**MARATONES DE PROGRAMACIÓN**

**ACIS- REDIS / ACM-ICPC**

**Cartilla de ejercicios resueltos, algoritmos de solución y ejemplos**

**Volumen 2**

**Grupo programación competitiva UECCI**

**NMA-ARTEMIS**

**Universidad ECCI**

**Ingeniería de sistemas**

**Tecnología de desarrollo informático**

**2019**

Contenido

[Walking Around Wisely (Programación dinámica) 3](#_Toc7974106)

[The Blocks Problem (Pila /Matematica) 3](#_Toc7974107)

[All-Star three Point Contest (String) 6](#_Toc7974108)

[Reaux! Sham! Beaux! (String) 8](#_Toc7974109)

[Train Swapping (Ordenamiento/Burbuja) 10](#_Toc7974110)

[Summing Digits (Mat. modular) 11](#_Toc7974111)

[Binary Search Tree (Árbol binario) 11](#_Toc7974112)

[Baskets Of Gold Coins (Mat/Gran cantidad de casos) 13](#_Toc7974113)

[Big Number (Kamenetsky / Números grandes / Factorial) 14](#_Toc7974114)

[Factorial Factors (Números grandes /Eratostenes) 14](#_Toc7974115)

[Sum It Up (Mat/Busqueda en profundidad) 15](#_Toc7974116)

[Roads In The North (Grafo/DFS) 16](#_Toc7974117)

[INCEPTION (Pila) 17](#_Toc7974118)

[Euclid Problem (Matemático/GCD) 18](#_Toc7974119)

[Bars (All sets) 18](#_Toc7974120)

# Walking Around Wisely (Programación dinámica)

ANSI C

#include <stdio.h>

int main() {

int t, n, m, sx, sy, ex, ey;

int x, y, dir, i, j;

char s[3];

scanf("%d", &t);

while(t--) {

scanf("%d", &n);

scanf("%d %d %d %d", &sx, &sy, &ex, &ey);

scanf("%d", &m);

int cant[35][35][2] = {};

long long dp[35][35] = {};

while(m--) {

scanf("%d %d %s", &x, &y, s);

if(s[0] == 'S')

y--, dir = 1;

else if(s[0] == 'W')

x--, dir = 0;

else if(s[0] == 'N')

dir = 1;

else dir = 0;

cant[x][y][dir] = 1;

}

dp[sx][sy] = 1;

for(i = sx; i <= ex; i++) {

for(j = sy; j <= ey; j++) {

if(!cant[i][j][0])

dp[i+1][j] += dp[i][j];

if(!cant[i][j][1])

dp[i][j+1] += dp[i][j];

}

}

printf("%lld\n", dp[ex][ey]);

}

return 0;

}

# The Blocks Problem (Pila /Matematica)

JAVA

import java.util.ArrayList;

import java.util.Scanner;

import java.util.Stack;

public class Main {

public static void main(String[] args) {

ArrayList<Stack<Integer>> listaPilas = new ArrayList<>();

Scanner sc = new Scanner(System.in);

int cantidad = sc.nextInt();

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

listaPilas.add(new Stack<>());

listaPilas.get(i).add(i);

}

String de = sc.next();

while (!de.equals("quit")) {

int posicionA = 0, posicionB = 0;

int uno = sc.nextInt();

String para = sc.next();

int dos = sc.nextInt();

posicionA = busquedaPosicion(uno, listaPilas);

posicionB = busquedaPosicion(dos, listaPilas);

if (posicionA!=posicionB) {

if ("move".equals(de)) {

if ("onto".equals(para)) {

//Primero devolver bloques encima de a y b a posiciones iniciales

//Mover el bloque a sobre el b

moverOriginalPosicion(uno, posicionA, listaPilas);

moverOriginalPosicion(dos, posicionB, listaPilas);

moverAaB(posicionB, posicionA, uno, listaPilas);

} else {

//es over

//devolver bloques que estan encima de a a iniciales

//colocar el bloque a sobre la pila que se encuentra el b

moverOriginalPosicion(uno, posicionA, listaPilas);

moverAaB(posicionB, posicionA, uno, listaPilas);

