|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Министерство образования Республики Беларусь  Учреждение образования «Полоцкий государственный университет имени Евфросинии Полоцкой» | |  | | | Министерство образования Республики Беларусь  Учреждение образования «Полоцкий государственный университет имени Евфросинии Полоцкой» | |
|  | Кафедра технологий программирования | |  | Кафедра технологий программирования | |
| Алгоритмы и структуры данных  Отчет по лабораторной работе №1  Вариант 10 | |  | | | Алгоритмы и структуры данных  Отчет по лабораторной работе №1 | |
| Выполнил | Купаленко А.Д., студент гр. 21-ИТ-1,ФИТ | | Выполнил | Купаленко А.Д., студент гр. 21-ИТ-1, ФИТ | |
| Проверил | Виноградова А.Д., Преподаватель кафедры ТП. | | Проверил | Виноградова А.Д., Преподаватель-стажер кафедры ТП. | |
| Полоцк  2022г. | |  | | | Полоцк | |

**Модуль № 2**

**“Числа Фибоначчи”**

**Задача 1:** Дано целое число 1≤*n*≤40, необходимо вычислить *n*-е число Фибоначчи.

**Решение**

#include <cassert>

#include <iostream>

long int fib (int n) {

if(n==1 || n==2)

return 1;

else

return fib(n-1)+fib(n-2);

}

int main ()

{

int n;

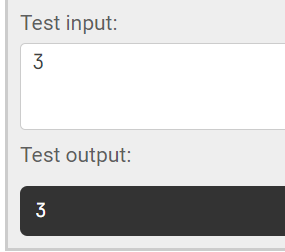
std::cin>>n;

std::cout << fib (n);

return 0;

}

*Листинг программы- Подсчет n-ого числа фибоначчи*

**

*Рисунок 1 - Результат работы программы*

**Задача 2:** Дано число 1≤*n*≤10^7 необходимо найти последнюю цифру *n*-го числа Фибоначчи.

**Решение**

#include <iostream>

long int fib (int n) {

int a=0, b=1, c=a+b;

if (n <= 1)

c = 1;

else

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

{

c = (a + b)%10;

a = b;

b = c;

}

return c;

}

int main ()

{

int n;

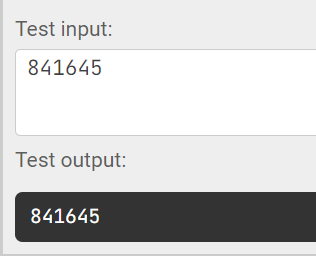
std::cin>>n;

std::cout << fib (n);

return 0;

}

*Листинг программы - Подсчет последнего числа числа фибоначчи*

**

*Рисунок 2 - Результат работы программы*

**Задача 3:** Даны целые числа 1≤n≤10^18 и 2≤m≤10^5, необходимо найти остаток от деления n-го числа Фибоначчи на m.

**Решение**

#include <iostream>

#include <vector>

int main()

{

long long n = 0;

long long m = 0;

std::cin >> n >> m;

std::vector<long long> fib(2);

fib[0] = 0;

fib[1] = 1;

long long t = 0;

for (long long j = 2; j < m\*m+1; ++j) {

fib.push\_back((fib[j - 1] + fib[j - 2]) % m);

++t;

if ((fib[j] == 1) && (fib[j - 1] == 0)) break;

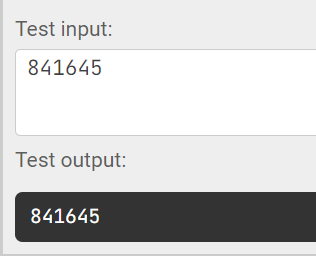
}

std::cout << fib[(n % t)] << std::endl;

return 0;

}

*Листинг программы- Решение задачи*

**

*Рисунок 3 - Результат работы программы*

**Задача 4:** По данным двум числам 1≤*a*,*b*≤2⋅10^2 найдите их наибольший общий делитель.

**Решение**

using System;

public class Program

{

public static void Main()

{

string input = Console.ReadLine();

int x = int.Parse(input.Split(' ')[0]);

int y = int.Parse(input.Split(' ')[1]);

