Algoritmica grafelor laboratorul 4
Tema: multimea intern stabila.

# Enunt

Sa se conceapa un algoritm care primeste ca parametru de intrare o lista reprezentand toate varfurile unui graf, iar pentru fiecare element din lista se memoreaza o lista de numere naturale reprezentand lista succesorilor acelui varf. Algoritmul va returna o multime de multimi reprezentand toate multimile interior stabile ale grafului dat ca intrare, multimile returnate contin doar varfuri.

# Dezvoltarea algoritmului

Pentru rezolvarea problemei va fi necesara impartrea in subprobleme. Este nevoie de o prima functie care primeste ca si parametru de intrare o multime de numere naturale si returneaza toate submultimile acelei multimi. O functie care verifica daca o submultime de varfuri ale unui graf dat este interor stabila, graful dat este reprezentat sub forma listei succesorilor (detaliere in enunt).

In final algoritmul propriu zis care mai intai determina toate submultimile multimii varfurilor dupa care pentru fiecare submultime determina daca aceasta este interior stabila. Toate multimile care sunt interior stabile sunt returnate de algoritm.

# Descrierea algoritmului

```
functie DeterminaMultimileInteriorStabile(listaSuccesorilor)
    multimiInteriorStabile ← CreazaLista()

    pentru submultime ∈ DeterminaSubmultimi(listaSuccesorilor.Varfuri) executa
        daca EsteInteriorStabila(submultime, listaSuccesorilor) atunci
            multimiInteriorStabile.Adauga(submultime)
        sfarsit daca
    sfarsit pentru

     DeterminaMultimileInteriorStabile ← multimiInteriorStabile
sfarsit functie

functie DeterminaSubmultimi(multime)
    submultimi ← CreazaMultime()
    multimiRamase ← CreazaMultime()
    multimiRamase.Adauga(multime)

    repeta
        multimeRamasa ← multimiRamase.EliminaPrimul()
        submultimi.Adauga(multimeRamasa)
        pentru element ∈ multimeRamasa
            multimiRamase.Adauga(multimeRamasa - {element})
        sfarsit pentru
    panacand multimiRamase.Length = 0

    DeterminaSubmultimi ← submultimi
sfarsit functie
```

```
functie EsteInteriorStabila(multimeVarfuri, listaSuccesorilor)
    i ← 0

    cattimp (i < listaSuccesorilor.Length
             si (i ∉ multimeVarfuri
                 sau listaSuccesorilor[i] ∩ multimeVarfuri = ∅)) executa
        i ← i + 1
    sfarsit cattimp

    EsteInteriorStabila ← (i = listaSuccesorilor.Length)
sfarsit functie
```

## Demostrarea corectitudinii

Generarea submultimilor.

Idea de baza pentru generarea submultimilor este aceea de a obtine n submultimi dintr-o multime data unde din fiecare submultime lipseste un alt element din multimea data, n este cardinalul multimii date. Acelasi procedeu se aplica si pe submultimile obtinute pana cand nu se mai pot elimina elemente (multimea din care trebuie sa se elimine este vida). Astfel se obtin toate submultimile unei multimi date.

Determinarea daca o submultime este interior stabila:

Avand o submultime de varfuri si lista succesorilor se verifica toate varfurile din submultime ca acestea sa nu aiba succesori in aceasi submultime (definitie curs).

Determinarea tuturor multimilor interior stabile:

Avand lista succesorilor se determina toate submultimile varfurilor dupa care intr-o colectie se plaseaza toate submultimile care indeplinesc conditia de multime interior stabila. Multimea vida este inclusa deoarece orice operatie de intersectie cu multimea vida are ca rezultat multimea vida.

