Министерство образования Республики Беларусь

Учреждение образования «Полоцкий государственный университет»

Факультет информационных технологий

Кафедра технологий программирования

**Отчёт по курсу «Алгоритмы: теория и практика. Методы »**

Выполнил: Шиковец Е.А.

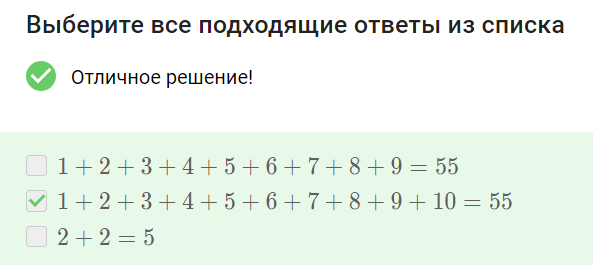
Группа 21-ИТ-3

Проверила: преподаватель

Виноградова А.Д.

Полоцк 2022

**Модуль 1. Обзор**

1. 
2. В данной задаче требуется вычислить сумму двух входных целых чисел, лежащих в отрезке от нуля до десяти. Никаких подвохов, это очевидная задача, предназначенная для того, чтобы познакомить вас с проверяющей системой. На следующем степе приведены решения данной задачи на нескольких языках программирования (вы можете прямо сейчас перейти туда и скопировать решение оттуда). В этой задаче, как и во всех задачах на программирование, не нужно проверять, что входные данные удовлетворяют требованиям, заявленным в условии. Другими словами, во всех тестах, на которых будет проверяться ваша программа, на вход будут подаваться два целых числа от 0 до 10.

#include <iostream>

int main(void) {

int a, b;

std::cin >> a >> b;

int result = a + b;

std::cout << result << std::endl;

return 0;

}

**Модуль 2. Введение**

1. Дано целое число 1 \le n \le 401≤*n*≤40, необходимо вычислить n*n*-е число Фибоначчи (напомним, что F\_0=0*F*0​=0, F\_1=1*F*1​=1 и F\_n=F\_{n-1}+F\_{n-2}*Fn*​=*Fn*−1​+*Fn*−2​ при n \ge 2*n*≥2).

#include <cassert>

#include <iostream>

using namespace std;

int main() {

int F0 = 0, F1 = 1, FT = 1;

int n;

cin >> n;

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

FT = F0 + F1;

F0 = F1;

F1 = FT;

}

cout << FT;

return 0;

}

2. Дано число 1 \le n \le 10^71≤*n*≤107, необходимо найти последнюю цифру n*n*-го числа Фибоначчи.

Как мы помним, числа Фибоначчи растут очень быстро, поэтому при их вычислении нужно быть аккуратным с переполнением. В данной задаче, впрочем, этой проблемы можно избежать, поскольку нас интересует только последняя цифра числа Фибоначчи: если 0 \le a,b \le 90≤*a*,*b*≤9 — последние цифры чисел F\_i*Fi*​ и F\_{i+1}*Fi*+1​ соответственно, то (a+b) \bmod{10}(*a*+*b*)mod10 — последняя цифра числа F\_{i+2}*Fi*+2​.

#include <iostream>

#include <stdio.h>

using namespace std;

int main()

{

int a = 1, b = 1, c = a + b, n;

cin >> n;

if (n <= 1)

c = 1;

else

for (int i = 3; i <= n; i++)

{

c = (a + b) % 10;

a = b;

b = c;

}

cout << c % 10;

return 0;

}

3.Даны целые числа 1 \le n \le 10^{18}1≤*n*≤1018 и 2 \le m \le 10^52≤*m*≤105, необходимо найти остаток от деления n*n*-го числа Фибоначчи на m*m*.

