using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace ConsoleApplication1

{

class Program

{

static void Main(string[] args)

{

int[] mas = new int[int.Parse(Console.ReadLine())];

string[] a = Console.ReadLine().Split();

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

{

mas[i] = int.Parse(a[i]);

if (mas[i] > mas[(i - 1) / 2])

{

Console.WriteLine("NO");

return;

}

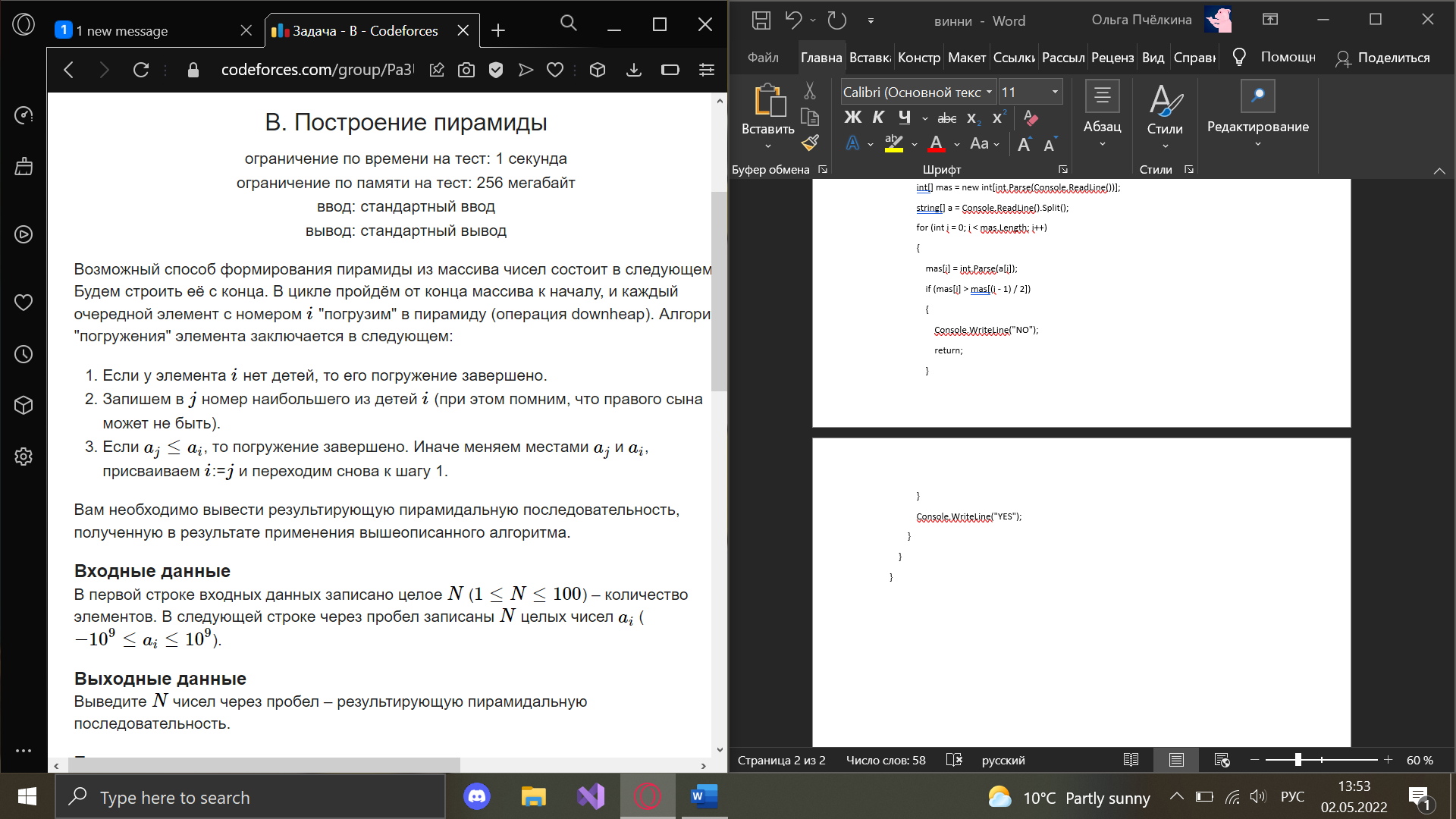
}

Console.WriteLine("YES");

