KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS

Algoritmų sudarymas ir analizė (P170B400)

Laboratorinių darbų ataskaita

Atliko:

IFF-1/1 gr. studentas

Vytenis Kriščiūnas

2023 m. kovo 21 d.

Priėmė:

Lekt. Vidmantas Rimavičius

KAUNAS 2023

TURINYS

1. Pirma 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).

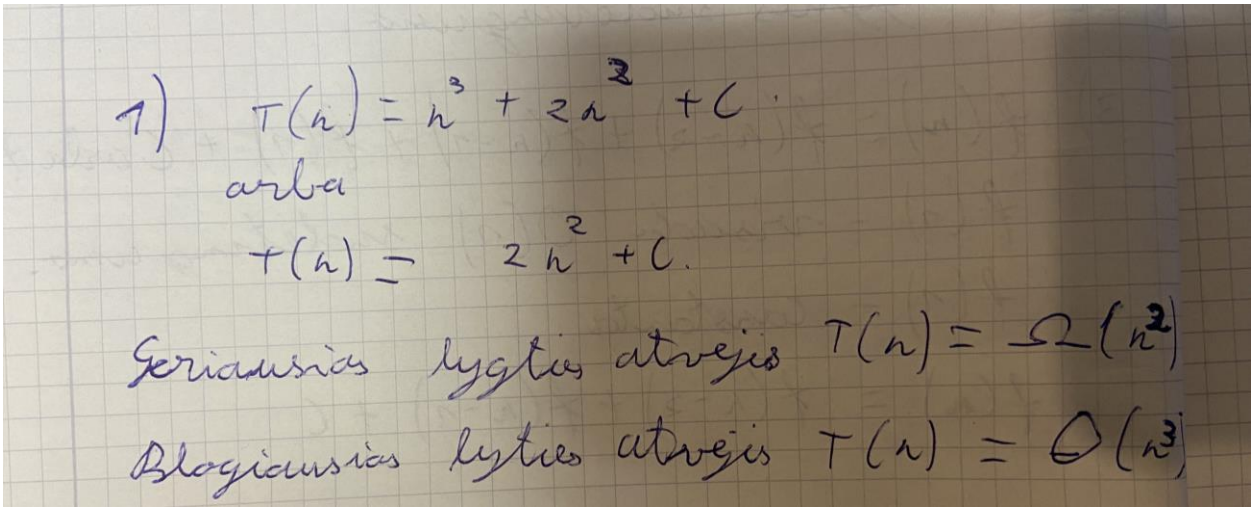
1.1 Pirmasis metodas

1.1.1 Kodo analizė

```
public static long methodToAnalysis1(int[] arr)
        {
            sk1++;
            long n = arr.Length;                    //          c1 | 1
            sk1++;
            long k = n;                             //          c2 | 1
            sk1++;
            for (int i = 0; i < n * 2 * n; i++)     //          c3 | 2*n*n
+ 1
            {
                sk1++;
                if (arr[0] > 0)                     //          c4 | 1
                {
                    sk1++;
                    for (int j = 0; j < n / 2; j++) //          c5 | (n/2 +
1) * 2*n*n
                    {
                        sk1++;
                        k -= 2;                     //          c6 |
1*n*n*n
                        sk1++;
                    }

                }
                sk1++;
            }
            sk1++;
            return k;                               //          c7 | 1
        }
```

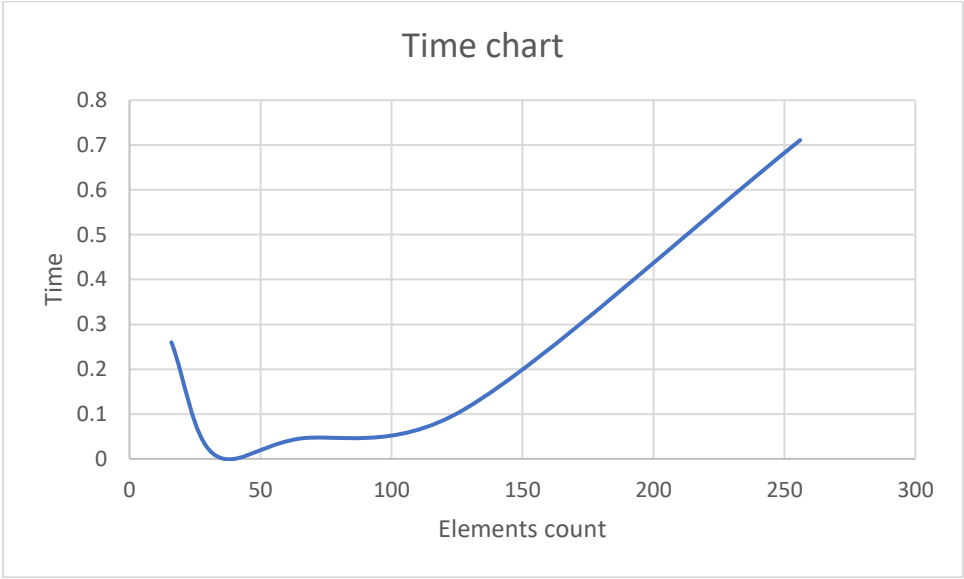
```
        // T(n) = n^3*(c5 + c6) + 2*n^2*(c3 + c5) + c1 + c2 + c3 + c4 + c7
, kai arr[0] > 0
        // T(n) = c1 + c2 + c3*(2*n^2 + 1) + c4 + c7 = 2*n^2*c3 + c1 + c2 +
c3 + c4 + c7, kai arr[0] <= 0
```