Console.WriteLine(NOD(x,y));

}

private static int NOD(int x, int y)

{

if(y == 0)

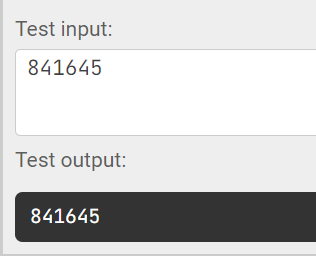
return x;

return NOD(y, x % y);

}

}

*Листинг программы- Решение задачи*

**

*Рисунок 3 - Результат работы программы*

**Модуль № 4**

**“Жадные алгоритмы”**

**Задача 1:** По данным *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≤*n*≤100 отрезков. Каждая из последующих *n* строк содержит по два числа, задающих начало и конец отрезка. Выведите оптимальное число *m* точек и сами *m* точек. Если таких множеств точек несколько, выведите любое из них.

**Решение**

using System;

using System.Collections.Generic;

public class MainClass

{

struct Segment : IComparable<Segment>

{

public int left;

public int right;

public Segment(string[] str)

{

left = int.Parse(str[0]);

right = int.Parse(str[1]);

}

public int CompareTo(Segment other)

{

return right.CompareTo(other.right);

}

public override string ToString()

{

return $"[{left}, {right}]";

}

}

public static void Main()

{

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

Segment[] array = new Segment[n];

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

{

array[i] = new Segment(Console.ReadLine().Split());

}

Array.Sort(array);

List<int> set = new List<int>(n);

int currentPoint = -1;

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

{

if (currentPoint < array[i].left)

{

currentPoint = array[i].right;

set.Add(currentPoint);

}

}

Console.WriteLine(set.Count);

foreach (int point in set)

{

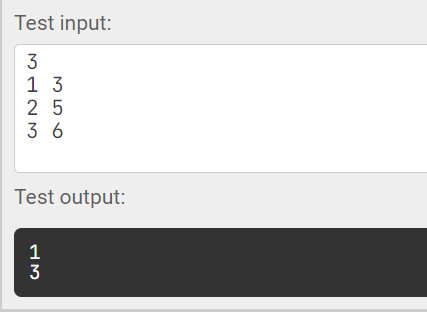
Console.Write($"{point} ");

}

}

**}**

*Листинг программы - Оптимальное количество точек*

**

*Рисунок 4 - Результат работы программы*

**Задача 2:** Первая строка содержит количество предметов и вместимость рюкзака. Каждая из следующих *n* строк задает стоимость и объём предмета. 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.

**Решение**

using System.Collections.Generic;

using System;

using System.Linq;

namespace ContinuousBackpack

{

static class Program

{

struct Item

{

public double Value;

public double Volume;

public double ValuePerVolume;

public Item(string[] input)

{

Value = double.Parse(input[0]);

Volume = double.Parse(input[1]);

ValuePerVolume = Value / Volume;

}

}

class Backpack

{

public double Capacity;

private double \_value = 0;

public Backpack(string capacity)

{

Capacity = double.Parse(capacity);

}

public double CalculateValue(List<Item> items)

{

while (items.Count > 0)

{

if (Capacity > 0)

{

if (Capacity >= items[0].Volume)

{

\_value += items[0].Value;

Capacity -= items[0].Volume;

}

else

{

\_value += Capacity \* items[0].ValuePerVolume;

break;

}

}

items = items.GetRange(1, items.Count - 1);

}

return \_value;

}

}

static void Main()

{

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

List<Item> items = new List<Item>();

Backpack backpack = new Backpack(backpackInput[1]);

for (int i = 0; i < int.Parse(backpackInput[0]); i++)

{

items.Add(new Item(Console.ReadLine().Split(' ')));

}

items = items.OrderByDescending(x => x.ValuePerVolume).ToList();

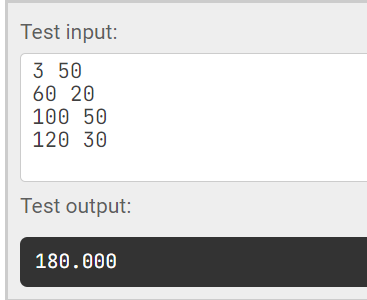
Console.WriteLine($"{backpack.CalculateValue(items):F3}");