}

}

}



using System;

using System.Collections.Generic;

using System.Collections;

namespace ConsoleApp1

{

class Program

{

static void Swap(ref int[] mas, int first, int second)

{

while (mas[first] < mas[second])

{

int buf = mas[first];

mas[first] = mas[second];

mas[second] = buf;

}

}

static void Main(string[] args)

{

Console.ReadLine();

int[] mas = Array.ConvertAll(Console.ReadLine().Split(), int.Parse);

int i = mas.Length - 1;

while(i >= 0)

{

int j = i \* 2 + 1;

while (true)

{

if (j + 1 < mas.Length)

{

if (mas[i] < mas[j] || mas[i] < mas[j + 1])

{

j = mas[j] < mas[j + 1] ? j + 1 : j;

Swap(ref mas, i, j);

i = j;

j = i \* 2 + 1;

}

else break;

}

else if (j < mas.Length)

{

if (mas[i] < mas[j])

{

Swap(ref mas, i, j);

i = j;

j = i \* 2 + 1;

}

else break;

}

else break;

}

i--;

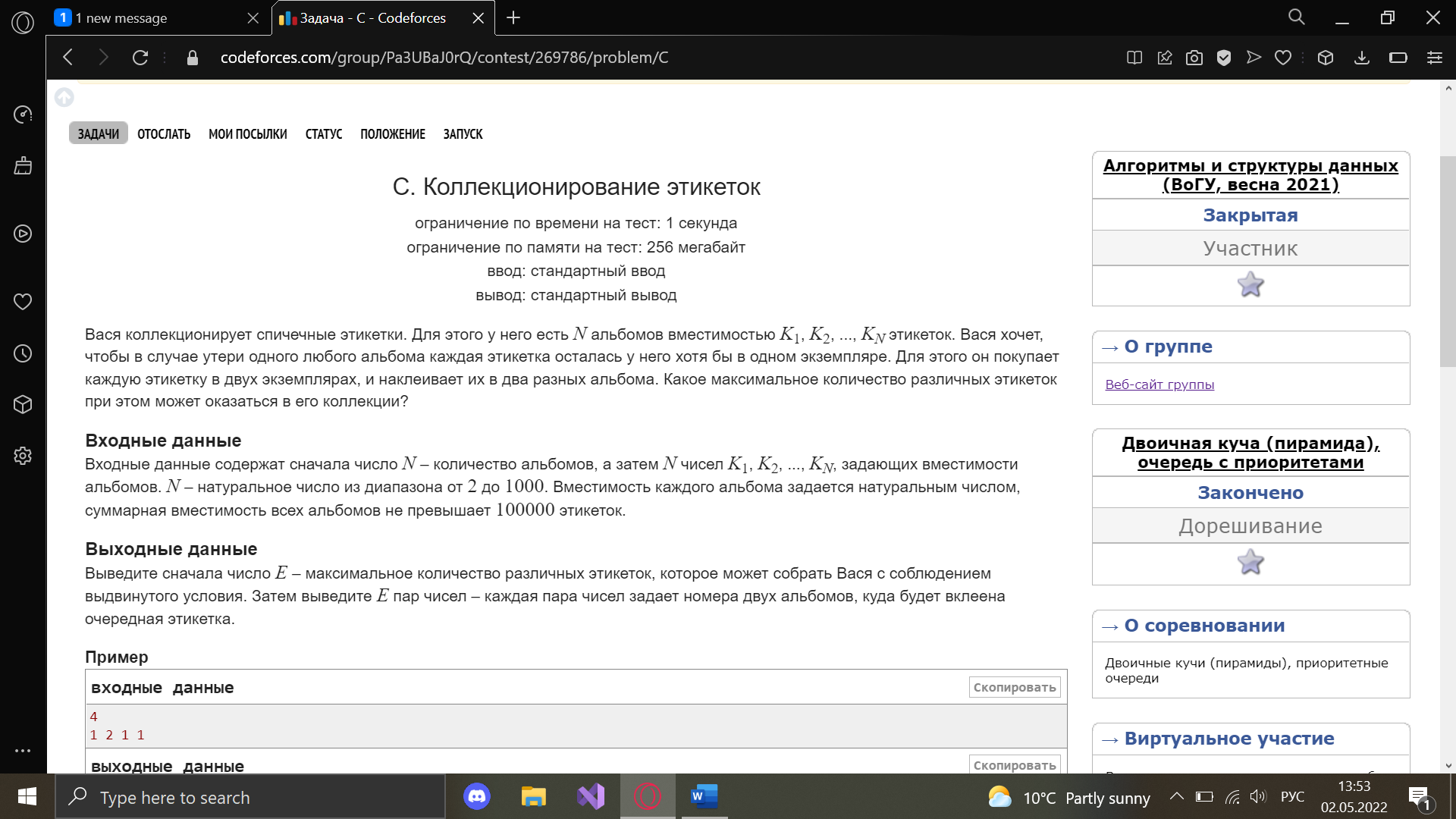
}

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

}

}

}



using System;

using System.Collections.Generic;

using System.Collections;

namespace ConsoleApp1

{

class Program

{

static void Swap(ref List<(int, int)> mas, int first, int second)

{

(int, int) buf = mas[first];

mas[first] = mas[second];

mas[second] = buf;

}

static void Balans(ref List<(int, int)> mas)

{

int i = mas.Count - 1;

while (i >= 0)

{

int j = i \* 2 + 1;

while (true)

{

if (j + 1 < mas.Count)

{

if (mas[i].Item2 < mas[j].Item2 || mas[i].Item2 < mas[j + 1].Item2)

{

