KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

Algoritmų sudarymas ir analizė (P170B400)

Laboratorinių darbų ataskaita

Atliko:

IFF-1/1 gr. studentas

Vytenis Kriščiūnas

2023 m. gegužės 2 d.

Priėmė:

Lekt. Vidmantas Rimavičius

KAUNAS 2023

TURINYS

[1. Pirma užduoties dalis 3](#_Toc133941262)

[1.1 Rekurentinės lygties sprendimas 3](#_Toc133941263)

[1.1.1 Sprendimo algoritmas 3](#_Toc133941264)

[1.1.2 Kodo analizė 3](#_Toc133941265)

[1.1.3 Asimptotinis sudėtingumas 4](#_Toc133941266)

[1.1.4 Eksperimentinis tyrimas 5](#_Toc133941267)

[1.2 Dinaminis lygties sprendimas 6](#_Toc133941268)

[1.2.1 Sprendimo algoritmas 6](#_Toc133941269)

[1.2.2 Kodo analizė 7](#_Toc133941270)

[1.2.3 Asimptotinis sudėtingumas 9](#_Toc133941271)

[1.2.4 Eksperimentinis tyrimas 10](#_Toc133941272)

[1.3 Programos kodas 10](#_Toc133941273)

[2. Antra užduoties dalis 15](#_Toc133941274)

[2.1 Pirmoji lygtis 16](#_Toc133941275)

[2.1.1 Sprendimas be lygiagretinimo 16](#_Toc133941276)

[2.1.1.1 Kodo analizė 16](#_Toc133941277)

[2.1.1.2 Lygties sprendimas 16](#_Toc133941278)

[2.1.1.3 Eksperimentinis tyrimas 16](#_Toc133941279)

[2.1.2 Sprendimas su lygiagretumu 17](#_Toc133941280)

[2.1.2.1 Kodo analizė 17](#_Toc133941281)

[2.1.2.2 Lygties sprendimas 18](#_Toc133941282)

[2.1.2.3 Eksperimentinis tyrimas 18](#_Toc133941283)

[2.2 Antroji lygtis 19](#_Toc133941284)

[2.2.1 Sprendimas be lygiagretinimo 19](#_Toc133941285)

[2.2.1.1 Kodo analizė 19](#_Toc133941286)

[2.2.1.2 Lygties sprendimas 20](#_Toc133941287)

[2.2.1.3 Eksperimentinis tyrimas 21](#_Toc133941288)

[2.2.2 Sprendimas su lygiagretumu 22](#_Toc133941289)

[2.2.2.1 Kodo analizė 22](#_Toc133941290)

[2.2.2.2 Lygties sprendimas 23](#_Toc133941291)

[2.2.2.3 Eksperimentinis tyrimas 24](#_Toc133941292)

[2.3 Programos kodas 26](#_Toc133941293)

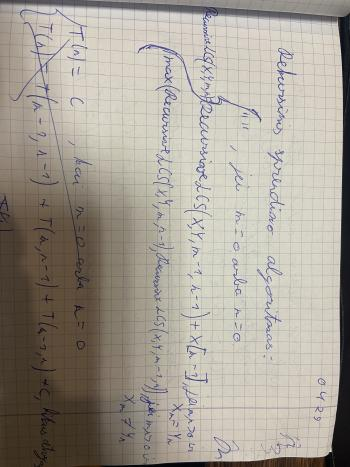
# Pirma užduoties dalis

* 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.

Simbolių sekoje surasti ilgiausią simbolių posekį, kurį būtų galima skaityti iš abiejų galų. Pvz. turime seką S="abaab". Ats.: ilgiausia simbolių seka, kurią galima skaityti iš abiejų galų "baab.

## Rekurentinės lygties sprendimas

### Sprendimo algoritmas



### Kodo analizė

public static String RecursiveLCS(char[] X, char[] Y, int m, int n)

{

sk1++;

if (m == 0 || n == 0) // c1 | 1

{

sk1++;

return ""; // c2 | 1

}

sk1++;

if (X[m - 1] == Y[n - 1]) // c3 | 1

{

sk1++;

return RecursiveLCS(X, Y, m - 1, n - 1) + X[m-1]; // T(m - 1, n - 1) + c4 | 1

}

sk1++;

return maxByLenght(RecursiveLCS(X, Y, m, n - 1), RecursiveLCS(X, Y, m - 1, n)); // T(m, n - 1) + T(m - 1, n) | 1

}

// T(n) = konstanta, jei m = 0 arba n = 0

// T(n) = T(m - 1, n - 1) + c1 + c3 + c4, m > 0 ir n > 0, Xm = Yn

// T(n) = T(m, n - 1) + T(m - 1, n) + c1 + c3, kitais atvejais

private static String maxByLenght(String a, String b)

