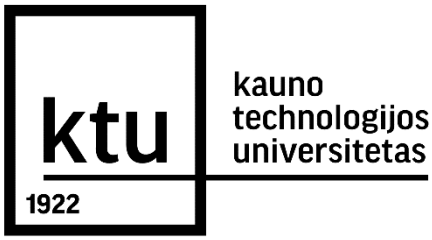
KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS



Modulio P170B400 „Algoritmų sudarymas ir analizė“

Laboratorinio darbo aprašas (ataskaita)
Antras laboratorinis darbas

Dėstytojas
lekt. MAKACKAS Dalius

Studentas
Matas Palujanskas IFF-1/8

KAUNAS 2023

TURINYS

1. Užduotis Nr. 1

1 užduoties dalis: Pateiktiems programinio kodo metodams „*methodToAnalysis(...)*“ (gautiems atlikus užduoties pasirinkimo testą):

- atlikite programinio kodo analizę, bei sudarykite rekurentinę lygį. Jei metodas neturi vidinių rekursinių kreipinių, apskaičiuokite pateikto metodo asimptotinį sudėtingumą. Jei metodo sudėtingumas priklauso nuo duomenų pateikiamų per parametrus – apskaičiuokite įverčius „iš viršaus“ ir „iš apačios“ *(2 balai)*.
- Metodams, kurie turi rekurentinių kreipinių išspręskite rekurentinę lygtį apskaičiuodami jos asimptotinį sudėtingumą *(1 balas)*.
- Atlikti eksperimentinį tyrimą (našumo testus: vykdymo laiką ir veiksmų skaičių) ir patikrinkite ar apskaičiuotas metodo asimptotinis sudėtingumas atitinka eksperimentinius rezultatus. Jei pateikto metodo asimptotinis sudėtingumas priklauso nuo duomenų, atitinkamai atliekant analizę reikia parinkti tokias testavimo duomenų imtis, kad rezultatai atspindėtų įvertinimus iš viršaus ir iš apačios *(1 balas)*.

Duoti kodai:

1.

```
public static long methodToAnalysis (int[] arr)
{
    long n = arr.Length;
    long k = n;
    if (arr[0] < 0) {
        for (int i = 0; i < n; i++)
        {
            if (i > 0)
            {
                for (int j = 0; j < n; j++)
                {
                    k -= 2;
                }
            }
        }
    }
    return k;
}
```

2.

```
public static long methodToAnalysis(int n)
{
    long k = 0;
    int[] arr = new int[n];
    Random randNum = new Random();
    for (int i = 0; i < n; i++)
    {
        arr[i] = randNum.Next(0, n);
        k += arr[i] + FF1(i);
    }

    return k;
}

public static long FF1(int n)
{
    if (n > 0)
    {
        return FF1(n - 1);
    }
    return n;
}
```

