KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS

Modulio P170B400 „Algoritmų sudarymas ir analizė“

Laboratorinio darbo aprašas (ataskaita)
Pirmas laboratorinis darbas

Dėstytojas
lekt. MAKACKAS Dalius

Studentas
Matas Palujanskas IFF-1/8

KAUNAS, 2023

TURINYS

1. UŽDUOTIS

1 užduoties dalis: Kiekvienai rekurentinei lygčiai (gautai atlikus užduoties pasirinkimo testą):

- Realizuoti metodą, kuris atitiktų pateiktos rekurentinės lygties sudėtingumą, t. y. programinio kodo rekursinių iškvietimų ir kiekvieno iškvietimo metu atliekamų veiksmų priklausomybę nuo duomenų. Metodas per parametrus turi priimti masyvą, kurio duomenų kiekis yra rekurentinės lygties kintamasis *n* (arba masyvą ir indeksų rėžius, kurie atitinkamai nurodo masyvo nagrinėjamų elementų indeksus atitinkamame iškvietime) (2 balai).
- Kiekvienam realizuotam metodui atlikti programinio kodo analizę, parodant jog jis atitinka pateiktą rekurentinę lygtį (1 balas).
- Išspręskite rekurentinę lygtį ir apskaičiuokite jos asimptotinį sudėtingumą (taikoma pagrindinė teorema, medžių ar kitas sprendimo metodas) (1 balas)
- Atlikti eksperimentinį tyrimą (našumo testus: vykdymo laiką ir veiksmų skaičių) ir patikrinkite ar apskaičiuotas metodo asimptotinis sudėtingumas atitinka eksperimentinius rezultatus (1 balas).

Individualaus užduoties varianto lygtys:

1) $T(n) = 2 * T\left(\frac{n}{6}\right) + n^4$

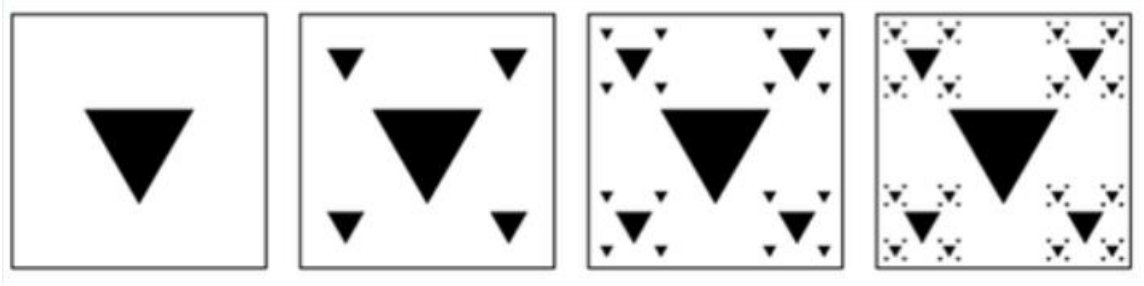
2) $T(n) = T\left(\frac{n}{9}\right) + T\left(\frac{n}{4}\right) + n^3$

3) $T(n) = T(n-3) + T(n-1) + n$

2 užduoties dalis: Naudojant rekursiją ir nenaudojant grafinių bibliotekų sudaryti nurodytos struktūros BMP formato (gautą atlikus užduoties pasirinkimo testą):

- Programos rezultatas BMP formato bylos demonstruojančios programos rekursijas. (3 balai)
- Eksperimentiškai nustatykite darbo laiko ir veiksmų skaičiaus priklausomybę nuo generuojamo paveikslėlio dydžio (taškų skaičiaus). Gautus rezultatus atvaizduokite grafikais. Grafiką turi sudaryti nemažiau kaip 5 taškai ir paveikslėlio taškų skaičius turi didėti proporcingai (kartais). (1 balas)
- Analitiškai įvertinkite procedūros, kuri generuoja paveikslėlį, veiksmų skaičių sudarydami rekurentinę lygtį ir ją išspręskite. Gautas rezultatas turi patvirtinti eksperimentinius rezultatus (našumo testus: vykdymo laiką ir veiksmų skaičių). (1 balas)

Individuali užduotis:



2. PIRMA DALIS. REKURENTINĖS LYGTYS

2.1. Pirma lygtis

$$T(n) = 2 * T\left(\frac{n}{6}\right) + n^4$$

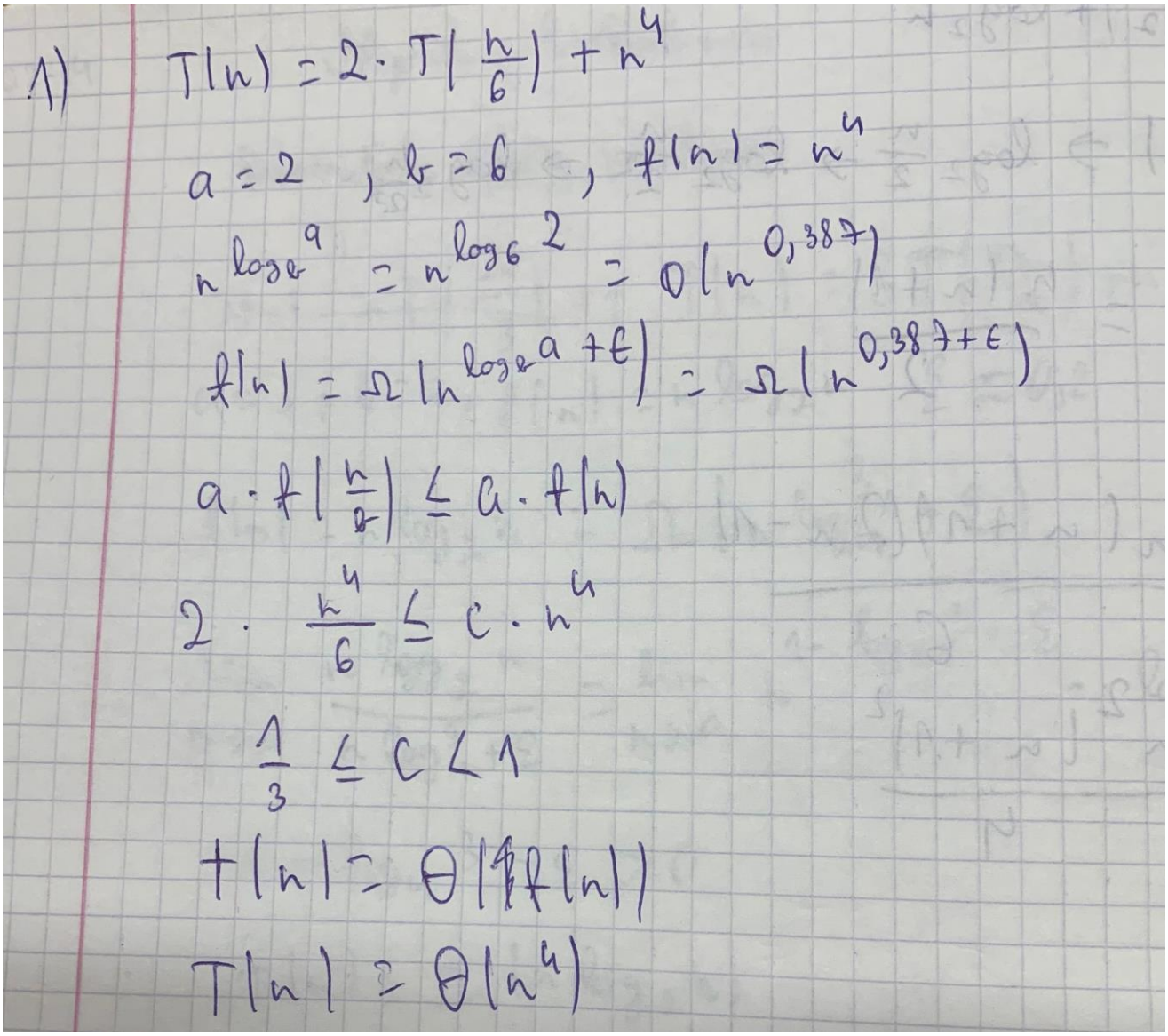
2.1.1. Programinio kodo analizė

T(k), kur k = n + 1

public static void Rec1(int[] A, int start, int end)	Laikas	Kiekis
{		
if (end <= 0)	c1	1
{		
return;	c2	1
}		
Rec1(A, start, end / 6);	$T\left(\frac{n}{6}\right)$	1
Rec1(A, start, end / 6);	$T\left(\frac{n}{6}\right)$	1
for (int i = start; i < end; i++)	c3	end - start + 1 = k
{		
for (int j = start; j < end; j++)	c3	k(k - 1)=$k^2 - k$
{		
for (int y = start; y < end; y++)	c3	$(k^2 - k)\,(k - 1) = k^3 - 2k^2 + k$
{		
for (int k = start; k < end; k++)	c3	$(k^3 - 2k^2 + k)\,(k - 1) = k^4 - 3k^3 + 3(k^2) - k$
{		
counter++;	c4	$(k - 1)\,(k - 1) = k^2 - 2k + 1$
}		
}		
}		
}		
}		

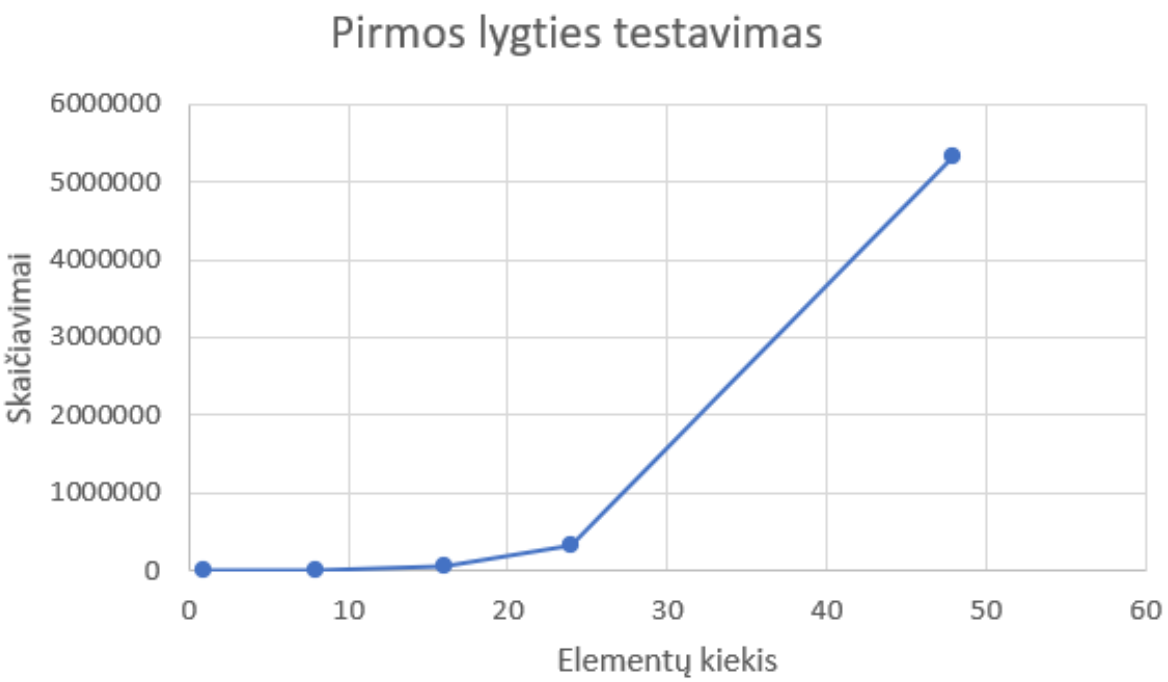
$$T(k) = \begin{cases} c1 + c2\,, kai\ k < 6 \\ 2T\left(\frac{k}{6}\right) + c1 + c3(2(k^2) - 2(k^3) + k^4) + c4(k^2 - 2k + 1)\,, kai\ k \geq 6 \end{cases}$$