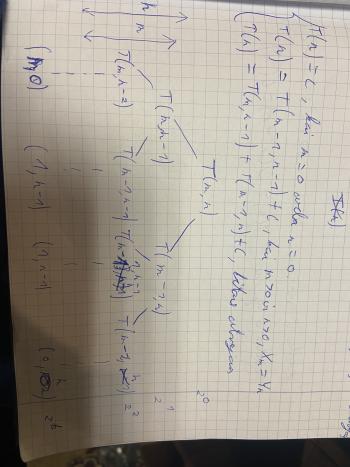
{

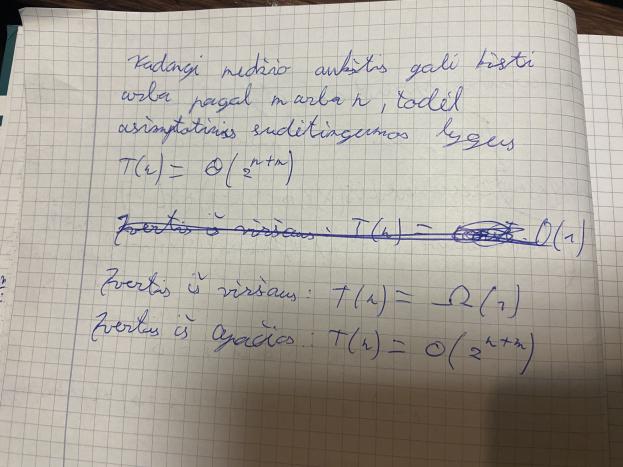
sk1++;

return a.Length > b.Length ? a : b; // c1 | 1

}

### Asimptotinis sudėtingumas





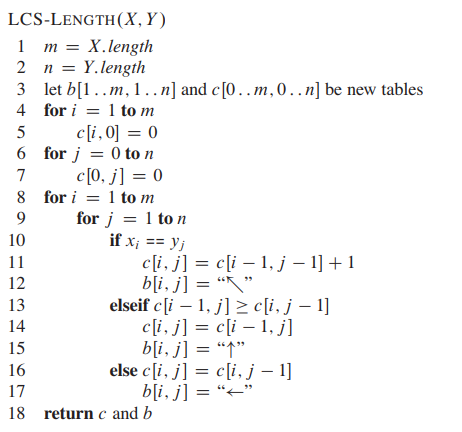
### Eksperimentinis tyrimas

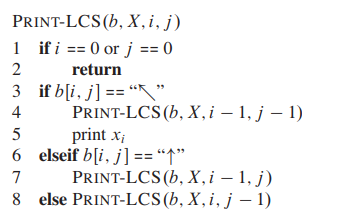
Iš viršaus rezultatai yra gaunami, kai simbolių masyvas yra tuščias, todėl pademonstruoti negalima.

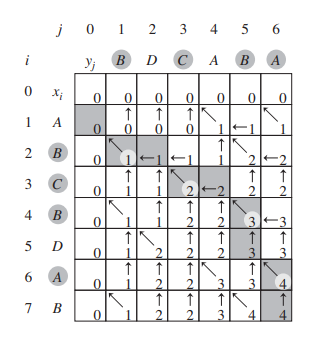
Iš apačios gauto atvejo rezultatai:

## Dinaminis lygties sprendimas

### Sprendimo algoritmas







### Kodo analizė

public static struk DinamicalLCSLenght(char[] X, char[] Y)

{

sk2+=3;

int m = X.Length; // c1 | 1

int n = Y.Length; // c2 | 1

struk s = new struk(m, n); // c3 | 1

sk2++;

for (int i = 1; i < m; i++) // c4 | n + 1

{

sk2++;

s.c[i, 0] = 0; // c5 | n

sk2++;

}

sk2++;

for (int j = 0; j < n; j++) // c6 | n + 1

{

sk2++;

s.c[0,j] = 0; // c7 | n

sk2++;

}

sk2++;

for (int i = 1; i < m; i++) // c8 | n + 1

{

sk2++;

for (int j = 1; j < n; j++) // c9 | n \* (n + 1)

{

sk2+=3;

if (Y[i] == X[j]) // c10 | n\*n

{

sk2+=3;

s.c[i, j] = s.c[i - 1, j - 1] + 1; // c11 | n\*n

s.b[i, j] = '\\'; // c12 | n\*n

}

else if (s.c[i - 1, j] >= s.c[i, j - 1]) // c13 | n\*n

{

sk2+=3;

s.c[i, j] = s.c[i - 1, j]; // c14 | n\*n

s.b[i, j] = '|'; // c15 | n\*n

}

else // c16 | n\*n

{

sk2 += 3;

s.c[i, j] = s.c[i, j - 1]; // c17 | n\*n

s.b[i, j] = '-'; // c18 | n\*n

}

sk2++;

