):

**Code:**

class Solution {

    public boolean isValid(String s) {

        Stack<Character> stack = new Stack<>();

        for (char c : s.toCharArray()) {

            if (c == '(' || c == '{' || c == '[') {

                stack.push(c);

            } else {

                if (stack.isEmpty() ||

                   (c == ')' && stack.pop() != '(') ||

                   (c == '}' && stack.pop() != '{') ||

                   (c == ']' && stack.pop() != '[')) {

                    return false;

                }

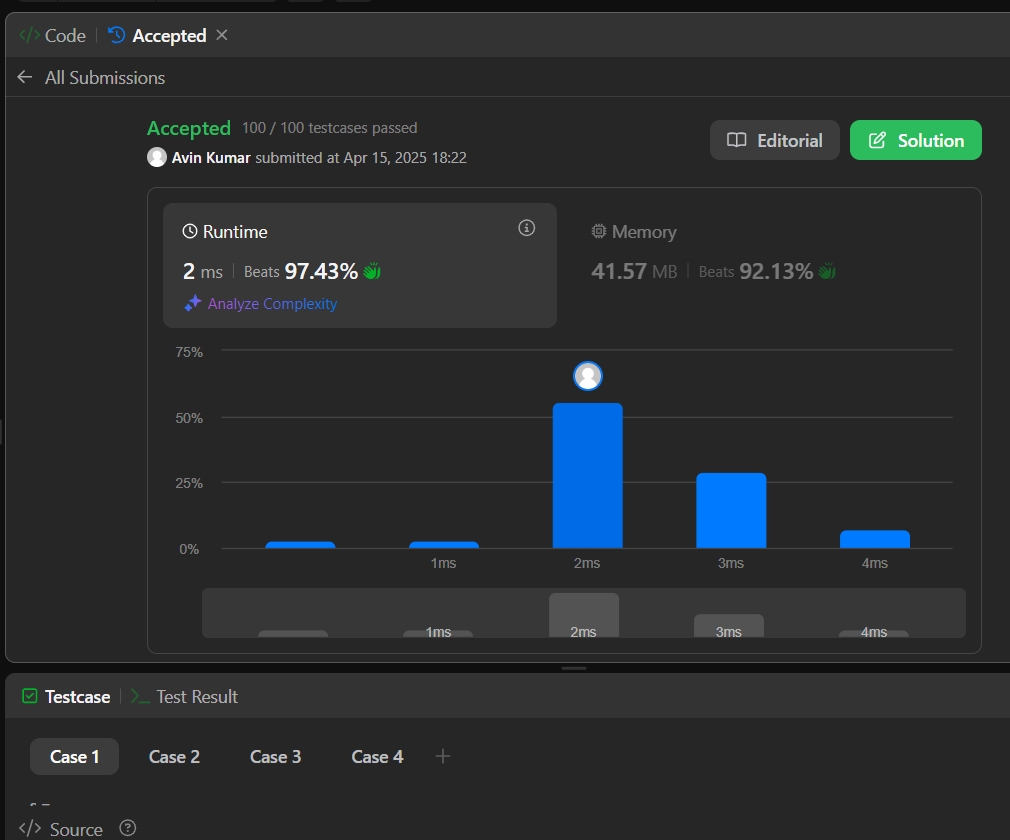
            }

        }

        return stack.isEmpty();

    }

}



[**29. Divide Two Integers**](https://leetcode.com/problems/divide-two-integers/):

**Code:**

class Solution {

public int divide(int dividend, int divisor) {

if(dividend == Integer.MIN\_VALUE && divisor == -1){

return Integer.MAX\_VALUE;

}

long a = Math.abs((long) dividend);

long b = Math.abs((long) divisor);

int sign = (dividend > 0) == (divisor > 0) ? 1 : -1;

long result = 0;

while(a >= b){

long temp = b;

long multi = 1;

while(a >= (temp << 1)){

temp <<= 1;

multi <<= 1;

}

a -= temp;

result += multi;