}

}

}

*Листинг программы- Решение задачи*

**

*Рисунок 5 - Результат работы программы*

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

**Решение**

#include <iostream>

#include <cmath>

int main()

{

unsigned int number = 0;

// Считываем число

std::cin >> number;

// Находим число k, такое что сумма чисел от 1 до k меньше либо равна нашему числу number

double kd = (-1.0 + std::sqrt(1.0 + 8.0 \* ((double)number))) / 2.0;

unsigned int k = (unsigned int)kd; // Отбрасываем дробную часть

// Выводим количество слагаемых

std::cout << k << std::endl;

// Выводим натуральные числа от 1 до k-1 и результат выражения от number - sum(k-1)

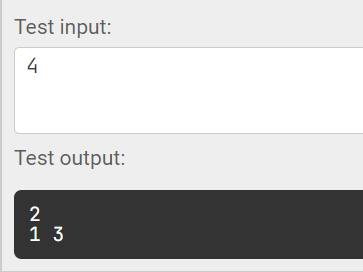
for (unsigned int i = 1; i <= k - 1; ++i) std::cout << i << " ";

std::cout << (number - (k - 1) \* k / 2) << std::endl;

return 0;

}

*Листинг программы - Решение задачи*

**

*Рисунок 6 - Результат работы программы*

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

**Решение**

#include <iostream>

#include <map>

#include <utility>

#include <algorithm>

#include <queue>

#include <iterator>

using namespace std;

using HuffCode = typename std::vector<bool>;

using HuffCodeMap = typename std::map<char, HuffCode>

class INode {

public:

const int f;

virtual ~INode() {

protected:

INode(int f) : f{f} {}

};

class InternalNode : public INode {

public:

INode \* const left;

INode \* const right;

InternalNode(INode \*c0, INode \*c1) : INode{c0->f + c1->f}, left{c0}, right{c1} {}

InternalNode(INode \*c) : INode{c->f}, left{c}, right{nullptr} {}

~InternalNode() {

delete left;

delete right;

}

};

class LeafNode : public INode {

public:

const char c;

LeafNode(int f, char c) : INode{f}, c{c} {}

};

struct NodeCmp {

bool operator()(const INode\* lhs, const INode\* rhs) const { return lhs->f > rhs->f; }

};

INode\* BuildTree(const map<char, int>& freq) {

std::priority\_queue<INode \*, std::vector<INode \*>, NodeCmp> trees;

for (auto cit = freq.cbegin(); cit != freq.end(); ++cit) {

trees.push(new LeafNode(cit->second, cit->first));

}

if (trees.size() == 1) {

INode \*childR = trees.top();

trees.pop();

INode \*parent = new InternalNode(childR);

trees.push(parent);

}

else {

while (trees.size() > 1) {

INode \*childR = trees.top();

trees.pop();

INode \*childL = trees.top();

trees.pop();

INode \*parent = new InternalNode(childR, childL);

trees.push(parent);

}

}

return trees.top();

}

void GenerateCodes(const INode\* node, const HuffCode& prefix, HuffCodeMap& outCodes) {

if (const LeafNode \*lf = dynamic\_cast<const LeafNode \*>(node)) {

outCodes[lf->c] = prefix;

}

else if (const InternalNode \*in = dynamic\_cast<const InternalNode \*>(node)) {

HuffCode leftPrefix = prefix;

leftPrefix.push\_back(false);

if (in->left) {

GenerateCodes(in->left, leftPrefix, outCodes);

}

HuffCode rightPrefix = prefix;

rightPrefix.push\_back(true);

if (in->right) {

GenerateCodes(in->right, rightPrefix, outCodes);

}

}

}

map<char, int> count\_freq(string& s) {

map<char, int> freq\_map;

for (auto& c : s) {

++freq\_map[c];

}

return freq\_map;

}

int main() {

string s;

vector<bool> code\_string;

vector<bool> prefix;

cin >> s;

auto frequencies = count\_freq(s);

INode \*root = BuildTree(frequencies);

HuffCodeMap codes;