#include <iostream>

#include <stdio.h>

#include <math.h>

using namespace std;

long period(int64\_t m)

{

long a = 0;

long b = 1;

long res = 0;

for (int i = 0; i < m\*6; i++)

{

long temp = 0;

temp = b;

b = (a + b) % m;

a = temp;

if (a == 0 && b == 1)

res = i + 1;

}

return res;

}

long module(int64\_t n, int64\_t m)

{

long pisPeriod = period(m);

n = n % pisPeriod;

long a = 0;

long b = 1;

if (n == 0)

return 0;

else if (n == 1)

return 1;

for (int i = 0; i < n - 1; i++)

{

long temp = 0;

temp = b;

b = (a + b) % m;

a = temp;

}

return b % m;

}

int main()

{

int64\_t n;

int64\_t m;

cin >> n >> m;

cout << (module(n, m));

return 0;

}

4. По данным двум числам 1 \le a, b \le 2 \cdot 10^91≤*a*,*b*≤2⋅109 найдите их наибольший общий делитель.

#include <cassert>

#include <iostream>

template <class Int>

Int gcd(Int A, Int B) {

assert(A > 0);

if (B > 0) {

return gcd(B, A % B);

}

return A;

}

int main(void) {

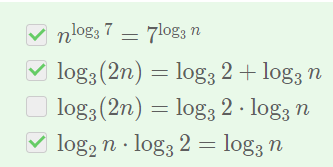
std::int64\_t A, B;

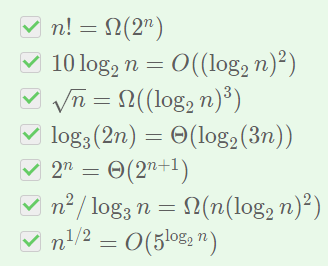
std::cin >> A >> B;

std::cout << gcd(A, B);

return 0;

}

5. 

6. 

7. 

8. 

**Модуль 4. Жадные алгоритмы**

1. По данным n*n* отрезкам необходимо найти множество точек минимального размера, для которого каждый из отрезков содержит хотя бы одну из точек.

В первой строке дано число 1 \le n \le 1001≤*n*≤100 отрезков. Каждая из последующих n*n* строк содержит по два числа 0 \le l \le r \le 10^90≤*l*≤*r*≤109, задающих начало и конец отрезка. Выведите оптимальное число m*m* точек и сами m*m* точек. Если таких множеств точек несколько, выведите любое из них.

#include <iostream>

#include <algorithm>

#include <utility>

#include <vector>

using std::vector;

struct Otrezok {

int start, end;

};

vector<int> optimal\_points(vector<Otrezok> &otrezki) {

std::sort(otrezki.begin(), otrezki.end(), [](const Otrezok &a, const Otrezok &b) -> bool {

return a.end < b.end;

});

vector<int> points;

int point = otrezki[0].end;

points.push\_back(point);

for (size\_t i = 1; i < otrezki.size(); ++i) {

if (point < otrezki[i].start || point > otrezki[i].end) {

point = otrezki[i].end;

points.push\_back(point);

}

}

return points;

}

int main() {

unsigned int n;

std::cin >> n;

vector<Otrezok> otrezki(n);

for (size\_t i = 0; i < otrezki.size(); ++i) {

std::cin >> otrezki[i].start >> otrezki[i].end;

}

vector<int> points = optimal\_points(otrezki);

std::cout << points.size() << "\n";

for (size\_t i = 0; i < points.size(); ++i) {

std::cout << points[i] << " ";

}

}

2. Первая строка содержит количество предметов 1 \le n \le 10^31≤*n*≤103 и вместимость рюкзака 0 \le W \le 2 \cdot 10^60≤*W*≤2⋅106. Каждая из следующих n*n* строк задаёт стоимость 0 \le c\_i \le 2\cdot 10^60≤*ci*​≤2⋅106 и объём 0 \lt w\_i \le 2\cdot 10^60<*wi*​≤2⋅106 предмета (n*n*, W*W*, c\_i*ci*​, w\_i*wi*​ — целые числа). Выведите максимальную стоимость частей предметов (от каждого предмета можно отделить любую часть, стоимость и объём при этом пропорционально уменьшатся), помещающихся в данный рюкзак, с точностью не менее трёх знаков после запятой.