2.1.2. Rekurentinės lygties sprendimas



Panaudotas teoremos 3 atvejis.

2.1.3. Eksperimentinis tyrimas




```
Function working time: 00:00:00.0000912
Elements amount: 1
Counter: 1

Function working time: 00:00:00.0000072
Elements amount: 8
Counter: 4098

Function working time: 00:00:00.0001170
Elements amount: 16
Counter: 65568

Function working time: 00:00:00.0006171
Elements amount: 24
Counter: 332288

Function working time: 00:00:00.0105143
Elements amount: 48
Counter: 5316612
```

2.2. Antra lygtis

$$T(n) = T\left(\frac{n}{9}\right) + T\left(\frac{n}{4}\right) + n^3$$

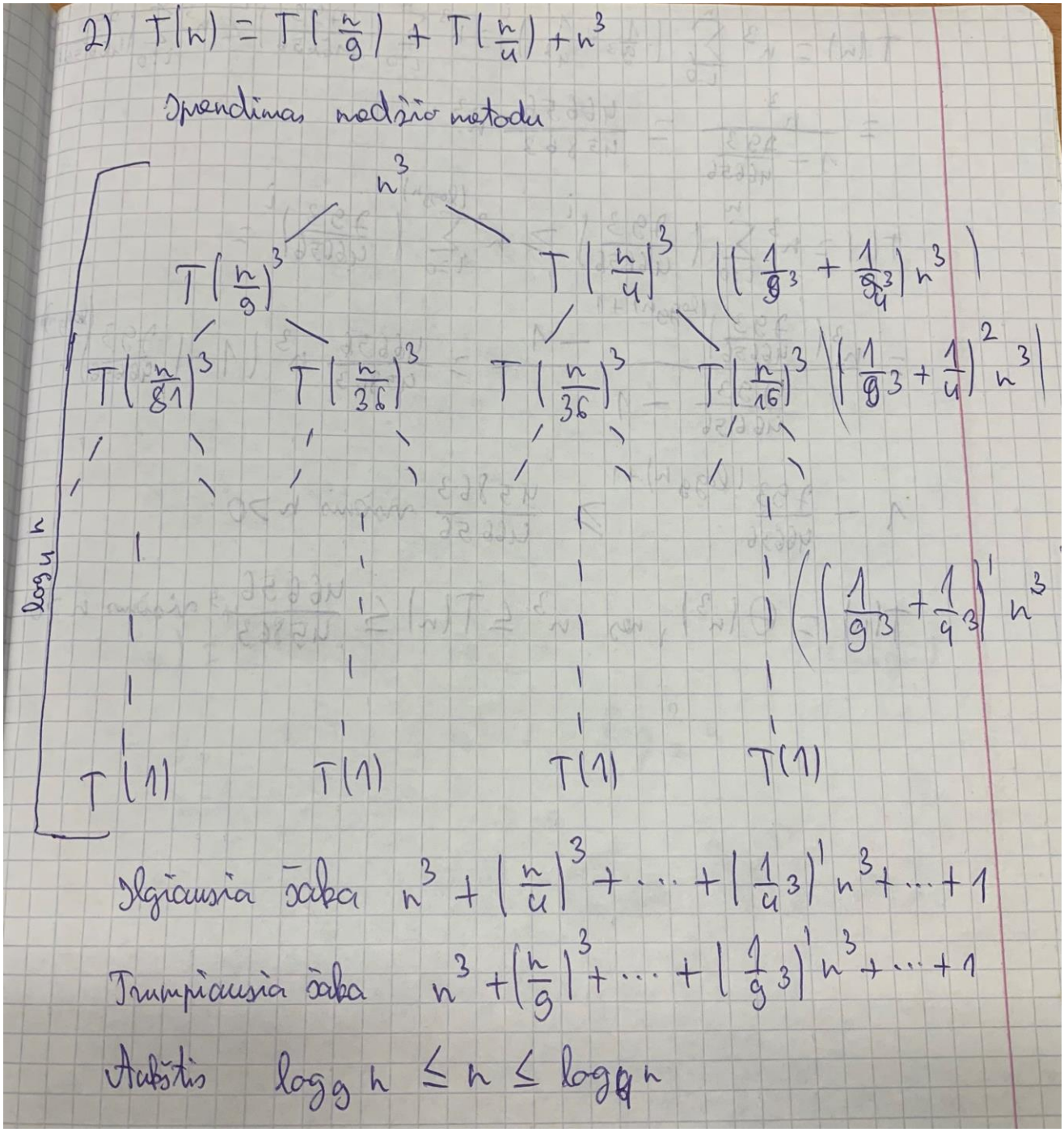
2.2.1. Programinio kodo analizė

T(k), kur k = n + 1

public static void Rec2(int[] A, int start, int end)	Laikas	Kiekis
{		
if (end <= 0)	c1	1
return;	c2	1
Rec2(A, start, end / 4);	$T\left(\frac{n}{4}\right)$	1
Rec2(A, start, end / 9);	$T\left(\frac{n}{9}\right)$	1
for(int i = 0; i < end; i++)	c3	$n-1+1+1 = n+1 = k$
{		
for(int j = 0; j < end; j++)	c3	$k * (n) = k^2 - k$
{		
for(int k = 0; k < end; k++)	c3	$k * (n)^2 = k^3 - 2k^2 + k$
{		
counter++;	c4	$(n)^3 = k^3 - 1 - 3k^2 + 3k$
}		
}		
}		
}		