GenerateCodes(root, prefix, codes);

delete root;

for (auto &i : s) {

for (const auto &code : codes[i]) {

code\_string.push\_back(code);

}

}

cout << frequencies.size() << ' ' << code\_string.size() << endl;

for (auto cit = codes.cbegin(); cit != codes.cend(); ++cit) {

std::cout << cit->first << ": ";

std::copy(cit->second.begin(), cit->second.end(),

std::ostream\_iterator<bool>(std::cout));

std::cout << std::endl;

}

for (int i = 0; i < code\_string.size(); ++i) {

cout << code\_string[i];

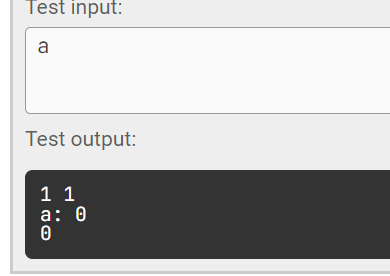
}

cout << endl;

return 0;

}

*Листинг программы - Решение задачи*

**

*Рисунок 7 - Результат работы программы*

**Задача 5:** Восстановите строку по её коду и беспрефиксному коду символов. В первой строке входного файла заданы два целых числа kk и ll через пробел — количество различных букв, встречающихся в строке, и размер получившейся закодированной строки, соответственно. В следующих kk 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.

**Решение**

#include <iostream>

#include <string>

#include <map>

using namespace std;

map<string, char> read\_codes(const int n) {

map<string, char> code\_map;

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

char c, delim;

string code;

cin >> c >> delim >> code;

code\_map[code] = c;

}

return code\_map;

}

int main() {

int n, size;

string enc\_str;

cin >> n >> size;

auto code\_map = read\_codes(n);

getline(cin, enc\_str);

string cur\_code;

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

cur\_code.push\_back(cin.get());

if (code\_map.find(cur\_code) != code\_map.end()) {

cout << code\_map[cur\_code];

cur\_code.clear();

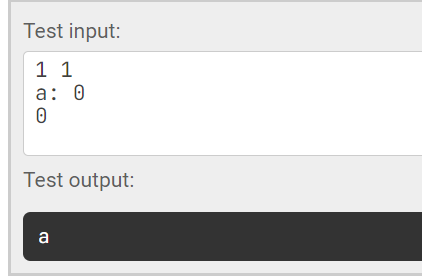
}

}

return 0;

}

*Листинг программы - Решение задачи*

**

*Рисунок 8 - Результат работы программы*

**Задача 6:** Первая строка входа содержит число операций n Каждая из последующих *n* строк задают операцию одного из следующих двух типов:

* Insert x — целое число;
* ExtractMax.

Первая операция добавляет число *x* в очередь с приоритетами, вторая — извлекает максимальное число и выводит его.

**Решение**

using System;

using System.Collections;

public class MainClass

{

public static void Main()

{

var input = Console.ReadLine();

var commandCount = int.Parse(input);

PriorityQueue<int> queue = new PriorityQueue<int>();

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

{

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

switch(commands[0])

{

case "Insert":

{

var value = int.Parse(commands[1]);

queue.Add(value, value); break;

}

case "ExtractMax": Console.WriteLine(queue.Poll()); break;

}

}

}

class PriorityQueue<T>

{

struct Item<T>

{

public T value;

public int priority;

public Item(T value, int priority)

{

this.value = value;

this.priority = priority;

}

public bool Compare(Item<T> item)

{

return priority > item.priority;

}

}

static ArrayList items = new ArrayList();

private void BinAdd(Item<T> item)

{

int i = -1;

int j = items.Count;

int avr;

while(i + 1 < j)

{

avr = (i + j) >> 1;

if(item.Compare((Item<T>)items[avr])) {

j = avr;

}

else i = avr;

}

items.Insert(++i, item);

}

public int Count

{

get

{

return items.Count;

}

}

public void Add(T new\_value, int new\_priority)

{

BinAdd(new Item<T>(new\_value, new\_priority));

}

public T Poll()

{

Item<T> top\_item = (Item<T>)items[0];

items.Remove(top\_item);