2. Algoritmai ir analizė

2.1. Kodas Nr. 1

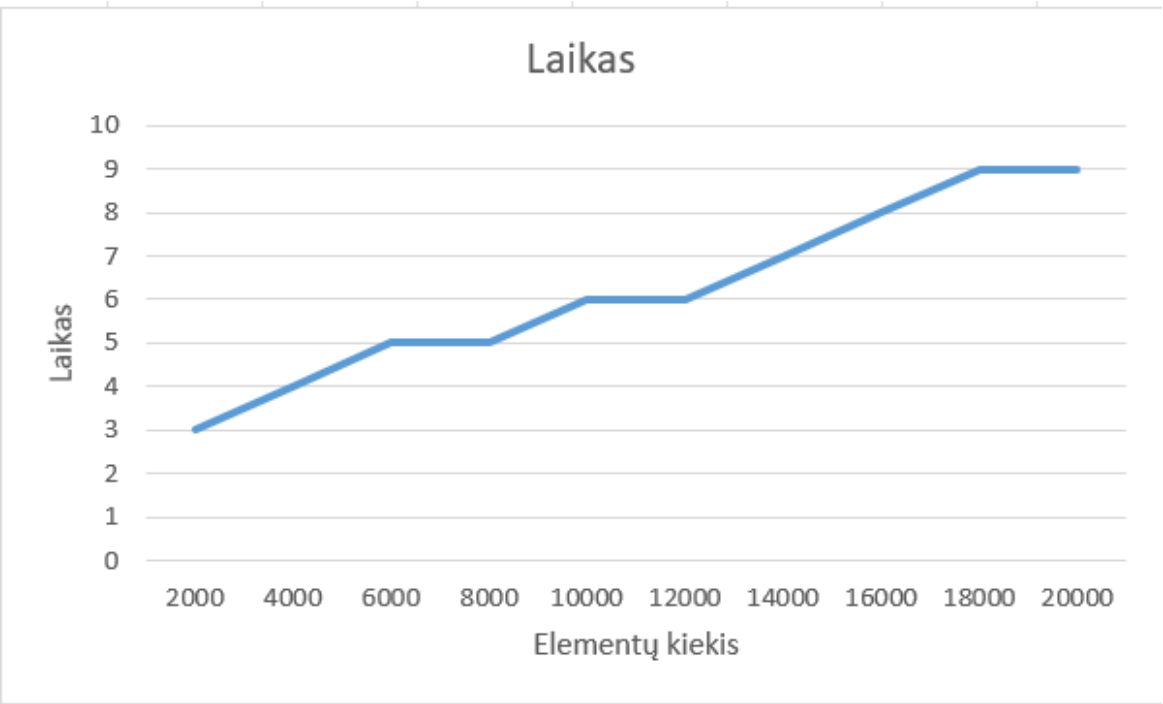
Kodas	Kaina	Kiekis
`public static long methodToAnalysis (int [] arr)`		
`{`		
`long n = arr.Length;`	c_1	1
`long k = n;`	c_2	1
`if (arr[0] < 0) {`	c_3	1
`for (int i = 0; i < n; i++)`	c_4	n
`{`		
`if (i > 0)`	c_5	n
`{`		
`for (int j = 0; j < n; j++)`	c_6	n * (n * γ);
`{`		
`k -= 2;`	c_7	n2 * γ;
`}`		
`}`		
`}`		
`}`		
`return k;`	c_8	1
`}`		

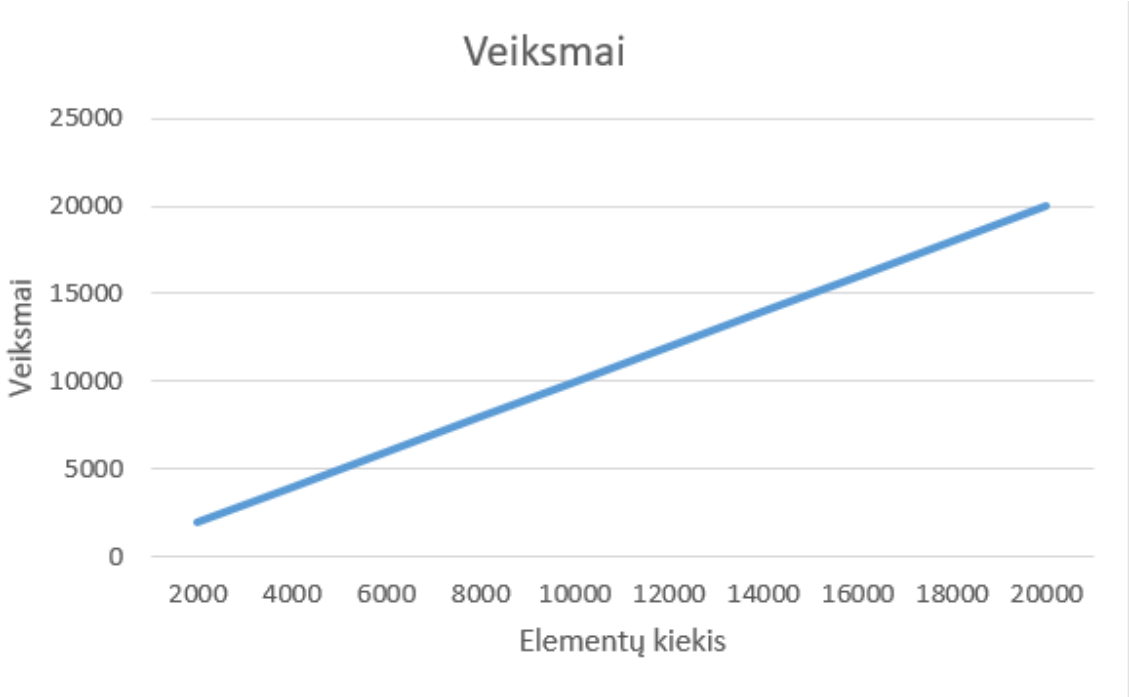
jei arr[0] < 0, tai γ=-1, kitu atveju γ=0.

