

**KAUNO TECHNOLOGIJOS**

**UNIVERSITETAS**

**INFORMATIKOS FAKULTETAS**

**ALGORITMŲ SUDARYMAS IR ANALIZĖ**

**P170B400**

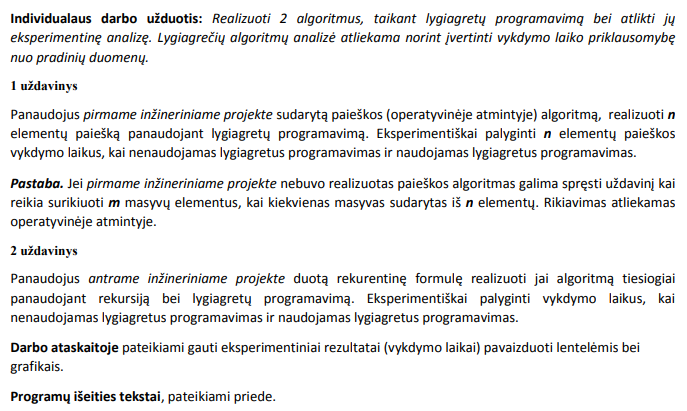
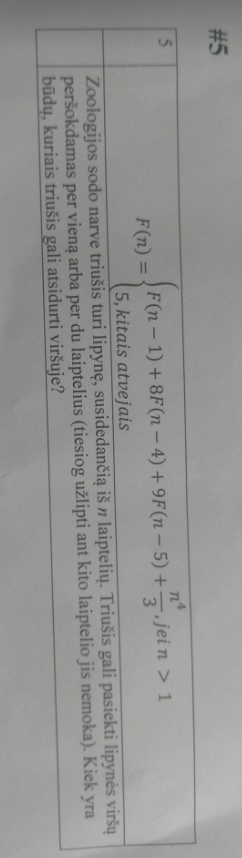
**Individualus darbas**

**Atliko:**

Tadas Laurinaitis IFF-6/8

**Priėmė:**

doc. Mikuckienė Irena



1 uždavinys:

Kai masyvų skaičius m = 5.

|  |  |  |
| --- | --- | --- |
| Elementu skaicius masyve | Rekursijos laiko trukmė (ms) | Lygiagretaus laiko trukmė(ms) |
| 1000 | 13 | 10 |
| 2500 | 72 | 33 |
| 5000 | 243 | 113 |
| 10000 | 891 | 464 |
| 20000 | 3602 | 1729 |
| 30000 | 7978 | 3875 |

2 uždavinys:

|  |  |  |
| --- | --- | --- |
| n reikšmė | Rekursinio sprendimo trukmė | Lygiagretaus sprendimo trukmė |
| 30 | 11 | 13 |
| 35 | 83 | 57 |
| 40 | 585 | 394 |
| 45 | 4375 | 2857 |
| 50 | 31920 | 21436 |

**Išvados:** Uždavinių sprendimas lygiagretaus programavimo būdu yra geras būdas paspartinti uždavinio sprendimo trukmę, kai tvarkomas labai didelis duomenų kiekis, lyginant su kitais sprendimo būdais.

Kai tvarkomas mažas duomenų kiekis, lygiagretaus programavimo naudojimas dažnu atveju gaunasi net gi lėtesnis negu kiti sprendimo būdai.Priedai:

**Pilnas programos kodas:**

using System;

using System.Diagnostics;

using System.Threading;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConsoleApp1

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Spauskite 1 - 1 uzduotis, 2 - 2 uzduotis ");

int n = Convert.ToInt32(Console.ReadLine());

switch (n)

{

case 1:

Testavimas1();

break;

case 2:

Testavimas2();

break;

default:

break;

}

}

//1 uzduotis

public static int[][] MakeArrayOfArrays(int m, int n)

{

int[][] arrayOfArrays = new int[m][];

Random rnd = new Random();

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

{

arrayOfArrays[i] = new int[n];

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

{

int number = rnd.Next(100);

arrayOfArrays[i][j] = number;

//Console.WriteLine(arrayOfArrays[i][j]);

}

//Console.WriteLine("---------------------------");

}

return arrayOfArrays;

}

public static void SortArray(int[][] arrayOfArrays, int m, int n)