}

} else {

//es pille

if ("onto".equals(para)) {

//devolver bloques que se encuentran encima de b

//mover el bloque a y lo que esta encima de el al b

moverOriginalPosicion(dos, posicionB, listaPilas);

moverTodoArriba(uno, dos, posicionA, posicionB, listaPilas);

} else {

//over

//colocar el bloque a y lo que esta encima de el encima del bloque b

moverTodoArriba(uno, dos, posicionA, posicionB, listaPilas);

}

}

}

de = sc.next();

}

imprimir(listaPilas);

}

public static void moverAaB(int posicionB, int posicionA, int a, ArrayList<Stack<Integer>> paraMover) {

paraMover.get(posicionB).add(paraMover.get(posicionA).peek());

paraMover.get(posicionA).pop();

}

public static int busquedaPosicion(int encontrar, ArrayList<Stack<Integer>> paraEncontrar) {//en donde esta

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

for (int j = 0; j < paraEncontrar.get(i).size(); j++) {

if (paraEncontrar.get(i).get(j) == encontrar) {

return i;

}

}

}

return 0;

}

public static void moverTodoArriba(int uno, int dos, int posicionA, int posicionB, ArrayList<Stack<Integer>> paraMoverTodo) {

ArrayList<Integer> paraPasar = new ArrayList<>();

boolean es = false;

while (es == false) {

if (paraMoverTodo.get(posicionA).peek() == uno) {

paraPasar.add(paraMoverTodo.get(posicionA).peek());

paraMoverTodo.get(posicionA).pop();

es = true;

break;

} else {

paraPasar.add(paraMoverTodo.get(posicionA).peek());

paraMoverTodo.get(posicionA).pop();

}

}

for (int i = paraPasar.size() - 1; i >= 0; i--) {

paraMoverTodo.get(posicionB).add(paraPasar.get(i));

}

}

public static void moverOriginalPosicion(int uno, int posicionA, ArrayList<Stack<Integer>> paraMover) {

boolean es = false;

while (es == false) {

if (paraMover.get(posicionA).peek() == uno) {

es = true;

break;

} else {

paraMover.get(paraMover.get(posicionA).peek()).add(paraMover.get(posicionA).peek());

paraMover.get(posicionA).pop();

}

}

}

public static void imprimir(ArrayList<Stack<Integer>> paraImprimir) {

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

System.out.print(i + ":");

for (int j = 0; j < paraImprimir.get(i).size(); j++) {

System.out.print(" "+paraImprimir.get(i).get(j) );

}

System.out.println("");

}

}

}

# All-Star three Point Contest (String)

JAVA

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.util.ArrayList;

import java.util.Collections;

public class all {

public all() throws IOException {

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

String line;

int cases = 1;

while ((line = reader.readLine()) != null) {

int lines = Integer.parseInt(line);

ArrayList<Player> players = new ArrayList<>();

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

String temp = reader.readLine();

int pos = temp.indexOf(";");

String name = temp.substring(0, pos);

pos++;

int points = 0;

while ((pos = temp.indexOf("1;", pos)) != -1) {

points++;

pos++;

}

if (temp.charAt(temp.length() - 1) == '1') {

points++;

}

temp = temp.replaceAll("[^1]", "");

points += temp.length();

Player player = new Player(name, points);

players.add(player);

}

Collections.sort(players);

System.out.println("Case " + cases + ":");

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

System.out.println(players.get(i).toString());

}

cases++;

}

}

public static void main(String[] args) {

try {

new all();

} catch (IOException e) {

e.printStackTrace();

}

}

private class Player implements Comparable<Player> {

private String name;

private int points;

public Player(String pName, int pPoints) {

this.name = pName;

this.points = pPoints;

}

public Player() {

this("", 0);

}

public String getName() {

return name;

}

public void setName(String pName) {

this.name = pName;

}

public int getPoints() {

return points;

}

public void setPoints(int pPoints) {

this.points = pPoints;

}

@Override

public int compareTo(Player pPlayer) {

if (this.name.toUpperCase().compareTo(pPlayer.getName().toUpperCase()) != 0 && this.points == pPlayer.getPoints()) {

return this.name.toUpperCase().compareTo(pPlayer.getName().toUpperCase());

} else {

if (this.points > pPlayer.getPoints()) {

return -1;

