

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

FUNDAMENTINIŲ MOKSLŲ FAKULTETAS

INFORMACINIŲ SISTEMŲ KATEDRA

ALGORITMAI IR DUOMENŲ STRUKTŪROS

Savarankiškas darbas nr. 1

Atliko: Prif 19/3 grupės stud.

Dovydas Tomašiūnas, Paulius Lėveris, Robert Paškevič

Priėmė: Dėst. Nijolė Čeikienė

Vilnius, 2020

Turinys

Darbo užduotis..................................................................................................................................... 3

Užduoties atlikimo algoritmas............................................................................................................. 3

Algoritmo schema ............................................................................................................................... 4

Programos kodas ................................................................................................................................. 5

Programos testavimas.......................................................................................................................... 10

Darbo užduotis

Duota kvadratinė matrica n x n, užpildyta atsitiktiniais sveikaisiais skaičiais. Reikia nustatyti

kiek neigiamų elementų yra po šalutine įstrižaine, rasti jų sumą ir vidurkį (šalutine įstrižainė – tai įstrižainė, kertanti kvadratinės matricos elementus nuo viršutinio dešiniojo kampo iki apatinio kairiojo). Sudarykite užduoties sprendimo algoritmą, nubraižykite jo schemą ir parašykite programą, realizuojančią sukurtą algoritmą.

Užduoties atlikimo algoritmas

Naudojant ciklą cikle pereiti per tuos n x n = n^2 matricos elementus, kurie yra po šalutine įstrižaine. Patikrinus, kad atrinkti elementai tikrai tenkina tokią salygą, patikrinti ir atrinkti tik neigiamus elementus: Pvz.:

1 2 3 4

3 5 5 6

-1 -2 3 7

-1 -2 -10 7

Turime 4 x 4 dimensijų matricą. Mums reikia atlikti veiksmus su neigiamais elementais, esančiais po šalutine įstrižaine.

