Kauno technologijos universitetas
Informatikos fakultetas

Objektinis programavimas 2 (P175B123)

Laboratorinių darbų ataskaita

<Aistis Jakutonis> <IFF3/1>

Studentas

Prof. Vacius Jusas

Dėstytojas

Kaunas 2024

TURINYS

1. Rekursija (L1)

1.1. Darbo užduotis

LD_22. Kelias tarp vietovių.

Gūdučių universiteto informatikos fakulteto I kurso studentai nutarė dalyvauti orientavimosi dviračiais varžybose. Jie sudarė komandą ir atvyko į vietovę Preivai, kur bus duotas startas. Buvo pranešta, kad finišas Balkuose. Kaip ir kitų komandų atstovai, jie gavo vietovės žemėlapį, kuriame pažymėti visi keliai ir surašyti kelių ilgiai. Padėkite studentams surasti trumpiausią kelią tarp nurodytų vietovių, jei žinoma, kad kelias tarp starto ir finišo vietovių gali būti tiesioginis (be tarpinių vietovių) arba tarpe jų gali būti ne daugiau kaip 5 tarpinės vietovės.

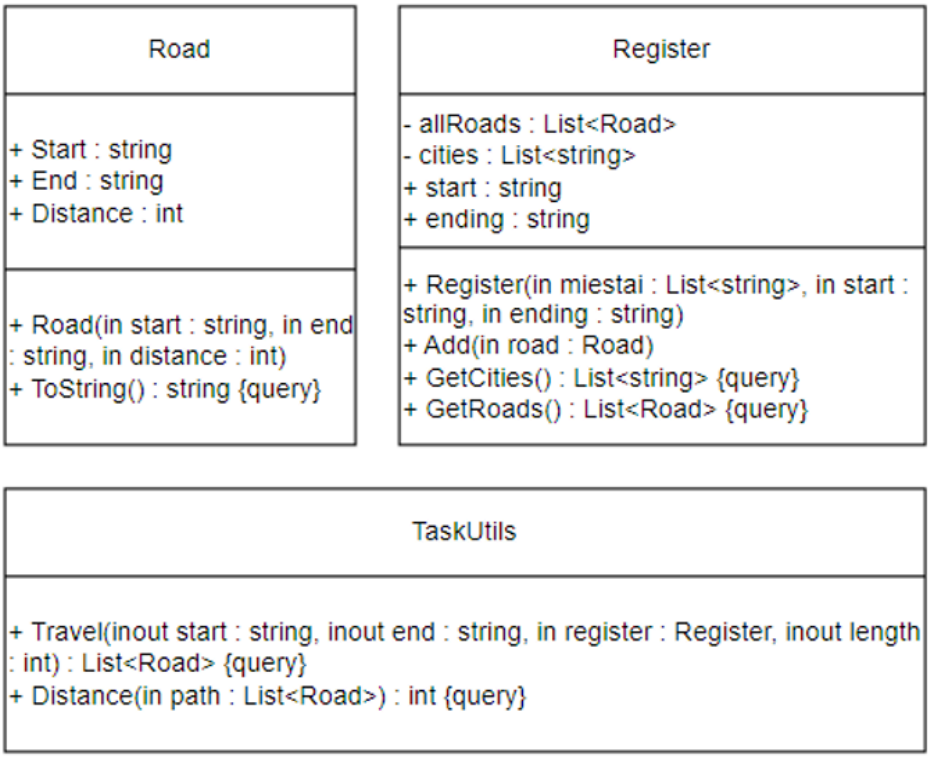
Duomenys. Tekstinio failo ‘U3.txt’ pirmoje eilutėje nurodytas vietovių skaičius N ($2 \leq N \leq 10$) ir visų kelių kiekis M ($1 \leq M \leq 50$). Tolimesnėse N eilutėse surašytos visos galimos vietovės po vieną eilutėje. Po to eilutėje surašytos starto ir finišo vietovių pavadinimai. Šiai eilutei iš viršaus ir apačios palikta po vieną tuščią eilutę. Po antros tuščios eilutės M eilutėse surašyti visi keliai po vieną eilutėje. Tokios eilutės struktūra: pradinė vietovė, galinė vietovė, atstumas tarp jų. Vietovės pavadinimas – iki 10 simbolių.