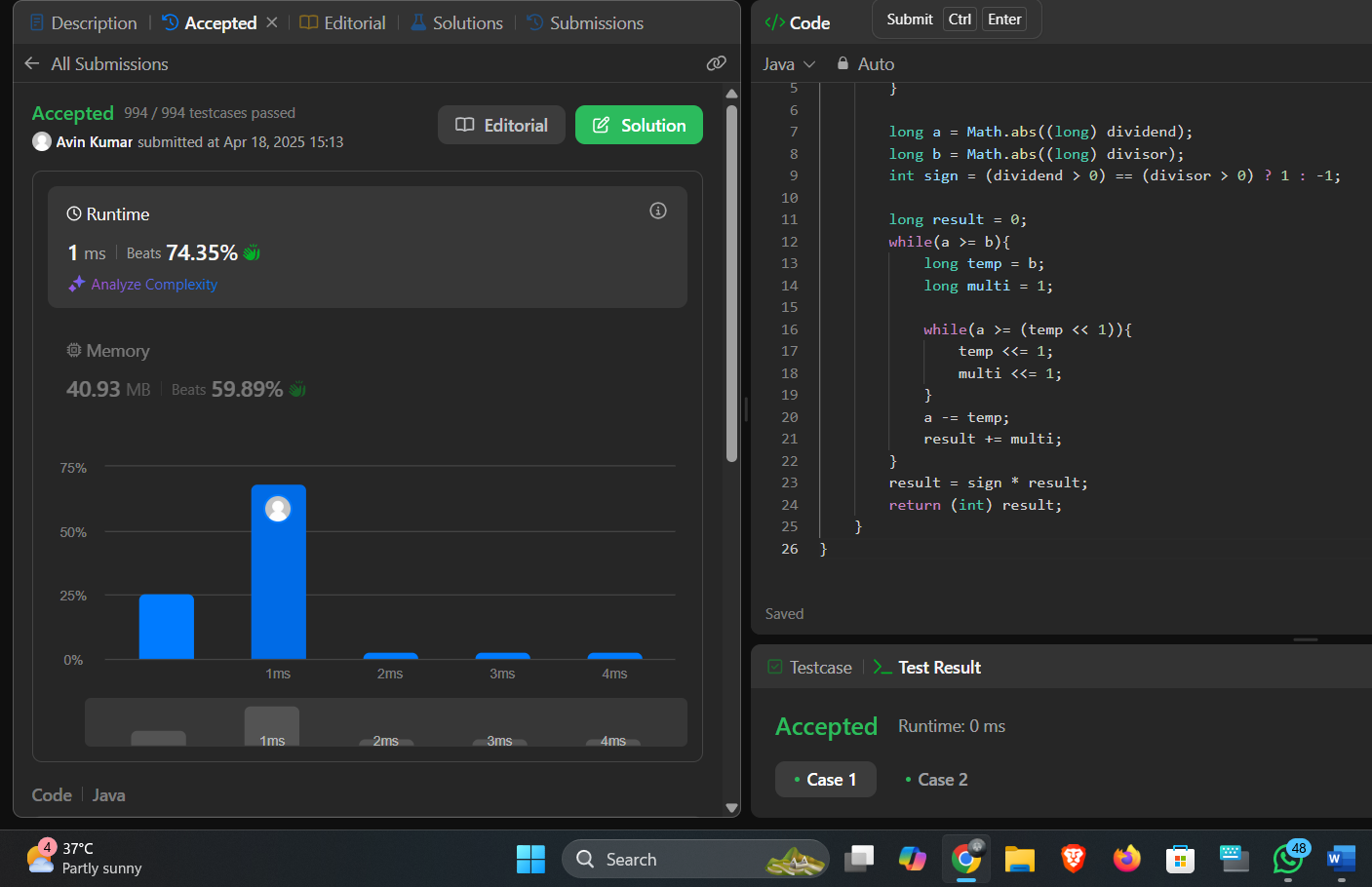
}

result = sign \* result;

return (int) result;

}

}



[**42. Trapping Rain Water**](https://leetcode.com/problems/trapping-rain-water/):

**Code:**

// Approach 1: Precompute Left and Right Max Arrays (O(N) Time, O(N) Space)

class Solution {

public int trap(int[] arr) {

int n = arr.length;

int[] lmax = new int[n];

lmax[0] = arr[0];

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

lmax[i] = Math.max(lmax[i - 1], arr[i]);

int[] rmax = new int[n];

rmax[n - 1] = arr[n - 1];

for (int i = n - 2; i >= 0; i--)

rmax[i] = Math.max(rmax[i + 1], arr[i]);

int ans = 0;