#include <iostream>

#include <algorithm>

#include <cassert>

#include <vector>

#include <cinttypes>

#include <ios>

struct Item final {

int weight;

int value;

};

double get\_max\_backpack\_value(int capacity, std::vector <Item> items) {

std::sort(items.begin(), items.end(), [](const Item &lhs, const Item &rhs) {

return static\_cast<std::int64\_t>(lhs.weight) \* rhs.value <

static\_cast<std::int64\_t>(rhs.weight) \* lhs.value;

});

double value = 0.0;

// take items while there is empty space in backpack

for (auto &item:items) {

if (capacity > item.weight) {

// can take full item and continue

capacity -= item.weight;

value += item.value;

} else {

value += item.value \* (static\_cast <double>(capacity) / item.weight);

break;

}

}

return value;

}

int main(void) {

std::ios\_base::sync\_with\_stdio(false);

int number\_of\_items;

int backpack\_capacity;

std::cin >> number\_of\_items >> backpack\_capacity;

std::vector <Item> items(number\_of\_items);

for (int i = 0; i < number\_of\_items; i++) {

std::cin >> items[i].value >> items[i].weight;

}

double max\_backpack\_value = get\_max\_backpack\_value(backpack\_capacity, std::move(items));

std::cout.precision(10);

std::cout << max\_backpack\_value << std::endl;

return 0;

}

3. По данному числу 1 \le n \le 10^91≤*n*≤109 найдите максимальное число k*k*, для которого n*n* можно представить как сумму k*k* различных натуральных слагаемых. Выведите в первой строке число k*k*, во второй — k*k* слагаемых.

using System;

using System.Collections.Generic;

using System.Linq;

public class MainClass

{

public static void Main()

{

int n = int.Parse(Console.ReadLine());

List<int> k = new List<int>();

int i = 1;

int sum = 0;

while (sum < n)

{

k.Add(i);

sum += i;

if (sum > n)

{

sum -= k.Last();

k.RemoveAt(k.Count - 1);

while (sum != n)

{

sum -= k[k.Count - 1];

k[k.Count - 1] = k[k.Count - 1] + 1;

sum += k[k.Count - 1];

}

}

i++;

}

Console.WriteLine(k.Count);

k.ForEach(x =>

{

Console.Write($"{x} ");

});

}

}

4. По данной непустой строке s*s* длины не более 10^4104, состоящей из строчных букв латинского алфавита, постройте оптимальный беспрефиксный код. В первой строке выведите количество различных букв k*k*, встречающихся в строке, и размер получившейся закодированной строки. В следующих k*k* строках запишите коды букв в формате "letter: code". В последней строке выведите закодированную строку.

using System.Net;

using System.Security.Cryptography.X509Certificates;

using System.Linq;

using System.Collections.Generic;

using System;

static class Program

{

static string Result = string.Empty;

private class Node

{

public char? Value;

public Node? Left, Right;

public char VAL;

public string Code = string.Empty;

public Node(char value)

{

Value = value;

VAL = value;

}

public void AddNeighbour(Node left, Node right)

{

Value = null;

Left = left;

Right = right;

}

public void AddToCode(string dir)

{

Code = dir + Code;

Left?.AddToCode(dir);

Right?.AddToCode(dir);

}

}

private static void Main()

{

string n = Console.ReadLine();

Dictionary<char, int> letters = new();

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

{

if (!letters.ContainsKey(n[i]))

letters.Add(n[i], n.Count(x => x == n[i]));

}

Dictionary<Node, int> sorted = new();

foreach (var c in letters.OrderBy(k => k.Value))

{

sorted.Add(new Node(c.Key), c.Value);

}

Dictionary<Node, int> saveNodes = new();

if (sorted.Count == 1)

{

var s = sorted.First();

s.Key.Code = "0";

saveNodes.Add(s.Key, s.Value);

}

while (sorted.Count != 1)

{

var left = MinBy(sorted);

sorted.Remove(left.Key);

var right = MinBy(sorted);

sorted.Remove(right.Key);

var temp = new Node('t');

temp.AddNeighbour(left.Key, right.Key);

sorted.Add(temp, left.Value + right.Value);

left.Key.AddToCode("1");

right.Key.AddToCode("0");

if (left.Key.Value is not null)