}

sk2++;

}

sk2++;

return s; // c19 | 1

}

// T(n) = c1 + c2 + c3 + c4\*n + c4 + c5\*n + c6\*n + c6 + c7\*n + c8\*n + c8 + c9\*n^2 + c9\*n + c10\*n^2 .. = n^2 + n + C

public static void DinamicalPrintLCS(char[,] b, char[] X, int i, int j)

{

sk2 += 4;

if (i == 0 || j == 0) // c1 | 1

{

sk2++;

return; // c2 | 1

}

if (b[i, j] == '\\') // c3 | 1

{

sk2+=2;

DinamicalPrintLCS(b, X, i - 1, j - 1); // T(i - 1, j - 1) | 1

Console.Write(X[j]); // c4 | 1

}

else if (b[i, j] == '|') // c5 | 1

{

sk2++;

DinamicalPrintLCS(b, X, i - 1, j); // T(i - 1, j) | 1

}

else // c6 | 1

{

sk2++;

DinamicalPrintLCS(b, X, i, j - 1); // T(i, j - 1) | 1

}

}

// T(n) = konstanta, jei i = 0 arba j = 0

// T(n) = T(i - 1, j - 1) + c1 + c3 + c4 , jei b[i, j] == '\\' ir i,j > 0

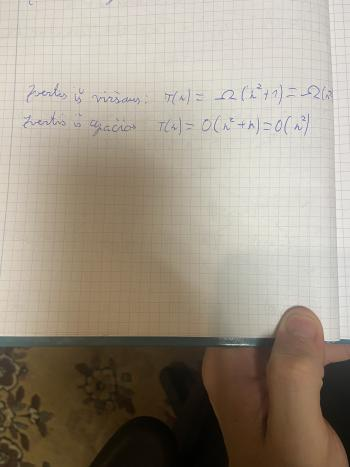
// T(n) = T(i - 1, j) + c1 + c3 + c5, jei b[i, j] == '|' ir i,j > 0

// T(n) = T(i, j - 1) + c1 + c3 + c5 + c6, kitais atvejais

// T(n) = O(1) - atvejis iš viršaus

// Kadangi šis rekursinis metodas veiks tol kol i,j > 0, o jie iš pradžių yra lygūs n, todėl T(n) = O(n + n) - atvejis iš apačios

### Asimptotinis sudėtingumas



### Eksperimentinis tyrimas

Iš viršaus rezultatai yra gaunami, kai simbolių masyvas yra tuščias, todėl pademonstruoti negalima.

Iš apačios gauto atvejo rezultatai:

## Programos kodas

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.Linq;

using System.Security.Cryptography.X509Certificates;

using System.Text;

using System.Text.Json;

using System.Threading.Tasks;

namespace Lab3

{

public class struk

{

public char[,] b;

public int[,] c;

public struk(int XLenght, int YLenght)

{

b = new char[XLenght, YLenght];

c = new int[XLenght, YLenght];

}

}

internal class Class1

{

private static long sk1;

private static long sk2;

public static void Main(string[] args)

{

char[] X = new char[0];

char[] Y = new char[0];

Console.WriteLine("Rekursinis sprendimas");

X = new char[2] { 'a', 'b' };

Y = new char[2] { 'b', 'a' };

Test1(X, Y);

X = new char[4] { 'a', 'a', 'b', 'b' };

Y = new char[4] { 'b', 'b', 'a', 'a' };

Test1(X, Y);

X = new char[8] { 'a', 'b', 'a', 'a', 'b', 'b', 'b', 'a' };

Y = new char[8] { 'a', 'b', 'b', 'b', 'a', 'a', 'b', 'a' };

Test1(X, Y);

X = new char[16] { 'a', 'b', 'b', 'a', 'a', 'b', 'a', 'a', 'b', 'b', 'b', 'a', 'a', 'a', 'b', 'a' };

Y = new char[16] { 'a', 'b', 'a', 'a', 'a', 'b', 'b', 'b', 'a', 'b', 'a', 'a', 'b', 'a', 'b', 'a' };

Test1(X, Y);

X = new char[32] { 'b', 'a', 'b', 'a', 'a', 'b', 'b', 'a', 'a', 'a', 'b', 'b', 'b', 'b', 'a', 'a', 'a', 'a', 'b', 'a', 'a', 'b', 'b', 'a', 'a', 'b', 'b', 'a', 'b', 'a', 'a', 'b' };

Y = new char[32] { 'b', 'a', 'a', 'b', 'a', 'a', 'b', 'b', 'a', 'a', 'b', 'b', 'a', 'a', 'b', 'a', 'b', 'a', 'a', 'a', 'a', 'b', 'b', 'b', 'b', 'a', 'a', 'a', 'a', 'b', 'a', 'b' };

Test1(X, Y);