j = mas[j].Item2 < mas[j + 1].Item2 ? j + 1 : j;

Swap(ref mas, i, j);

i = j;

j = i \* 2 + 1;

}

else break;

}

else if (j < mas.Count)

{

if (mas[i].Item2 < mas[j].Item2)

{

Swap(ref mas, i, j);

i = j;

j = i \* 2 + 1;

}

else break;

}

else break;

}

i--;

}

}

static void ChangeHeap(ref List<(int, int)> mas)

{

mas[0] = (mas[0].Item1, mas[0].Item2 - 1);

mas[1] = (mas[1].Item1, mas[1].Item2 - 1);

Console.WriteLine($"{mas[0].Item1} {mas[1].Item1}");

bool check = false;

if(mas[0].Item2 == 0)

{

mas.RemoveAt(0);

check = true;

}

if (mas.Count > 1 && check)

{

if (mas[0].Item2 == 0)

{

mas.RemoveAt(0);

}

}

else if(mas.Count > 1)

{

if (mas[1].Item2 == 0)

{

mas.RemoveAt(1);

}

}

var buf = mas[0];

mas[0] = mas[mas.Count - 1];

mas[mas.Count - 1] = buf;

Balans(ref mas);

}

static void Main(string[] args)

{

Console.ReadLine();

int[] mas = Array.ConvertAll(Console.ReadLine().Split(), int.Parse);

List<(int, int)> result = new List<(int, int)>();

int count = 0;

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

{

result.Add((i + 1, mas[i]));

count += mas[i];

}

Balans(ref result);

if (count - result[0].Item2 >= result[0].Item2)

{

Console.WriteLine(count / 2);

for (int i = 0; i < count / 2; i++)

{

ChangeHeap(ref result);

}

}

else

{

Console.WriteLine(count - result[0].Item2);

int j = count - result[0].Item2;

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

{

ChangeHeap(ref result);