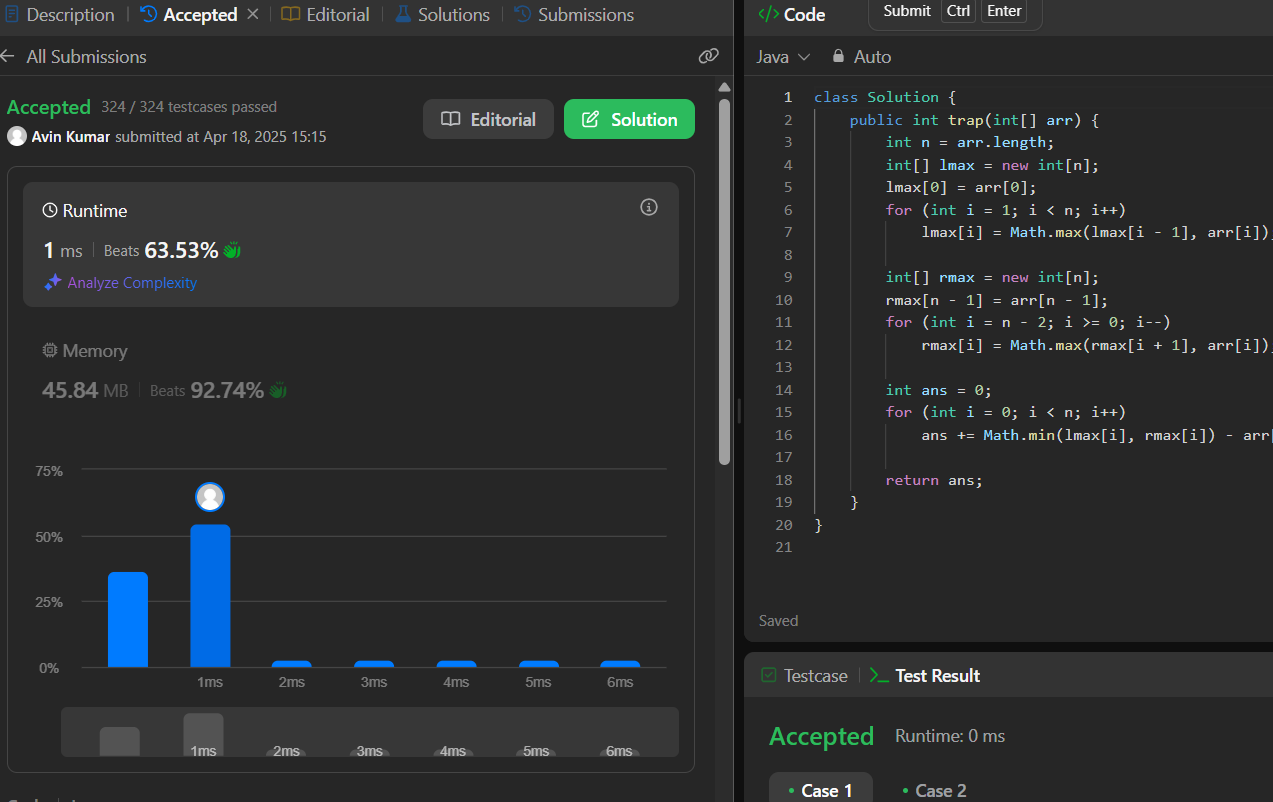
for (int i = 0; i < n; i++)

ans += Math.min(lmax[i], rmax[i]) - arr[i];

return ans;

}

}



[**2071. Maximum Number of Tasks You Can Assign**](https://leetcode.com/problems/maximum-number-of-tasks-you-can-assign/):

**Code:**

class Solution {

    public int maxTaskAssign(int[] tasks, int[] workers, int pills, int strength) {

        Arrays.sort(tasks);

        Arrays.sort(workers);

        int low=0;

        int high=tasks.length;

        int ans=0;

        while(low<=high)

        {

            int mid=(low+high)/2;

            if(isPossible(tasks,workers,pills,strength,mid))

            {

                ans=mid;

                low=mid+1;

            }else{

                high=mid-1;

            }

        }

        return ans;

    }

    public boolean isPossible(int tasks[],int workers[],int pills,int strength,int n)

    {

        if(Math.min(workers.length,tasks.length)<n)

            return false;

        TreeMap<Integer,Integer> tm=new TreeMap<>();

        for(int i=0;i<n;i++)

        {

            tm.put(tasks[i],tm.getOrDefault(tasks[i],0)+1);

        }

        for(int j=workers.length-n;j<workers.length;j++)

        {

            int smallest=tm.firstKey();

            if(workers[j]<smallest)

            {

                if(pills==0)

                    return false;

                pills--;

                if(tm.floorKey(workers[j]+strength)!=null)

                    smallest=tm.floorKey(workers[j]+strength);

                else

                    return false;