} else if (this.points < pPlayer.getPoints()) {

return 1;

} else {

return 0;

}

}

}

@Override

public String toString() {

return name + " " + points;

}

}

}

# Reaux! Sham! Beaux! (String)

Python

lista=["Kamen","Rock","Pierre","Stein","Ko","Koe","Sasso","Roccia","Guu","Kamien","Piedra",

"Nuzky","Scissors","Ciseaux","Schere","Ollo","Olloo","Forbice","Choki","Nozyce", "Tijera",

"Papir","Paper","Feuille","Papier","Papir","Carta","Rete","Paa","Papier","Papel"];

#rock 0-10 scissors 11-20 paper 21-30

cantidad=1

bolean=False

while (bolean!=True):

t1=input()

if(t1=="."):

break

else:

x=t1.split(" ")

sera=x[0]

person1= x[1]

t2=input()

x=t2.split(" ")

person2=x[1]

conta=0

contb=0

vs=input()

vss=vs.split(" ")

a=vss[0]

while (a!="-"):

if a[0]==("."):

bolean=True

break

else:

b=vss[1]

num1=lista.index(a)

num2=lista.index(b)

if(num1>=0) &(num1<=10):

uno="roca"

elif (num1>=11) &(num1<=20):

uno="tijeras"

elif(num1>=21) &(num1<=30):

uno="papel"

if(num2>=0) &(num2<=10):

dos="roca"

elif (num2>=11) &(num2<=20):

dos="tijeras"

elif(num2>=21) &(num2<=30):

dos="papel"

if uno=="roca":

if dos=="tijeras":

conta=conta+1

if uno=="tijeras":

if dos=="papel":

conta=conta+1

if uno=="papel":

if dos=="roca":

conta=conta+1

if dos=="roca":

if uno=="tijeras":

contb=contb+1

if dos=="tijeras":

if uno=="papel":

contb=contb+1

if dos=="papel":

if uno=="roca":

contb=contb+1

vs=input()

vss=vs.split(" ")

a=vss[0]

print("Game #"+str(cantidad)+":");

if conta>contb:

if conta==1:

print(person1+": "+str(conta)+" point")

else:

print(person1+": "+str(conta)+" points")

if contb==1:

print(person2+": "+str(contb)+" point")

else:

print(person2+": "+str(contb)+" points")

print("WINNER: "+str(person1))

elif contb>conta:

if conta==1:

print(person1+": "+str(conta)+" point")

else:

print(person1+": "+str(conta)+" points")

if contb==1:

print(person2+": "+str(contb)+" point")

else:

print(person2+": "+str(contb)+" points")

print("WINNER: "+str(person2))

elif conta==contb:

if conta==1:

print(person1+": "+str(conta)+" point")

else:

print(person1+": "+str(conta)+" points")

if contb==1:

print(person2+": "+str(contb)+" point")

else:

print(person2+": "+str(contb)+" points")

print("TIED GAME")

cantidad=cantidad+1

print()

if bolean==True:

break

Train Swapping (Ordenamiento/Burbuja)

JAVA

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int cases, cant;

int[] vec;

cases = sc.nextInt();

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

cant = sc.nextInt();

vec = new int[cant];

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

vec[j] = sc.nextInt();

}

int res = burbuja(vec);

System.out.println("Optimal train swapping takes "+res+" swaps.");

}

}

public static int burbuja(int[] vec) {

int aux;

int cont = 0;

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

for (int j = 0; j < vec.length - 1; j++) {

if (vec[j] > vec[j + 1]) {

aux = vec[j + 1];

vec[j + 1] = vec[j];

vec[j] = aux;

cont++;

}

}

}

return cont;

}

}

# Summing Digits (Mat. modular)

Python

##########################

def revisar(a,b):

while(True):

if a<10:

b=b+(a%10)

return b

b=b+(a%10)

a=int(a/10)

#########################

numero=int(input())

while (numero != 0) :

res=revisar(numero,0)

while (res>9):

res=revisar(res,0)

print (res)

numero=int(input())

##########################3

# Binary Search Tree (Árbol binario)

JAVA

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

ArbolBinario arbolito = new ArbolBinario();

int numeroNodo;

while (sc.hasNext()) {

numeroNodo = sc.nextInt();

arbolito.agregarNodo(numeroNodo);