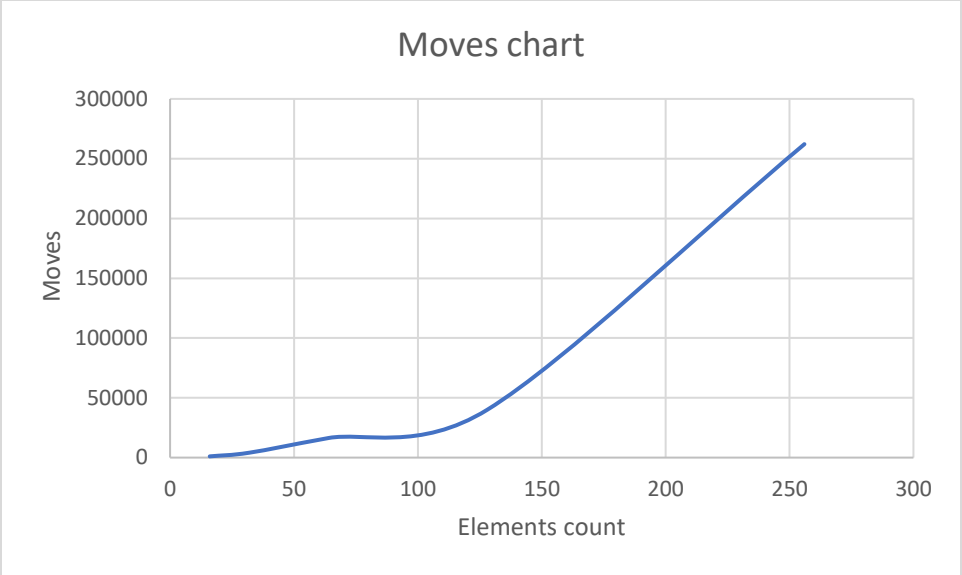
1.1.2 Lygties sprendimas



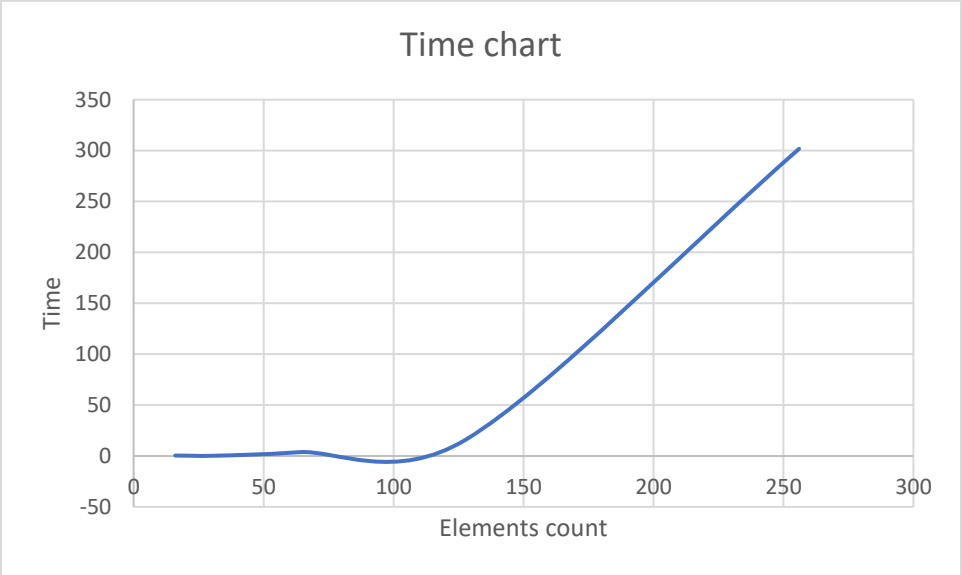
1.1.3 Eksperimentinis tyrimas

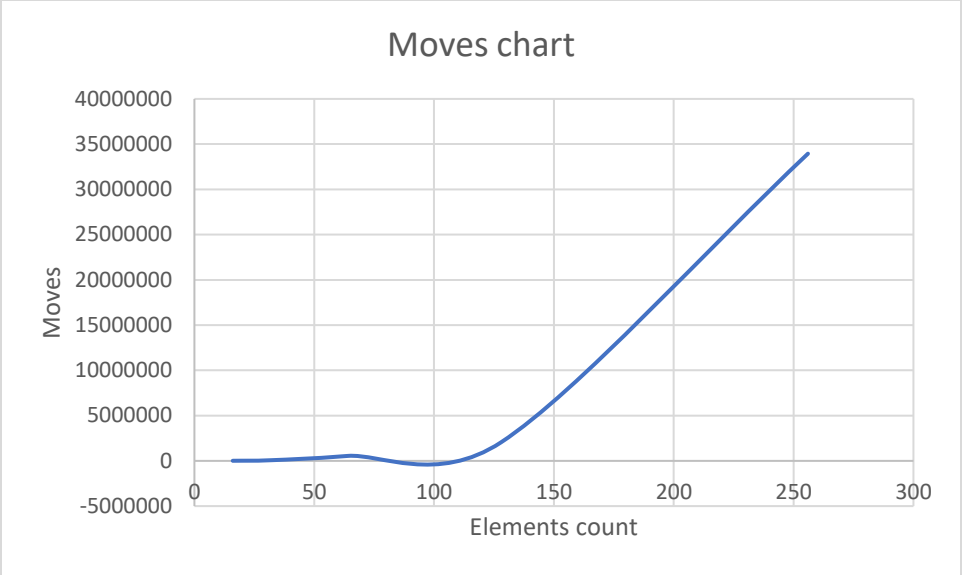
Geriausio atvejo rezultatai:





Blogiausio atvejo rezultatai:





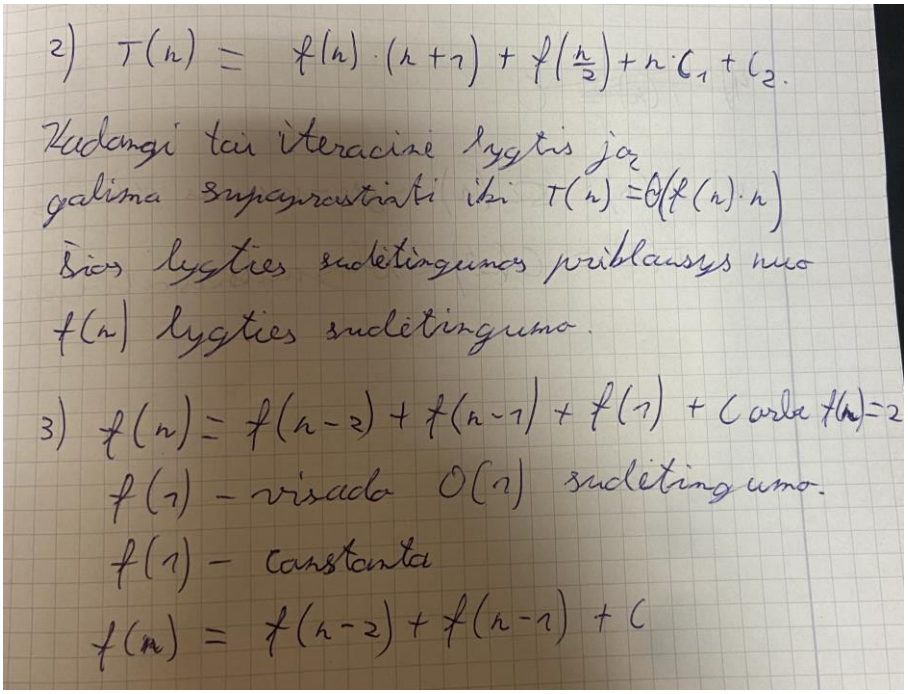
1.2 Antrasis metodas

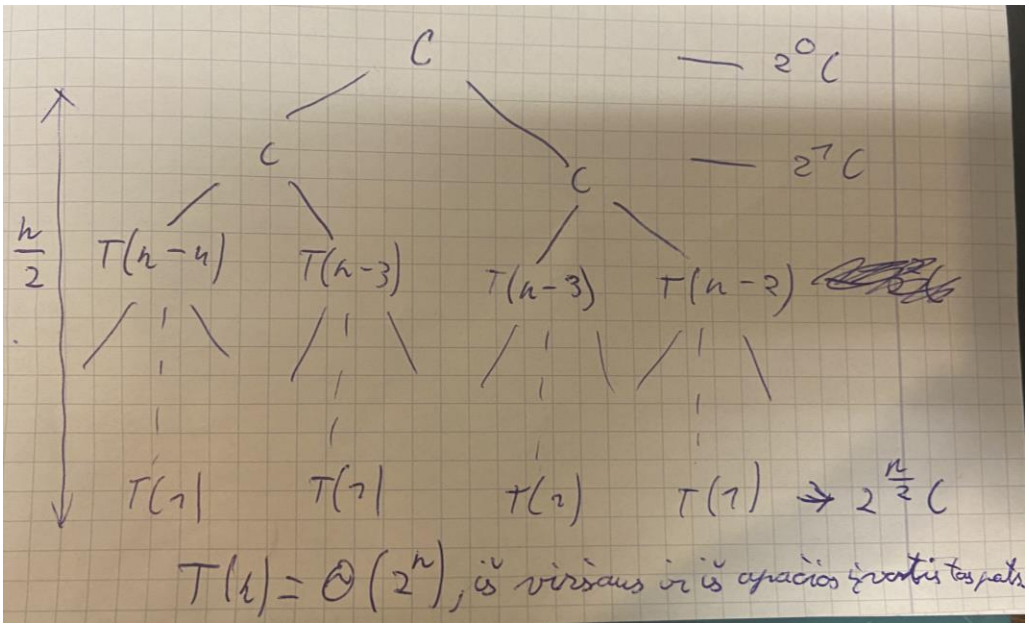
1.2.1 Kodo analizė

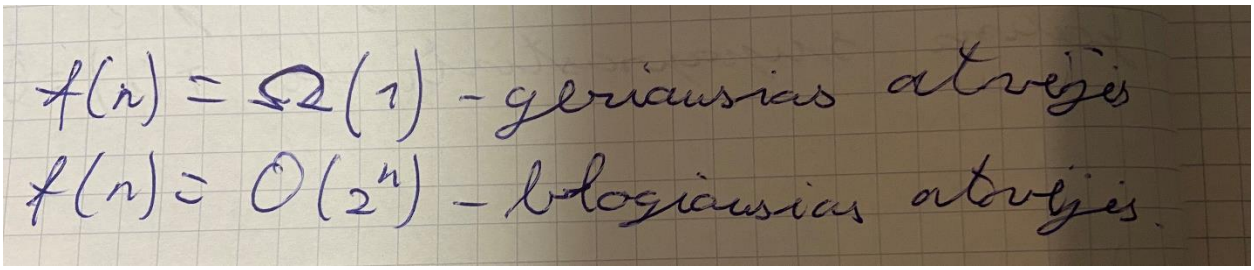
```
public static long methodToAnalysis2(int n, int[] arr)
        {
            sk2++;
            long k = 0;                                 //              c1 | 1
            sk2++;
            for (int i = 0; i < n; i++)                 //              c2 | n + 1
            {
                sk2++;
                k += k;                                 //              c3 | n
                sk2++;
                k += FF9(i, arr);                       //     (c4 + f(n))| n
                sk2++;
            }
            sk2++;
            k += FF9(n, arr);                           //       c5 + f(n) | 1
            sk2++;
            k += FF9(n / 2, arr);                       //     c6 + f(n/2) | 1
            sk2++;
            return k;                                   //              c7 | 1

        }
        // T(n) = f(n)*(n + 1) + f(n/2) + n*(c2 + c3 + c4) + c1 + c2 + c5 +
c6 + c7

        public static long FF9(int n, int[] arr)
        {
            sk2++;
            if (n > 1 && arr.Length > n && arr[0] < 0)
//          c1 | 1
            {
                sk2++;
                return FF9(n - 2, arr) + FF9(n - 1, arr) + FF9(n / n, arr);
//          f(n-2) + f(n-1) + f(1) + c2 | 1
```

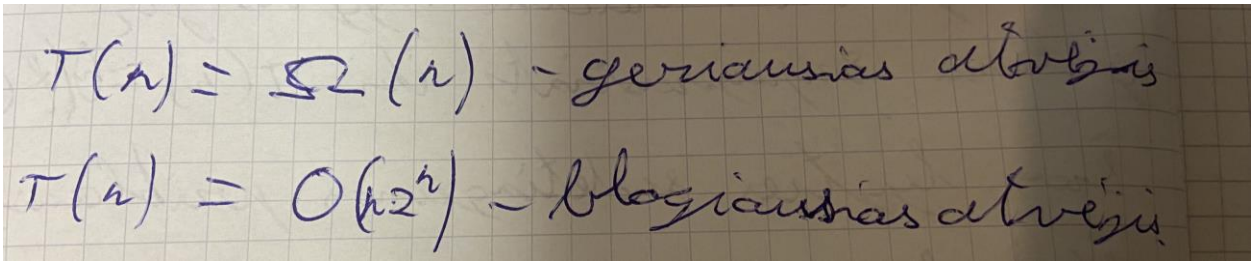
```
            }
            sk2++;
            return n;
//          c3 | 1

        }
        // f(n) = c1 + c2                                    , kai n <= 1 arba
arr.Length <= n arba arr[0] >= 0
            // f(n) = f(n-2) + f(n-1) + f(1) + c1 + c2 + c3, kai n > 1 ir
arr.Length > n ir arr[0] < 0
```