$$T(n) = \begin{cases} c_1 + c_2 + c_3 + c_8, kai\ \gamma = 0 \\ (c_1 + c_2) + c_3 + c_4 * n + c_5 * n + c_6(n * \gamma) + c_7 * n^2 * \gamma + c_8, kai\ \gamma = -1 \end{cases}$$

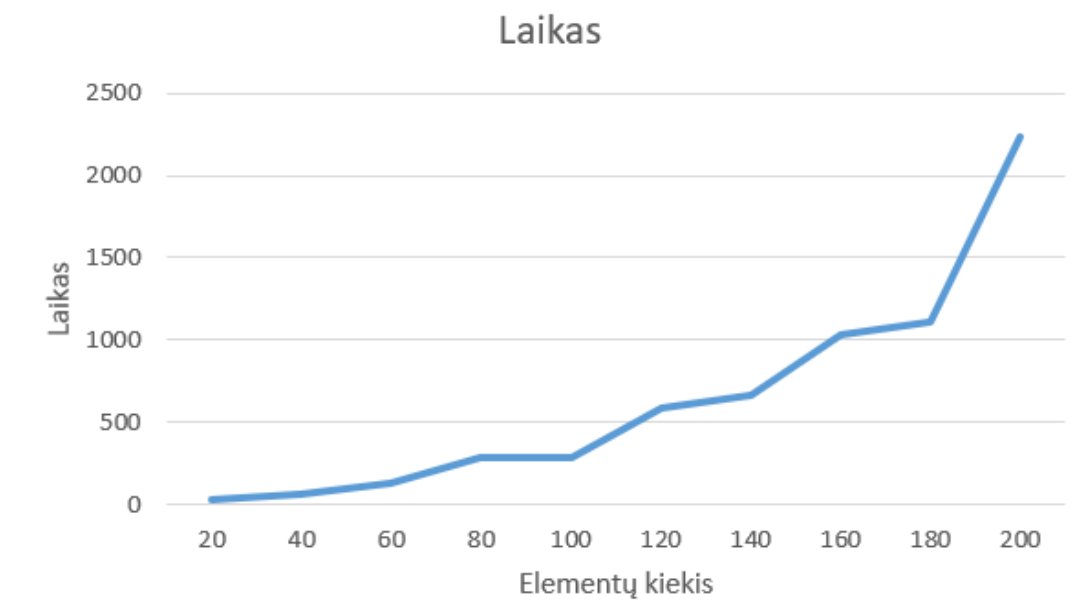
$$T(n) = \begin{cases} O(1), kai\ \gamma = 0 \\ O(n^2), kai\ \gamma = -1 \end{cases}$$

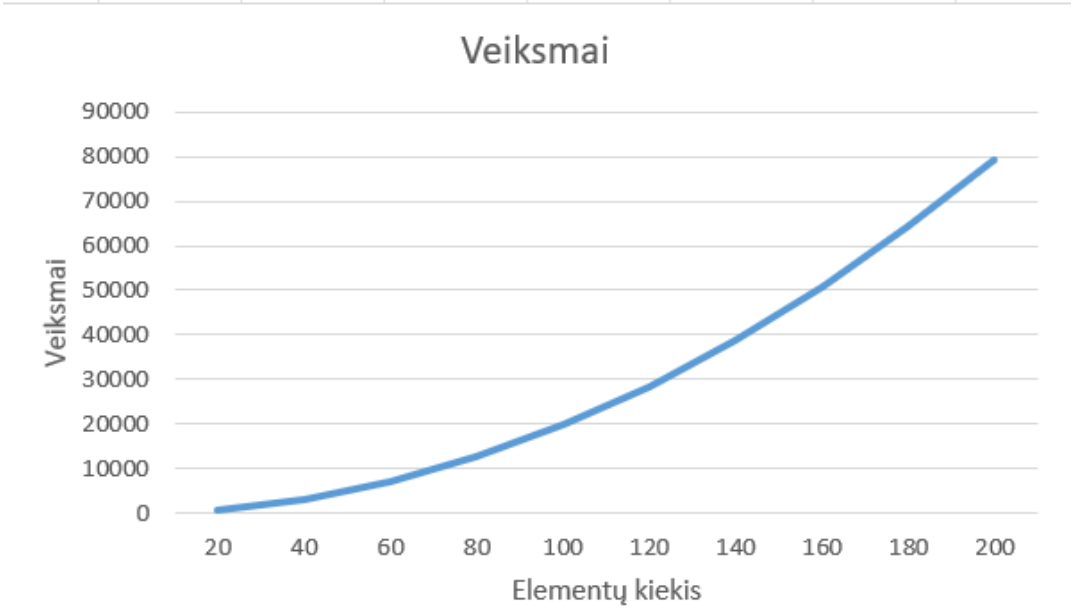
- Rezultatų atvaizdavimas grafikuose:
  Kai γ=0:





- Kai γ= -1:


Laikas
2500
2000
1500
1000
500
0
Laikas
20
40
60
80
100
120
140
160
180
200
Elementų kiekis

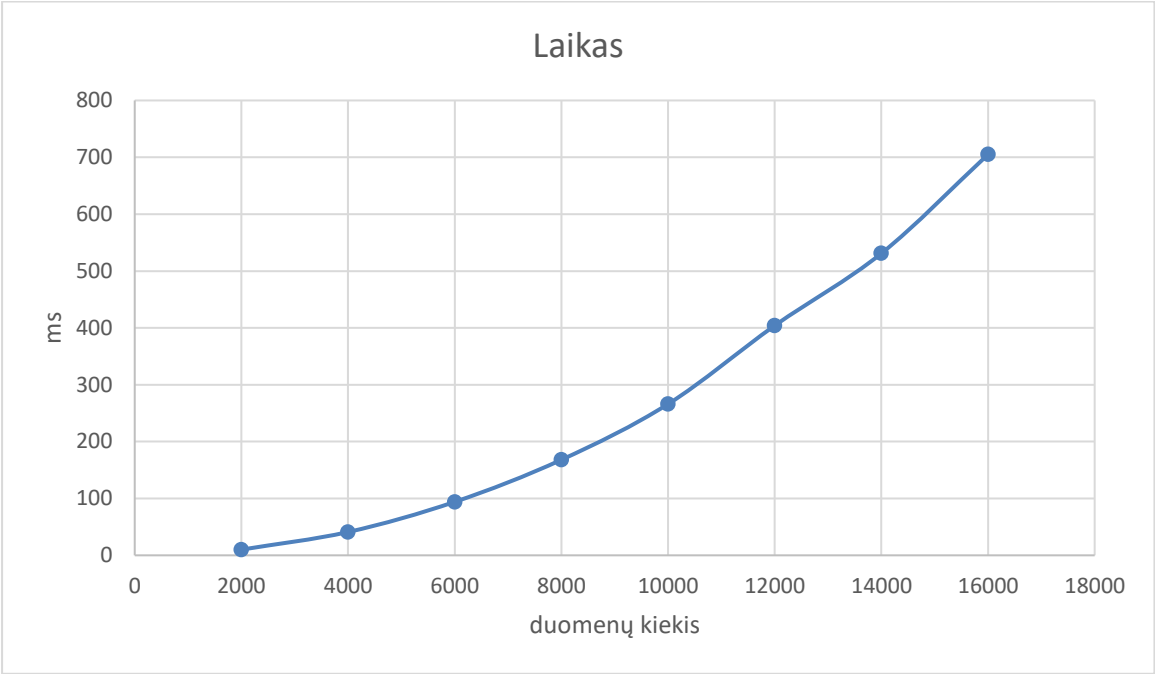

Veiksmai
90000
80000
70000
60000
50000
40000
30000
20000
10000
0
Veiksmai
20
40
60
80
100
120
140
160
180
200
Elementų kiekis