}

arbolito.postOrden(arbolito.raiz);

}

public static class NodoArbol {

int dato;

NodoArbol hijoIzquierdo, hijoDerecho;

public NodoArbol(int d) {

this.dato = d;

this.hijoDerecho = null;

this.hijoIzquierdo = null;

}

}

public static class ArbolBinario {

NodoArbol raiz;

public ArbolBinario() {

raiz = null;

}

public void agregarNodo(int d) {

NodoArbol nuevo = new NodoArbol(d);

if (raiz == null) {

raiz = nuevo;

} else {

NodoArbol auxiliar = raiz;

NodoArbol padre;

while (true) {

padre = auxiliar;

if (d < auxiliar.dato) {

auxiliar = auxiliar.hijoIzquierdo;

if (auxiliar == null) {

padre.hijoIzquierdo = nuevo;

return;

}

} else {

auxiliar = auxiliar.hijoDerecho;

if (auxiliar == null) {

padre.hijoDerecho = nuevo;

return;

}

}

}

}

}

public void postOrden(NodoArbol r) { //izquierda derecha raiz

if (r != null) {

postOrden(r.hijoIzquierdo);

postOrden(r.hijoDerecho);

System.out.println(r.dato);

}

}

}

}

# Baskets Of Gold Coins (Mat/Gran cantidad de casos)

C++

#include <iostream>

#include <cstdint>

#include <string>

#include <sstream>

using namespace std;

int main()

{

int64\_t N = 0;

int64\_t w = 0;

int64\_t d = 0;

int64\_t difference = 0;

int64\_t basketWithCoin = 0;

int64\_t sumWeightGiven = 0;

int64\_t sumWeightExpected = 0;

string input;

while(getline(cin, input))

{

stringstream stream(input);

stream >> N >> w >> d >> sumWeightGiven;

if(N < 2)

{

return 0;

}

sumWeightExpected = 0;

basketWithCoin = 0;

difference = 0;

for(int64\_t i = 1; i < N; i++)

{

sumWeightExpected += (i \* w);

}

difference = (sumWeightExpected - sumWeightGiven);

if(sumWeightExpected == sumWeightGiven)

{

basketWithCoin = N;

}

else

{

basketWithCoin = (difference / d);

}

cout << basketWithCoin << endl;

}

return 0;

}

# Big Number (Kamenetsky / Números grandes / Factorial)

C++

#include <bits/stdc++.h>

using namespace std;

long long findDigits(int n)

{

if (n < 0)

return 0;

if (n <= 1)

return 1;

// Kamenetsky formula

double x = ((n \* log10(n / M\_E) +

log10(2 \* M\_PI \* n) /

2.0));

return floor(x) + 1;

}

int main()

{

int t,n;

cin>>t;

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

cin>>n;

cout << findDigits(n) << endl;

}

return 0;

}

# Factorial Factors (Números grandes /Eratostenes)

C++;

#include <iostream>

#include <limits.h>

#include <cmath>

using namespace std;

const int N = 1000001;

unsigned long long nFactors[N];

void sieve() {

int i, j, t;

for (i = 0; i < N; ++i)

nFactors[i] = 0;

unsigned long long cum = 0;

for (i = 2; i < N; ++i) {

if (nFactors[i] == 0) {

nFactors[i] = 1;

if (i < INT\_MAX / 2)

for (j = 2 \* i; j > 0 && j < N; j += i) {

t = j;

do

++nFactors[j];

while ((t /= i) % i == 0);

}

}

nFactors[i] = (cum += nFactors[i]);

}

}

int main() {

sieve();

int n;

while (cin >> n)

cout << nFactors[n] << endl;

return 0;

}

# Sum It Up (Mat/Busqueda en profundidad)

JAVA

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.util.HashMap;

import java.util.StringTokenizer;

class Main {

public static int targetSum = 0;

public static HashMap<String, Boolean> map;

public static boolean hasSolution;

public static void dfs(int[] values, boolean[] flag, int currSum, int min, String str) {

if (currSum == targetSum) {

hasSolution = true;

