**KAUNO TECHNOLOGIJOS UNIVERSITETAS**

**INFORMATIKOS FAKULTETAS**

**Modulio P170B400 „Algoritmų sudarymas ir analizė“**

**Laboratorinio darbo aprašas (ataskaita)**

**Trečias laboratorinis darbas**

**Dėstytojas**

Lekt. Dalius Makackas

Lekt. Tadas Kraujalis

**Studentas**

Martynas Kuliešius IFF-1/9

**KAUNAS, 2023**

Turinys:

Contents

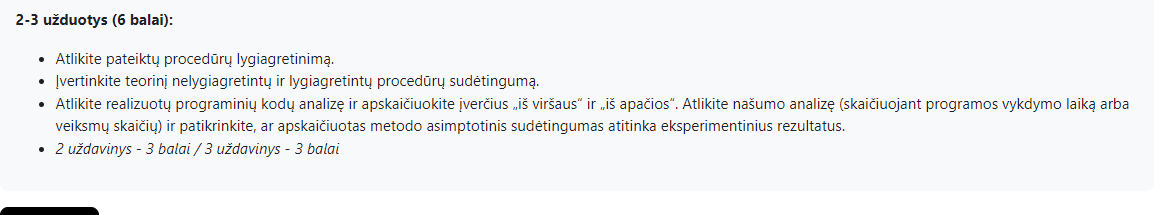
[1. Užduotis 1 3](#_Toc134043531)

[a. Pirmoji užduotis 3](#_Toc134043532)

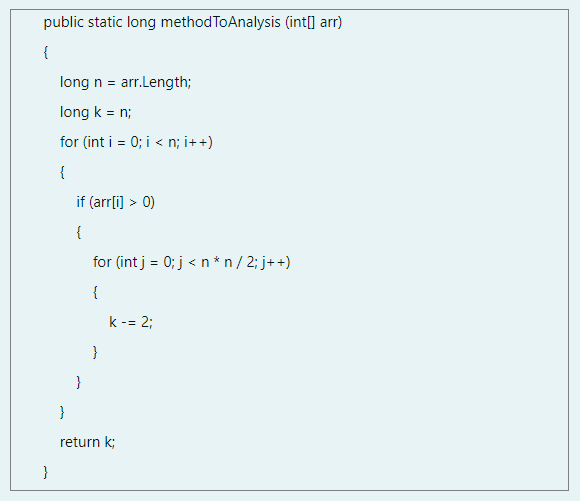
[b. Antroji užduotis 8](#_Toc134043533)