## Cod sursa (C#)

```
using System;
using System.Collections.Generic;
using System.Linq;

namespace AlgoritmicaGrafelor.Laborator4.Multimi
{
    public static class MultimiInteriorStabile
    {
        static public IreadOnlyList<IReadOnlyList<int>>
            Toate(IReadOnlyList<IReadOnlyList<int>> listaSuccesorilor)
        {
            if (listaSuccesorilor != null)
                return _DeterminaSubmultimi(_Varfuri(listaSuccesorilor)).Where(submultime
                            => _EsteInteriorStabila(submultime, listaSuccesorilor)).ToList();
            else
                throw new ArgumentNullException("listaSuccesorilor");
        }
```

```
static private IreadOnlyList<IReadOnlyList<int>>
    _DeterminaSubmultimi(IReadOnlyCollection<int> multime)
{
    if (multime != null)
    {
        IEqualityComparer<IEnumerable<int>> listElementsEqualityComparer
            = new ListElementsEqualityComparer<int>();
        HashSet<IReadOnlyList<int>> submultimi
            = new HashSet<IReadOnlyList<int>>(listElementsEqualityComparer);
        HashSet<IReadOnlyList<int>> multimiRamase
            = new HashSet<IReadOnlyList<int>>(listElementsEqualityComparer)
              {
                  multime.ToList()
              };

        do
        {
            IReadOnlyList<int> multimeRamasa = multimiRamase.First();
            multimiRamase.Remove(multimeRamasa);
            submultimi.Add(multimeRamasa);

            foreach (int element in multimeRamasa)
                multimiRamase.Add(multimeRamasa.Except(new[] { element }).ToList());
        } while (multimiRamase.Count > 0);

        return submultimi.ToList();
    }
    else
        throw new ArgumentNullException("multime");
}

static private IreadOnlyList<int>
    _Varfuri(IReadOnlyList<IReadOnlyList<int>> listaSuccesorilor)
{
    List<int> varfuri = new List<int>();

    for (int varf = 0; varf < listaSuccesorilor.Count; varf++)
        varfuri.Add(varf);

    return varfuri;
}

static private bool
    _EsteInteriorStabila(IReadOnlyList<int> multimeVarfuri,
                         IReadOnlyList<IReadOnlyList<int>> listaSuccesorilor)
{
    int varf = 0;

    while (varf < listaSuccesorilor.Count
           && (!multimeVarfuri.Contains(varf)
               || listaSuccesorilor[varf].Intersect(multimeVarfuri).Count() == 0))
        varf++;

    return (varf == listaSuccesorilor.Count);
}
```

```
        private sealed class ListElementsEqualityComparer<T>
            : IEqualityComparer<IEnumerable<T>>
        {
            public bool Equals(IEnumerable<T> x, IEnumerable<T> y)
            {
                return x.SequenceEqual(y);
            }

            public int GetHashCode(IEnumerable<T> obj)
            {
                return obj.Aggregate(0, (currentHashCode, element)
                                        => (currentHashCode ^ element.GetHashCode()));
            }
        }

    }
}
```