1.2.2 Lygties sprendimas



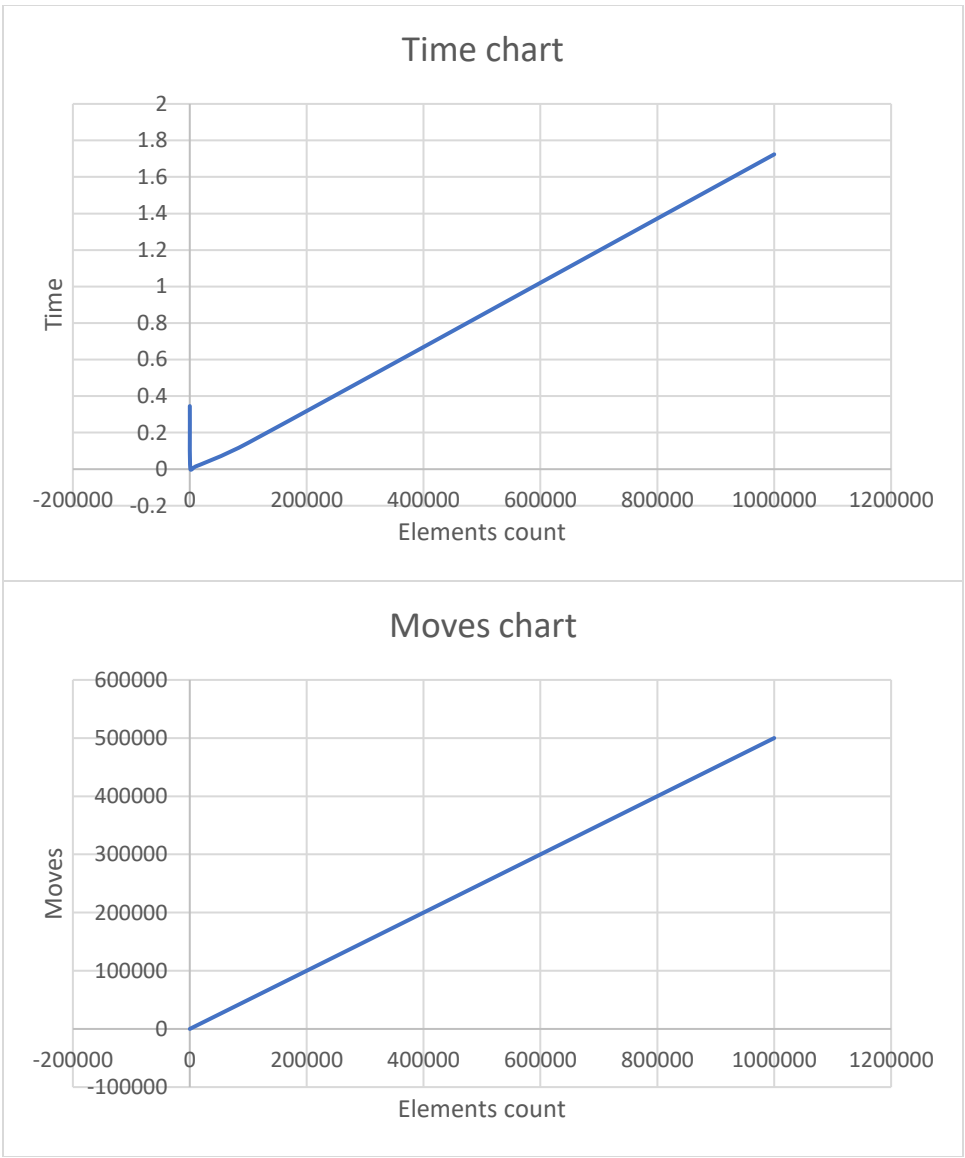




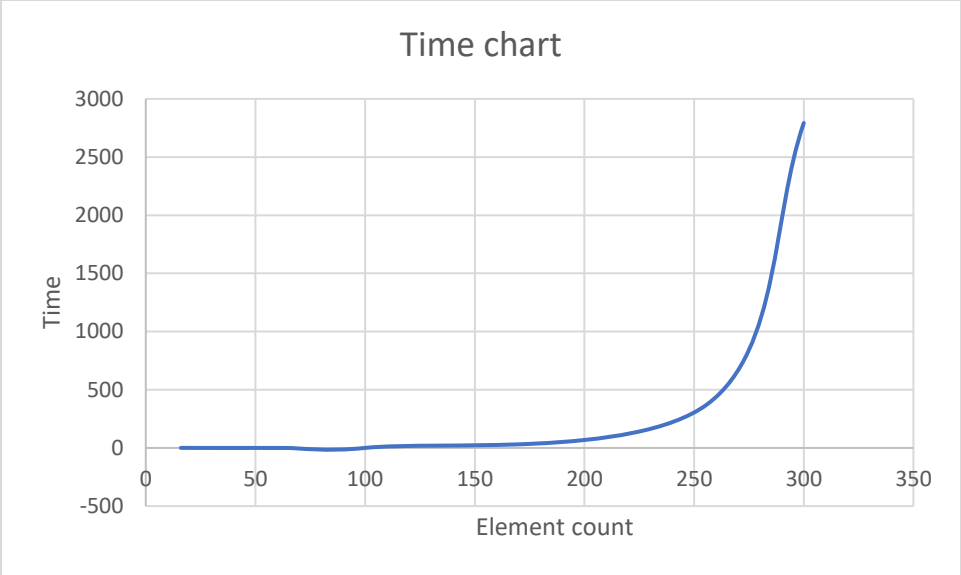


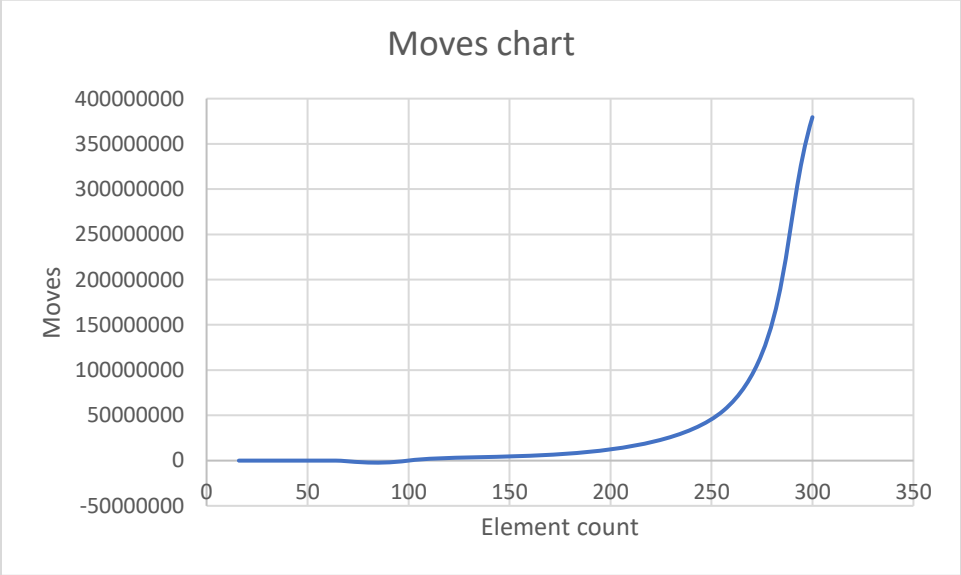
1.2.3 Eksperimentinis tyrimas

Geriausio atvejo rezultatai:



Blogiausio atvejo rezultatai:





1.3 Programos kodas

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace lab2
{
internal class Class1
{
private static int sk1;
private static int sk2;
public static void Main(string[] args)
{
```

```
int[] arr = new int[0];
long n = 0;
Console.WriteLine("Pirmoji lygtis: ");
arr = new int[16];
arr[0] = 1;
Test1(arr);
arr = new int[32];
arr[0] = 1;
Test1(arr);
arr = new int[64];
arr[0] = 1;
Test1(arr);
arr = new int[100];
arr[0] = 1;
Test1(arr);
arr = new int[256];
arr[0] = 1;
Test1(arr);
Console.WriteLine();

//Console.WriteLine("Antroji lygtis: ");
//n = 2;
//arr = new int[16];
//arr[0] = -1;
//Test2(n, arr);
//n = 4;
//arr = new int[32];
//arr[0] = -1;
//Test2(n, arr);
//n = 8;
//arr = new int[64];
//arr[0] = -1;
//Test2(n, arr);
//n = 16;
//arr = new int[100];
//arr[0] = -1;
//Test2(n, arr);
//n = 32;
//arr = new int[256];
//arr[0] = -1;
//Test2(n, arr);
//n = 36;
//arr = new int[300];
//arr[0] = -1;
//Test2(n, arr);
//Console.WriteLine();
}

public static void Test1(int[] arr)
{
Stopwatch time = new Stopwatch();
sk1 = 0;
time.Start();
methodToAnalysis1(arr);
time.Stop();
Console.WriteLine("Duomenu kiekis: {0}", arr.Length);
```

```
Console.WriteLine("Elapsed time: {0} microsec.",
time.Elapsed.TotalMilliseconds);
Console.WriteLine("Veiksmu skaicius: {0}", sk1);

}

public static void Test2(long n, int[] arr)
{
Stopwatch time = new Stopwatch();
sk2 = 0;
time.Start();
methodToAnalysis2(n, arr);
time.Stop();
Console.WriteLine("Duomenu kiekis: {0} ir n reiksme: {1}", arr.Length,
n);
Console.WriteLine("Elapsed time: {0} microsec.",
time.Elapsed.TotalMilliseconds);
Console.WriteLine("Veiksmu skaicius: {0}", sk2);

}

public static long methodToAnalysis1(int[] arr)
{
sk1++;
long n = arr.Length;                    //            c1 | 1
sk1++;
long k = n;                             //            c2 | 1
sk1++;
for (int i = 0; i < n * 2 * n; i++)     //            c3 | 2*n*n + 1
{
sk1++;
if (arr[0] > 0)                     //            c4 | 1
{
sk1++;
for (int j = 0; j < n / 2; j++) //            c5 | (n/2 + 1) * 2*n*n
{
sk1++;
k -= 2;                     //            c6 | 1*n*n*n
sk1++;
}

}
sk1++;
}
sk1++;
return k;                               //            c7 | 1
}
// T(n) = n^3*(c5 + c6) + 2*n^2*(c3 + c5) + c1 + c2 + c3 + c4 + c7
, kai arr[0] > 0
// T(n) = c1 + c2 + c3*(2*n^2 + 1) + c4 + c7 = 2*n^2*c3 + c1 + c2 + c3
+ c4 + c7, kai arr[0] <= 0

public static long methodToAnalysis2(long n, int[] arr)
{
sk2++;
long k = 0;                             //            c1 | 1
sk2++;
```

```
for (int i = 0; i < n; i++)              //            c2 | n + 1
{
sk2++;
k += k;                                  //            c3 | n
sk2++;
k += FF9(i, arr);                        //    (c4 + f(n))| n
sk2++;
}
sk2++;
k += FF9(n, arr);                            //      c5 + f(n) | 1
sk2++;
k += FF9(n / 2, arr);                        //    c6 + f(n/2) | 1
sk2++;
return k;                                    //             c7 | 1

}
// T(n) = f(n)*(n + 1) + f(n/2) + n*(c2 + c3 + c4) + c1 + c2 + c5 + c6
+ c7

public static long FF9(long n, int[] arr)
{
sk2++;
if (n > 1 && arr.Length > n && arr[0] < 0)                          //
c1 | 1
{
sk2++;
return FF9(n - 2, arr) + FF9(n - 1, arr) + FF9(n / n, arr); //
f(n-2) + f(n-1) + f(1) + c2 | 1

}
sk2++;
return n;                                                           //
c3 | 1