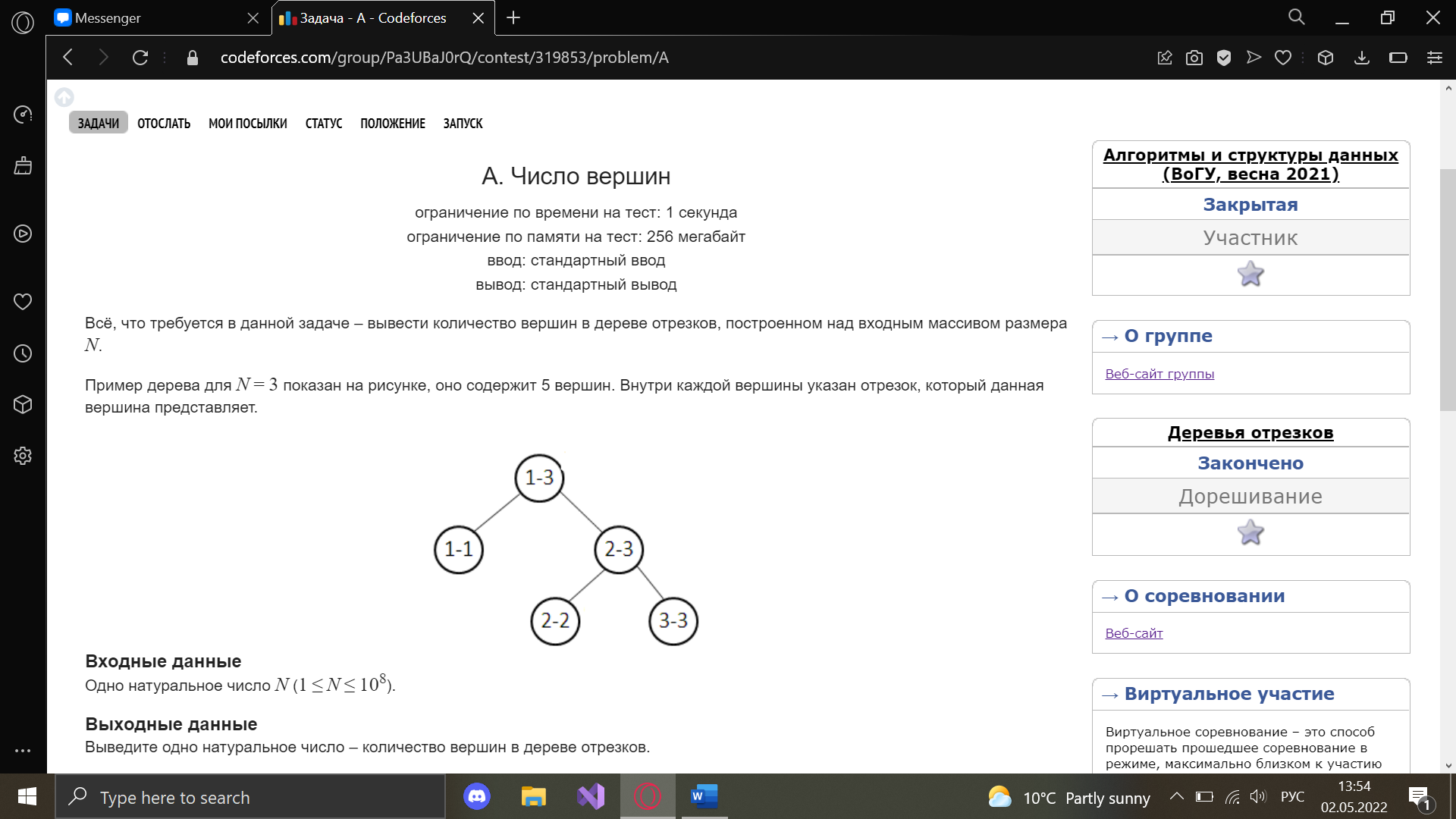
}

}

}

}

}



using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace ConsoleApplication1

{

class Program

{

static void Main(string[] args)

{

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

int countSegment = 0;

int begin = 1;

Segment(ref begin, ref count, ref countSegment);

Console.WriteLine(countSegment);

}

static void Segment(ref int begin, ref int end, ref int countSegment)

{

if (begin != end)

{

if (begin + 1 != end)

{

countSegment++;

int buf = (end + begin) / 2;

int begin\_2 = buf + 1;

Segment(ref begin, ref buf, ref countSegment);

Segment(ref begin\_2, ref end, ref countSegment);

