**UNIVERSIDAD NACIONAL DE COLOMBIA.**

**SEDE BOGOTÁ.**

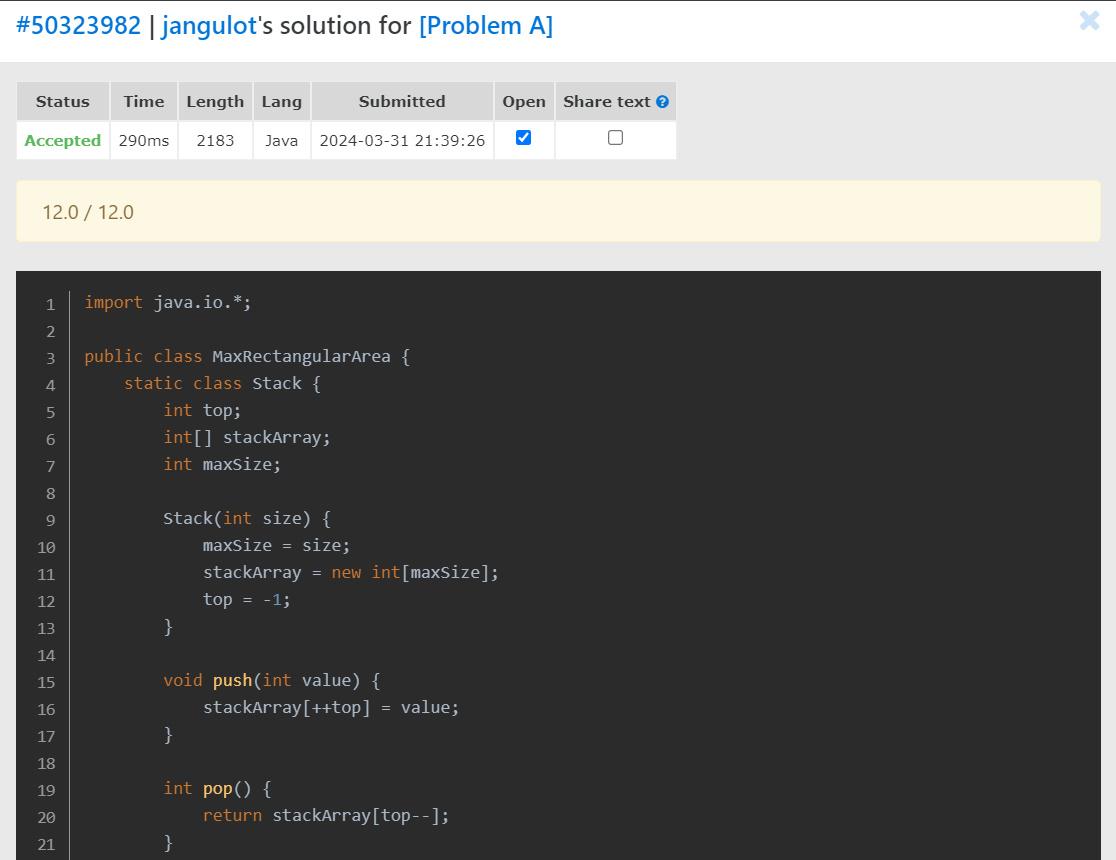
**ESTRUCTURA DE DATOS GRUPO 4**

**JAIME DARLEY ANGULO TENORIO**

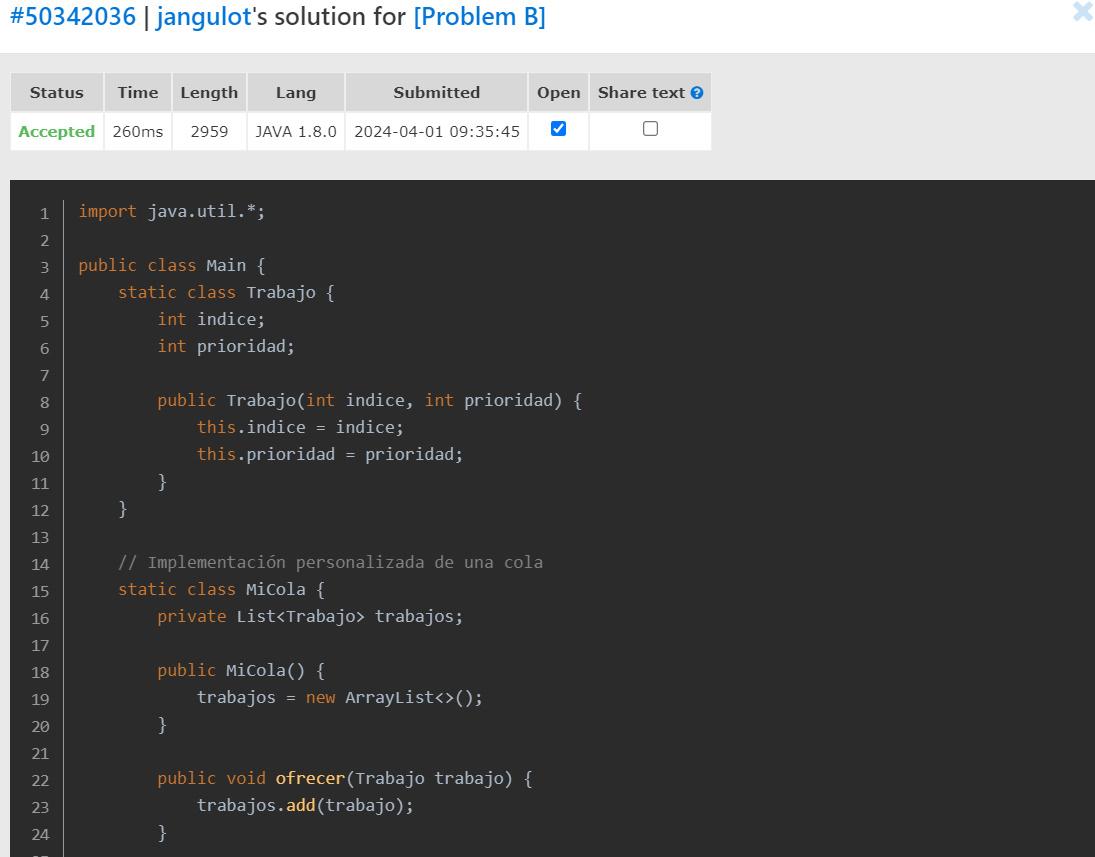
**TALLER 1**

1. Aqui esta una captura de pantalla de los 3 ejercicios aprobados por la máquina virtual

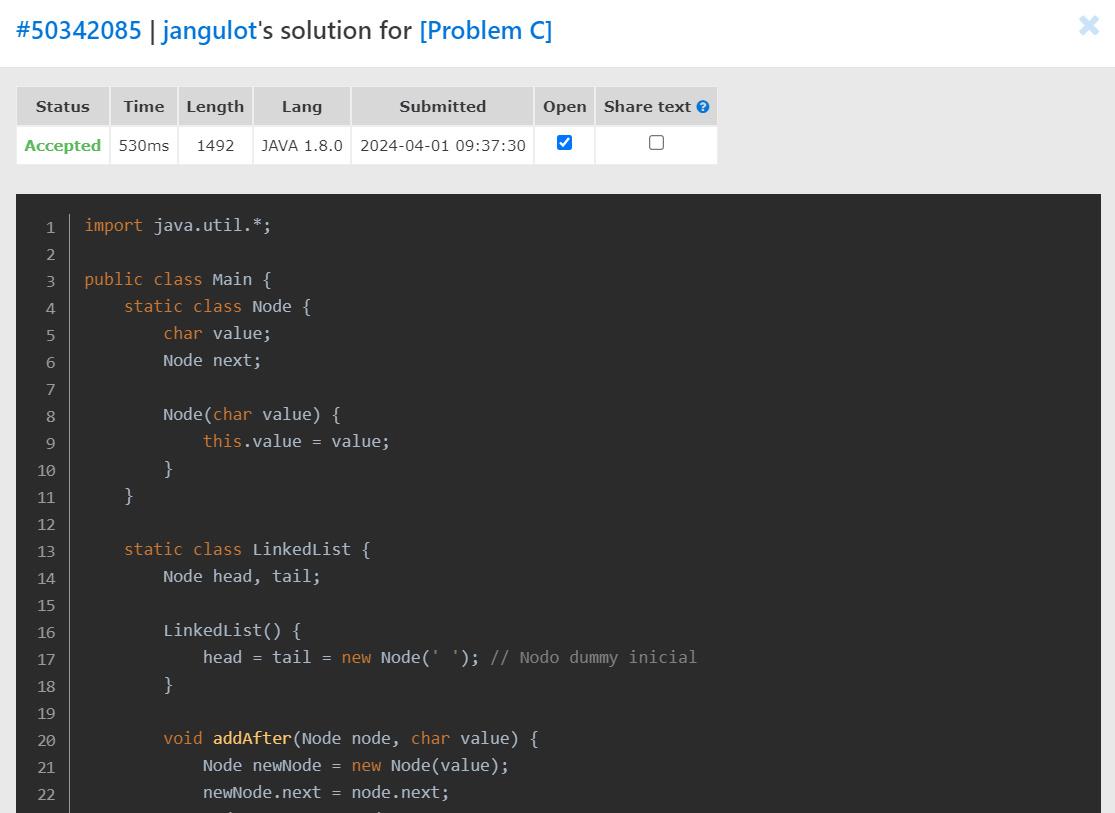
Problema A - Buildings and Trees



Problema B - Printing Time



Problema C - Weird Text



Ahora el código de cada ejercicio

Ejercicio A - Buildings and Trees:

import java.io.\*;

public class MaxRectangularArea {

static class Stack {

int top;

int[] stackArray;

int maxSize;

Stack(int size) {

maxSize = size;

stackArray = new int[maxSize];

top = -1;

}

void push(int value) {

stackArray[++top] = value;

}

int pop() {

return stackArray[top--];

}

int peek() {

return stackArray[top];

}

boolean isEmpty() {

return (top == -1);

