**Министерство образования и науки Российской Федерации**

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ,   
МЕХАНИКИ И ОПТИКИ**

Факультет программной инженерии и компьютерной техники

Кафедра информатики и прикладной математики   
Направление подготовки 09.03.04 Программная инженерия

Дисциплина «Алгоритмы и структуры данных»

**ОТЧЁТ**

по лабораторной работе №6  
неделя шестая

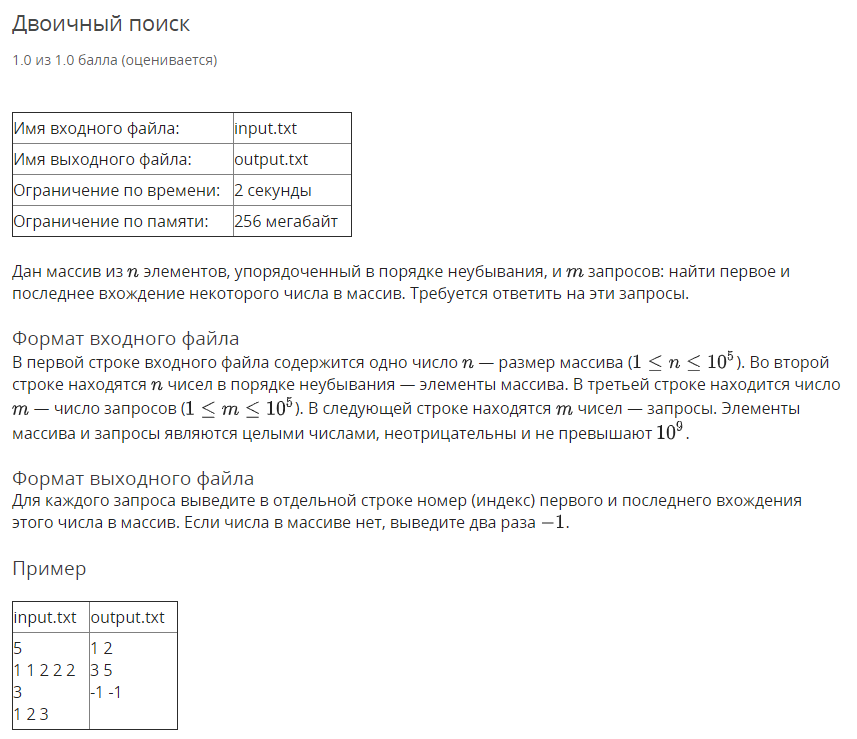
Выполнил:  
Айгузин Иван Олегович   
P3218

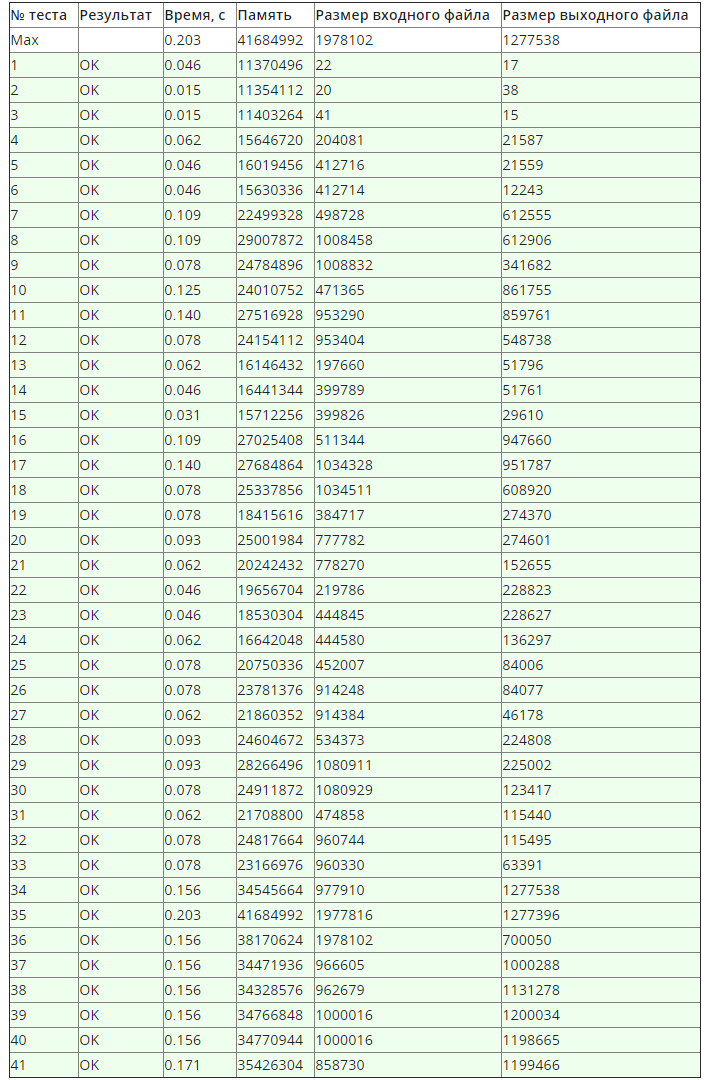
Преподаватели:

Романов Алексей Андреевич  
Волчек Дмитрий Геннадьевич

Санкт-Петербург

2018





using System;

using System.IO;

using System.Linq;

namespace Week06.Task01 {

public sealed class Program {

private static StreamReader \_in;

private static StreamWriter \_out;

private static void Main(string[] args) {

if (!args.Contains("console")) {

SetupIO();

}

Run();

if (args.Contains("console")) {

Console.ReadLine();

}

DisposeIO();

}

private static void Run() {

Console.ReadLine(); // skip line

var a = ReadIntList();

Console.ReadLine();

var b = ReadIntList();

foreach (var x in b) {

var l = BinarySearchLeft(x);

if (l == -1) {

Console.WriteLine("-1 -1");

continue;

}

Console.WriteLine($"{l + 1} {BinarySearchRight(x)}");

}

int BinarySearchRight(int x) {

var (l, r) = (0, a.Length);

while (l < r) {

var m = (l + r) / 2;

if (x < a[m]) {

r = m;

}

else {

l = m + 1;

}

}

return l;

}

int BinarySearchLeft(int x) {

var (l, r) = (0, a.Length - 1);

while (l < r) {

var m = (l + r) / 2;

if (a[m] < x) {

l = m + 1;

}

else {

r = m;

}

}

return l < a.Length && a[l] == x ? l : -1;

}

}

private static int[] ReadIntList() {

return Console.ReadLine()

.Split(' ')

.Select(int.Parse)

.ToArray();

}

private static void SetupIO() {

\_in = new StreamReader("input.txt");

\_out = new StreamWriter("output.txt");

Console.SetIn(\_in);

Console.SetOut(\_out);

}

private static void DisposeIO() {

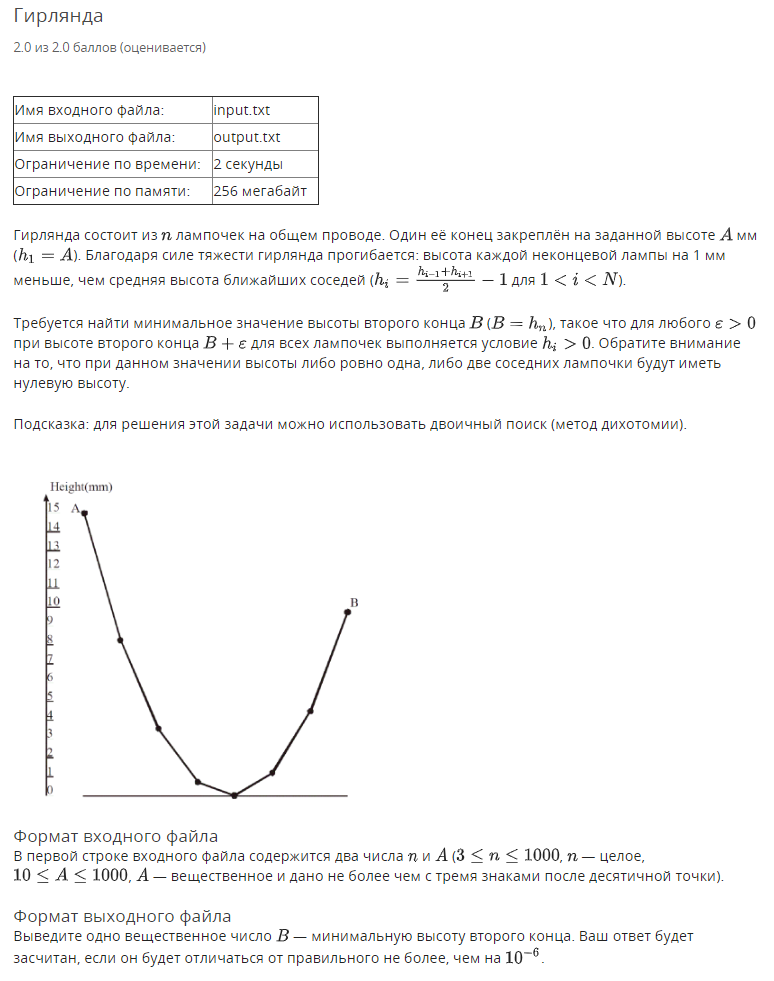
\_in?.Dispose();