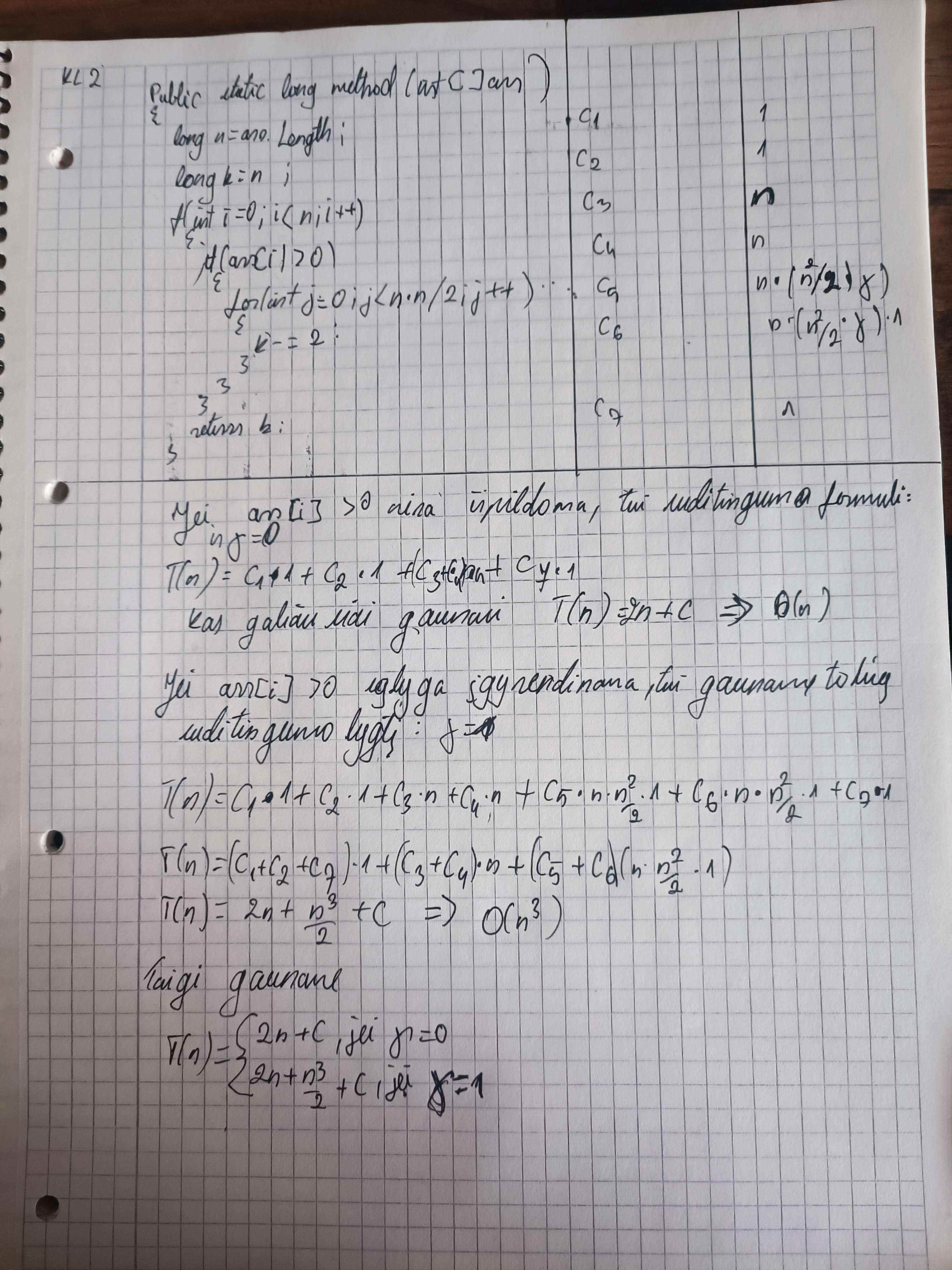
[2. Užduotis 2 16](#_Toc134043534)

# Užduotis 1

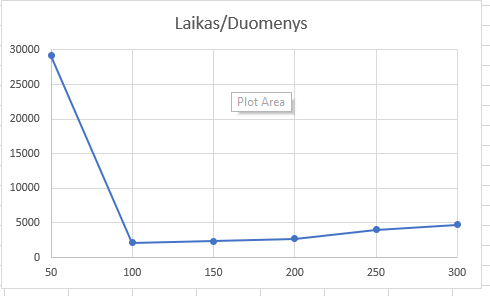


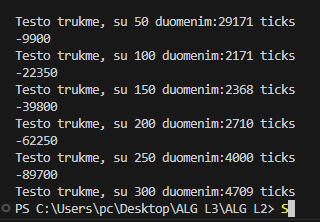
# Pirmoji užduotis





Laiko tyrimas su pasirinktais duomenimis:





Programinis kodas:

using System.Diagnostics;

namespace ALG\_L2

{

    class Program

    {

        static void Main(string[] args)

        {

            string path = "Data.csv";

            Stopwatch sw = new Stopwatch();

            int[] arr=Read(path);

            for(int i=50; i<=300;i+=50)

            {

                sw.Start();

                Console.WriteLine(methodToAnalysis(arr, i));

                sw.Stop();

                Console.WriteLine("Testo trukme, su {0} duomenim:" + sw.ElapsedTicks + " ticks", i);

                sw.Reset();

            }

        }

         static int[] Read(string path)

        {

            int[] arr = new int[300];

            string[] lines = File.ReadAllLines(path);

            foreach(string line in lines)

            {

                string[] bit = line.Split(';');

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

                {

                    arr[i]=int.Parse(bit[i]);

                }

            }

            return arr;

        }

        public static long methodToAnalysis(int[] arr, int ind)

        {

            long n = ind;

            long k = n;

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

            {

                if (arr[i] > 0)

                {

                    for (int j = 0; j < n \* n / 2; j++)

                    {

                        k -= 2;