}

void clear() {

top = -1;

}

}

public static int maxRectangularArea(char[][] forest) {

int n = forest.length;

int m = forest[0].length;

int[] heights = new int[m];

Stack stack = new Stack(m);

int maxArea = 0;

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

for (int j = 0; j < m; j++) {

if (forest[i][j] == '.') {

heights[j]++;

} else {

heights[j] = 0;

}

}

stack.clear();

for (int j = 0; j <= m; j++) {

while (!stack.isEmpty() && (j == m || heights[j] < heights[stack.peek()])) {

int height = heights[stack.pop()];

int width = stack.isEmpty() ? j : j - stack.peek() - 1;

maxArea = Math.max(maxArea, height \* width);

}

stack.push(j);

}

}

return maxArea;

}

public static void main(String[] args) throws IOException {

BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));

String[] nm = reader.readLine().split(" ");

int n = Integer.parseInt(nm[0]);

int m = Integer.parseInt(nm[1]);

char[][] forest = new char[n][m];

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

String line = reader.readLine();

for (int j = 0; j < m; j++) {

forest[i][j] = line.charAt(j);

}

}

int maxArea = maxRectangularArea(forest);

System.out.println(maxArea);

}

}

Ejercicio B - Printing Time

import java.util.\*;

public class Main {

static class Trabajo {

int indice;

int prioridad;

public Trabajo(int indice, int prioridad) {

this.indice = indice;

this.prioridad = prioridad;

}

}

// Implementación personalizada de una cola

static class MiCola {

private List<Trabajo> trabajos;

public MiCola() {

trabajos = new ArrayList<>();

}

public void ofrecer(Trabajo trabajo) {

trabajos.add(trabajo);

}

public Trabajo extraer() {

if (estaVacia()) {

return null;

}

Trabajo trabajo = trabajos.get(0);

trabajos.remove(0);

return trabajo;