## Date de test

```
using System.Collections.Generic;
using System.Linq;
using AlgoritmicaGrafelor.Laborator4.Multimi;
using Microsoft.VisualStudio.TestTools.UnitTesting;

namespace AlgoritmicaGrafelor.Laborator4.Tests
{
    [TestClass]
    public class MultimiInteriorStabileTests
    {
        static private readonly IReadOnlyList<IReadOnlyList<int>> _onePeekGraph
            = new[]
            {
                new List<int>{ }
            };
        static private readonly IReadOnlyList<IReadOnlyList<int>> _twoPeekGraph
            = new[]
            {
                new List<int>{ 1 },
                new List<int>{ }
            };
        static private readonly IReadOnlyList<IReadOnlyList<int>> _fourPeekGraph
            = new[]
            {
                new List<int>{ 1, 2 },
                new List<int>{ 3 },
                new List<int>{ 3 },
                new List<int>{ }
            };
        static private readonly IReadOnlyList<IReadOnlyList<int>> _fivePeekGraphWithACycle
            = new[]
            {
                new List<int>{ 1 },
                new List<int>{ 2, 4 },
                new List<int>{ 3 },
                new List<int>{ 1 },
                new List<int>{ }
            };
```

```
[TestMethod]
public void TestForOnePeekGraph()
{
    IReadOnlyList<IReadOnlyList<int>> multimiInteriorStabile
        = MultimiInteriorStabile.Toate(_onePeekGraph);

    Assert.AreEqual(2, multimiInteriorStabile.Count);
    Assert.IsTrue(multimiInteriorStabile[0].Count == 1
                  || multimiInteriorStabile[1].Count == 0);
    if (multimiInteriorStabile[0].Count == 1)
    {
        Assert.AreEqual(0, multimiInteriorStabile[0][0]);
        Assert.AreEqual(0, multimiInteriorStabile[1].Count);
    }
    else
    {
        Assert.AreEqual(0, multimiInteriorStabile[1][0]);
        Assert.AreEqual(0, multimiInteriorStabile[0].Count);
    }
}

[TestMethod]
public void TestForTwoPeekGraph()
{
    IReadOnlyList<IReadOnlyList<int>> multimiInteriorStabile
        = MultimiInteriorStabile.Toate(_twoPeekGraph).OrderBy(multime
              => multime.Count).ToList();

    Assert.AreEqual(3, multimiInteriorStabile.Count);
    Assert.AreEqual(0, multimiInteriorStabile.Where(multime => multime.Count == 1)
                                             .Select(multime => multime.First())
                                             .Except(new[] { 0, 1 })
                                             .Count());
}

[TestMethod]
public void TestForFourPeekGraph()
{
    IReadOnlyList<IReadOnlyList<int>> multimiInteriorStabile
        = MultimiInteriorStabile.Toate(_fourPeekGraph).OrderBy(multime
              => multime.Count).ToList();

    Assert.AreEqual(7, multimiInteriorStabile.Count);
    Assert.AreEqual(0, multimiInteriorStabile.Where(multime => multime.Count == 1)
                                             .Select(multime => multime.First())
                                             .Except(new[] { 0, 1, 2, 3 })
                                             .Count());

    var misCuDouaElement
        = multimiInteriorStabile.Where(multime => multime.Count == 2)
              .Select(multime => multime.OrderBy(element => element))
              .OrderBy(multime => string.Join(", ", multime.Select(element
                  => element.ToString()))).ToList();
    Assert.AreEqual(2, misCuDouaElement.Count);
    Assert.IsTrue(new[] { 0, 3 }.SequenceEqual(misCuDouaElement[0]));
    Assert.IsTrue(new[] { 1, 2 }.SequenceEqual(misCuDouaElement[1]));
}
```

```
        [TestMethod]
        public void TestForFivePeekGraphWithACycle()
        {
            IReadOnlyList<IReadOnlyList<int>> multimiInteriorStabile
                = MultimiInteriorStabile.Toate(_fivePeekGraphWithACycle).OrderBy(multime
                      => multime.Count).ToList();

            Assert.AreEqual(13, multimiInteriorStabile.Count);
            Assert.AreEqual(0, multimiInteriorStabile.Where(multime => multime.Count == 1)
                                                     .Select(multime => multime.First())
                                                     .Except(new[] { 0, 1, 2, 3, 4 })
                                                     .Count());

            var misCuDouaElement = multimiInteriorStabile.Where(multime
                                                                => multime.Count == 2)
                    .Select(multime => multime.OrderBy(element => element))
                    .OrderBy(multime
                        => string.Join(", ", multime.Select(element => element.ToString())))
                    .ToList();
            Assert.AreEqual(5, misCuDouaElement.Count);
            Assert.IsTrue(new[] { 0, 2 }.SequenceEqual(misCuDouaElement[0]));
            Assert.IsTrue(new[] { 0, 3 }.SequenceEqual(misCuDouaElement[1]));
            Assert.IsTrue(new[] { 0, 4 }.SequenceEqual(misCuDouaElement[2]));
            Assert.IsTrue(new[] { 2, 4 }.SequenceEqual(misCuDouaElement[3]));
            Assert.IsTrue(new[] { 3, 4 }.SequenceEqual(misCuDouaElement[4]));
            var misCuTreiElement = multimiInteriorStabile.Where(multime
                                                                => multime.Count == 3)
                                    .Select(multime => multime.OrderBy(element
                                                                => element))
                                    .OrderBy(multime
                                        => string.Join(", ", multime.Select(element
                                                                => element.ToString())))
                                    .ToList();

            Assert.AreEqual(2, misCuTreiElement.Count);
            Assert.IsTrue(new[] { 0, 2, 4 }.SequenceEqual(misCuTreiElement[0]));
            Assert.IsTrue(new[] { 0, 3, 4 }.SequenceEqual(misCuTreiElement[1]));
        }
    }
}
```

Aici se afla grafele intr-o reprezentare grafica pentru a vizualiza mai usor datele de test. Acestea apar in aceasi ordine in care apar declarate mai sus.