Console.WriteLine();

//Console.WriteLine("Dinaminis sprendimas");

//X = new char[2] { 'a', 'b' };

//Y = new char[2] { 'b', 'a' };

//Test2(X, Y);

//X = new char[4] { 'a', 'a', 'b', 'b' };

//Y = new char[4] { 'b', 'b', 'a', 'a' };

//Test2(X, Y);

//X = new char[8] { 'a', 'b', 'a', 'a', 'b', 'b', 'b', 'a' };

//Y = new char[8] { 'a', 'b', 'b', 'b', 'a', 'a', 'b', 'a' };

//Test2(X, Y);

//X = new char[16] { 'a', 'b', 'b', 'a', 'a', 'b', 'a', 'a', 'b', 'b', 'b', 'a', 'a', 'a', 'b', 'a' };

//Y = new char[16] { 'a', 'b', 'a', 'a', 'a', 'b', 'b', 'b', 'a', 'b', 'a', 'a', 'b', 'a', 'b', 'a' };

//Test2(X, Y);

//X = new char[32] { 'b', 'a', 'b', 'a', 'a', 'b', 'b', 'a', 'a', 'a', 'b', 'b', 'b', 'b', 'a', 'a', 'a', 'a', 'b', 'a', 'a', 'b', 'b', 'a', 'a', 'b', 'b', 'a', 'b', 'a', 'a', 'b' };

//Y = new char[32] { 'b', 'a', 'a', 'b', 'a', 'a', 'b', 'b', 'a', 'a', 'b', 'b', 'a', 'a', 'b', 'a', 'b', 'a', 'a', 'a', 'a', 'b', 'b', 'b', 'b', 'a', 'a', 'a', 'a', 'b', 'a', 'b' };

//Test2(X, Y);

//Console.WriteLine();

////Testavimui

//char[] X = new char[6];

//char[] Y = new char[6];

//Y[1] = 'a';

//Y[2] = 'b';

//Y[3] = 'a';

//Y[4] = 'a';

//Y[5] = 'b';

//int ii = 1;

//for (int i = Y.Length - 1; i > 0; i--)

//{

// X[ii] = Y[i];

// ii++;

//}

////Rekursinis

//String rec = RecursiveLCS(X, Y, X.Length, Y.Length);

//Console.WriteLine(rec);

////Dinaminis

//struk S = DinamicalLCSLenght(X, Y);

//DinamicalPrintLCS(S.b, X, X.Length - 1, Y.Length - 1);

}

public static void Test1(char[] X, char[] Y)

{

Stopwatch time = new Stopwatch();

sk1 = 0;

time.Start();

RecursiveLCS(X, Y, X.Length, Y.Length);

time.Stop();

Console.WriteLine("Duomenu kiekis: {0}", X.Length \* Y.Length);

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

Console.WriteLine("Veiksmu skaicius: {0}", sk1);

}

public static void Test2(char[] X, char[] Y)

{

Stopwatch time = new Stopwatch();

sk2 = 0;

time.Start();

struk S = DinamicalLCSLenght(X, Y);

DinamicalPrintLCS(S.b, X, X.Length - 1, Y.Length - 1);

time.Stop();

Console.WriteLine("Duomenu kiekis: {0}", X.Length \* Y.Length);

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

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

}

public static String RecursiveLCS(char[] X, char[] Y, int m, int n)

{

sk1++;

if (m == 0 || n == 0) // c1 | 1

{

sk1++;

return ""; // c2 | 1

}

sk1++;

if (X[m - 1] == Y[n - 1]) // c3 | 1

{

sk1++;

return RecursiveLCS(X, Y, m - 1, n - 1) + X[m-1]; // T(m - 1, n - 1) + c4 | 1

}

sk1++;

return maxByLenght(RecursiveLCS(X, Y, m, n - 1), RecursiveLCS(X, Y, m - 1, n)); // T(m, n - 1) + T(m - 1, n) | 1

}

// T(n) = konstanta, jei m = 0 arba n = 0

// T(n) = T(m - 1, n - 1) + c1 + c3 + c4, m > 0 ir n > 0, Xm = Yn

// T(n) = T(m, n - 1) + T(m - 1, n) + c1 + c3, kitais atvejais

private static String maxByLenght(String a, String b)

{

sk1++;

return a.Length > b.Length ? a : b; // c1 | 1

}

public static struk DinamicalLCSLenght(char[] X, char[] Y)

{

sk2+=3;

int m = X.Length; // c1 | 1

int n = Y.Length; // c2 | 1

struk s = new struk(m, n); // c3 | 1

sk2++;

for (int i = 1; i < m; i++) // c4 | n + 1

{

sk2++;

s.c[i, 0] = 0; // c5 | n

sk2++;