{

saveNodes.Add(left.Key, left.Value);

}

if (right.Key.Value is not null)

{

saveNodes.Add(right.Key, right.Value);

}

sorted = SortByValue(sorted);

}

Dictionary<char, string> map = new();

foreach (var sn in saveNodes.OrderByDescending(x => x.Value))

{

map.Add(sn.Key.VAL,sn.Key.Code);

}

string res = String.Empty;

foreach (var v in n)

{

res += map[v];

}

Console.WriteLine($"{letters.Count} {res.Length}");

foreach (var sn in saveNodes.OrderByDescending(x => x.Value))

{

Console.WriteLine($"{sn.Key.Value}: {sn.Key.Code}");

}

Console.WriteLine(res);

}

private static Dictionary<Node, int> SortByValue(Dictionary<Node, int> letters)

{

Dictionary<Node, int> sorted = new();

foreach (var c in letters.OrderBy(k => k.Value))

{

sorted.Add(c.Key, c.Value);

}

return sorted;

}

private static KeyValuePair<Node, int> MinBy(Dictionary<Node, int> letters)

{

Dictionary<Node, int> sorted = new();

foreach (var c in letters.OrderBy(k => k.Value))

{

return new KeyValuePair<Node, int>(c.Key, c.Value);

}

return new KeyValuePair<Node, int>();

}

}

5. Восстановите строку по её коду и беспрефиксному коду символов.

В первой строке входного файла заданы два целых числа k*k* и l*l* через пробел — количество различных букв, встречающихся в строке, и размер получившейся закодированной строки, соответственно. В следующих k*k* строках записаны коды букв в формате "letter: code". Ни один код не является префиксом другого. Буквы могут быть перечислены в любом порядке. В качестве букв могут встречаться лишь строчные буквы латинского алфавита; каждая из этих букв встречается в строке хотя бы один раз. Наконец, в последней строке записана закодированная строка. Исходная строка и коды всех букв непусты. Заданный код таков, что закодированная строка имеет минимальный возможный размер.

В первой строке выходного файла выведите строку s*s*. Она должна состоять из строчных букв латинского алфавита. Гарантируется, что длина правильного ответа не превосходит 10^4104 символов.

using System;

using System.Collections.Generic;

using System.Linq;

public class Program

{

public static void Main()

{

Dictionary<string, char> map = new Dictionary<string, char>();

string lengths = Console.ReadLine();

var l = lengths.Split(" ");

int uniqueChars = Int16.Parse(l.First());

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

{

var input = Console.ReadLine().Split(": ");

map.Add(input.Last(), input.First()[0]);

}

string coded = Console.ReadLine();

string decoded = String.Empty;

string token = string.Empty;

foreach (var ch in coded)

{

token += ch;

if (map.ContainsKey(token))

{

decoded += map[token];

token = String.Empty;

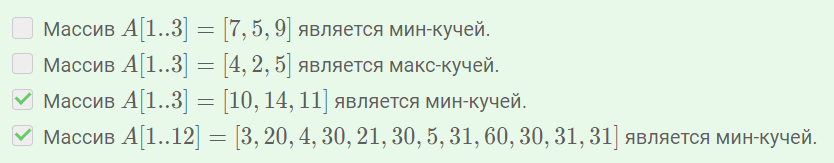
}

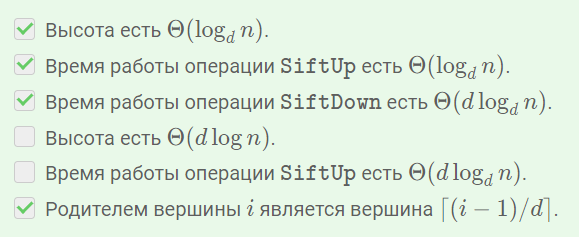
}

Console.WriteLine(decoded);

}

}

6. 

7. 

