public class Main {//longest increasing subsequence

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

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

int n = Integer.parseInt(br.readLine());

int[] arr = new int[n];

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

arr[i] = Integer.parseInt(br.readLine());

}

int omax = 0;

int[] dp = new int[arr.length];

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

Integer max = null;

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

if(arr[j] <= arr[i]){

if(max == null || dp[j] > max){

max = dp[j];

}

}

}

if(max != null){

dp[i] = max + 1;

} else {

dp[i] = 1;

}

if(dp[i] > omax){

omax = dp[i];

} }

System.out.println(omax);

}

LIS Sum

public class Main {

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

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

int n = Integer.parseInt(br.readLine());

int[] arr = new int[n];

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

arr[i] = Integer.parseInt(br.readLine());

}

int omax = Integer.MIN\_VALUE;

int[] dp = new int[arr.length];

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

Integer max = null;

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

if(arr[j] <= arr[i]){

if(max == null || dp[j] > max){

max = dp[j];

}}}

if(max != null){

dp[i] = max + arr[i];

} else {

dp[i] = arr[i];

}

if(dp[i] > omax){

omax = dp[i];

}

}

System.out.println(omax);

}}

INcrewsing ande Decre LIS

int[] lis = new int[arr.length];

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

Integer max = null;

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

if(arr[j] <= arr[i]){

if(max == null || lis[j] > max){

max = lis[j];

}

}

}

if(max != null){

lis[i] = max + 1;

} else {

lis[i] = 1;

}

}

int[] lds = new int[arr.length];

for(int i = arr.length - 1; i >= 0; i--){

Integer max = null;

for(int j = arr.length - 1; j > i; j--){

if(arr[j] <= arr[i]){

if(max == null || lds[j] > max){

max = lds[j];

}

}

}

if(max != null){

lds[i] = max + 1;

} else {

lds[i] = 1;

}

}

int omax = 0;

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

if(lis[i] + lds[i] - 1> omax){

omax = lis[i] + lds[i] - 1;

}

}

System.out.println(omax);

}

}

///loverlaping brideges

public class Main {

public static class Bridge implements Comparable<Bridge> {

int n;

int s;

public int compareTo(Bridge o){

if(this.n != o.n){

return this.n - o.n;

} else {

return this.s - o.s;

}

}

}

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

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

int n = Integer.parseInt(br.readLine());

Bridge[] brdgs = new Bridge[n];

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

String str = br.readLine();

brdgs[i] = new Bridge();

brdgs[i].n = Integer.parseInt(str.split(" ")[0]);

brdgs[i].s = Integer.parseInt(str.split(" ")[1]);

}

Arrays.sort(brdgs);

int[] lis = new int[brdgs.length];

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

Integer max = null;

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

if(brdgs[j].s <= brdgs[i].s){

if(max == null || lis[j] > max){

max = lis[j];

}

}

}

if(max != null){

lis[i] = max + 1;

} else {

lis[i] = 1;

}

}

int omax = 0;

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

if(lis[i] > omax){

omax = lis[i];

}

}

System.out.println(omax);

}}

//Rushin Doll

import java.io.\*;

import java.util.\*;

public class Main {

public static class Envelope implements Comparable<Envelope> {

int w;

int h;

public int compareTo(Envelope o){

return this.w - o.w;

}

}

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

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

int n = Integer.parseInt(br.readLine());

Envelope[] envlps = new Envelope[n];

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

String str = br.readLine();

envlps[i] = new Envelope();

envlps[i].w = Integer.parseInt(str.split(" ")[0]);

envlps[i].h = Integer.parseInt(str.split(" ")[1]);

}

Arrays.sort(envlps);

int[] lis = new int[envlps.length];

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

Integer max = null;

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

if(envlps[j].h < envlps[i].h && envlps[j].w < envlps[i].w){

if(max == null || lis[j] > max){

max = lis[j];

}

}

}

if(max != null){

lis[i] = max + 1;

} else {

lis[i] = 1;

}

}

int omax = 0;

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

if(lis[i] > omax){

omax = lis[i];

}

}

System.out.println(omax);

}

}