                    }

                }

            }

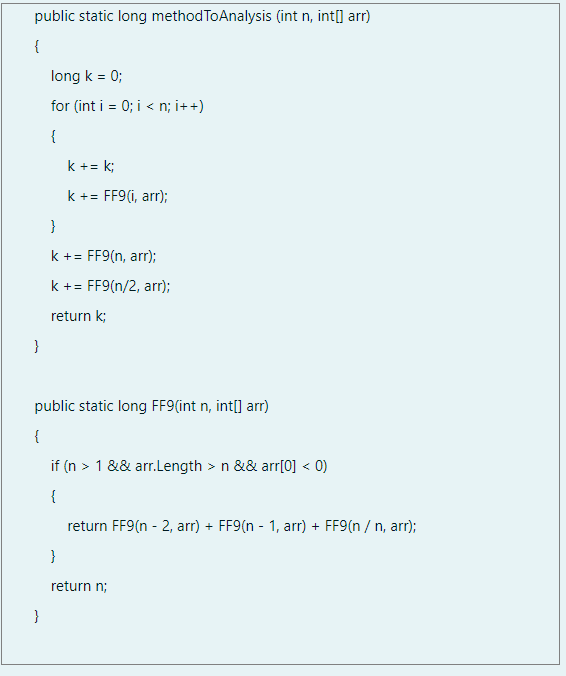
            return k;