i/j

I = 1 j = 0

Item = matrix[1][4-1-j]; Item -> 6

I -> 2; j -> 0; Item -> 7

I -> 2; j -> 1; Item -> 3

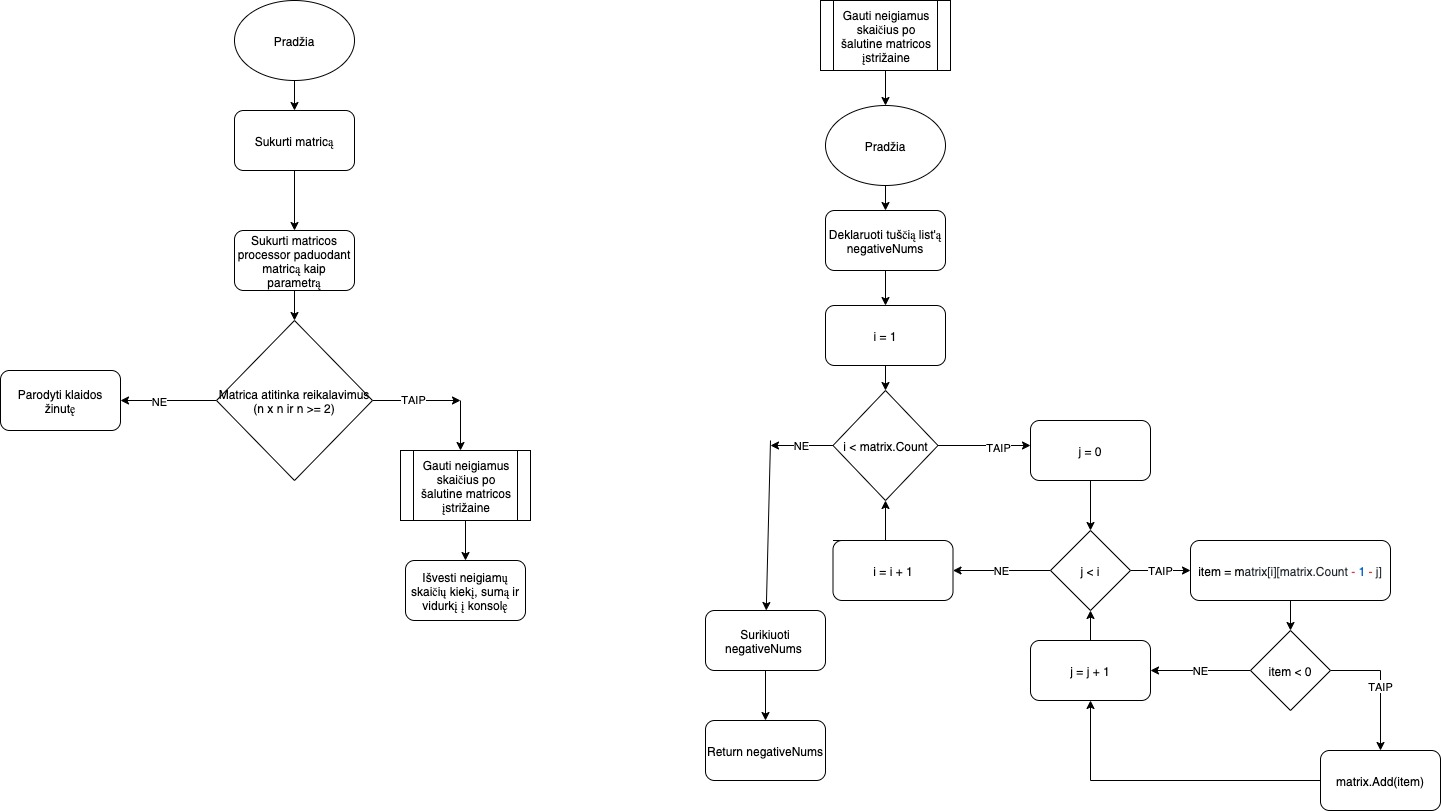
I -> 3; j -> 0; Item -> 7

I -> 3; j -> 1; Item -> -10 (NegativeNums++);

I -> 3; j -> 2; Item -> -2 (NegativeNums++).

Taigi kaip matome, einame pro visus šalutinės įstrižainės elementus (matrika[i][4-1-j]); Aptikę, kad skaičius neigiamas (Item < 0), padidiname neigiamu elementu skaitliuka. Vėliau žinodami, kiek iš viso yra neigiamų skaičių, nesunkiai galėsime apskaičiuoti jų sumą ir vidurkį.

Algoritmo schema



Programos kodas

Failas: Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

namespace Matrix

{

class Program

{

static void Main(string[] args)

{

var matrix = new List<List<int>>();

var l1 = new List<int> { 1, 2, 3, 4};

var l2 = new List<int> {3, 5, 5, 6};

var l3 = new List<int> { -1, -2, 3, 7 };

var l4 = new List<int> { -1, -2, -10, 7 };

matrix.Add(l1);

matrix.Add(l2);

matrix.Add(l3); matrix.Add(l4);

try

{

var matrixProcessor = new MatrixProcessor(matrix);

var negativeNums = matrixProcessor.GetNegativeNumbersUnderDiagonal();

OutputResult(negativeNums);

}

catch(Exception e)

{

Console.WriteLine(e.Message);

}

}

public static List<int> GetNegativeNumbers(List<List<int>> matrix)

{

var negativeNums = new List<int>();

for (int i = 1; i < matrix.Count; i++)

{

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

{

var item = matrix[i][matrix.Count - 1 - j];

if (item < 0)

{

negativeNums.Add(item);

}

}

}

return negativeNums;

}

private static void OutputResult(List<int> negativeNums)

{

Console.Write("Negative numbers under matrix diagonal: ");

foreach (var num in negativeNums)

{

Console.Write($"{num} ");

}

Console.WriteLine($"\nSum: {negativeNums.Sum()}");

Console.WriteLine($"Average: {negativeNums.Average()}");

}

}

}

Failas: MatrixProcessor.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace Matrix

{

public class MatrixProcessor

{

private List<List<int>> Matrix { get; set; }

public MatrixProcessor(List<List<int>> matrix)

{

if (IsValidMatrix(matrix))

{

Matrix = matrix;

}

else

{

throw new ArgumentException("Invalid matrix (has to be n x n and n > 2)");

}

}

public List<int> GetNegativeNumbersUnderDiagonal()

{

var negativeNums = new List<int>();

for (int i = 1; i < Matrix.Count; i++)

{

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

{

var item = Matrix[i][Matrix.Count - 1 - j];

Console.WriteLine($"i: {i}; j: {j}; item: {item}");

if (item < 0)

{

negativeNums.Add(item);

}

}

}

return negativeNums;

}

private bool IsValidMatrix(List<List<int>> matrix)

{

var matrixCount = matrix.Count;

if (matrixCount <= 2)

{

return false;

}

foreach (var row in matrix)

{

if (row.Count != matrix.Count)

{

return false;

}

}

return true;

}

}

}

Programos testavimas

Programos veikimą tikrina UnitTest’ai, faile MatrixProcessor.cs aprašytas metodas isValidMatrix (matrix) tikrina,, ar programai paduota teisinga matrica. Jei taip, tuomet programa tvarkingai atlieka savo darbą, priešingu atveju informuoja vartotoją apie tai, jog neteisingai įvesta matrica ir baigia darbą.