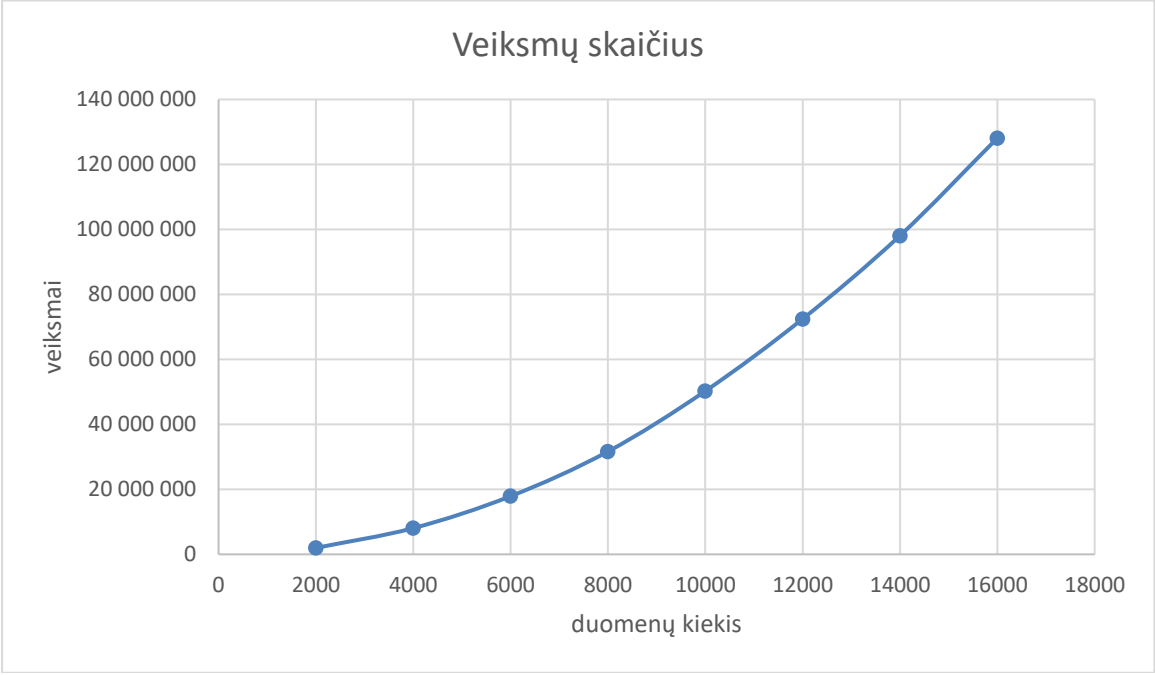
2.2. Kodas Nr. 2

Kodas	Kaina	Kiekis
`public static long methodToAnalysis2(int n)`		
`{`		
`long k = 0;`	c_1	1
`int[] arr = new int[n];`	c_2	1
`Random randNum = new Random();`	c_3	1
`for (int i = 0; i < n; i++)`	c_4	n
`{`		
`arr[i] = randNum.Next(0, n);`	c_5	n
`k += arr[i] + FF1(i);`	F(n)	n
`}`		
`return k;`	c_6	1
`}`		
`public static long FF1(int n)`		
`{`		
`if (n > 0)`	c_7	1
`{`		
`return FF1(n - 1);`	F(n-1)	1
`}`		
`return n;`	c_8	1
`}`		

$$T(n) = (c_1 + c_2 + c_3) + (c_4 + c_5) * n + F(n) * n + c_6 + c_7 + \mathrm{F(n-1)} + c_8$$
$$F(n) = F(n-1) + c_6, O(n)$$
$$T(n) = n^2 + n + c, O(n^2)$$

Rekursinis algoritmas vykdys mažinimus vienetu tol kol pasieks 0, todėl sudėtingumas yra O(n).


Laikas
800
700
600
500
400
300
200
100
0
ms
0
2000
4000
6000
8000
10000
12000
14000
16000
18000
duomenų kiekis


Veiksmų skaičius
140 000 000
120 000 000
100 000 000
80 000 000
60 000 000
40 000 000
20 000 000
0
veiksmai
0
2000
4000
6000
8000
10000
12000
14000
16000
18000
duomenų kiekis