            }

            if(tm.get(smallest)==1)

                tm.remove(smallest);

            else

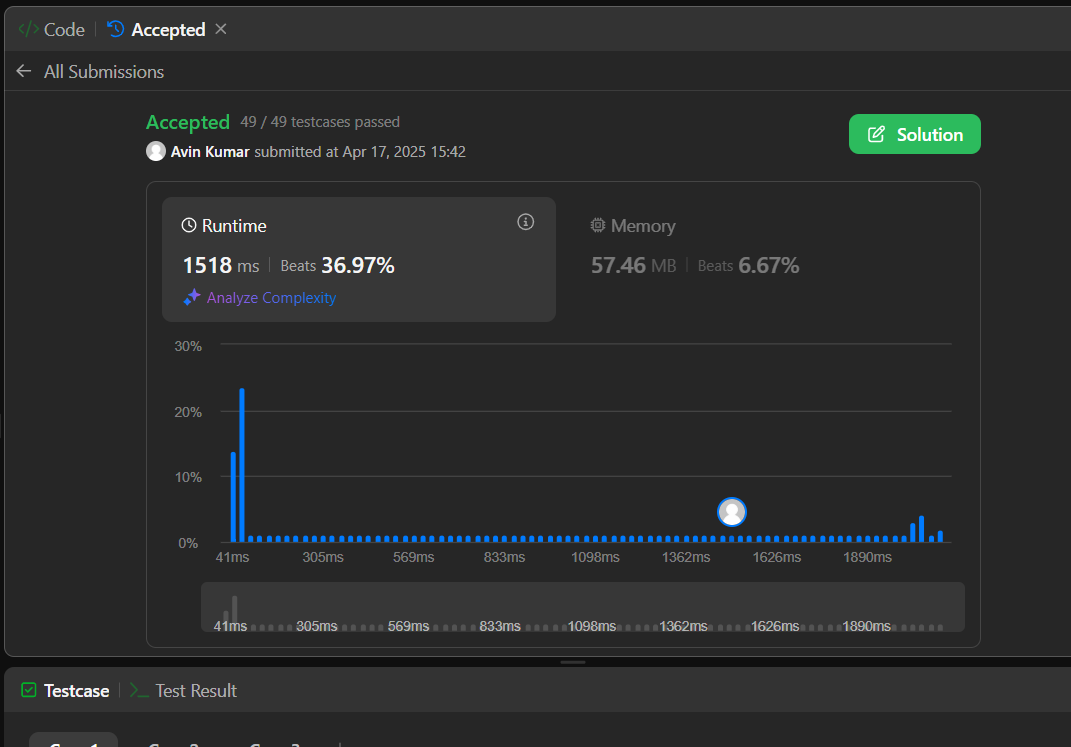
                tm.put(smallest,tm.get(smallest)-1);

        }

        return true;

    }

}



[**297. Serialize and Deserialize Binary Tree**](https://leetcode.com/problems/serialize-and-deserialize-binary-tree/):

**Code:**

public class Codec {

public String serialize(TreeNode root) {

if (root == null) return "";

Queue<TreeNode> q = new LinkedList<>();

StringBuilder res = new StringBuilder();

q.add(root);

while (!q.isEmpty()) {

TreeNode node = q.poll();

if (node == null) {

res.append("n ");

continue;

}

res.append(node.val + " ");

q.add(node.left);

q.add(node.right);

}

return res.toString();

}

public TreeNode deserialize(String data) {

if (data == "") return null;

Queue<TreeNode> q = new LinkedList<>();

String[] values = data.split(" ");

TreeNode root = new TreeNode(Integer.parseInt(values[0]));

q.add(root);

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

TreeNode parent = q.poll();

if (!values[i].equals("n")) {

TreeNode left = new TreeNode(Integer.parseInt(values[i]));

parent.left = left;

q.add(left);

}

if (!values[++i].equals("n")) {

TreeNode right = new TreeNode(Integer.parseInt(values[i]));

parent.right = right;

q.add(right);