$$T(k) = \begin{cases} c1 + c2\,, kai\ k < 9 \\ c1 + T\left(\frac{k}{9}\right) + T\left(\frac{k}{4}\right) + c3(k^3 - 2(k^2) + k) + c4(k^3 - 1 - 3k^2 + 3k)\,, kai\ k \geq 9 \end{cases}$$

2.2.2. Rekurentinės lygties sprendimas



$$T(n) = n^3 \sum_{i=0}^{n} \left(\frac{1}{9^3} + \frac{1}{4^3} \right)^i = n^3 \sum_{i=0}^{n} \left(\frac{793}{46656} \right)^i < n^3 \sum_{i=0}^{\infty} \left(\frac{793}{46656} \right)^i =$$

$$= \frac{n^3}{1 - \frac{793}{46656}} = \frac{46656}{45863} n^3$$

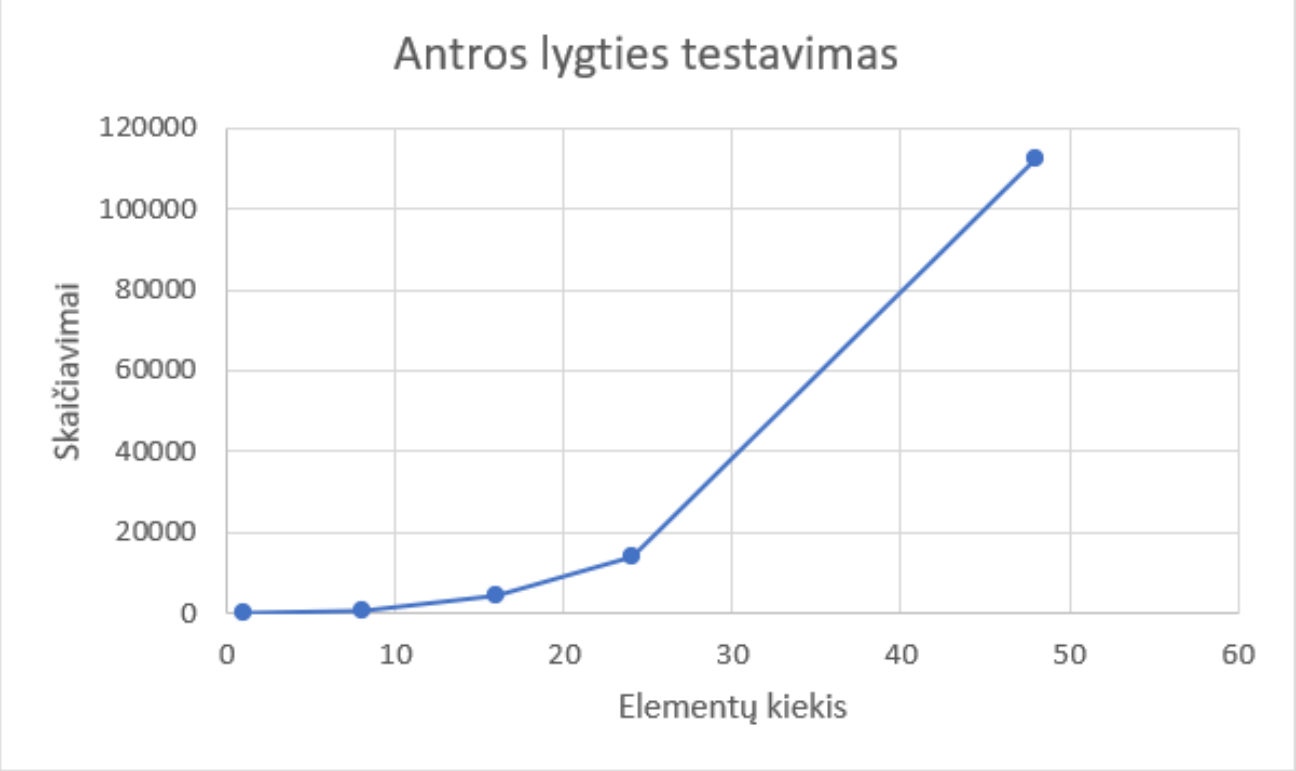
$$T(n) = n^3 \sum_{i=0}^{n} \left(\frac{793}{46656} \right)^i \geq n^3 \sum_{i=0}^{(\log_9 n)} \left(\frac{793}{46656} \right)^i =$$

$$= \frac{n^3 \left(\frac{793}{46656} \right)^{(\log_9 n)+1} - 1}{\frac{793}{46656} - 1} = \frac{46656}{45863} n^3 \left(1 - \left(\frac{793}{46656} \right)^{(\log_9 n)+1} \right)$$

$$1 - \frac{793}{46656}^{(\log_9 n)+1} \geq \frac{45863}{46656} \text{ visiems } n > 0$$

$T(n) = \Theta(n^3)$, nes $n^3 \leq T(n) \leq \frac{46656}{45863} n^3$ visiems $n > 0$.