}

else

{

countSegment += 3;

}

}

else

{

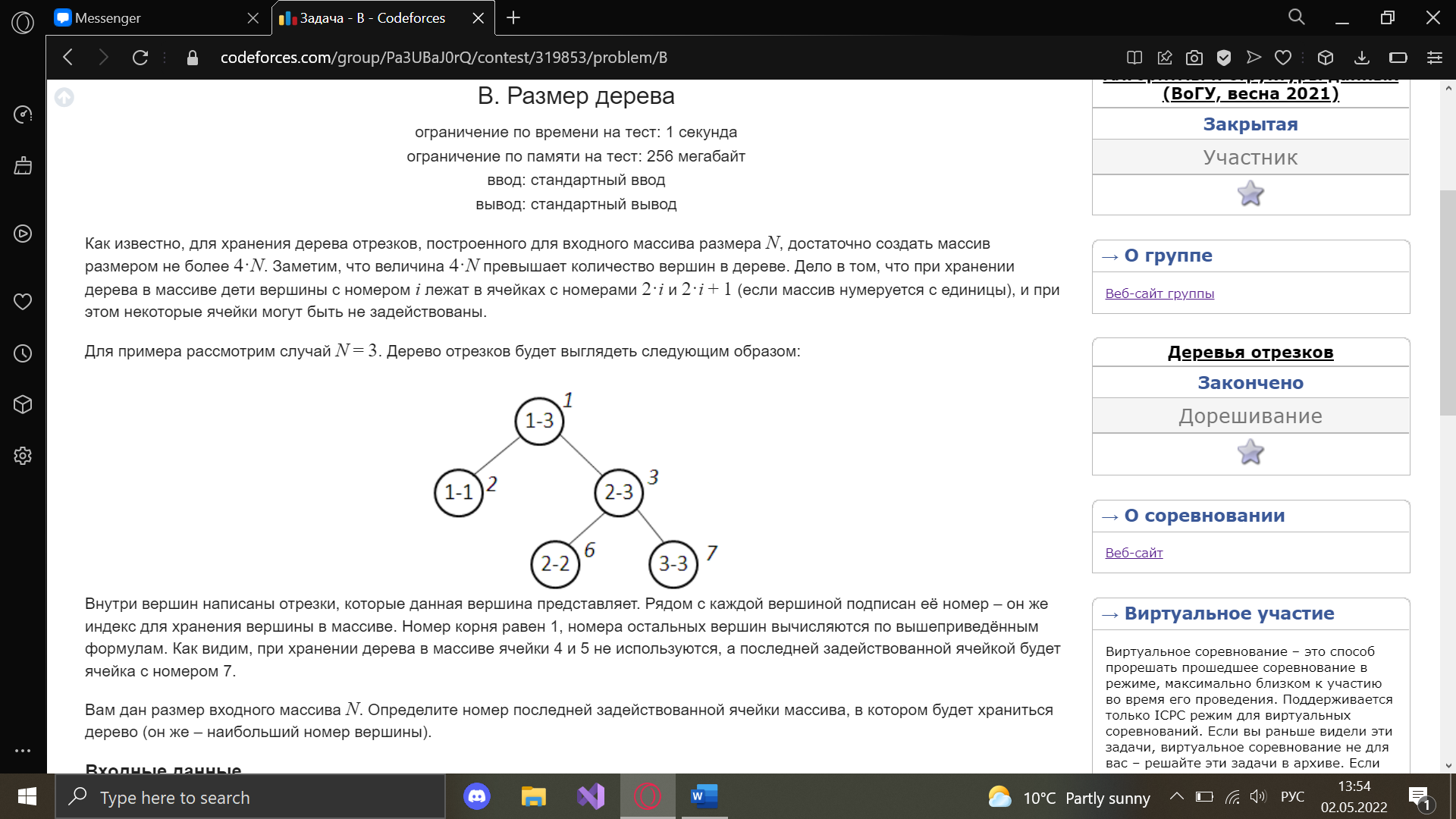
countSegment++;

}

}

}

}



using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace ConsoleApplication1