\_out?.Dispose();

}

}

}





using System;

using System.Globalization;

using System.IO;

using System.Linq;

using System.Threading;

namespace Week06.Task02 {

public sealed class Program {

private static StreamReader \_in;

private static StreamWriter \_out;

private static void Main(string[] args) {

if (!args.Contains("console")) {

SetupIO();

}

Thread.CurrentThread.CurrentCulture = CultureInfo.InvariantCulture;

Run();

if (args.Contains("console")) {

Console.ReadLine();

}

DisposeIO();

}

private static void Run() {

var list = ReadList();

var (n, a) = (int.Parse(list[0]), double.Parse(list[1]));

var (l, r) = (0.0, a);

var h = new double[n];

h[0] = a;

while (r - l > 0.000000000001) {

h[1] = (r + l) / 2;

var ok = true;

for (var i = 2; i < n; i++) {

h[i] = 2 \* h[i - 1] - h[i - 2] + 2;

if (h[i] < 0) {

ok = false;

break;

}

}

if (ok) {

r = h[1];

}

else {

l = h[1];

}

}

Console.WriteLine($"{h[n - 1]:F8}");

}

private static string[] ReadList() {

return Console.ReadLine()

.Split(' ')

.ToArray();

}

private static void SetupIO() {

\_in = new StreamReader("input.txt");

\_out = new StreamWriter("output.txt");

Console.SetIn(\_in);

Console.SetOut(\_out);

}

private static void DisposeIO() {

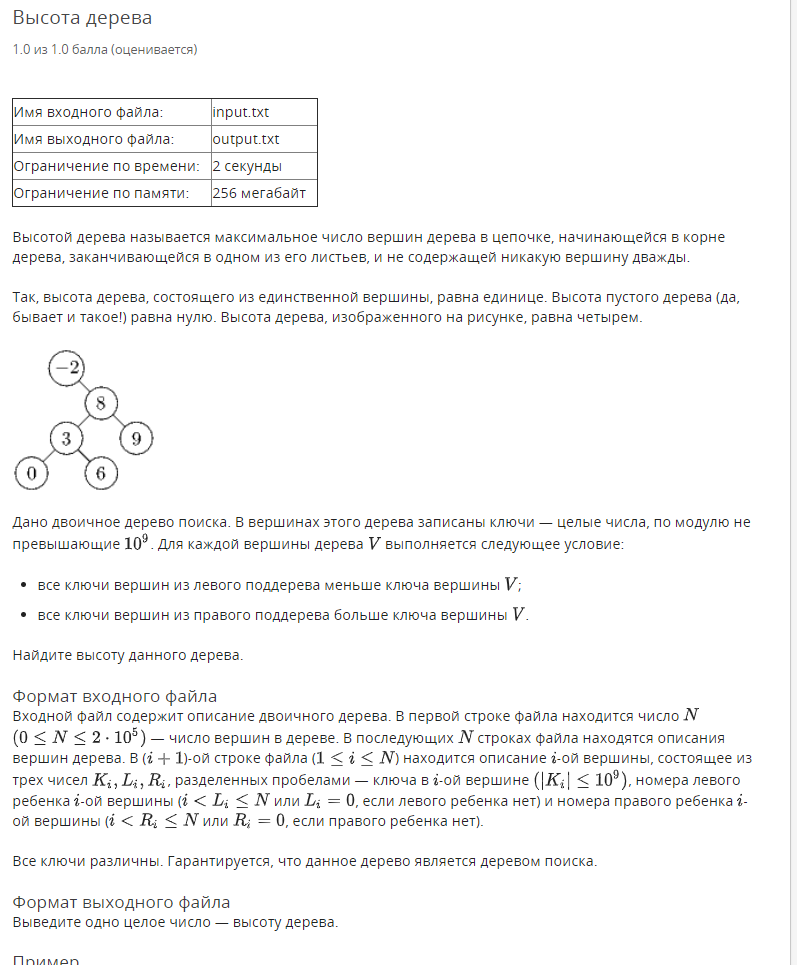
\_in?.Dispose();