}
// f(n) = c1 + c2                                  , kai n <= 1 arba
arr.Length <= n arba arr[0] >= 0
// f(n) = f(n-2) + f(n-1) + f(1) + c1 + c2 + c3, kai n > 1 ir
arr.Length > n ir arr[0] < 0
}
}
```

2. Antra užduoties dalis

- 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.

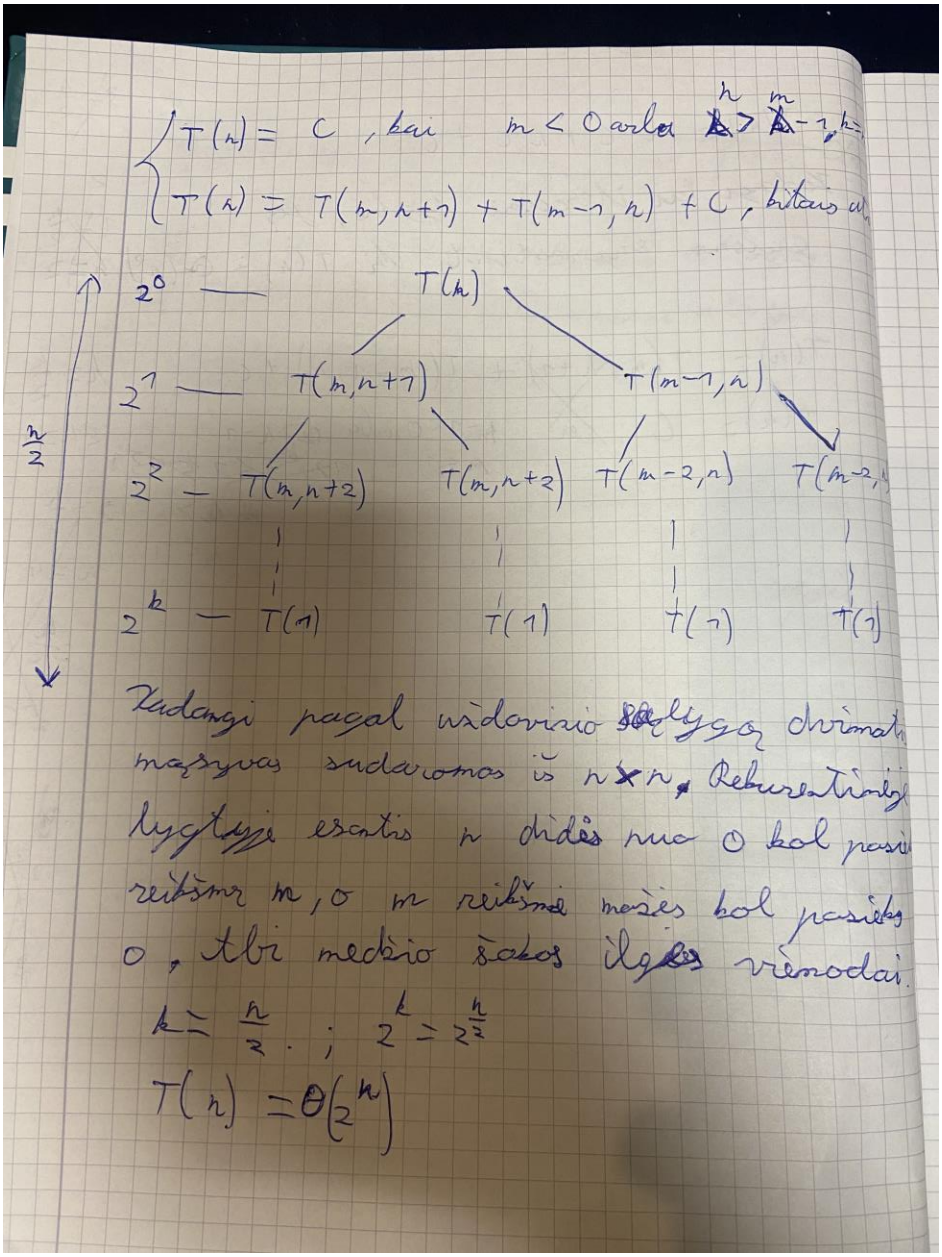
Kvadrato formos regione kelininkai planuoja nutiesti kelią nuo regiono pietvakarinio iki šiaurės rytų kampo. Regionas suskirstytas į $n \times n$ kvadratėlių. Žemės tinkamumas kelio tiesimui kiekviename kvadratėlyje yra išreiškiamas tinkamumo indeksu (sveiku teigiamu skaičiumi). Kuo skaičius mažesnis, tuo žemė tinkamesnė kelio tiesimui. Kelią reikia nutiesti taip, kad kelio tiesimui naudotų kvadratėlių tinkamumo indeksų suma būtų kuo mažesnė. Kelias gali būti tiesiamas tik į rytus arba į šiaurę (kelias negali būti tiesiamas įstrižai, būti nukreiptas į vakarus ar pietus).

2.1 Rekurentinės lygties sprendimas

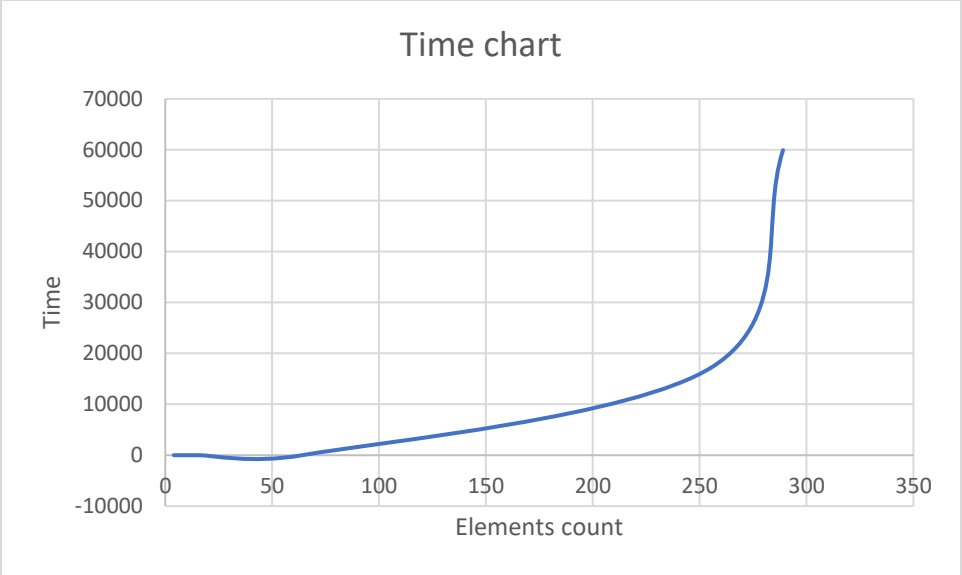
2.1.1 Kodo analizė

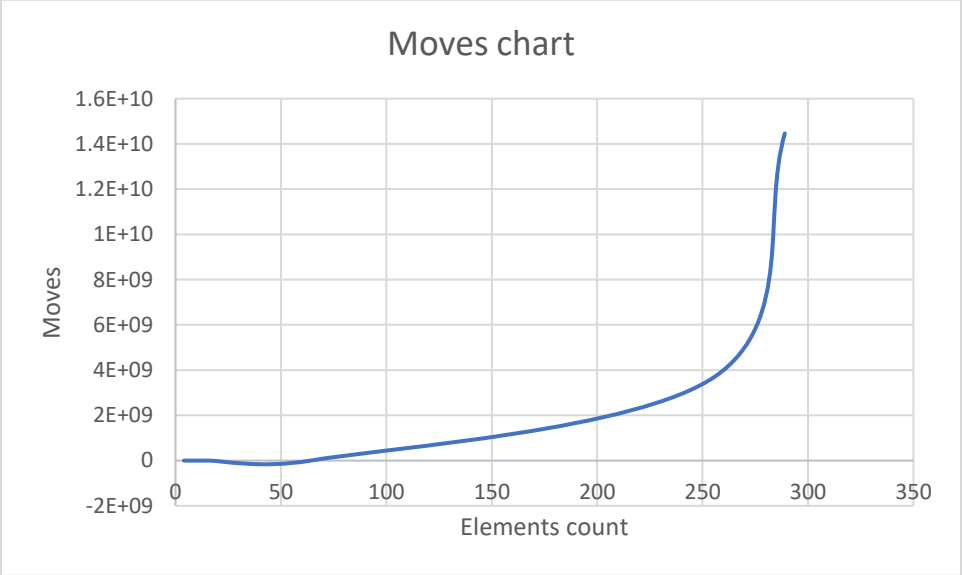
```
public static int getMinCostPath1(int[,] A, int start_row, int start_col, int
end_row, int end_col)
        {
            sk1++;
            if (start_row < end_row || start_col > end_col)
// c1 | 1
            {
                sk1++;
                return 999999;
// c2 | 1
            }
            sk1++;
            if (start_row == end_row && start_col == end_col)
// c3 | 1
            {
                sk1++;
                return A[start_row, start_col];
// c4 | 1
            }
            sk1+=3;
            //Recursive Calls
            int x = getMinCostPath1(A, start_row, start_col + 1, end_row,
end_col);              // T(m, n + 1) | 1
            int y = getMinCostPath1(A, start_row - 1, start_col, end_row,
end_col);              // T(m - 1, n) | 1
            int minimum = Math.Min(x, y);
// c5 | 1
            sk1++;
            return A[start_row, start_col] + minimum;
// c6 | 1