System.out.println(str.substring(0, str.length() - 1));

} else if (currSum < targetSum) {

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

if (!flag[i]) {

String temp = str + values[i] + "+";

if (map.get(temp) == null) {

flag[i] = true;

map.put(temp, true);

dfs(values, flag, currSum + values[i], i, temp);

flag[i] = false;

}

}

}

}

}

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

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

String s;

while (!(s = br.readLine()).equals("0 0")) {

StringTokenizer st = new StringTokenizer(s);

targetSum = Integer.parseInt(st.nextToken());

int n = Integer.parseInt(st.nextToken());

int[] values = new int[n];

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

values[i] = Integer.parseInt(st.nextToken());

}

map = new HashMap<>();

hasSolution = false;

System.out.println("Sums of " + targetSum + ":");

dfs(values, new boolean[n], 0, 0, "");

if (!hasSolution) {

System.out.println("NONE");

}

}

}

}

# Roads In The North (Grafo/DFS)

C++

#include <iostream>

#include <algorithm>

#include <cstdio>

#include <cstring>

#include <vector>

using namespace std;

#define For(i, a, b) for (int i=a; i<=b; i++)

#define maxN 10001

int n, ans;

vector<int> adj[maxN];

vector<int> adjw[maxN];

bool Free[maxN];

int DFS(int u, int val) {

int tmpmax = 0, tmp;

bool leaf = true;

for (int i=0, size=adj[u].size(); i<size; i++) {

int v = adj[u][i];

if (!Free[v]) continue;

Free[v] = false;

tmp = DFS(v, adjw[u][i]);

ans = max(ans, tmpmax+tmp);

tmpmax = max(tmpmax, tmp);

leaf = false;

}

return tmpmax+val;

}

int main() {

int u, v, w;

char s[30];

while (gets(s)) {

For (i, 0, maxN-1) {

Free[i] = true;

adj[i].clear(); adjw[i].clear();

}

n = 0;

do {

if (strlen(s)==0) break;

sscanf(s, "%d %d %d", &u, &v, &w);

adj[u].push\_back(v); adjw[u].push\_back(w);

adj[v].push\_back(u); adjw[v].push\_back(w);

n = max(n, u); n = max(n, v);

} while (gets(s));

ans = 0;

Free[1] = false;

DFS(1, 0);

printf("%d\n", ans);

}

}

# Inception (Pila)

JAVA

import java.util.Scanner;

import java.util.Stack;

public class inception {

static Stack<String> pila= new Stack<>();

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int queries= sc.nextInt();

sc.nextLine();

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

String entrada = sc.nextLine();

switch (entrada.charAt(0)) {

case 'S':

int index = entrada.indexOf(" ");

String nombre = entrada.substring(index+1);

pila.push(nombre);

break;

case 'K':

if (!pila.isEmpty()) {

pila.pop();

} break;

case 'T':

if(!pila.isEmpty()){

System.out.println(pila.peek());

}else{

System.out.println("Not in a dream");

} break;

default:

break;

}

}

}

}

# Euclid Problem (Matemático/GCD)

ANSI C

#include <stdio.h>

int gcd(int a, int b) {

int tmp, flag = 0;

int x1 = 1, y1 = 0, x2 = 0, y2 = 1;

while(a%b) {

if(flag) {

x2 -= a/b\*x1;

y2 -= a/b\*y1;

} else {

x1 -= a/b\*x2;

y1 -= a/b\*y2;

}

tmp = a, a = b, b = tmp%b;

flag ^= 1;

}

if(flag)

printf("%d %d", x1, y1);

else

printf("%d %d", x2, y2);

printf(" %d\n", b);

return b;

}

int main() {

int A, B;

while(scanf("%d %d", &A, &B) == 2) {

gcd(A, B);

}

return 0;

}

# Bars (All sets)

C++

#include <bits/stdc++.h>

#include <cstdlib>

using namespace std ;

int t,n,p;

bool allSets(int set[]){

if(n==0)return true;

int suma=0;

int r=p;

vector<int> numeros;

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

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

numeros.push\_back(set[j]);

}

}

if(!numeros.size()==0){

for (int j=0;j<numeros.size();j++){

try{

suma+=numeros.at(j);

if(suma==n){

return true;