}

public boolean estaVacia() {

return trabajos.isEmpty();

}

}

// Implementación personalizada de una cola de prioridad (mayor prioridad primero)

static class MiColaPrioridad {

private List<Integer> prioridades;

public MiColaPrioridad() {

prioridades = new ArrayList<>();

}

public void ofrecer(int prioridad) {

prioridades.add(prioridad);

Collections.sort(prioridades, Collections.reverseOrder());

}

public int consultar() {

if (estaVacia()) {

return Integer.MIN\_VALUE;

}

return prioridades.get(0);

}

public void extraer() {

if (!estaVacia()) {

prioridades.remove(0);

}

}

public boolean estaVacia() {

return prioridades.isEmpty();

}

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int numCasosPrueba = scanner.nextInt();

for (int t = 0; t < numCasosPrueba; t++) {

int n = scanner.nextInt(); // número de trabajos en la cola

int m = scanner.nextInt(); // posición de tu trabajo

MiCola cola = new MiCola();

MiColaPrioridad colaPrioridad = new MiColaPrioridad();

// Leer las prioridades de los trabajos en la cola

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

int prioridad = scanner.nextInt();

cola.ofrecer(new Trabajo(i, prioridad));

colaPrioridad.ofrecer(prioridad);

}

int minutos = 0;

while (!cola.estaVacia()) {

Trabajo trabajoActual = cola.extraer();

if (trabajoActual.prioridad == colaPrioridad.consultar()) {

minutos++;

colaPrioridad.extraer();

if (trabajoActual.indice == m) {

System.out.println(minutos);

break;

}

} else {

cola.ofrecer(trabajoActual);

}

}

}

}

}

Ejercicio C - Weird Text

import java.util.\*;

public class Main {

static class Node {

char value;

Node next;

Node(char value) {

this.value = value;

}

}

static class LinkedList {

Node head, tail;

LinkedList() {

head = tail = new Node(' '); // Nodo dummy inicial

}

void addAfter(Node node, char value) {

Node newNode = new Node(value);

newNode.next = node.next;

node.next = newNode;

if (node == tail) {

tail = newNode;

}

}

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

while (scanner.hasNext()) {

String line = scanner.nextLine();

LinkedList list = new LinkedList();

Node cursor = list.head;

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

if (c == '[') {

cursor = list.head;

} else if (c == ']') {

cursor = list.tail;

} else {

list.addAfter(cursor, c);

cursor = cursor.next;

}

}

StringBuilder result = new StringBuilder();

Node node = list.head.next;

while (node != null) {

result.append(node.value);

node = node.next;

}

System.out.println(result.toString());

}

}

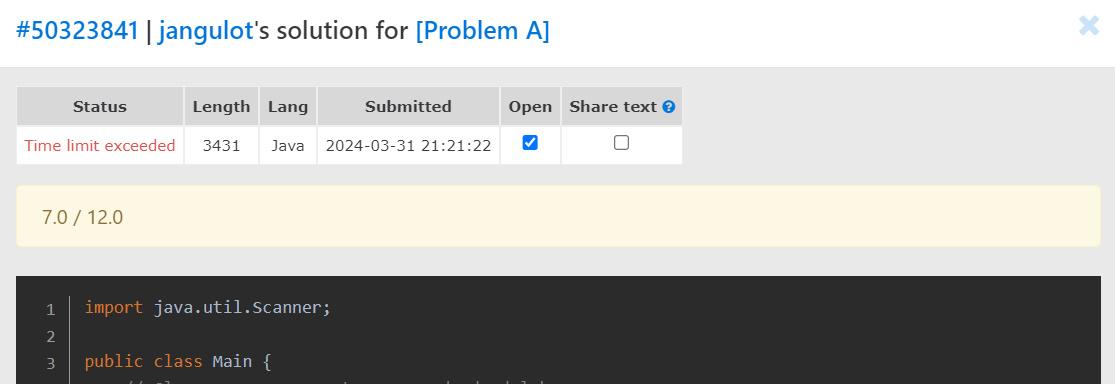
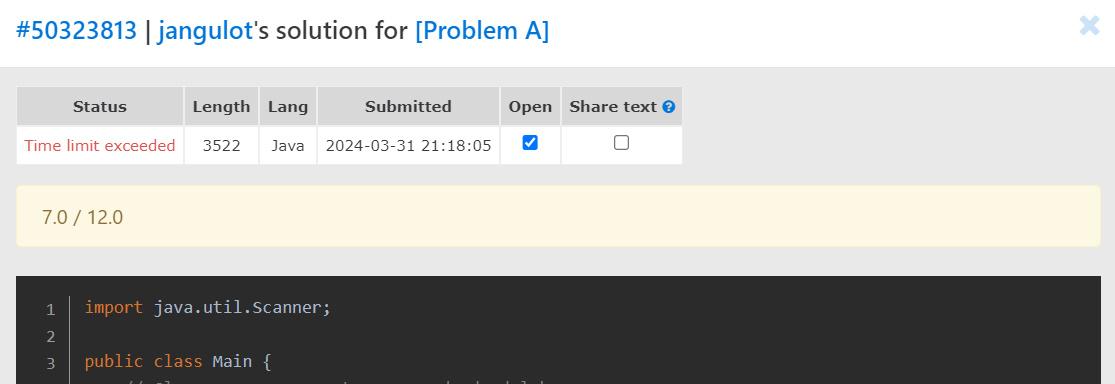
}