        }
        // T(n) = constanta, jei tenkinama nors viena if salyga
              // T(n) = T(m, n + 1) + T(m - 1, n) + c1 + c3 + c5 + c6, kitais
atvejais
```

2.1.2 Lygties sprendimas



2.1.3 Eksperimentinis tyrimas





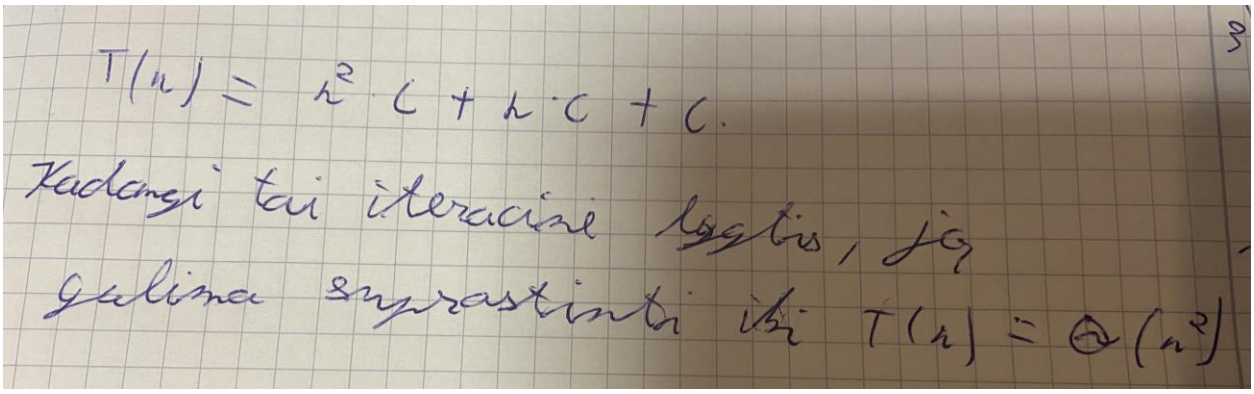
2.2 Iteracinės lygties sprendimas

2.2.1 Kodo analizė

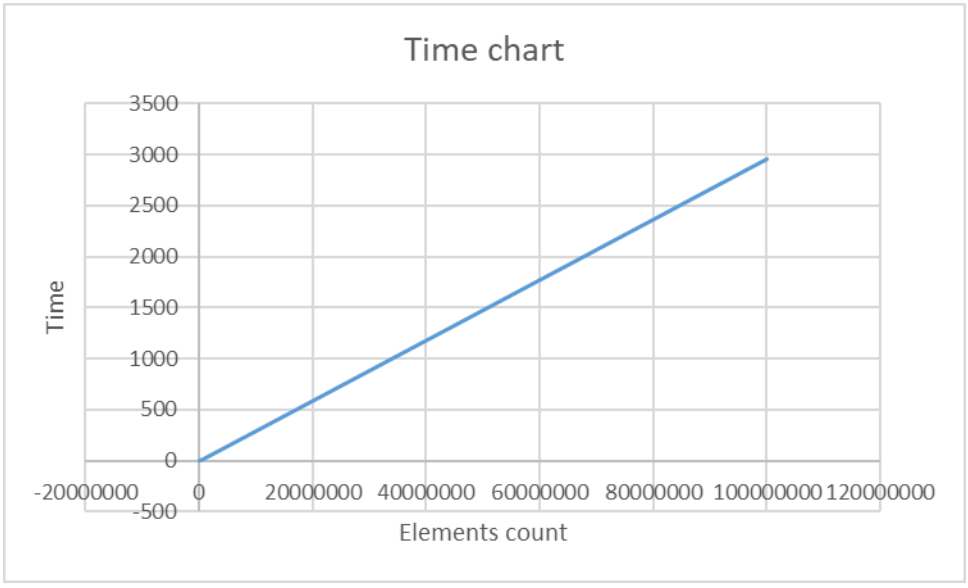
```
public static int minCost(int[,] A, int n)
        {
            sk2++;
            int[,] tc = new int[n + 1,n + 1];
// c1 | 1
            sk2++;
            tc[n,0] =  A[n,0];
// c2 | 1
            sk2++;
```

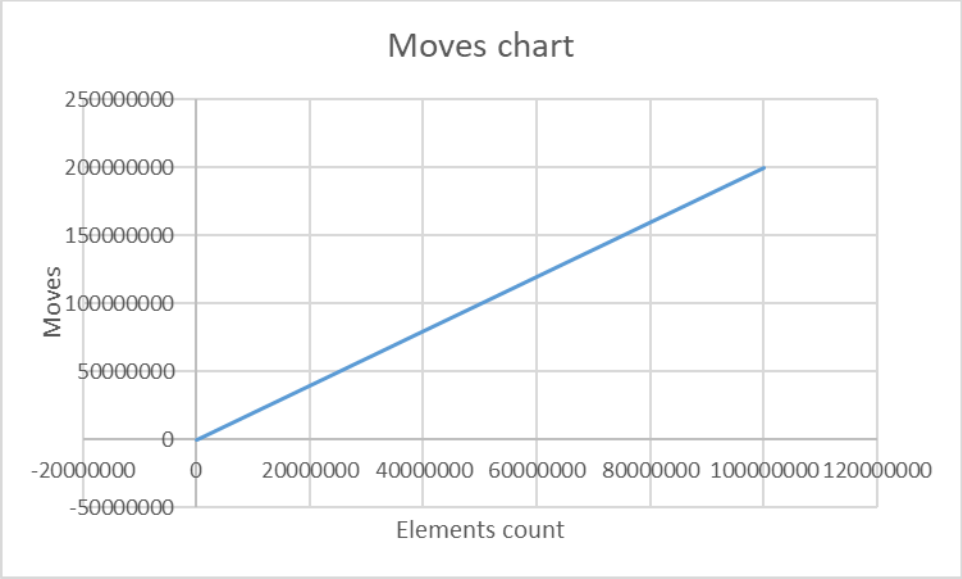
```
            /* Initialize first column of total cost(tc) array
             */
            for (int i = n-1; i >= 0; i--)
// c3 | n + 1
            {
                sk2++;
                tc[i,0] = tc[i + 1,0] + A[i,0];
// c4 | n
                sk2++;
            }
            sk2++;
            /* Initialize first row of tc array */
            for (int j = 1; j <= n; j++)
// c5 | n + 1
            {
                sk2++;
                tc[n,j] = tc[n,j - 1] + A[n,j];
// c6 | n
                sk2++;
            }
            sk2++;
            /* Construct rest of the tc array */
            for (int i = n-1; i >= 0; i--)
// c7 | n + 1
            {
                sk2++;
                for (int j = 1; j <= n; j++)
// c8 | (n + 1) * n
                {
                    sk2++;
                    tc[i,j] = Math.Min(tc[i + 1,j], tc[i,j - 1]) + A[i,j];
// c9 | 3*n*n
                    sk2++;
                }
                sk2++;
            }
            sk2++;
            return tc[0,n];
// c10 | 1
        }
              // T(n) = n^2*(c8 + c9) + n*(c3 + c4 + c5 + c6 + c7 + c8) + c1 +
c2 + c3 + c5 + c7 + c10
```

2.2.2 Lygties sprendimas



2.2.3 Eksperimentinis tyrimas





2.3 Programos kodas

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using System.Runtime.InteropServices;
using System.Text;
using System.Threading.Tasks;

namespace lab2
{
    internal class Class2
    {
        private static long sk1;
        private static long sk2;
        public static void Main(string[] args)
        {
            int n = 0;
            int[,] A = new int[0, 0];

            Console.WriteLine("Pirmoji lygtis");
            n = 2;
            A = new int[2, 2];
            Test1(A, n);
            n = 4;
            A = new int[4, 4];
            Test1(A, n);
            n = 8;
            A = new int[8, 8];
            Test1(A, n);
            n = 16;
            A = new int[16, 16];
            Test1(A, n);
            n = 17;
            A = new int[17, 17];
            Test1(A, n);
```

```
            Console.WriteLine();

            //Console.WriteLine("Antroji lygtis");
            //n = 10;
            //A = new int[10, 10];
            //Test2(A, n);
            //n = 100;
            //A = new int[100, 100];
            //Test2(A, n);
            //n = 1000;
            //A = new int[1000, 1000];
            //Test2(A, n);
            //n = 10000;
            //A = new int[10000, 10000];
            //Test2(A, n);
            //Console.WriteLine();

            //Irodymas, kad algoritmai veikia
            //A = new int[4, 4]
            //{
            //    { 5, 3, 4, 9 },
            //    { 8, 2, 9, 4 },
            //    { 10, 0, 6, 0 },
            //    { 11, 1, 7, 8 }
            //};
            //int rezult1 = getMinCostPath1(A, n - 1, 0, 0, n - 1);
            //Console.WriteLine(rezult1);