2.2.3. Eksperimentinis tyrimas




```
Function working time: 00:00:00.0000967
Elements amount: 1
Counter: 1

Function working time: 00:00:00.0000011
Elements amount: 8
Counter: 520

Function working time: 00:00:00.0000074
Elements amount: 16
Counter: 4162

Function working time: 00:00:00.0000241
Elements amount: 24
Counter: 14049

Function working time: 00:00:00.0001925
Elements amount: 48
Counter: 112474
```

2.3. Trečia lygtis

$$T(n) = T(n-3) + T(n-1) + n$$

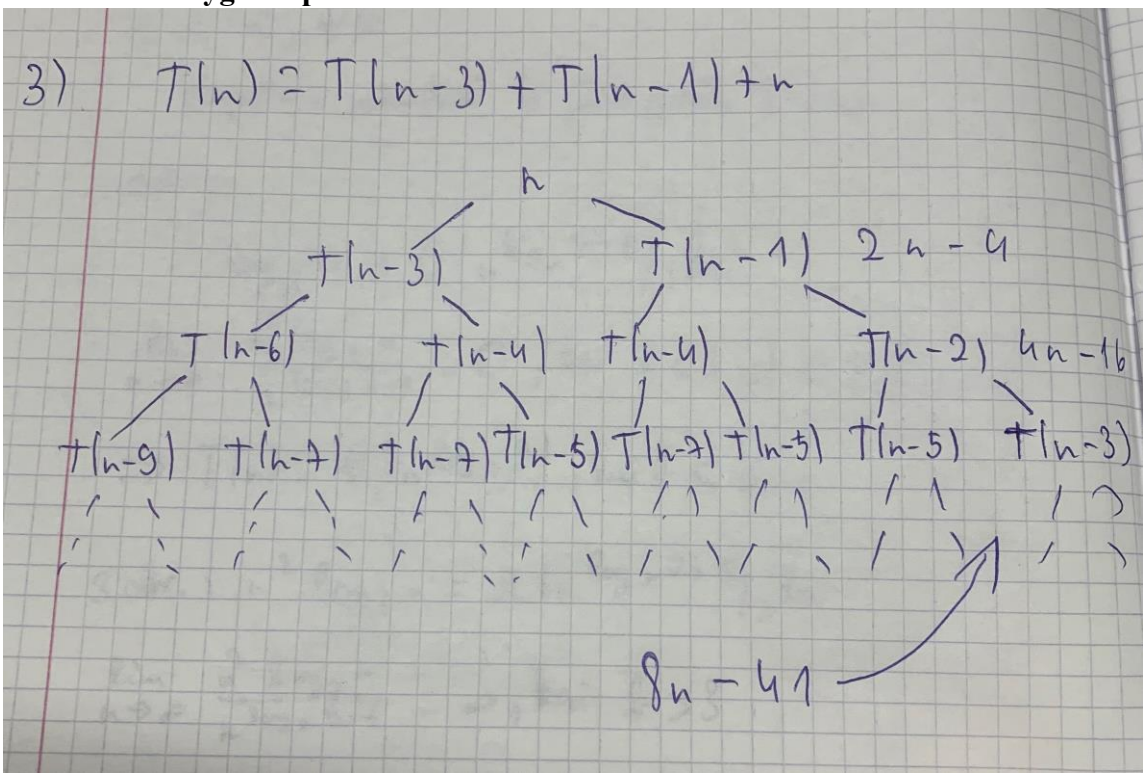
T(k), kur k = n + 1

2.3.1. Programinio kodo analizė

public static void Rec3(int[] A, int start, int end)	Laikas	Kiekis
{		
if (end <= 0)	c1	1
return;	c2	1
Rec3(A, start, end - 3);	T(n - 3)	1
Rec3(A, start, end - 1);	T(n – 1)	1
for (int i = 0; i < end; i++)	c3	n – 1 + 1 + 1 = n + 1
{		
counter++;	c4	n
}		
}		

$$T(k) = \begin{cases} c1 + c2\,, kai\ k < 4 \\ c1 + T(k-3) + T(k-1) + c3 * k + c4(k-1) \geq 4 \end{cases}$$

2.3.2. Rekurentinės lygties sprendimas



S_n - tojo lygio suma

$$\begin{cases} S_0 = 0 \\ S_n = 2S_{n-1} + 4 \cdot 2^{n-1} \end{cases} \qquad S_n = 4n2^{n-1}$$

$$S_{n+1} = 2 \cdot 4n2^{n-1} + 4 \cdot 2^n = 4(n+1)2^n$$

$$T(n) = \sum_{i=0}^{n} (2^i n - 4i\,2^{i-1}) = n\sum_{i=0}^{n} 2^i - 4\sum_{i=0}^{n} i\,2^{i-1} =$$