}

sk2++;

for (int j = 0; j < n; j++) // c6 | n + 1

{

sk2++;

s.c[0,j] = 0; // c7 | n

sk2++;

}

sk2++;

for (int i = 1; i < m; i++) // c8 | n + 1

{

sk2++;

for (int j = 1; j < n; j++) // c9 | n \* (n + 1)

{

sk2+=3;

if (Y[i] == X[j]) // c10 | n\*n

{

sk2+=3;

s.c[i, j] = s.c[i - 1, j - 1] + 1; // c11 | n\*n

s.b[i, j] = '\\'; // c12 | n\*n

}

else if (s.c[i - 1, j] >= s.c[i, j - 1]) // c13 | n\*n

{

sk2+=3;

s.c[i, j] = s.c[i - 1, j]; // c14 | n\*n

s.b[i, j] = '|'; // c15 | n\*n

}

else // c16 | n\*n

{

sk2 += 3;

s.c[i, j] = s.c[i, j - 1]; // c17 | n\*n

s.b[i, j] = '-'; // c18 | n\*n

}

sk2++;

}

sk2++;

}

sk2++;

return s; // c19 | 1

}

// T(n) = c1 + c2 + c3 + c4\*n + c4 + c5\*n + c6\*n + c6 + c7\*n + c8\*n + c8 + c9\*n^2 + c9\*n + c10\*n^2 .. = n^2 + n + C

public static void DinamicalPrintLCS(char[,] b, char[] X, int i, int j)

{

sk2 += 4;

if (i == 0 || j == 0) // c1 | 1

{

sk2++;

return; // c2 | 1

}

if (b[i, j] == '\\') // c3 | 1

{

sk2+=2;

DinamicalPrintLCS(b, X, i - 1, j - 1); // T(i - 1, j - 1) | 1

Console.Write(X[j]); // c4 | 1

}

else if (b[i, j] == '|') // c5 | 1

{

sk2++;

DinamicalPrintLCS(b, X, i - 1, j); // T(i - 1, j) | 1

}

else // c6 | 1

{

sk2++;

DinamicalPrintLCS(b, X, i, j - 1); // T(i, j - 1) | 1

}

}

// T(n) = konstanta, jei i = 0 arba j = 0

// T(n) = T(i - 1, j - 1) + c1 + c3 + c4 , jei b[i, j] == '\\' ir i,j > 0

// T(n) = T(i - 1, j) + c1 + c3 + c5, jei b[i, j] == '|' ir i,j > 0

// T(n) = T(i, j - 1) + c1 + c3 + c5 + c6, kitais atvejais

// T(n) = O(1) - atvejis iš viršaus

// Kadangi šis rekursinis metodas veiks tol kol i,j > 0, o jie iš pradžių yra lygūs n, todėl T(n) = O(n + n) - atvejis iš apačios

}

}

# Antra užduoties dalis

* Atlikite pateiktų procedūrų lygiagretinimą.
* Įvertinkite teorinį nelygiagretintų ir lygiagretintų procedūrų sudėtingumą.
* 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.
* 2 uždavinys - 3 balai / 3 uždavinys - 3 balai

## Pirmoji lygtis

### Sprendimas be lygiagretinimo

#### Kodo analizė

public static long methodToAnalysis1(int[] arr)

{

sk1++;

long n = arr.Length; // c1 | 1

sk1++;

long k = n; // c2 | 1

sk1++;

if (arr[0] > 0) // c3 | 1

{

sk1++;

for (int i = 0; i < n; i++) // c4 | n + 1

{

sk1++;

for (int j = 0; j < n; j++) // c5 | (n + 1) \* n

{

sk1++;

k -= 2; // c6 | n \* n

sk1++;

}

sk1++;

}

}

sk1++;

return k; // c7 | 1

}

// T(n) = konstanta, jei arr[0] <= 0

// T(n) = n^2 \* (c5 + c6) + n \* (c4 + c5) + c4 + c1 + c2 + c3 , jei arr[0] > 0

#### Lygties sprendimas

Įvertis iš viršaus: T(n) = Ω(1), kai arr[0] <=0

Įvertis iš apačios: T(n) = O(n^2), kitais atvejais

#### Eksperimentinis tyrimas

Iš viršaus gauto atvejo asimptotinis sudėtingumas visada bus lygūs konstantai.

Iš apačios gauto atvejo rezultatai:

### Sprendimas su lygiagretumu

#### Kodo analizė

public static long ParallelmethodToAnalysis1(int[] arr)

{

object monitor = new object(); // c1 | 1

sk1++;

long n = arr.Length; // c2 | 1

sk1++;

long k = n; // c3 | 1

sk1++;

if (arr[0] > 0) // c4 | 1

{

sk1++;

Parallel.For(0, n, i => // c5 | 1

{

sk1++;

Parallel.For(0, n, j => // c6 | 1

{

sk1++;

lock (monitor) k -= 2; // c7 | 1

sk1++;

});

sk1++;

});

}

sk1++;

return k; // c8 | 1

}

// T(n) = konstanta

#### Lygties sprendimas

Kadangi For ciklai veikia lygiagrečiai, teoriškai, metodo asimptotinis sudėtingumas ir iš viršaus ir iš apačios turėtų būti: T(n) = Θ(1).