**Модуль 6. Разделяй и властвуй**

1. В первой строке даны целое число 1 \le n \le 10^51≤*n*≤105 и массив A[1 \ldots n]*A*[1…*n*] из n*n* различных натуральных чисел, не превышающих 10^9109, в порядке возрастания, во второй — целое число 1 \le k \le 10^51≤*k*≤105 и k*k* натуральных чисел b\_1, \ldots, b\_k*b*1​,…,*bk*​, не превышающих 10^9109. Для каждого i*i* от 1 до k*k* необходимо вывести индекс 1 \le j \le n1≤*j*≤*n*, для которого A[j]=b\_i*A*[*j*]=*bi*​, или -1−1, если такого j*j* нет.

using System;

using System.Collections.Generic;

public class MainClass

{

private static Dictionary<int,int> FillList(){

string[] input = Console.ReadLine().Split(' ');

Dictionary<int,int> list = new Dictionary<int,int>();

for(int i = 1; i <= int.Parse(input[0]); i++){

list.Add(int.Parse(input[i]), i);

}

return list;

}

private static Dictionary<int,int> FillListUn(){

string[] input = Console.ReadLine().Split(' ');

Dictionary<int,int> list = new Dictionary<int,int>();

for(int i = 1; i <= int.Parse(input[0]); i++){

list.Add(i, int.Parse(input[i]));

}

return list;

}

public static void Main()

{

Dictionary<int,int> x = FillList();

Dictionary<int,int> y = FillListUn();

foreach(KeyValuePair<int,int> i in y){

if (x.ContainsKey(i.Value)){

Console.Write($"{x[i.Value]} ");

}

else

{

Console.Write("-1 ");

}

}

}

}

2. Первая строка содержит число 1 \le n \le 10^51≤*n*≤105, вторая — массив A[1\ldots n]*A*[1…*n*], содержащий натуральные числа, не превосходящие 10^9109. Необходимо посчитать число пар индексов 1 \le i \lt j \le n1≤*i*<*j*≤*n*, для которых A[i] \gt A[j]*A*[*i*]>*A*[*j*]. (Такая пара элементов называется инверсией массива. Количество инверсий в массиве является в некотором смысле его мерой неупорядоченности: например, в упорядоченном по неубыванию массиве инверсий нет вообще, а в массиве, упорядоченном по убыванию, инверсию образуют каждые два элемента.)

using System;

using System.Linq;

static class Program

{

private static long inverseCount;

public static int[] MergeSort(int[] array)

{

inverseCount = 0;

if (array == null) return null!;

if (array.Length == 0) return array;

return MergeSort(array, 0, array.Length - 1);

}

static int[] MergeSort(int[] array, int low, int high)

{

if (low > high - 1) return new int[] { array[low] };

int mid = low + (high - low) / 2;

return Merge(MergeSort(array, low, mid), MergeSort(array, mid + 1, high));

}

static int[] Merge(int[] array1, int[] array2)

{

int cursor1 = 0;

int cursor2 = 0;

int[] merged = new int[array1.Length + array2.Length];

int mergedCursor = 0;

while (cursor1 < array1.Length && cursor2 < array2.Length)

{

if (array1[cursor1] <= array2[cursor2])

merged[mergedCursor++] = array1[cursor1++];

else

{

merged[mergedCursor++] = array2[cursor2++];

inverseCount += (array1.Length - cursor1);

}

}

while (cursor1 < array1.Length)

merged[mergedCursor++] = array1[cursor1++];

while (cursor2 < array2.Length)

merged[mergedCursor++] = array2[cursor2++];

return merged;

}

static void Main()

{

var inputN = int.Parse(Console.ReadLine()!);

string stringInput = Console.ReadLine()!;

int[] array = stringInput.Split(new[] { ' ' }, StringSplitOptions.RemoveEmptyEntries).Select(int.Parse).ToArray();

MergeSort(array);

System.Console.WriteLine(inverseCount);

}

}

3. В первой строке задано два целых числа 1 \le n \le 500001≤*n*≤50000 и 1 \le m \le 500001≤*m*≤50000 — количество отрезков и точек на прямой, соответственно. Следующие n*n* строк содержат по два целых числа a\_i*ai*​ и b\_i*bi*​ (a\_i \le b\_i*ai*​≤*bi*​) — координаты концов отрезков. Последняя строка содержит m*m* целых чисел — координаты точек. Все координаты не превышают 10^8108 по модулю. Точка считается принадлежащей отрезку, если она находится внутри него или на границе. Для каждой точки в порядке появления во вводе выведите, скольким отрезкам она принадлежит.