$$= n(2^{n+1} - 1) - 11(n2^n - 2^n + 1)$$

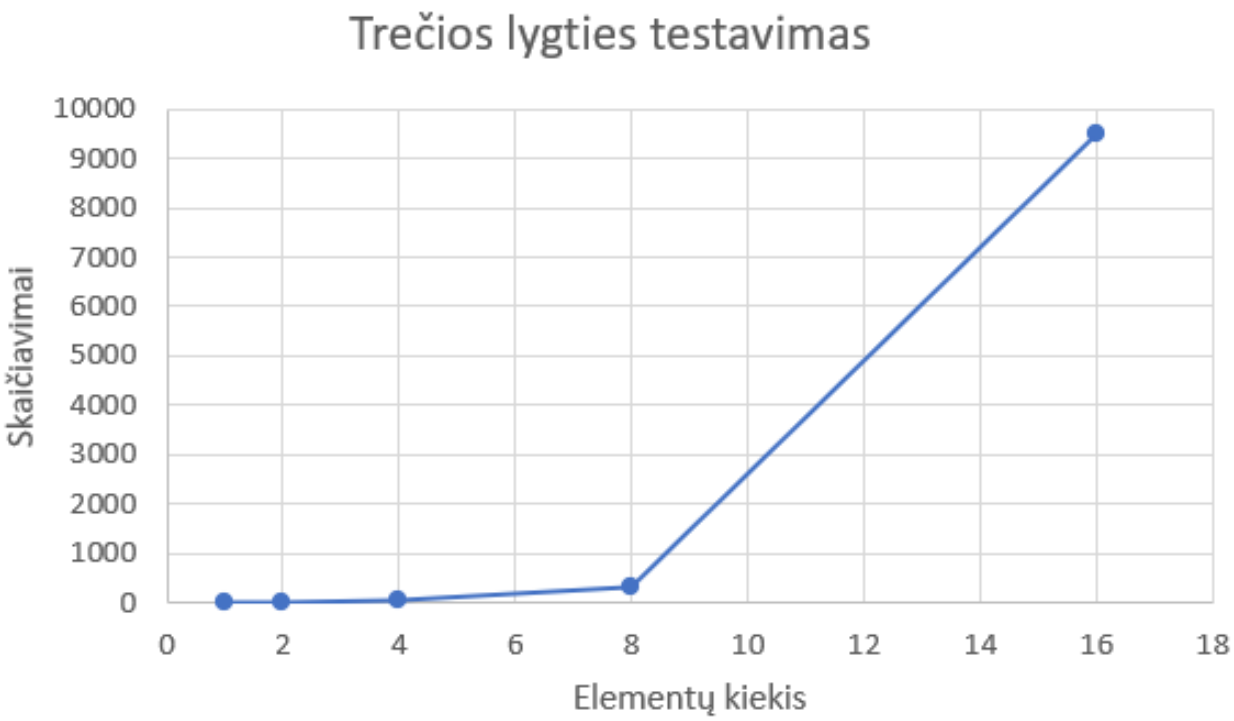
Apatinis vertinimas, kai $n = \frac{n}{3}$

$$T(n) = \Omega(n2^{\frac{n}{3}})$$

Viršutinis, kai $n = \frac{n}{1} = n$

$$T(n) = O(n2^n)$$

2.3.3. Eksperimentinis tyrimas




```
Function working time: 00:00:00.0002240
Elements amount: 1
Counter: 1

Function working time: 00:00:00.0000005
Elements amount: 2
Counter: 5

Function working time: 00:00:00.0000008
Elements amount: 4
Counter: 31

Function working time: 00:00:00.0000032
Elements amount: 8
Counter: 316

Function working time: 00:00:00.0000542
Elements amount: 16
Counter: 9513
```