**¿Cuál fue el ejercicio más difícil?**

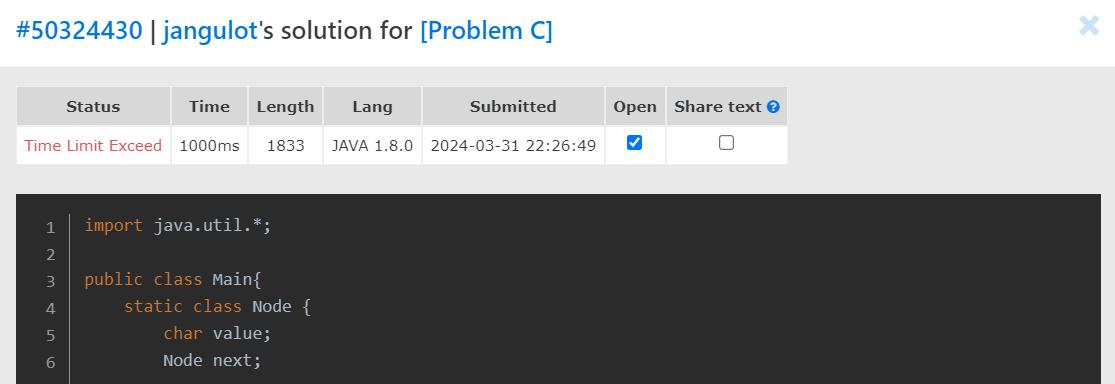
Por mucho el ejercicio más difícil fue el Ejercicio A - Buildings and Trees, quizás sea porque fue el primer ejercicio, porque los otros dos se hicieron fáciles después en términos de tiempo, el problema principal fue de tiempo excedido.

**¿Qué problemas tuvieron y cómo los solucionaron**?

El mayor principalmente de tiempo, el programa pasaba del límite de tiempo exigido por la página aquí algunas capturas del tiempo que tomó los primeros intentos



No Sé porque en el problema C dice que se pasó del límite si el tiempo fue justo el tiempo que pedían osea 1000ms, quizás la diferencia fue tan mínima que la pagina solo lo redondeó



La solucionar esto en el problema A se uso lista enlazadas para implementar la estructura de datos Pila al principio y la clase Scanner para la lectura de datos, la solución simplemente se implementa la Pila con arreglos y se cambió el Scanner por BufferedReader y ya funcionó con normalidad.

Para el problema C la diferencia de tiempo excedido fue tan mínima que solo hice cambio tambien fue asi, se usó lista enlazadas había métodos que no se estaba utilizando para en la resolución del problema simplemente se quitaron ya función normal