#include <iostream>

#include <algorithm>

#include <cmath>

int main()

{

int n, k;

std::cin >> n >> k;

int \*L = new int[n];

int \*R = new int[n];

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

std::cin >> L[i] >> R[i];

std::sort(L, L + n);

std::sort(R, R + n);

for (int i = 0, x, cnt = 0; i < k; ++i)

{

std::cin >> x;

int left = 0, right = n - 1, m;

while (left <= right)

{

m = (left + right) / 2; //середина

if (L[m] > x)

right = m - 1; //справа отрезки правее точки

else

left = m + 1;

}

cnt = right + 1;

left = 0;

right = n - 1;

while (left <= right)

{

m = (left + right) / 2; //середина

if (R[m] < x)

left = m + 1; //слева отрезки закончились раньше x

else

right = m - 1;

}

cnt -= right + 1;

std::cout << cnt << " ";

}

delete[] L, R;

return 0;

}

4. Первая строка содержит число 1 \le n \le 10^41≤*n*≤104, вторая — n*n* натуральных чисел, не превышающих 10. Выведите упорядоченную по неубыванию последовательность этих чисел.

using System;

using System.Linq;

using System.Collections.Generic;

public class MainClass

{

public static void Main()

{

Console.ReadLine();

var inputString = Console.ReadLine()!.Split(' ');

var input = inputString.Select(x => int.Parse(x)).ToList();

var counts = new Dictionary<int, int>();

foreach (var x in input.Where(x => !counts.ContainsKey(x)))

{

counts.Add(x, input.Count(y => y == x));

}

var output = new List<int>();

for (var k = 0; k < 11; k++)

{

if (!counts.ContainsKey(k)) continue;

for (var j = 0; j < counts[k]; j++)

output.Add(k);