{

class Program

{

static void Main(string[] args)

{

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

int exists = 1;

int existsMax = 1;

Segment(1, length, exists, ref existsMax);

Console.WriteLine(existsMax);

}

static void Segment(int begin, int end, int exists, ref int existsMax)

{

if (begin != end)

{

if (begin + 1 != end)

{

int buf = (begin + end) % 2 == 0? (begin + end) / 2 - 1 : (begin + end) / 2;

existsMax = existsMax < exists \* 2 ? existsMax \* 2 : existsMax;

Segment(begin, buf, exists \* 2, ref existsMax);

existsMax = existsMax < exists \* 2 + 1 ? exists \* 2 + 1 : existsMax;

Segment(buf + 1, end, exists \* 2 + 1, ref existsMax);

}

else

{

existsMax = existsMax < exists \* 2 ? exists \* 2 : existsMax;

existsMax = existsMax < exists \* 2 + 1 ? exists \* 2 + 1 : existsMax;

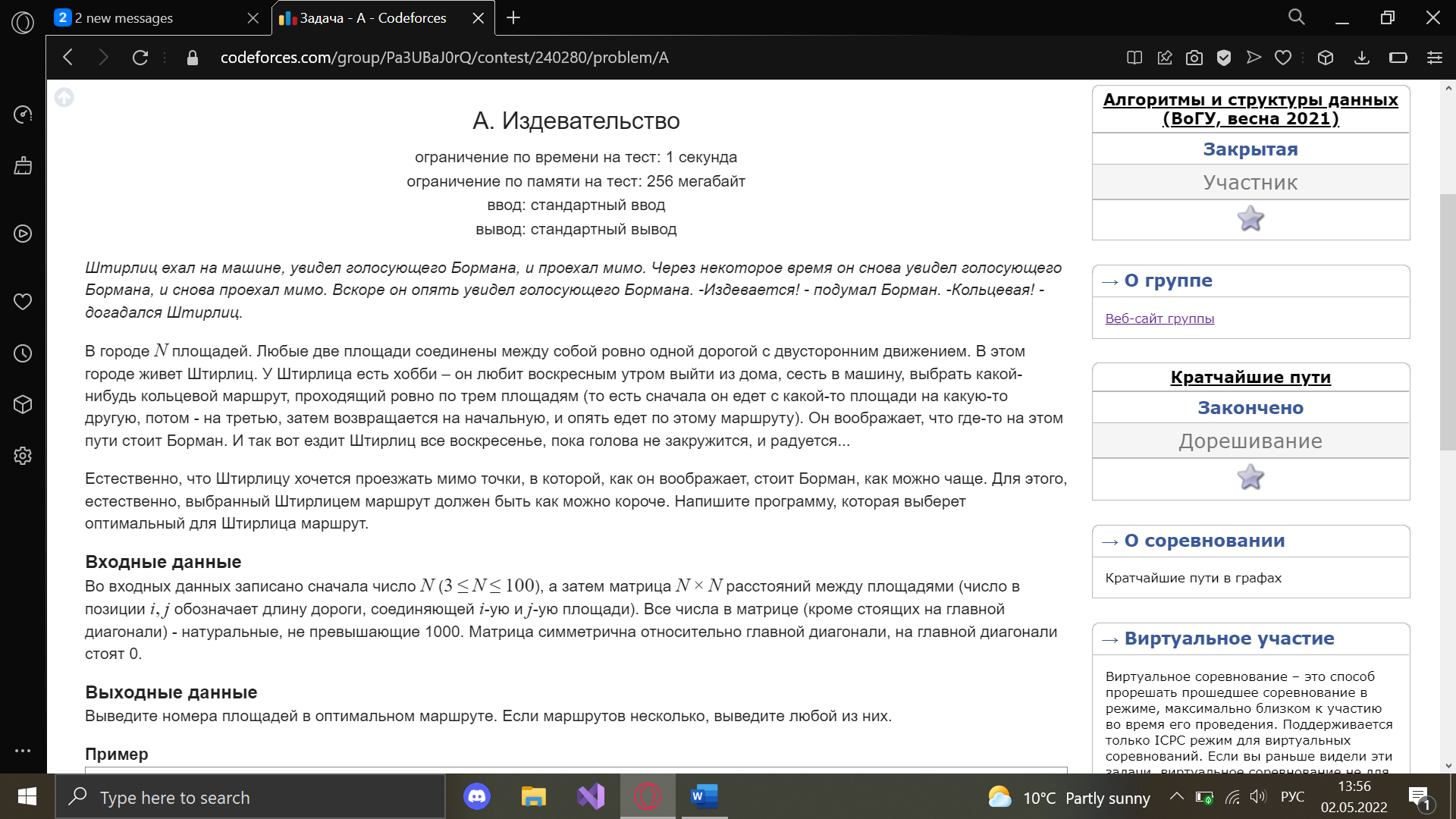
}

}

}

}

}



using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace ConsoleApplication1

{

class Program

{

static void Main(string[] args)

{

int n;

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

int[,] mas = new int[n, n];

string[] s;

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

{

s = Console.ReadLine().Split();

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

{

mas[i, j] = int.Parse(s[j]);

}

}

int sum = -1;

int area\_1 = 0, area\_2 = 0, area\_3 = 0;

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

{

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

{

for (int k = j + 1; k < n; k++)

{

if (sum == -1)

{

sum = mas[i, j] + mas[i, k] + mas[j, k];

area\_1 = i;

area\_2 = j;

area\_3 = k;

}

else if (sum > mas[i, j] + mas[i, k] + mas[j, k])

{

sum = mas[i, j] + mas[i, k] + mas[j, k];

area\_1 = i;

area\_2 = j;

area\_3 = k;