#### Eksperimentinis tyrimas

Iš viršaus gauto atvejo, kai arr[0] <= 0 asimptotinis sudėtingumas visada bus lygūs konstantai.

Iš apačios gauto atvejo rezultatai neatitinka teorinių skaičiavimų:

## Antroji lygtis

### Sprendimas be lygiagretinimo

#### Kodo analizė

public static long methodToAnalysis2(int n, int[] arr)

{

sk2++;

long k = 0; // c1 | 1

sk2++;

Random randNum = new Random(); // c2 | 1

sk2++;

for (int i = 0; i < n; i++) // c3 | n + 1

{

sk2++;

k += arr[i] + FF3(i, arr); // (c4 + f(n)) | n

sk2++;

}

sk2++;

return k; // c5 | 1

}

// T(n) = n\*(c3 + c4 + konstanta) + c1 + c2 + c3 + c5, kai FF3() metodo rezultatas lygus konstantai

// T(n) = 2^n\*n + n\*(c3 + c4) + c1 + c2 + c3 + c5, kitais atvejais

public static long FF3(int n, int[] arr)

{

sk2++;

if (n > 0 && arr.Length > n && arr[n] > 0) // c1 | 1

{

sk2++;

return FF3(n - 1, arr) + FF3(n - 3, arr); // c2 + f(n - 1) + f(n - 3) | 1

}

sk2++;

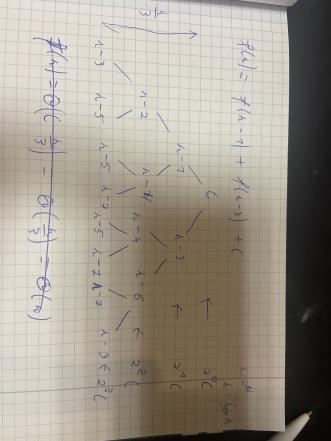
return n; // c3 | 1

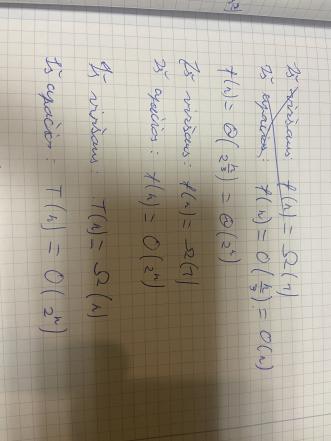
}

// f(n) = konstanta, kai n <=0, arr.Length <= n ir arr[n] <=0

// f(n) = f(n - 1) + f(n - 3) + c1 + c2 + c3, kitais atvejais

#### Lygties sprendimas





#### Eksperimentinis tyrimas

Iš viršaus gauto atvejo rezultatai:

Iš apačios gauto atvejo rezultatai:

### Sprendimas su lygiagretumu

#### Kodo analizė

public static long ParallelmethodToAnalysis2(int n, int[] arr)

{

object monitor = new object(); // c1 | 1

sk2++;

long k = 0; // c2 | 1

sk2++;

Random randNum = new Random(); // c3 | 1

sk2++;

Parallel.For(0, n, i => // c4 | 1

{

sk2++;

lock (monitor) k += arr[i] + ParallelFF3(i, arr); // (c5 + f(n)) | 1

sk2++;

});

sk2++;

return k; // c6 | 1

}

// T(n) = konstanta + c1 + c2 + c3 + c4 + c5 + c6, kai FF3() metodo rezultatas lygus konstantai

// T(n) = 2^n + c1 + c2 + c3 + c4 + c5 + c6, kitais atvejais

public static long ParallelFF3(int n, int[] arr)

{

sk2++;

if (n > 0 && arr.Length > n && arr[n] > 0) // c1 | 1

{

sk2+=5;

Task<long> task = Task<long>.Factory.StartNew((local\_n) => { return ParallelFF3((int)local\_n - 1, arr); }, n); // f(n - 1) | 1

long y = ParallelFF3(n - 3, arr); // f(n - 3) | 1

Task.WaitAll(task); // c2 | 1

long x = task.Result; // c3 | 1

return x + y; // c4 | 1

}

sk2++;