3. Kodas

```
using System.Diagnostics;

namespace Pirma
{
    class Program
    {
        static void Main(string[] args)
        {
            int size = 2000;
                  int n = 10;
                  long k;
                  Console.WriteLine("Pirma rekurentine: ");
                  Console.WriteLine("1 - arr[0] = 0");
                  for (int i = 0; i < n; i++)
                  {
                        Stopwatch stopwatch = new Stopwatch();
                        int[] A = new int[size];
                        A[0] = 0;
                        stopwatch.Start();
                        k = methodToAnalysis1(A);
                        stopwatch.Stop();
                        Console.WriteLine($"Function working time:
{stopwatch.Elapsed}");
                        Console.WriteLine($"Elements amount: {size}");
                        Console.WriteLine($"Counter: {k}");
                        Console.WriteLine();
                        GC.Collect();
                        size += 2000;
                  }

                  size = 20;
                  Console.WriteLine("Pirma rekurentine: ");
                  Console.WriteLine("1 - arr[0] = 1");
                  for (int i = 0; i < n; i++)
                  {
                        Stopwatch stopwatch = new Stopwatch();
                        int[] A = new int[size];
                        A[0] = -1;
                        stopwatch.Start();
                        k = methodToAnalysis1(A);
                        stopwatch.Stop();
                        Console.WriteLine($"Function working time:
{stopwatch.Elapsed}");
                        Console.WriteLine($"Elements amount: {size}");
                        Console.WriteLine($"Counter: {k}");
                        Console.WriteLine();
                        GC.Collect();
                        size += 20;
                  }

                  n = 8;
                  size = 2000;
                  Console.WriteLine("Antra rekurentine:");
                  for (int i = 0; i < n; i++)
                  {
                        Stopwatch stopwatch = new Stopwatch();
                        stopwatch.Start();
                        k = methodToAnalysis2(size);
                        stopwatch.Stop();
```

```
                        Console.WriteLine($"Function working time:
{stopwatch.Elapsed}");
                        Console.WriteLine($"Elements amount: {size}");
                        Console.WriteLine($"Counter: {k}");
                        Console.WriteLine();
                        GC.Collect();
                        size += 2000;
                    }           for (int i = 0; i < n; i++)

            }
        }

        public static long methodToAnalysis1(int[] arr)

        {

            long n = arr.Length;
            long k = n * n;
            for (int i = 0; i < n * n; i++)
            {
                if (arr[0] > 0)
                {
                    for (int j = 0; j < n; j++)
                    {
                        k -= 2;
                    }
                    for (int j = 0; j < n * n / 2; j++)
                    {
                        k += 3;
                    }
                }
            }

            return k;

        }

        public static long methodToAnalysis2(int n)

        {

            long k = 0;
            int[] arr = new int[n];
            Random randNum = new Random();
            for (int i = 0; i < n; i++)
            {
                arr[i] = randNum.Next(0, n);
                k += arr[i] + FF1(i);
            }
            return k;

        }

        public static long FF1(int n)

        {
            if (n > 0)
            {
                return FF1(n - 1);
            }
            return n;
        }
    }
}
```

4. Užduotis Nr. 2

2 užduoties dalis (6 balai):

- Pateikite rekursinį uždavinio sprendimo algoritmą (rekursinis sąryšis su paaiškinimais), bei realizuokite programinį kodą sprendžiantį nurodytą uždavinį (rekursinis sprendimas netaikant dinaminio programavimo).
- Pritaikykite dinaminio programavimo metodologiją pateiktam uždaviniui (pateikti paaiškinimą), bei realizuokite programinį kodą sprendžiantį nurodytą uždavinį (taikant dinaminį programavimą).
- Atlikite realizuotų programinių kodų analizę ir apskaičiuokite įverčius „iš viršaus“ ir „iš apačios“. Atlikite našumo analizę (skaičiuojant programos vykdymo laiką arba veiksmų skaičių) ir patikrinkite, ar apskaičiuotas metodo asimptotinis sudėtingumas atitinka eksperimentinius rezultatus.

Užduotis:

Du broliai rado lobį, kurį sudaro n daiktų ir nori jį pasidalinti tarpusavyje. Visi daiktai turi būti pasidalinti, nei vienas daiktas negali būti padalintas į dvi dalis. Kiekvienas daiktas turi savo vertę, išreikštą sveikuoju skaičiumi. Daiktus reikia padalinti taip, kad broliams tenkančių daiktų verčių sumos būtų kuo panašesnės.

5. Algoritmai

Sprendimas naudojant rekursiją:

Kodas	Kaina	Kiekis
`static void CanDivide(int index, int sum1, int sum2)`		
`{`		
`if (index == values.Length)`	c_1	1
`{`		
`int difference = Math.Abs(sum1 - sum2);`	c_2	1
`if (difference < minDifference)`	c_3	1
`{`		
`minDifference = difference;`	c_4	1
`firstBrother.Clear();`	c_5	1
`secondBrother.Clear();`	c_6	1
`for (int i = 0; i < selected.Length; i++)`	c_7	n
`{`		
`if (selected[i])`	c_8	n
`{`		
`firstBrother.Add(values[i]);`	c_9	n
`}`		
`else`		
`{`		
`secondBrother.Add(values[i]);`	c_{10}	n
`}`		
`}`		
`}`		
`return;`	c_{11}	1
`}`		
`selected[index] = true;`	c_{12}	1
`CanDivide(index + 1, sum1 + values[index], sum2);`	T(n)	1
`selected[index] = false;`	c_{13}	1
`CanDivide(index + 1, sum1, sum2 + values[index]);`	T(n)	1
`}`		

T(n) = (c1+c2+c3+c4+c5+c6+(c7+c8+c9+c10)*n+c11+c12+T(n)+c13+T(n)

Gaunamas sudėtingumas yra O(n).