}

}

}

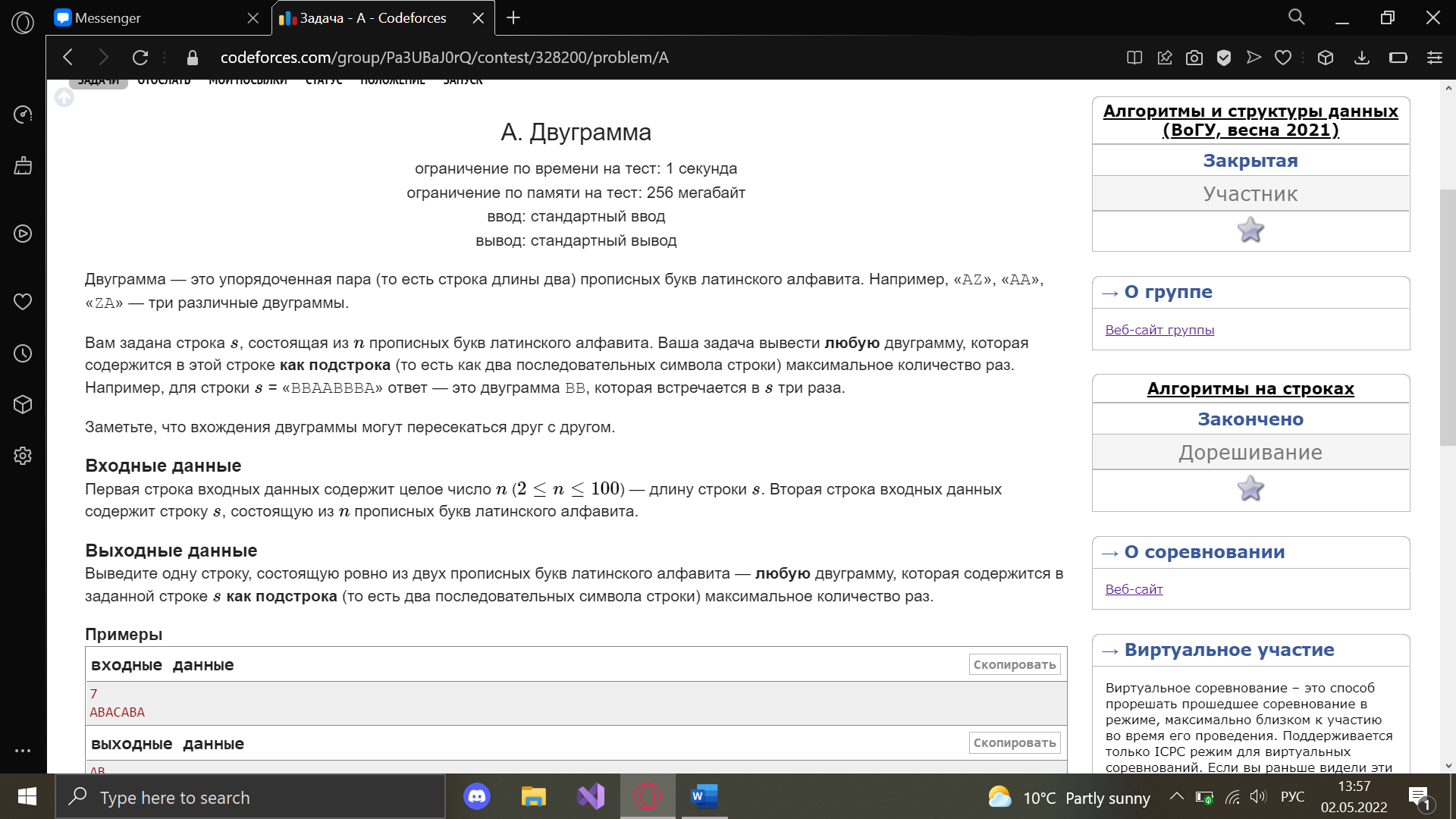
}

Console.WriteLine($"{area\_1 + 1} {area\_2 + 1} {area\_3 + 1}");

}

}

}



using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace ConsoleApplication1

{

class Program

{

static void Main(string[] args)

{

int n = 0;

int max = 0;

string b = "";

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

Dictionary<string, int> m = new Dictionary<string, int>();

string a = Console.ReadLine();

string ans = "";

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

{

b = a.Substring(i, 2);

if(!m.ContainsKey(b))

{

m.Add(b, 0);

}

m[b]++;

if (max < m[b])

{

ans = b;

max = m[b];

}

}

Console.WriteLine(ans);

}

}

}

ЭКЗАМЕН НАИЛУЧШИЙ ПУТЬ

#include <bits/stdc++.h>

using namespace std;

typedef vector <vector <pair<int,int> > > graph;

bool bfs(int n, int a, int b, const graph &gr, int Max, vector<int> &way) {

vector<int> where(n+1, 0);

vector<bool> used(n+1, false);

queue<int> q;

q.push(a);

used[a]=true;

while(!q.empty()){

int u = q.front(); q.pop();

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