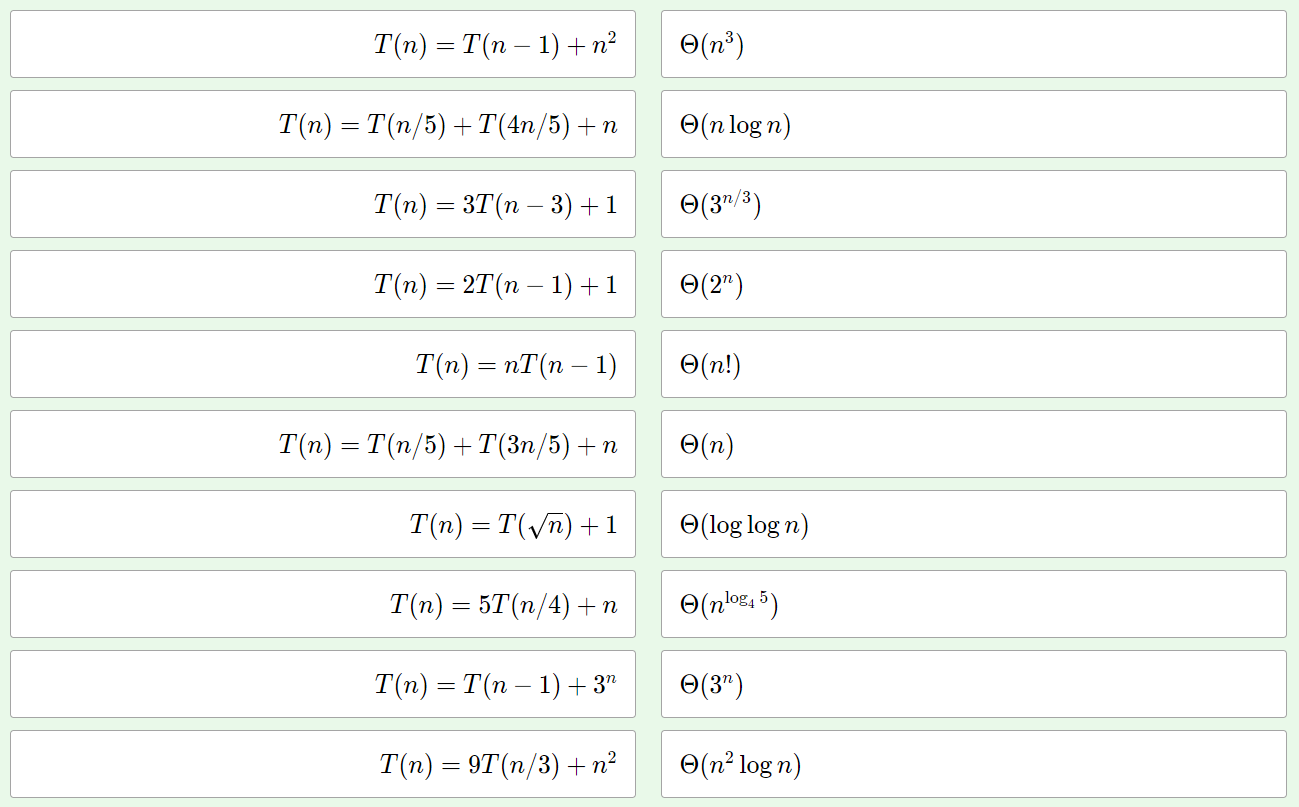
}

Console.WriteLine($"{string.Join(" ", output)}");

}

}

5. 

6. 

**Модуль 8. Динамическое программирование**

1. Дано целое число 1 \le n \le 10^31≤*n*≤103 и массив A[1\ldots n]*A*[1…*n*] натуральных чисел, не превосходящих 2 \cdot 10^92⋅109. Выведите максимальное 1 \le k \le n1≤*k*≤*n*, для которого найдётся подпоследовательность 1 \le i\_1 \lt i\_2 \lt \ldots \lt i\_k \le n1≤*i*1​<*i*2​<…<*ik*​≤*n* длины k*k*, в которой каждый элемент делится на предыдущий (формально: для  всех 1 \le j \lt k1≤*j*<*k*, A[i\_j]\, | \,A[i\_{j+1}]*A*[*ij*​]∣*A*[*ij*+1​]).

import sys

def get\_max\_sequence\_len(array: list, n: int) -> int:

# Max len sequence

max\_len = 0

# Array of sequence length

len\_arr = [-1] \* n

for i in range(0, n):

# Set len 1 for first element

len\_arr[i] = 1

for p in range(i - 1, -1, -1):

if array[p] != 0 and array[i] >= array[p] and array[i] % array[p] == 0 and len\_arr[p] > (len\_arr[i] - 1):

len\_arr[i] = len\_arr[p] + 1

# Set max len

if len\_arr[i] > max\_len:

max\_len = len\_arr[i]

return max\_len

def main():

n = int(sys.stdin.readline())

array = list(map(int, sys.stdin.readline().split()))

print(get\_max\_sequence\_len(array, n))

if \_\_name\_\_ == '\_\_main\_\_':

main()

2. Дано целое число 1 \le n \le 10^51≤*n*≤105и массив A[1 \ldots n]*A*[1…*n*], содержащий неотрицательные целые числа, не превосходящие 10^9109. Найдите наибольшую невозрастающую подпоследовательность в A*A*. В первой строке выведите её длину k*k*, во второй — её индексы 1\le i\_1 \lt i\_2\lt \ldots \lt i\_k \le n1≤*i*1​<*i*2​<…<*ik*​≤*n* (таким образом, A[i\_1] \ge A[i\_2] \ge \ldots \ge A[i\_n]*A*[*i*1​]≥*A*[*i*2​]≥…≥*A*[*in*​]).

import sys

def get\_max\_sequence\_len(array: list, n: int) -> list:

# Infinitive

inf = 10 \*\* 10

max\_sequence\_end\_number = [inf] + [-inf] \* n

# Previous sequence node

prev\_sequence\_node = [] \* (n + 1)

# Fill max\_sequence\_end\_number and prev\_sequence\_node

for i in range(n):

left, right = 0, n

while right - left > 1:

middle = (left + right) // 2

if max\_sequence\_end\_number[middle] < array[i]:

right = middle

else:

left = middle

max\_sequence\_end\_number[right] = array[i]

prev\_sequence\_node.append([right, i, array[i]])

# Find correct end of array max\_sequence\_end\_number

i = n

while max\_sequence\_end\_number[i] == -inf:

i = i - 1

# Restore correct array from prev\_sequence\_node

counter = len(max\_sequence\_end\_number[1: i + 1])

result\_array = []

for element in reversed(prev\_sequence\_node):

if element != -1 and element[0] == counter:

counter -= 1

result\_array.append(element[1] + 1)

return list(reversed(result\_array))

def main():

n = int(sys.stdin.readline())

array = list(map(int, sys.stdin.readline().split()))

result = get\_max\_sequence\_len(array, n)

print(len(result), '\n', ' '.join(map(str, result)), end='', sep='')

if \_\_name\_\_ == '\_\_main\_\_':

main()

3. Первая строка входа содержит целые числа 1 \le W \le 10^41≤*W*≤104 и 1 \le n \le 3001≤*n*≤300 — вместимость рюкзака и число золотых слитков. Следующая строка содержит n*n* целых чисел 0 \le w\_1, \ldots, w\_n \le 10^50≤*w*1​,…,*wn*​≤105, задающих веса слитков. Найдите максимальный вес золота, который можно унести в рюкзаке.

import sys

def find\_max\_bag\_width(bag\_width: int, item\_count: int, items: list) -> int:

# data\_matrix = [[0] \* (bag\_width + 1)] \* (item\_count + 1)

data\_matrix = [[0] \* (bag\_width + 1) for item in range(item\_count + 1)]

for i in range(1, item\_count + 1):

for w in range(1, bag\_width + 1):

data\_matrix[i][w] = data\_matrix[i - 1][w]

# If we can store this item

if w >= items[i - 1][0]:

# print(i, w)

data\_matrix[i][w] = max(data\_matrix[i][w], data\_matrix[i - 1][w - items[i - 1][0]] + items[i - 1][1])

return data\_matrix[item\_count][bag\_width]

def main():

bag\_width, item\_count = map(int, sys.stdin.readline().split())

items = [[width, width] for width in map(int, sys.stdin.readline().split())]

print(find\_max\_bag\_width(bag\_width, item\_count, items))

if \_\_name\_\_ == '\_\_main\_\_':

main()

4. У вас есть примитивный калькулятор, который умеет выполнять всего три операции с текущим числом x*x*: заменить x*x* на 2x2*x*, 3x3*x* или x+1*x*+1. По данному целому числу 1 \le n \le 10^51≤*n*≤105 определите минимальное число операций k*k*, необходимое, чтобы получить n*n* из 1. Выведите k*k* и последовательность промежуточных чисел.

import sys

def step\_calculate(n: int):

min\_operations = [-1]

inf = 10 \*\* 10

for i in range(1, n + 1):

x, y, z = min\_operations[i - 1] + 1, inf, inf

if i % 2 == 0:

y = min\_operations[i // 2] + 1

if i % 3 == 0:

z = min\_operations[i // 3] + 1

min\_operations.append(min(x, y, z))

result, num = [n], n

operations\_count = min\_operations[-1]

for i in range(n - 1, -1, -1):

if min\_operations[i] == operations\_count - 1 and (num / 3 == i or num / 2 == i or num == i + 1):

result.append(i)

operations\_count -= 1

num = i

return min\_operations[-1], reversed(result[:-1])

def main():

n = int(sys.stdin.readline())

count, result = step\_calculate(n)

print(count)

print(' '.join(map(str, result)))

if \_\_name\_\_ == '\_\_main\_\_':

main()