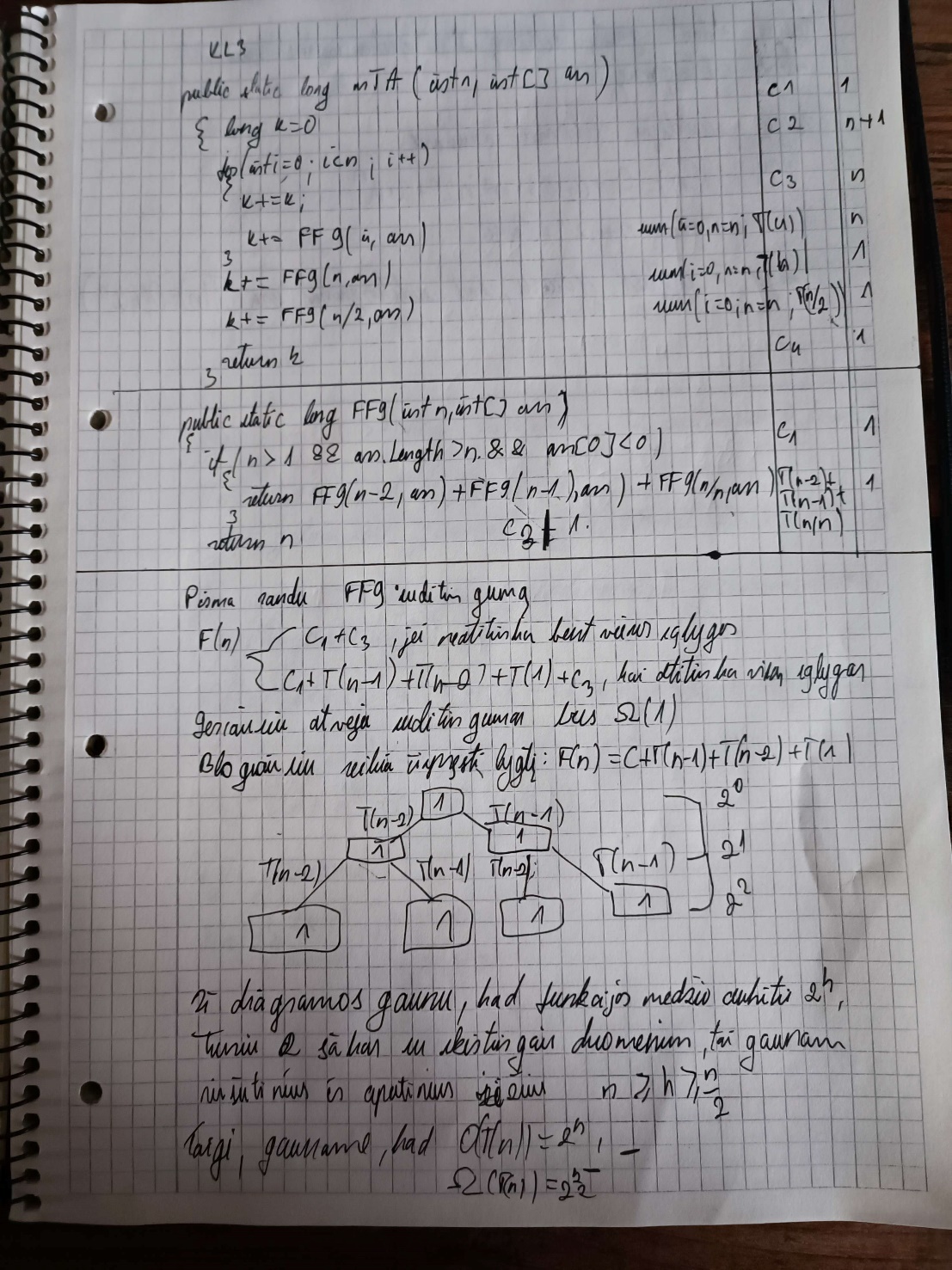
        }

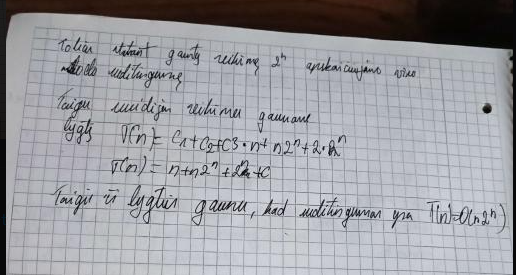
    }

}

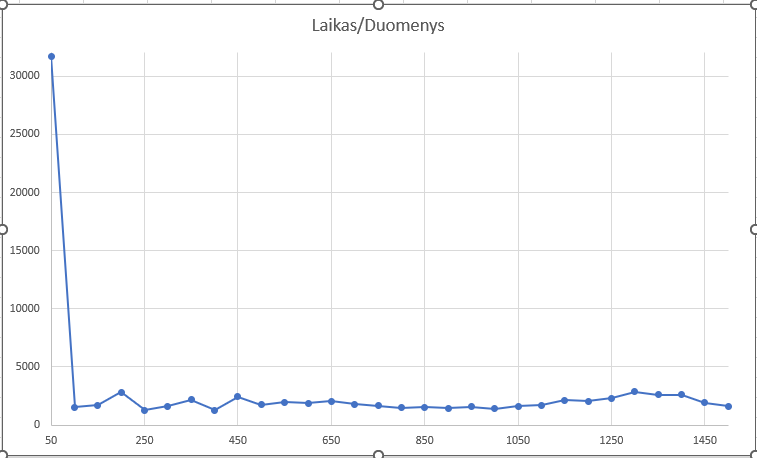
# Antroji užduotis

Skaičiavimai:





Laiko testavimas šiai programai yra kiek komplikuotas, nes net su dideliu duomenų kiekiu, metodas atlieka darbus labai sparčiai:





Programinis kodas:

using System.Diagnostics;

namespace ALG\_L2

{

    class Program

    {

        static void Main(string[] args)

        {

            string path = "Data.csv";

            Stopwatch sw = new Stopwatch();

            int[] arr=Read(path);

            for(int i=50; i<=1500;i+=50)

            {

                sw.Start();

                Console.WriteLine(methodToAnalysis(i, arr));

                sw.Stop();

                Console.WriteLine("Testo trukme, su {0} duomenim:" + sw.ElapsedTicks + " ticks", i);

                sw.Reset();

            }

        }

         static int[] Read(string path)

        {

            int[] arr = new int[1500];

            string[] lines = File.ReadAllLines(path);

            foreach(string line in lines)

            {

                string[] bit = line.Split(';');

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

                {

                    arr[i]=int.Parse(bit[i]);

                }

            }

            return arr;

        }

        public static long methodToAnalysis (int n, int[] arr)

        {

            long k = 0;

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

            {

                k += k;

                k += FF9(i, arr);

            }

            k += FF9(n, arr);

            k += FF9(n/2, arr);

            return k;

        }

        public static long FF9(int n, int[] arr)

        {

            if (n > 1 && arr.Length > n && arr[0] < 0)

            {

                return FF9(n - 2, arr) + FF9(n - 1, arr) + FF9(n / n, arr);

            }

            return n;

        }

    }

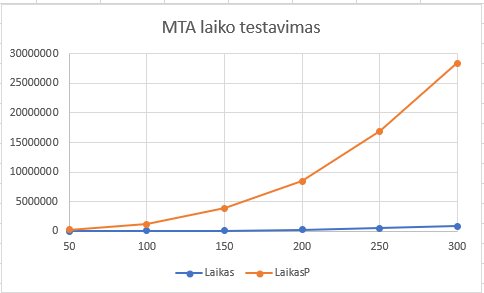
}

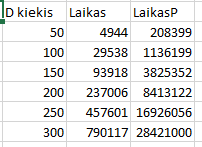
# Paraleliškumo tyrimas:

Teoriškai, vykdant abi funkcijas vieną po kitos turėtume gauti laiko sudėtingumą atitinkantį arba O( n+n2n ) arba O( 2n+n3/2 + n2n). Logiškai mąstant, paralelizmo principą pritaikius savo metodams, jie teoriškai turėtų veikti sparčiau, nes jų veikimas yra padalinamas kelioms procesoriaus dalims, bet ką pastebėjau, kad paralelizmo principus pritaikius, šie metodai atliko savo arbą panašiai arba lėčiau negu netaikant paralelizmo principo. Toliau parodytose laiko diagramose, paralelizmo principą pritaikytų metodų laiką žymiu LaikasP, paprastai spręstų metodų Laikas.