if (used[gr[u][i].first]) {continue;}

if (gr[u][i].second < Max) {continue;}

int v = gr[u][i].first;

q.push(v);

used[v] = true;

where[v] = u;

if (v==b) {break;}

}

}

if (where[b]==0) {return false;}

way.resize(0);

while (b>0)

{

way.push\_back(b);

b = where[b];

}

reverse(way.begin(), way.end());

return true;

}

int main(){

int n, m, a, b;

cin >> n >> m >> a >> b;

graph gr(n+1);

set<int> values;

for (int i=1; i<=m; i++) {

int u, v, c1, c2;

cin >> u >> v >> c1 >> c2;

gr[u].push\_back(make\_pair(v, c1));

gr[v].push\_back(make\_pair(u, c2));

values.insert(c1);

values.insert(c2);

}

vector<int> vec(values.size());

vector<int> bestWay;

copy(values.begin(), values.end(), vec.begin());

int best = 0, left = 0, right = vec.size() - 1;

while(left <= right)

{

int mid = (left + right) / 2;

vector<int> way;

if(bfs(n, a, b, gr, vec[mid], way))

{

best = vec[mid];

left = mid + 1;

bestWay = way;

left = mid + 1;

}

else

{

right = mid - 1;

}

}

cout << best << endl;

if(best > 0)

{

cout << a;

for(int i = 1; i < bestWay.size(); i++)

{

cout << " " << bestWay[i];

}

}

return 0;

}