3. ANTRA DALIS. BMP FORMATO BYLOS

3.1. Programinis kodas

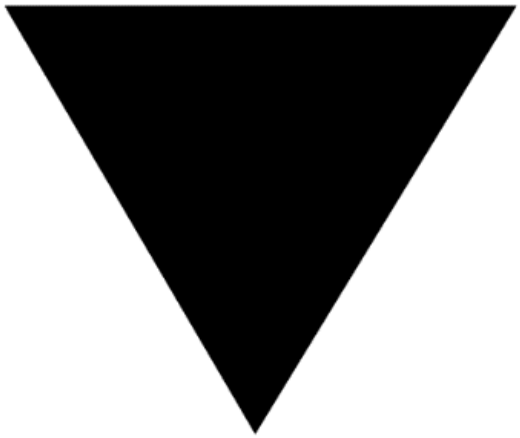
```
using System;
using System.IO;

namespace Laboras
{
    internal class Renderer
    {
        private const int pictureSize = 3000;

        static void Main(string[] args)
        {
            Renderer renderer = new Renderer("Triangle", pictureSize, pictureSize,
0xFFFFFF);
            // X Y coords, Width, Height, Color
            renderer.InitializeFractal();
            renderer.Print();
        }

        // Color format is ARGB (to define recomended hex: 0xAARRGGBB), coordinates
start from bottom left corner, 1 unit is 1 pixel
        public Renderer(string OutputName, ushort Width, ushort Height, uint
FillingColor)
        {
            this.Width = Width;
            this.Height = Height;
            Buffer = new uint[Width * Height];

            Array.Fill(Buffer, FillingColor);

            this.OutputName = OutputName;
            if (!OutputName.Contains(".bmp"))
                this.OutputName += ".bmp";
        }

        private void InitializeFractal()
        {
            double xW = 33 * (pictureSize / 113);
            double yW = 71 * (pictureSize / 113);

            double xE = 82 * (pictureSize / 113);
            double yE = 71 * (pictureSize / 113);

            double xS = 57 * (pictureSize / 113);
            double yS = 30 * (pictureSize / 113);

            DrawFullTriangle(xW, yW, xE, yE, xS, yS);
            int recursionLength = -1;
            double xLength = xE - xW;
            while (xLength > 3)
            {
                xLength = xLength * ((double)15 / (double)49);
                recursionLength++;
            }
            MoveInDifferentDirections(recursionLength, xW, yW, xE, yE, xS, yS);

        }

        //Moving triangle in different directions
```

```
        private void MoveInDifferentDirections(int recursionLength, double _xW, double
_yW, double _xE, double _yE, double _xS, double _yS)
        {
            if (recursionLength <= 0) return;

            double xW_SW = _xW - (_xE - _xW) * ((double)21 / (double)49);
            double yW_SW = _yW - (_yW - _yS) * ((double)48 / (double)42);
            double xE_SW = xW_SW + (_xE - _xW) * ((double)15 / (double)49);
            double xS_SW = (xE_SW + xW_SW) / (double)2;
            double yS_SW = _yW - (_yW - _yS) * ((double)61 / (double)42);

            double xW_NE = _xW + (_xE - _xW) * ((double)57 / (double)49);
            double yW_NE = _yW + (_yW - _yS) * ((double)27 / (double)42);
            double xE_NE = xW_NE + (_xE - _xW) * ((double)15 / (double)49);
            double xS_NE = (xE_NE + xW_NE) / (double)2;
            double yS_NE = _yW + (_yW - _yS) * ((double)14 / (double)42);

            //Second picture
            DrawFullTriangle(xW_SW, yW_SW, xE_SW, yW_SW, xS_SW, yS_SW);       // South
West Triangle
            DrawFullTriangle(xW_NE, yW_NE, xE_NE, yW_NE, xS_NE, yS_NE);       // North
East Triangle
            DrawFullTriangle(xW_NE, yW_SW, xE_NE, yW_SW, xS_NE, yS_SW);       // South
East Triangle
            DrawFullTriangle(xW_SW, yW_NE, xE_SW, yW_NE, xS_SW, yS_NE);       // North
West Triangle

            //Third picture
            //MoveInAllDirections(recursionLength - 2, xW_SW, yW_SW, xE_SW, yW_SW,
xS_SW, yS_SW);   // South West Triangle
            //MoveInAllDirections(recursionLength - 2, xW_NE, yW_NE, xE_NE, yW_NE,
xS_NE, yS_NE);   // North East Triangle
            //MoveInAllDirections(recursionLength - 2, xW_NE, yW_SW, xE_NE, yW_SW,
xS_NE, yS_SW);   // South East Triangle
            //MoveInAllDirections(recursionLength - 2, xW_SW, yW_NE, xE_SW, yW_NE,
xS_SW, yS_NE);   // North West Triangle

            //Fourth picture
            MoveInDifferentDirections(recursionLength - 1, xW_SW, yW_SW, xE_SW, yW_SW,
xS_SW, yS_SW);   // South West Triangle
            MoveInDifferentDirections(recursionLength - 1, xW_NE, yW_NE, xE_NE, yW_NE,
xS_NE, yS_NE);   // North East Triangle
            MoveInDifferentDirections(recursionLength - 1, xW_NE, yW_SW, xE_NE, yW_SW,
xS_NE, yS_SW);   // South East Triangle
            MoveInDifferentDirections(recursionLength - 1, xW_SW, yW_NE, xE_SW, yW_NE,
xS_SW, yS_NE);   // North West Triangle

        }

        //Drawing triangle
        public void DrawFullTriangle(double X0, double Y0, double X1, double Y1,
double X2, double Y2,
            double Precision = 0.5, uint Color = 0)
        {
            double Length = Math.Sqrt(Math.Pow(X1 - X0, 2) + Math.Pow(Y1 - Y0, 2));

            double XStep = (X1 - X0) / (Length / Precision);
            double YStep = (Y1 - Y0) / (Length / Precision);

            double XRun = X0;
            double YRun = Y0;
            for (double i = 0; i < Length; i += Precision)
            {
                XRun += XStep;
                YRun += YStep;
```

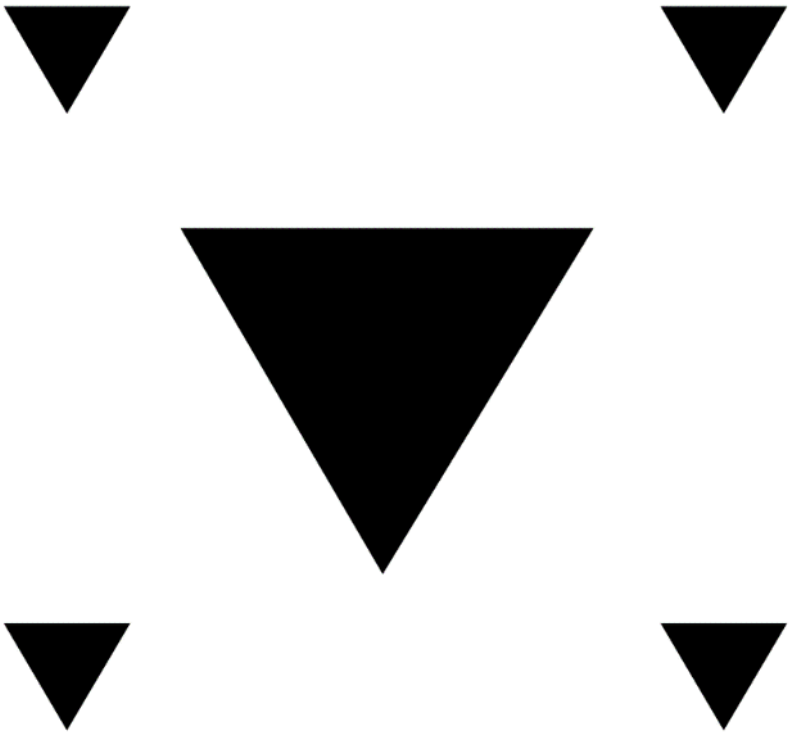
```
                DrawLine(XRun, YRun, X2, Y2, Precision, Color);
            }
        }

        //Drawing lines
        public void DrawLine(double X0, double Y0, double X1, double Y1, double
Precision = 0.5, uint Color = 0)
        {
            double Length = Math.Sqrt(Math.Pow(X0 - X1, 2) + Math.Pow(Y0 - Y1, 2));

            double XStep = (X1 - X0) / (Length / Precision);
            double YStep = (Y1 - Y0) / (Length / Precision);

            double XRun = X0;
            double YRun = Y0;
            for (double i = 0; i < Length; i += Precision)
            {
                XRun += XStep;
                YRun += YStep;

                SetPixel(XRun, YRun, Color);
            }
        }

        //Setting pixels
        private void SetPixel(double X, double Y, uint Color)
        {
            int Pixel = GetPixel(X, Y);
            if (Pixel < 0)
                return;

            Buffer[Pixel] = Color;
        }

        //Getting pixels
        private int GetPixel(double X, double Y)
        {
            int Pixel = ((int)Math.Round(Y) * Width) + (int)Math.Round(X);
            if (Pixel > Buffer.Length)
                return -1;

            if (X < 0)
                return -1;
            else if (X > Width)
                return -1;

            return Pixel;
        }
```

```
        //Printing picture to file
        public void Print()
        {
            using (FileStream File = new FileStream(OutputName, FileMode.Create,
FileAccess.Write))
            {

                File.Write(new byte[] { 0x42, 0x4D }); // BM
                File.Write(BitConverter.GetBytes(Height * Width * sizeof(uint) +
0x1A)); // Size
                File.Write(BitConverter.GetBytes(0)); // Reserved (0s)
                File.Write(BitConverter.GetBytes(0x1A)); // Image Offset (size of the
header)

                File.Write(BitConverter.GetBytes(0x0C)); // Header size (size is 12
bytes)
                File.Write(BitConverter.GetBytes(Width)); // Width
                File.Write(BitConverter.GetBytes(Height)); // Height
                File.Write(BitConverter.GetBytes((ushort)1)); // Color plane
                File.Write(BitConverter.GetBytes((ushort)32)); // bits per pixel

                byte[] Converted = new byte[Buffer.Length * sizeof(uint)];

                System.Buffer.BlockCopy(Buffer, 0, Converted, 0, Converted.Length);

                File.Write(Converted);
                File.Close();
            }
        }

        private readonly uint[] Buffer;
        private readonly ushort Width;
        private readonly ushort Height;
        private readonly string OutputName;
    }
}
```