{

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

{

for (int j = 0; j < n - 1; j++)

{

for (int k = j + 1; k < n; k++)

{

if (arrayOfArrays[i][j] > arrayOfArrays[i][k])

{

int temp = arrayOfArrays[i][j];

arrayOfArrays[i][j] = arrayOfArrays[i][k];

arrayOfArrays[i][k] = temp;

}

}

}

}

}

public static int LygiagretusRikiavimas(int[][] arrayOfArrays, int m, int n)

{

int completed = 0;

Task<int>[] tasks = new Task<int>[m];

for (var j = 0; j < tasks.Length; j++)

tasks[j] = Task<int>.Factory.StartNew(

(object p) =>

{

for (int i = 0; i < n - 1; i++)

{

for (int q = i + 1; q < n; q++)

{

if (arrayOfArrays[completed][i] > arrayOfArrays[completed][q])

{

int temp = arrayOfArrays[completed][i];

arrayOfArrays[completed][i] = arrayOfArrays[completed][q];

arrayOfArrays[completed][q] = temp;

}

}

}

completed++;

return completed;

}, j);

int hh = 0;

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

{

hh += tasks[i].Result;

}

return hh;

}

public static void Testavimas1()

{

int m = 5;

int n = 10000;

int[][] arrayOfArrays1 = MakeArrayOfArrays(m, n);

int[][] arrayOfArrays2 = arrayOfArrays1;

Stopwatch watch = new Stopwatch();

watch.Start();

SortArray(arrayOfArrays1, m, n);

watch.Stop();

Console.WriteLine("Paprastas sortinimas uztruko: " +watch.ElapsedMilliseconds +" milisekundziu.");

watch.Reset();

watch.Start();

int lr = LygiagretusRikiavimas(arrayOfArrays2, m, n);

watch.Stop();

Console.WriteLine("Lygiagretus sortinimas uztruko: " + watch.ElapsedMilliseconds + " milisekundziu.");

}

//2 uzduotis

class CustomData

{

public double TNum;

public double TResult;

}

public static void Testavimas2()

{

int n = 50;

var stopWatch = new Stopwatch();

stopWatch.Start();

double fibnum = F1(n);

stopWatch.Stop();

Console.WriteLine("Time in milliseconds for sequential F(n): {0,6:N0} ",

stopWatch.ElapsedMilliseconds);

Console.WriteLine("F( {0,4:N0} ) = {1,9:N0}", n, fibnum);

stopWatch.Reset();

stopWatch.Start();

fibnum = F2(n);

stopWatch.Stop();

Console.WriteLine("Time in milliseconds for parallel F(n): {0,6:N0} ",

stopWatch.ElapsedMilliseconds);

Console.WriteLine("F( {0,4:N0} ) = {1,9:N0}", n, fibnum);

}

static double F1(int n)

{

double sum = 0;

//Console.WriteLine("n pradzia: " + n + ", o suma: " + sum);

if (n > 1)

{

sum += F1(n - 1);

sum += 8 \* F1(n - 4);

sum += 9 \* F1(n - 5);

sum += (Math.Pow(n, 4)) / 3;

return sum;

}

else

{

return 5;

}

}

static double F2(int n)

{

double fibnum = 0;

if (n < 6) fibnum = F1(n);

else

{

//fibnum = F1(n - 3) + 3 \* F1(n - 4) + 3 \* F1(n - 5) + F1(n - 6);

int countCPU = 4;

Task[] tasks = new Task[countCPU];

for (var j = 0; j < countCPU; j++)

tasks[j] = Task.Factory.StartNew(

(Object p) =>

{

var data = p as CustomData; if (data == null) return;

if (data.TNum == 0)

data.TResult = F1(n - 1);

else if (data.TNum == 1)

data.TResult = F1(n - 4);

else if (data.TNum == 2)

data.TResult = F1(n - 5);

//data.TResult = F1(n - data.TNum - 3);

},

new CustomData() { TNum = j });

Task.WaitAll(tasks);

fibnum = (tasks[0].AsyncState as CustomData).TResult

+ 8 \* (tasks[1].AsyncState as CustomData).TResult

+ 9 \* (tasks[2].AsyncState as CustomData).TResult

+ ((Math.Pow(n, 4)) / 3);

}

return fibnum;

}

}

}