Kodas naudojant rekursiją:

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Drawing;

class TreasureDivisionRecursion
{
	static int[] values;
	static bool[] selected;
	static List<int> firstBrother, secondBrother;
	static int minDifference = int.MaxValue;

	/// <summary>
	/// Recursive solution
	/// </summary>
	/// <param name="index"></param>
	/// <param name="sum1"></param>
	/// <param name="sum2"></param>
	static void CanDivide(int index, int sum1, int sum2)
	{
		Stopwatch stopwatch = new Stopwatch();
		stopwatch.Start();

		if (index == values.Length)
		{
			int difference = Math.Abs(sum1 - sum2);
			if (difference < minDifference)
			{
				minDifference = difference;
				firstBrother.Clear();
				secondBrother.Clear();

				for (int i = 0; i < selected.Length; i++)
				{
					if (selected[i])
					{
						firstBrother.Add(values[i]);
					}
					else
					{
						secondBrother.Add(values[i]);
					}
				}
			}
			return;
		}

		// Try adding the item to the first brother's share
		selected[index] = true;
		CanDivide(index + 1, sum1 + values[index], sum2);

		// Try adding the item to the second brother's share
		selected[index] = false;
		CanDivide(index + 1, sum1, sum2 + values[index]);
		stopwatch.Stop();
```

```
            Console.WriteLine($"Function working time:
{stopwatch.Elapsed}");
            Console.WriteLine();
      }

      static void Main()
      {
            values = new int[] { 2, 4, 5, 3 };
            selected = new bool[values.Length];
            firstBrother = new List<int>();
            secondBrother = new List<int>();

            CanDivide(0, 0, 0);