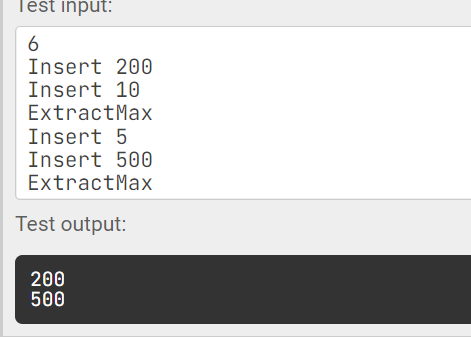
return top\_item.value;

}

}

}

*Листинг программы - Решение задачи*

**

*Рисунок 9 - Результат работы программы*

**Модуль № 6**

**“Разделяй и властвуй”**

**Задача 1:** В первой строке даны целое число 1≤n≤10 и массив A[1…n] из n различных натуральных чисел, в порядке возрастания, во второй — целое число, не превышающих 10^9. Для каждого ii от 1 до kk необходимо вывести индекс 1≤j≤n, для которого A[j]=b\_iA[j]=b, или -1−1, если такого j вывести “нет”.

**Решение**

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 ");

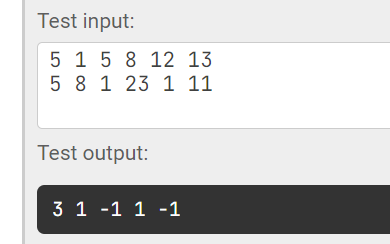
}

}

}

}

*Листинг программы- Двоичный поиск*

**

*Рисунок 10 - Результат работы программы*

**Задача 2:** Первая строка содержит число 1≤n≤10^5, вторая — массив A[1\ldots n]A[1…n], содержащий натуральные числа, не превосходящие 10^9. 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.)

**Решение**

using System;

using System.Collections.Generic;

using System.Linq;

public class MainClass

{

class Program

{

static long inversion;

static void Main()

{

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

List<int> array = Console.ReadLine().Split(' ').Select(x => int.Parse(x)).ToList();

//страшный код для наращивания нулей вначале

if (Math.Pow(2, ((int)Math.Log(arrayLength, 2))) != arrayLength) //проверка на степень двойки

{

int numberZeros = (int)Math.Pow(2, (int)Math.Log(arrayLength, 2)+1) - arrayLength; //сколько нулей надо добавить

int[] arrayNull = new int[numberZeros];

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

buff.AddRange(arrayNull);

buff.AddRange(array);

array = buff;

}

List<int> sortArray = IterativeMergeSort(array);

Console.WriteLine(inversion);

Console.ReadLine();

}

static List<int> IterativeMergeSort(List<int> array)

{

Queue<List<int>> queueArrays = new Queue<List<int>>();

for(int i = 0; i<array.Count; i++)

{

queueArrays.Enqueue(new List<int> { array[i] });

}

while (queueArrays.Count > 1)

{

queueArrays.Enqueue(Merge(queueArrays.Dequeue(), queueArrays.Dequeue()));

}

return queueArrays.Dequeue();

}

static List<int> Merge(List<int> arrayOne, List<int> arrayTwo)

{

List<int> arraySum = new List<int>(arrayOne.Count + arrayTwo.Count);

for(int i = 0; i < arraySum.Capacity; i++)

{

if(arrayOne.Count == 0)

{

arraySum.AddRange(arrayTwo);

break;

}

if(arrayTwo.Count == 0)

{

arraySum.AddRange(arrayOne);

break;

}

if(arrayOne.First() <= arrayTwo.First())

{

arraySum.Add(arrayOne.First());

arrayOne.RemoveAt(0);

}

else

{

arraySum.Add(arrayTwo.First());

arrayTwo.RemoveAt(0);

inversion += arrayOne.Count;

}

}

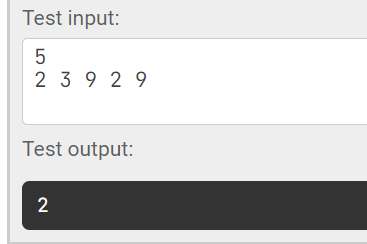
return arraySum;

}

}

}

*Листинг программы- Подсчет инверсий*

**

*Рисунок 11 - Результат работы программы*

**Задача 3:** В первой строке задано два целых числа 1≤n≤50000 и 1≤m≤50000 — количество отрезков и точек на прямой, соответственно. Следующие n строк содержат по два целых числа a i— координаты концов отрезков. Последняя строка содержит m целых чисел — координаты точек. Все координаты не превышают 10^8 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.