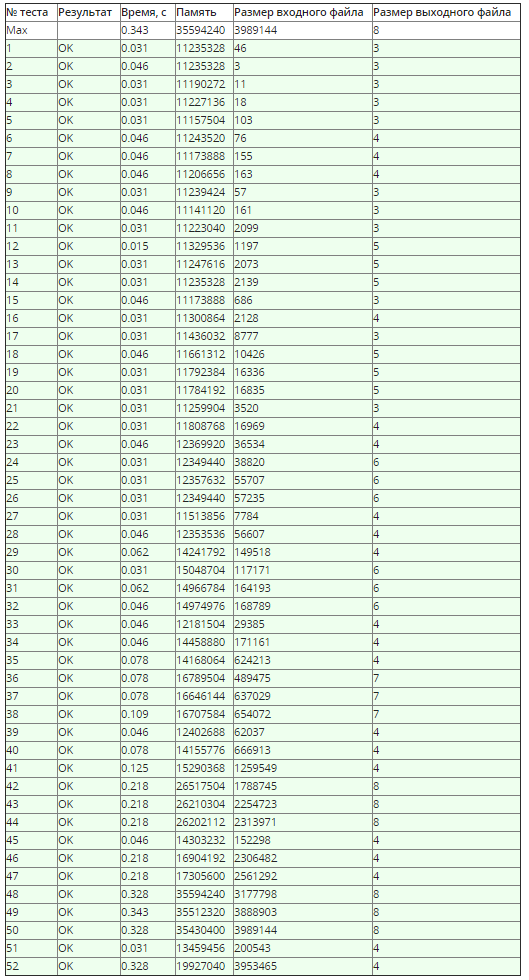
\_out?.Dispose();

}

}

}





using System;

using System.IO;

using System.Linq;

using System.Threading;

namespace Week06.Task03 {

public sealed class TreeNode {

public TreeNode(int left, int right) {

Left = left;

Right = right;

}

public int Left { get; set; }

public int Right { get; set; }

}

public sealed class Program {

private static StreamReader \_in;

private static StreamWriter \_out;

private static TreeNode[] \_tree;

private static void Main(string[] args) {

if (!args.Contains("console")) {

SetupIO();

}

var thread = new Thread(Run, int.MaxValue / 10);

thread.Start();

thread.Join();

if (args.Contains("console")) {

Console.ReadLine();

}

DisposeIO();

}

private static void Run() {

var n = ReadIntList()[0];

if (n == 0) {

Console.WriteLine(0);

return;

}

\_tree = new TreeNode[n];

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

var a = ReadIntList();

\_tree[i] = new TreeNode(a[1] - 1, a[2] - 1);

}

Console.WriteLine(CountDepth(0) + 1);

}

private static int CountDepth(int i) {

var d = 0;

if (\_tree[i].Left != -1) {

d = Math.Max(CountDepth(\_tree[i].Left) + 1, d);

}

if (\_tree[i].Right != -1) {

d = Math.Max(CountDepth(\_tree[i].Right) + 1, d);

}

return d;

}

private static int[] ReadIntList() {

return Console.ReadLine()

.Split(' ')

.Select(int.Parse)

.ToArray();

}

private static void SetupIO() {

\_in = new StreamReader("input.txt");

\_out = new StreamWriter("output.txt");

Console.SetIn(\_in);

Console.SetOut(\_out);