            //int rezult2 = minCost(A, n-1);
            //Console.WriteLine(rezult2);


        }

        public static void Test1(int[,] A, int n)
        {
            Stopwatch time = new Stopwatch();
            sk1 = 0;
            time.Start();
            getMinCostPath1(A, n - 1, 0, 0, n - 1);
            time.Stop();
            Console.WriteLine("Duomenu kiekis: {0}", n*n);
            Console.WriteLine("Elapsed time: {0} microsec.",
time.Elapsed.TotalMilliseconds);
            Console.WriteLine("Veiksmu skaicius: {0}", sk1);

        }

        public static void Test2(int[,] A, int n)
        {
            Stopwatch time = new Stopwatch();
            sk2 = 0;
            time.Start();
            minCost(A, n - 1);
            time.Stop();
            Console.WriteLine("Duomenu kiekis: {0}", n * n);
            Console.WriteLine("Elapsed time: {0} microsec.",
time.Elapsed.TotalMilliseconds);
```

```
            Console.WriteLine("Veiksmu skaicius: {0}", sk2);

        }

        public static int getMinCostPath1(int[,] A, int start_row, int
start_col, int end_row, int end_col)
        {
            sk1++;
            if (start_row < end_row || start_col > end_col)
// c1 | 1
            {
                sk1++;
                return 999999;
// c2 | 1
            }
            sk1++;
            if (start_row == end_row && start_col == end_col)
// c3 | 1
            {
                sk1++;
                return A[start_row, start_col];
// c4 | 1
            }
            sk1+=3;
            //Recursive Calls
            int x = getMinCostPath1(A, start_row, start_col + 1,
end_row, end_col);              // T(m, n + 1) | 1
            int y = getMinCostPath1(A, start_row - 1, start_col,
end_row, end_col);              // T(m - 1, n) | 1
            int minimum = Math.Min(x, y);
// c5 | 1
            sk1++;
            return A[start_row, start_col] + minimum;
// c6 | 1

        }
        // T(n) = constanta, jei tenkinama nors viena if salyga
        // T(n) = T(m, n + 1) + T(m - 1, n) + c1 + c3 + c5 + c6, kitais
atvejais

        public static int minCost(int[,] A, int n)
        {
            sk2++;
            int[,] tc = new int[n + 1,n + 1];
// c1 | 1
            sk2++;
            tc[n,0] =  A[n,0];
// c2 | 1
            sk2++;
            /* Initialize first column of total cost(tc) array
             */
            for (int i = n-1; i >= 0; i--)
// c3 | n + 1
            {
                sk2++;
                tc[i,0] = tc[i + 1,0] + A[i,0];
// c4 | n
```

```
                sk2++;
            }
            sk2++;
            /* Initialize first row of tc array */
            for (int j = 1; j <= n; j++)
// c5 | n + 1
            {
                sk2++;
                tc[n,j] = tc[n,j - 1] + A[n,j];
// c6 | n
                sk2++;
            }
            sk2++;
            /* Construct rest of the tc array */
            for (int i = n-1; i >= 0; i--)
// c7 | n + 1
            {
                sk2++;
                for (int j = 1; j <= n; j++)
// c8 | (n + 1) * n
                {
                    sk2++;
                    tc[i,j] = Math.Min(tc[i + 1,j], tc[i,j - 1]) +
A[i,j];                         // c9 | 3*n*n
                    sk2++;
                }
                sk2++;
            }
            sk2++;
            return tc[0,n];
// c10 | 1
        }
        // T(n) = n^2*(c8 + 3*c9) + n*(c3 + c4 + c5 + c6 + c7 + c8) +
c1 + c2 + c3 + c5 + c7 + c10
    }
}
```