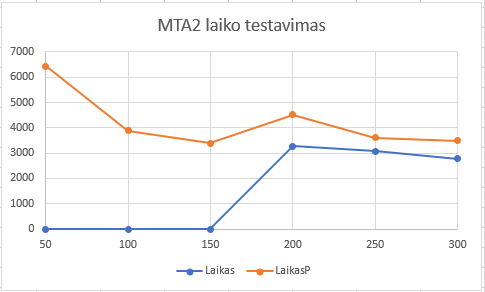
***Kadangi kai kuriais kartais metodų sparta buvo per greita milisekundėm, visus laikus rodau tikais arba 10000 tikų į vieną milisekundę***

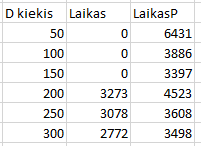
Pirmo metodo analizė:



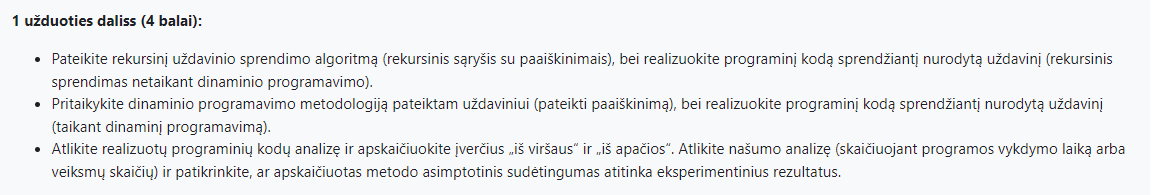


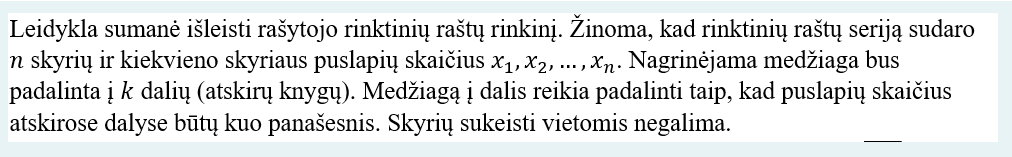
Antro metodo analizė:





# Užduotis 2





Dinaminis sprendimas:

using System;

using System.Collections.Generic;

using System.Diagnostics;

class Program

{

    static void Main()

    {

        var stopWatch = new Stopwatch();

        stopWatch.Start();

        int[] chapters = { 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 125, 150, 175, 200, 250, 300 }; // Pavyzdiniam testui, duota 16 skyriu puslapiu kiekiai ir viso puslapiu: 1750

        int numParts = 6; // Pavyzdiniui testui i kiek daliu padalinti rinktiniu rastu rinkini

        List<int> partPages = DivideIntoParts(chapters, numParts);

        PrintPartPages(partPages);

        stopWatch.Stop();

        Console.WriteLine("Laikas milisekundėmis: {0} ms", stopWatch.ElapsedMilliseconds.ToString());

    }

    static List<int> DivideIntoParts(int[] chapters, int numParts)

    {

        int totalPages = 0;

        foreach (int pages in chapters)

        {

            totalPages += pages;

        }

        Console.WriteLine(totalPages);

        int targetPagesPerPart = totalPages / numParts;

        int currentPartPages = 0;

        int currentPartIndex = 1;

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

        foreach (int pages in chapters)

        {

            currentPartPages += pages;

            if (currentPartPages > targetPagesPerPart)

            {

                currentPartPages -= pages;

                partPages.Add(currentPartPages);

                currentPartPages = pages;

                currentPartIndex++;

            }

        }

        partPages.Add(currentPartPages);

        return partPages;

    }

    static void PrintPartPages(List<int> partPages)

    {

        Console.WriteLine("Puslapių kiekis kiekvienoje dalyje:");

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

        {

            Console.WriteLine("Dalis {0}: {1} psl", i + 1, partPages[i]);

        }

    }

}

Asimptotinio sudėtingumo sprendimas