}

private static void DisposeIO() {

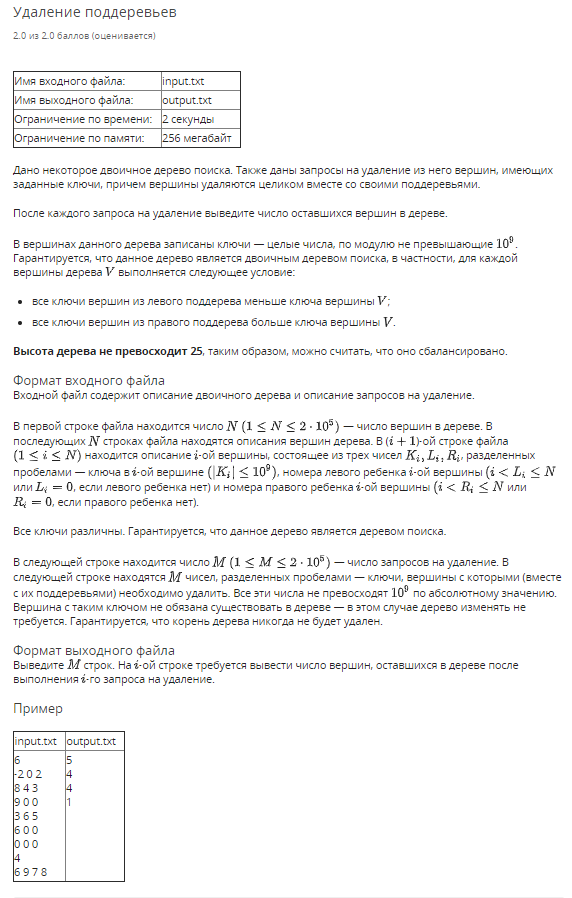
\_in?.Dispose();

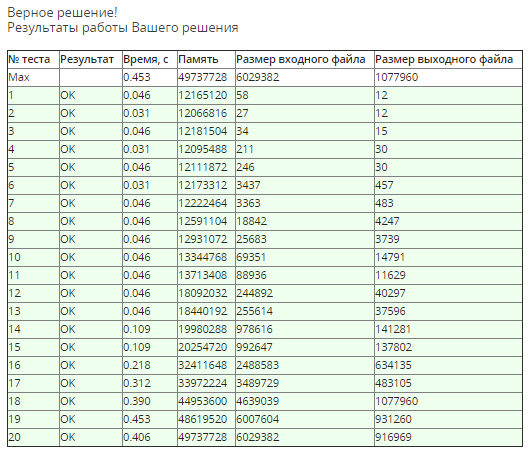
\_out?.Dispose();

}

}

}





using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

namespace Week06.Task04 {

public sealed class TreeNode {

public int Key { get; set; }

public int Left { get; set; }

public int Right { get; set; }

public int Parent { get; set; } = -1;

}

public sealed class Program {

private static StreamReader \_in;

private static StreamWriter \_out;

private static List<TreeNode> \_tree;

private static void Main(string[] args) {

if (!args.Contains("console")) {

SetupIO();

}

Run();

if (args.Contains("console")) {

Console.ReadLine();

}

DisposeIO();

}

private static void Run() {

var n = ReadIntList()[0];

ReadTree(n);

Console.ReadLine(); // skip line

foreach (var m in ReadIntList()) {

var nodeIndex = FindNodeIndexByKey(m);

if (nodeIndex == -1) {

Console.WriteLine(n);

continue;

}

var parent = \_tree[\_tree[nodeIndex].Parent];

if (parent.Left == nodeIndex) {

parent.Left = -1;

}

else {

parent.Right = -1;

}

n -= CountNodes(nodeIndex);

Console.WriteLine(n);

}

}

private static void ReadTree(int n) {

\_tree = Enumerable.Range(0, n)

.Select(x => new TreeNode())

.ToList();

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

var a = ReadIntList();

var node = \_tree[i];

(node.Key, node.Left, node.Right) =

(a[0], a[1] - 1, a[2] - 1);

if (node.Left != -1) {

\_tree[node.Left].Parent = i;

}

if (node.Right != -1) {

\_tree[node.Right].Parent = i;

}

}

}

private static int FindNodeIndexByKey(int key) {

var i = 0;

while (\_tree[i].Key != key) {

if (key < \_tree[i].Key) {

if (\_tree[i].Left == -1) {

return -1;

}

i = \_tree[i].Left;

continue;

}

if (\_tree[i].Right == -1) {

return -1;

}

i = \_tree[i].Right;

}

return i;

}

private static int CountNodes(int i) {

var d = 1;

if (\_tree[i].Left != -1) {

d += CountNodes(\_tree[i].Left);

}

if (\_tree[i].Right != -1) {

d += CountNodes(\_tree[i].Right);

}

return d;

}

private static int[] ReadIntList() {

return Console.ReadLine()

.Split(' ')

.Select(int.Parse)

.ToArray();

}

private static void SetupIO() {

\_in = new StreamReader("input.txt");

\_out = new StreamWriter("output.txt");

Console.SetIn(\_in);

Console.SetOut(\_out);

}

private static void DisposeIO() {

\_in?.Dispose();

\_out?.Dispose();

}

}

}