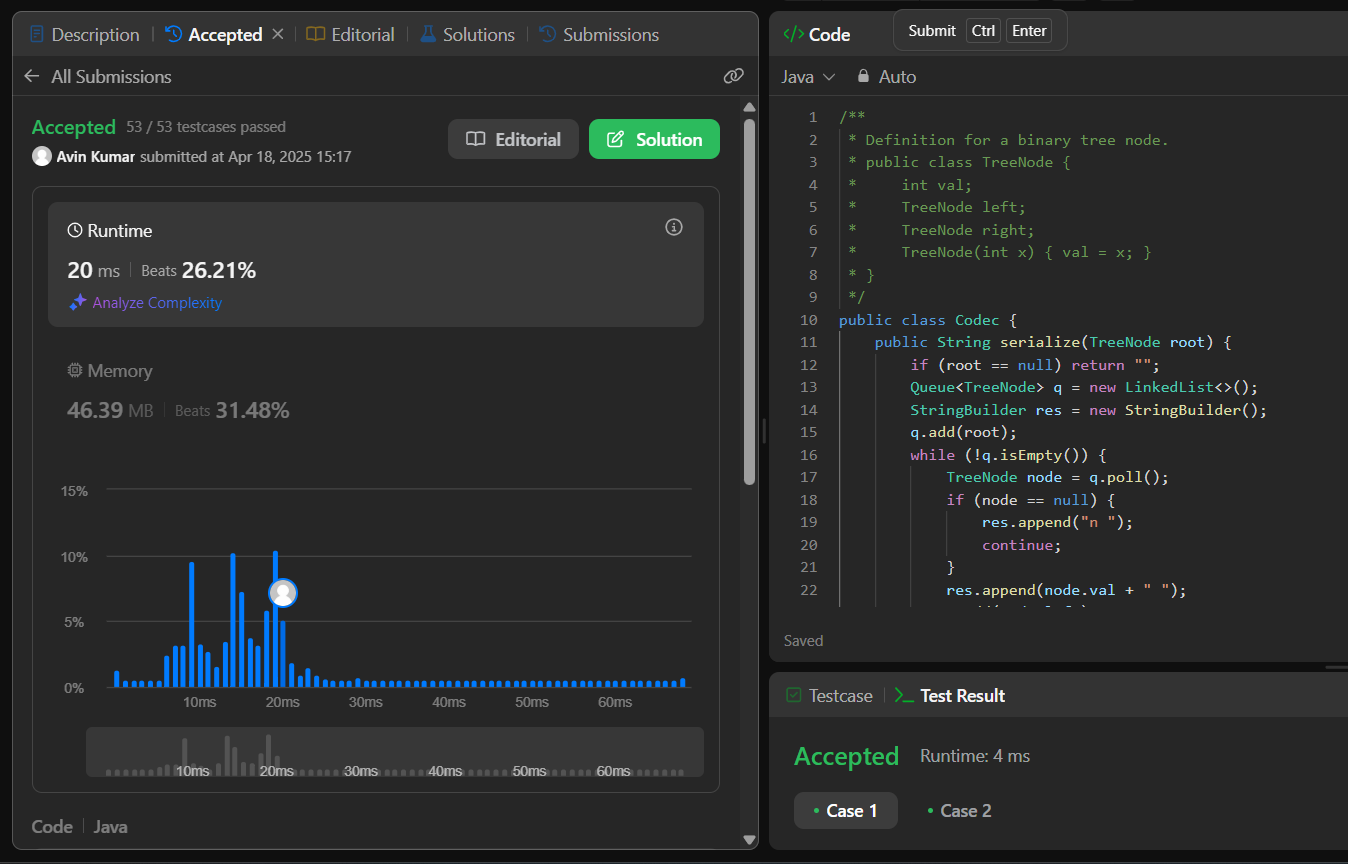
}

}

return root;

}

}



[**146. LRU Cache**](https://leetcode.com/problems/lru-cache/):

**Code:**

class LRUCache {

class Node {

int key;

int val;

Node prev;

Node next;

Node(int key, int val) {

this.key = key;

this.val = val;

}

}

Node head = new Node(-1, -1);

Node tail = new Node(-1, -1);

int cap;

HashMap<Integer, Node> m = new HashMap<>();

public LRUCache(int capacity) {

cap = capacity;

head.next = tail;

tail.prev = head;

}

private void addNode(Node newnode) {

Node temp = head.next;

newnode.next = temp;

newnode.prev = head;

head.next = newnode;

temp.prev = newnode;

}

private void deleteNode(Node delnode) {

Node prevv = delnode.prev;

Node nextt = delnode.next;

prevv.next = nextt;

nextt.prev = prevv;

}

public int get(int key) {

if (m.containsKey(key)) {

Node resNode = m.get(key);

int ans = resNode.val;

m.remove(key);

deleteNode(resNode);

addNode(resNode);

m.put(key, head.next);

return ans;

}

return -1;

}

public void put(int key, int value) {

if (m.containsKey(key)) {

Node curr = m.get(key);

m.remove(key);

deleteNode(curr);

}

if (m.size() == cap) {

m.remove(tail.prev.key);

deleteNode(tail.prev);

}

addNode(new Node(key, value));

m.put(key, head.next);

}

}