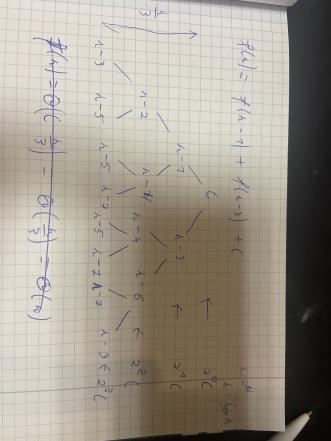
return n; // c5 | 1

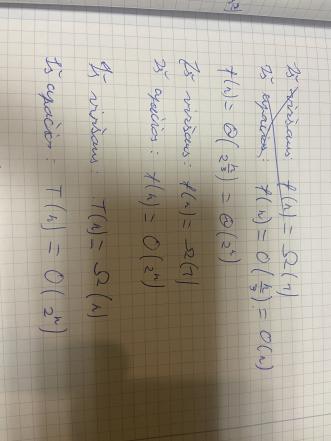
}

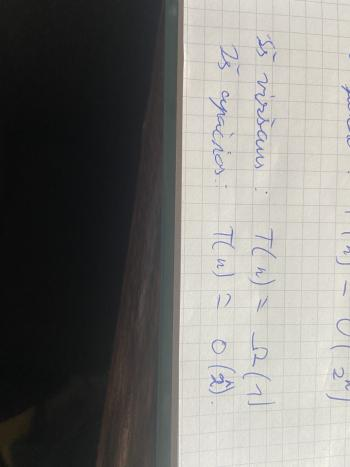
// f(n) = konstanta, kai n <=0, arr.Length <= n ir arr[n] <=0

// f(n) = f(n - 1) + f(n - 3) + c1 + c2 + c3 + c4 + c5, kitais atvejais

#### Lygties sprendimas







#### Eksperimentinis tyrimas

Iš viršaus gauto atvejo rezultatai neatitinka teorinių skaičiavimų:

Iš apačios gauto atvejo rezultatai:

## Programos kodas

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace lab3\_antra\_dalis

{

internal class Class1

{

private static int sk1;

private static int sk2;

public static void Main(string[] args)

{

var random = new Random();

//Console.WriteLine("Pirmoji lygtis: ");

//Console.WriteLine("Be lygiagretumo: ");

//var arr = Enumerable.Range(0, 16).Select(i => random.Next(0, 16)).ToArray();

//Test1Normal(arr);

//arr = Enumerable.Range(0, 32).Select(i => random.Next(0, 32)).ToArray();

//Test1Normal(arr);

//arr = Enumerable.Range(0, 64).Select(i => random.Next(0, 64)).ToArray();

//Test1Normal(arr);

//arr = Enumerable.Range(0, 128).Select(i => random.Next(0, 128)).ToArray();

//Test1Normal(arr);

//arr = Enumerable.Range(0, 256).Select(i => random.Next(0, 256)).ToArray();

//Test1Normal(arr);

//Console.WriteLine();

//Console.WriteLine("Su lygiagretumu: ");

//arr = Enumerable.Range(0, 16).Select(i => random.Next(0, 16)).ToArray();

//Test1Parallel(arr);

//arr = Enumerable.Range(0, 32).Select(i => random.Next(0, 32)).ToArray();

//Test1Parallel(arr);

//arr = Enumerable.Range(0, 64).Select(i => random.Next(0, 64)).ToArray();

//Test1Parallel(arr);

//arr = Enumerable.Range(0, 128).Select(i => random.Next(0, 128)).ToArray();

//Test1Parallel(arr);

//arr = Enumerable.Range(0, 256).Select(i => random.Next(0, 256)).ToArray();

//Test1Parallel(arr);

Console.WriteLine("Antroji lygtis: ");

Console.WriteLine("Be lygiagretumo: ");

int n = 8;

var arr = Enumerable.Range(0, 16).Select(i => random.Next(0, 16)).ToArray();

Test2Normal(n, arr);

n = 16;

arr = Enumerable.Range(0, 32).Select(i => random.Next(0, 32)).ToArray();

Test2Normal(n, arr);

n = 32;

arr = Enumerable.Range(0, 64).Select(i => random.Next(0, 64)).ToArray();

Test2Normal(n, arr);

n = 40;

arr = Enumerable.Range(0, 128).Select(i => random.Next(0, 128)).ToArray();

Test2Normal(n, arr);

Console.WriteLine();

Console.WriteLine("Su lygiagretumu: ");

n = 8;

arr = Enumerable.Range(0, 16).Select(i => random.Next(0, 16)).ToArray();

Test2Parallel(n, arr);

n = 16;

arr = Enumerable.Range(0, 32).Select(i => random.Next(0, 32)).ToArray();

Test2Parallel(n, arr);

n = 32;

arr = Enumerable.Range(0, 64).Select(i => random.Next(0, 64)).ToArray();

Test2Parallel(n, arr);

n = 40;

arr = Enumerable.Range(0, 128).Select(i => random.Next(0, 128)).ToArray();

Test2Parallel(n, arr);