}

} catch(int e){

}

}

suma=0;

}

suma=0;

numeros.clear();

return false;

}

int main(){

cin>>t;

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

cin>>n;

cin>>p;

int barras[p];

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

int aux;

cin>>aux;

barras[j]=aux;

}

if(printsets(barras)){

cout<<"YES"<<endl;

}else{

cout<<"NO"<<endl;

}

}

}

# Budget Travel (Programación dinámica)

ANSI C

#include <stdio.h>

int n;

double target, size, mp, pos[99], price[99], min;

void run(double cost, double loc, double left, double now) {

  if (now == n - 1) {

    left -= (target - loc) / mp;

    if (left >= 0 && cost < min) min = cost;

    return;

  }

  int i;

  for (i = now + 1; i < n; i++)

    if (left \* mp >= (pos[i] - loc)) {

      double newLeft = left - (pos[i] - loc) / mp;

      run(cost, pos[i], newLeft, i);

      run(cost + (size - newLeft) \* price[i] \* 0.01 + 2, pos[i], size, i);

    }

}

int main() {

  int T = 1;

  while (scanf("%lf", &target) && target >= 0) {

    double cost;

    scanf("%lf%lf%lf%d", &size, &mp, &cost, &n);

    int i;

    for (i = 0; i < n; i++)

      scanf("%lf%lf", &pos[i], &price[i]);

    min = 2e9;

    run(cost, 0, size, -1);

    printf("Data Set #%d\nminimum cost = $%.2lf\n", T++, min);

  }

  return 0;

}

# Mountain Road (Programación dinamica)

C++

#include <stdio.h>

#include <string.h>

#include <math.h>

#include <queue>

#include <functional>

#include <deque>

#include <assert.h>

#include <algorithm>

using namespace std;

#define eps 1e-8

#define MAXN 256

#define INF 0x3f3f3f3f

int dp[MAXN][MAXN][2];

int main() {

int testcase, N;

char s[MAXN];

scanf("%d", &testcase);

while (testcase--) {

scanf("%d", &N);

vector< pair<int, int> > A, B;

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

int start, pass;

scanf("%s %d %d", s, &start, &pass);

if (s[0] == 'A')

A.push\_back(make\_pair(start, pass));

else

B.push\_back(make\_pair(start, pass));

}

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

for (int j = 0; j <= B.size(); j++) {

dp[i][j][0] = dp[i][j][1] = INF;

}

}

dp[0][0][0] = dp[0][0][1] = 0;

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

for (int j = 0; j <= B.size(); j++) {

if (dp[i][j][0] != INF)

for (int k = i, start = dp[i][j][0], end = 0; k < A.size(); k++) {

start = max(start, A[k].first);

end = max(end, start + A[k].second);

dp[k+1][j][1] = min(dp[k+1][j][1], end);

start = start + 10, end = end + 10;

}

if (dp[i][j][1] != INF)

for (int k = j, start = dp[i][j][1], end = 0; k < B.size(); k++) {

start = max(start, B[k].first);

end = max(end, start + B[k].second);

dp[i][k+1][0] = min(dp[i][k+1][0], end);

start = start + 10, end = end + 10;

}

}

}

int ret = min(dp[A.size()][B.size()][0], dp[A.size()][B.size()][1]);

printf("%d\n", ret);

}

return 0;

}

# Three powers (Números grandes / Bitwise)

JAVA

import java.io.\*;

import java.util.Scanner;

import java.math.BigInteger;

public class Main

{

BigInteger[] P = new BigInteger[ 70 ];

Main()

{

Scanner sc = new Scanner( System.in );

P[ 0 ] = BigInteger.ONE;

for (int i=1; i<70; i++) P[ i ] = P[ i-1 ].multiply( BigInteger.valueOf( 3 ) );

long n;

while ( ( n = sc.nextLong()) != 0 )

{

n--;

System.out.print("{");

boolean f = false;

for (int i=0; n!=0; i++, n >>= 1)

{

if ( (n & 1) == 1 )

{

if ( f ) System.out.print(",");

f = true;

System.out.print( " "+P[ i ] );

}

}

System.out.println(" }");

}

}

public static void main(String[] args)

{

new Main();

}

}