**Решение**

using System;

using System.Linq;

class Program

{

static void Main(string[] args)

{

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

int numSegments = int.Parse(str[0]);

int[] arrSegStart = new int[numSegments];

int[] arrSegEnd = new int[numSegments];

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

{

int[] arrSegments = Console.ReadLine().Split(' ').Select(x => int.Parse(x)).ToArray();

arrSegStart[i] = arrSegments[0];

arrSegEnd[i] = arrSegments[1];

}

int[] points = Console.ReadLine().Split(' ').Select(x => int.Parse(x)).ToArray();

QuickSort(arrSegStart, 0, arrSegStart.Length - 1);

QuickSort(arrSegEnd, 0, arrSegEnd.Length - 1);

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

{

int startPoints = BinarySearch(arrSegStart, true, points[i]);

int endPoints = BinarySearch(arrSegEnd, false, points[i]);

int result = startPoints - endPoints;

Console.Write(result + " ");

}

Console.ReadLine();

}

static int BinarySearch(int[] array, bool SubArrStart, int point)

{

int start = 0;

int end = array.Length - 1;

while (start <= end)

{

int median = (start + end) / 2;

if (array[median] == point && SubArrStart == true) start = median + 1;

else if (array[median] < point) start = median + 1;

else end = median - 1;

}

return start;

}

static void QuickSort(int[] array, int left, int right)

{

int rnd = new Random().Next(left, right);

Swap(array, left, rnd);

int pivot = array[left];

int pivotStart = left;

int pivotEnd = left;

int i = left + 1;

int j = right;

while (pivotEnd < j)

{

if (array[i] == pivot)

{

pivotEnd++; i++;

}

else if (array[i] < pivot)

{

Swap(array, i++, pivotStart++);

pivotEnd++;

}

else

Swap(array, i, j--);

}

if (pivotStart - 1 > left) QuickSort(array, left, pivotStart - 1);

if (pivotEnd + 1 < right) QuickSort(array, pivotEnd + 1, right);

}

static void Swap<T>(T[] items, int indexLeft, int indexRight)

{

if (indexLeft != indexRight)

{

T temp = items[indexLeft];

items[indexLeft] = items[indexRight];

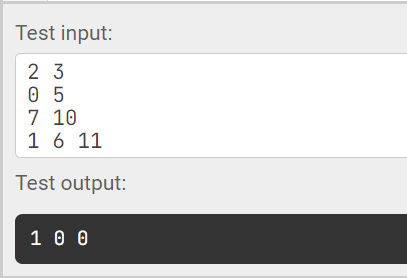
items[indexRight] = temp;

}

}

}

*Листинг программы - Точки и отрезки*

**

*Рисунок 12 - Результат работы программы*

**Задача 4:** Первая строка содержит число 1≤n≤10^4,вторая — n натуральных чисел, не превышающих 10. Выведите упорядоченную по не убыванию последовательность этих чисел.

**Решение**

using System;

using System.Linq;

public class MainClass

{

public static void Main()

{

Console.ReadLine();

var input = Console.ReadLine().Split(' ').Select(int.Parse).ToArray();

var counts = new int[11];

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

{

counts[input[i]]++;

}

int index = 0;

var output = new int[input.Length];

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

{

for(int j=0; j < counts[i]; j++)

{

output[index++] = i;

}

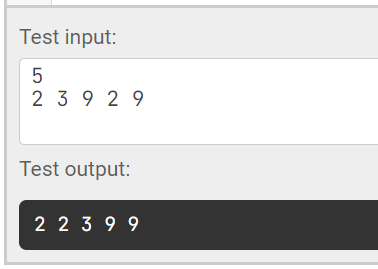
}

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

}

}

*Листинг программы- сортировка подсчётом*

**

*Рисунок 13 - Результат работы программы*