            if (minDifference == 0)
            {
                  Console.WriteLine("The treasure can be divided equally:");
            }
            else
            {
                  Console.WriteLine("The treasure cannot be divided equally,
but can be divided similarly:");
            }
            Console.WriteLine("First brother's share: " + string.Join(", ",
firstBrother));
            Console.WriteLine("Second brother's share: " + string.Join(", ",
secondBrother));
      }
}
```

Sprendimas naudojant dinaminio programavimo metodologiją:

Kodas	Kaina	Kiekis
`public static bool CanDivide(int[] items)`		
`{`		
`int n = items.Length;`	c_1	1
`int S = 0;`	c_2	1
`for (int i = 0; i < n; i++)`	c_3	n
`{`		
`S += items[i];`	c_4	n
`}`		
`if (S % 2 != 0)`	c_5	1
`{`		
`return false;`	c_6	1
`}`		
`bool[,] DP = new bool[n + 1, S / 2 + 1];`	c_7	1
`for (int i = 0; i <= n; i++)`	c_8	n
`{`		
`DP[i, 0] = true;`	c_9	n
`}`		
`for (int i = 1; i <= n; i++)`	c_{10}	n
`{`		
`for (int j = 1; j <= S / 2; j++)`	c_{11}	n^2
`{`		
`if (items[i - 1] > j)`	c_{12}	n^2
`{`		
`DP[i, j] = DP[i - 1, j];`	c_{13}	n^2
`}`		
`else`		
`{`		
`DP[i, j] = DP[i - 1, j] || DP[i - 1, j - items[i - 1]];`	c_{14}	n^2
`}`		
`}`		
`}`		
`return DP[n, S / 2];`	c_{15}	1
`}`		

T(n) = (c1+c2+(c3+c4)*n+c5+c6+c7+(c8+c9+c10)*n+(c11+c12+c13+c14)* n^2+c15

Gaunamas sudėtingumas yra O(n^2).

Kodas naudojant dinaminio programavimo metodologiją:

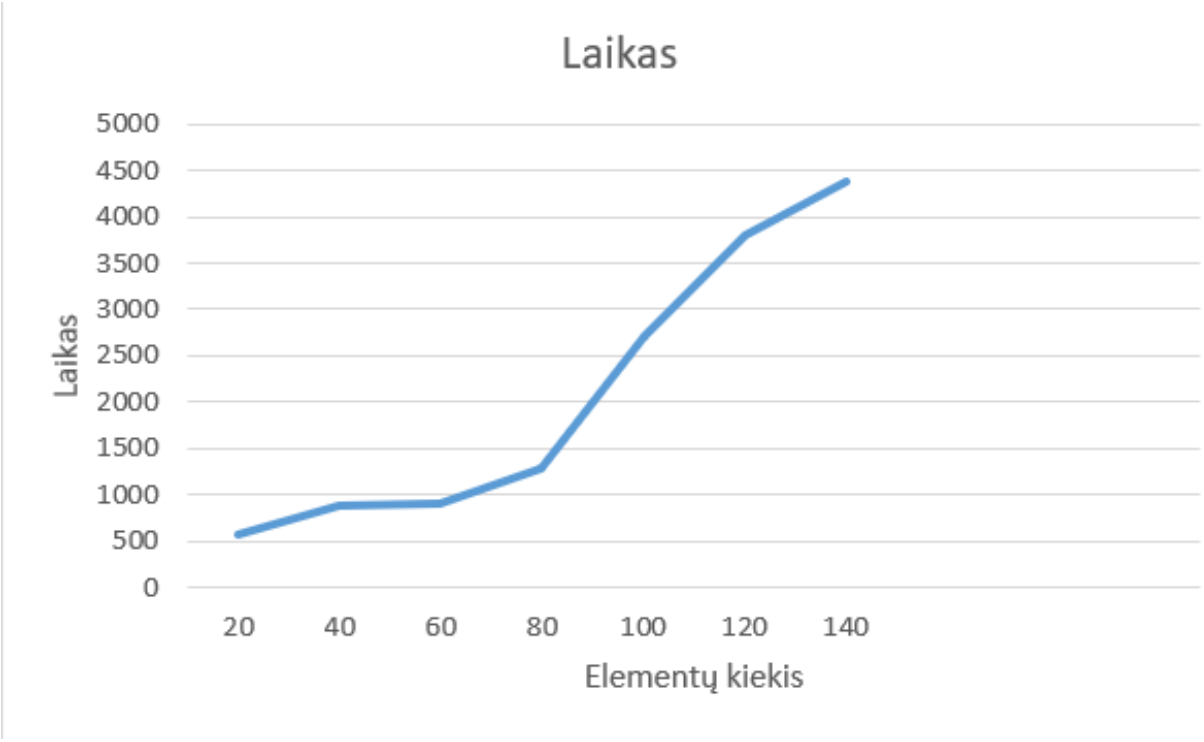
```
using System;

public class TreasureDivision
{
	public static bool CanDivide(int[] items)
	{
		int n = items.Length;
		int S = 0;
		for (int i = 0; i < n; i++)
		{
			S += items[i];
		}
		if (S % 2 != 0)
		{
			// The total sum is odd, so the items cannot be divided
into two groups with equal sums.
			return false;
		}
		bool[,] DP = new bool[n + 1, S / 2 + 1];
		// Initialize the base case where the sum is zero.
		for (int i = 0; i <= n; i++)
		{
			DP[i, 0] = true;
		}
		// Compute the remaining cases.
		for (int i = 1; i <= n; i++)
		{
			for (int j = 1; j <= S / 2; j++)
			{
				if (items[i - 1] > j)
				{
					DP[i, j] = DP[i - 1, j];
				}
				else
				{
					DP[i, j] = DP[i - 1, j] || DP[i - 1, j - items[i
- 1]];
				}
			}
		}
		return DP[n, S / 2];
	}

	public static void Main(string[] args)
	{
		int[] items = new int[] { 2, 4, 5, 6, 7 };
		if (CanDivide(items))
		{
			Console.WriteLine("The items can be divided into two groups
with equal sums.");
		}
		else
		{
			Console.WriteLine("The items cannot be divided into two
groups with equal sums.");
		}
	}
}
```

6. Rezultatai

Apskaičiuotas algoritmo darbo laikas, su skirtingu kiekiu duomenų ir gauti tokie rezultatai:



Prilyginus grafą O(n^2), matoma, jog sutampa kilimo tendencija, todėl galima spręsti, jog toks yra ir šio algoritmo sudėtingumas.