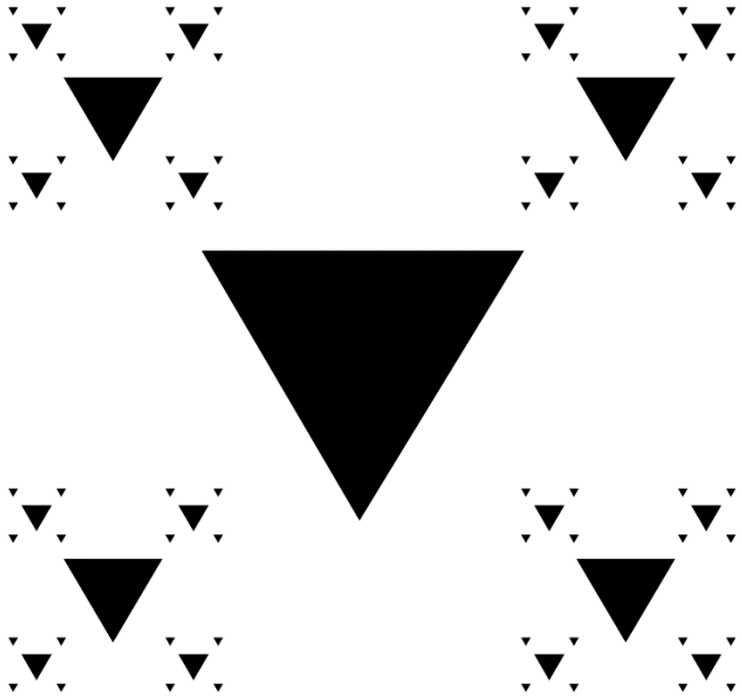
3.2. Programos formuojami rezultatai



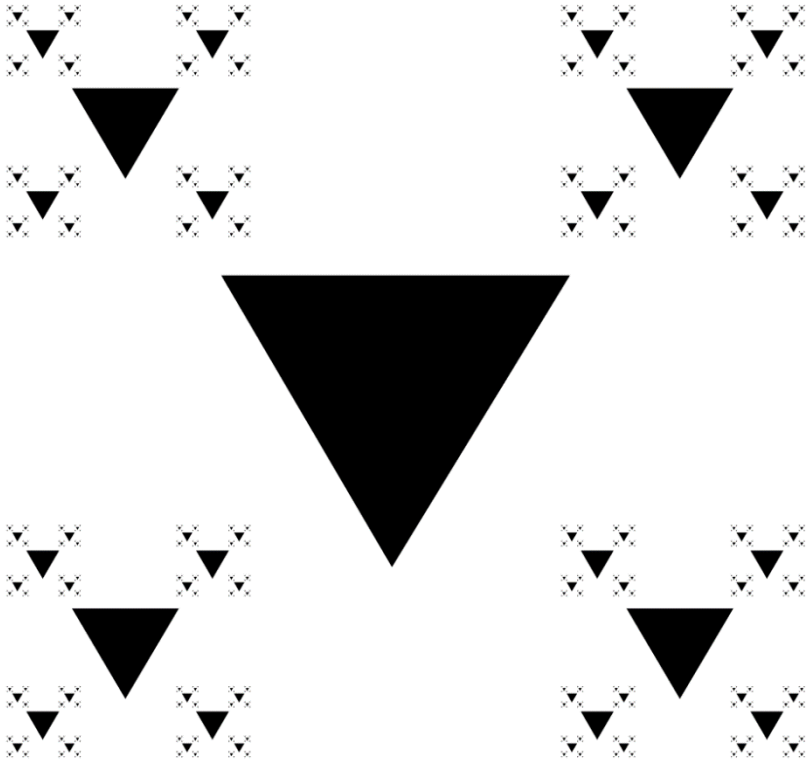
pav. 1



pav. 2



pav. 3



pav. 4