}

public static void Test1Normal(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 Test1Parallel(int[] arr)

{

Stopwatch time = new Stopwatch();

sk1 = 0;

time.Start();

ParallelmethodToAnalysis1(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 Test2Normal(int n, int[] arr)

{

Stopwatch time = new Stopwatch();

sk2 = 0;

time.Start();

methodToAnalysis2(n, arr);

time.Stop();

Console.WriteLine("Duomenu kiekis: {0}", arr.Length);

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

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

}

public static void Test2Parallel(int n, int[] arr)

{

Stopwatch time = new Stopwatch();

sk2 = 0;

time.Start();

ParallelmethodToAnalysis2(n, arr);

time.Stop();

Console.WriteLine("Duomenu kiekis: {0}", arr.Length);

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++;

if (arr[0] > 0) // c3 | 1

{

sk1++;

for (int i = 0; i < n; i++) // c4 | n + 1

{

sk1++;

for (int j = 0; j < n; j++) // c5 | (n + 1) \* n

{

sk1++;

k -= 2; // c6 | n \* n

sk1++;

}

sk1++;

}

}

sk1++;

return k; // c7 | 1

}

// T(n) = konstanta, jei arr[0] <= 0

// T(n) = n^2 \* (c5 + c6) + n \* (c4 + c5) + c4 + c1 + c2 + c3 , jei arr[0] > 0

public static long ParallelmethodToAnalysis1(int[] arr)

{

object monitor = new object(); // c1 | 1

sk1++;

long n = arr.Length; // c2 | 1

sk1++;

long k = n; // c3 | 1

sk1++;

if (arr[0] > 0) // c4 | 1

{

sk1++;

Parallel.For(0, n, i => // c5 | 1

{

sk1++;

Parallel.For(0, n, j => // c6 | 1

{

sk1++;

lock (monitor) k -= 2; // c7 | 1

sk1++;

});

sk1++;

});

}

sk1++;

return k; // c8 | 1

}

// T(n) = konstanta

public static long methodToAnalysis2(int n, int[] arr)

{

sk2++;

long k = 0; // c1 | 1

sk2++;

Random randNum = new Random(); // c2 | 1

sk2++;

for (int i = 0; i < n; i++) // c3 | n + 1

{

sk2++;

k += arr[i] + FF3(i, arr); // (c4 + f(n)) | n

sk2++;

}

sk2++;

return k; // c5 | 1

}

// T(n) = n\*(c3 + c4 + konstanta) + c1 + c2 + c3 + c5, kai FF3() metodo rezultatas lygus konstantai

// T(n) = 2^n\*n + n\*(c3 + c4) + c1 + c2 + c3 + c5, kitais atvejais

public static long FF3(int n, int[] arr)

{

sk2++;

if (n > 0 && arr.Length > n && arr[n] > 0) // c1 | 1

{

sk2++;

return FF3(n - 1, arr) + FF3(n - 3, arr); // c2 + f(n - 1) + f(n - 3) | 1

}

sk2++;

return n; // c3 | 1

}

// f(n) = konstanta, kai n <=0, arr.Length <= n ir arr[n] <=0

// f(n) = f(n - 1) + f(n - 3) + c1 + c2 + c3, kitais atvejais

public static long ParallelmethodToAnalysis2(int n, int[] arr)

{

object monitor = new object(); // c1 | 1

sk2++;

long k = 0; // c2 | 1

sk2++;

Random randNum = new Random(); // c3 | 1

sk2++;

Parallel.For(0, n, i => // c4 | 1

{

sk2++;

lock (monitor) k += arr[i] + ParallelFF3(i, arr); // (c5 + f(n)) | 1

sk2++;

});

sk2++;

return k; // c6 | 1

}

// T(n) = konstanta + c1 + c2 + c3 + c4 + c5 + c6, kai FF3() metodo rezultatas lygus konstantai

// T(n) = 2^n + c1 + c2 + c3 + c4 + c5 + c6, kitais atvejais

public static long ParallelFF3(int n, int[] arr)

{

sk2++;

if (n > 0 && arr.Length > n && arr[n] > 0) // c1 | 1

{

sk2+=5;

Task<long> task = Task<long>.Factory.StartNew((local\_n) => { return ParallelFF3((int)local\_n - 1, arr); }, n); // f(n - 1) | 1

long y = ParallelFF3(n - 3, arr); // f(n - 3) | 1

Task.WaitAll(task); // c2 | 1

long x = task.Result; // c3 | 1

return x + y; // c4 | 1

}

sk2++;

return n; // c5 | 1

}

// f(n) = konstanta, kai n <=0, arr.Length <= n ir arr[n] <=0

// f(n) = f(n - 1) + f(n - 3) + c1 + c2 + c3 + c4 + c5, kitais atvejais

}

}