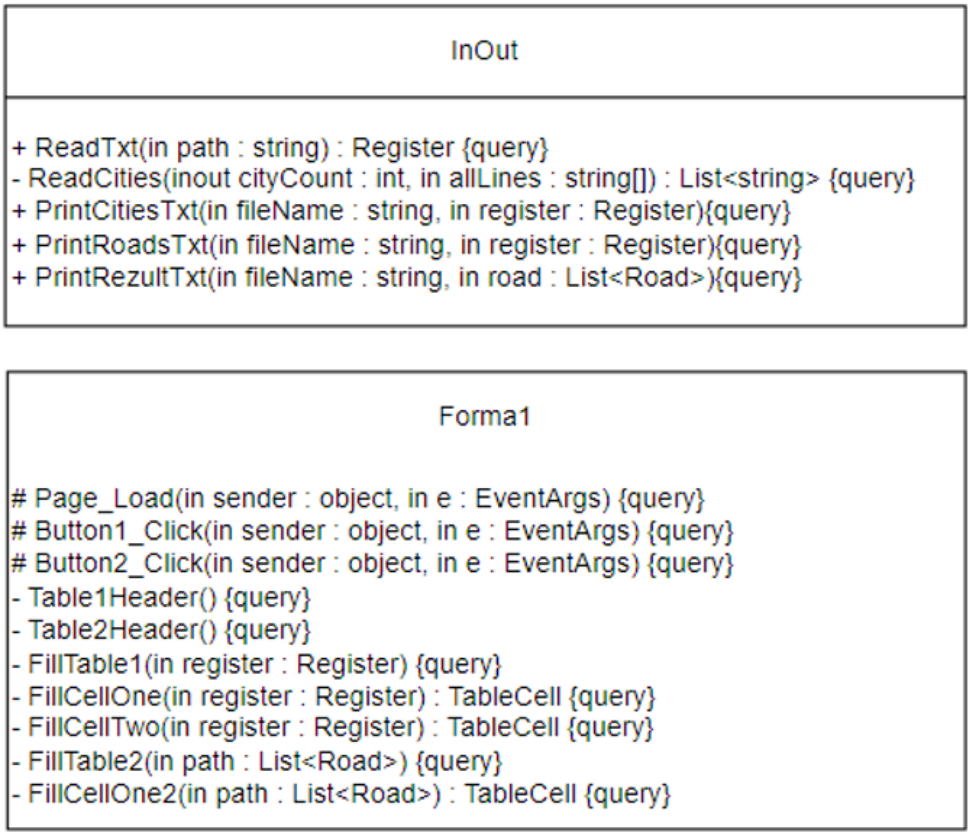
1.2. Grafinės vartotojo sąsajos schema

1.3. Sąsajoje panaudotų komponentų keičiamos savybės

Komponentas	Savybė	Reikšmė
Button	ID	Button1
Button	OnClick	Button1_Click
Button	Text	Pradėti
Button	Width	103px
Button	ID	Button2
Button	OnClick	Button2_Click
Button	Text	Apskaičiuoti
Button	Width	107px
Table	ID	Table1
Table	BorderColor	Black
Table	BorderStyle	Solid
Table	BorderWidth	1px
Table	Width	390px
Table	ID	Table2
Table	BorderColor	Black
Table	BorderStyle	Solid
Table	BorderWidth	1px
Table	Width	390px

1.4. Klasių diagrama





1.5. Programos vartotojo vadovas

Programos darbiniame aplanke atidarome App_Data aplanką, jame sukuriame failą U3.txt, kuriame pateikiame duomenis apie orientavimosi varžybas: Vietovių skaičius bei galimų maršrutų kiekis; išvardijame vietoves bei maršrutus bei jų ilgius; įrašome pradinę ir galutinę vietovę.

Įjungę programą, pirmiausia užkrauname duomenų failus. Tai padarome paspausdami mygtuką „Pradėti“. Paspaudę mygtuką patikriname ar duomenys buvo įvesti teisingai. Jei lentelėje duomenys teisingi, spaudžiame mygtuką „Apskaičiuoti“. Paspaudus mygtuką programa apdoros duomenis ir į ekraną išves lentelę su rezultatais.

1.6. Programos tekstas

```
using System;

namespace LD22_kelias_tarp_vietoviu
{
  /// <summary>
  /// Constructor class
  /// </summary>
  public class Road
  {
    public string Start {  get; }
    public string End { get; }
    public int Distance { get; }

    public Road(string start,
      string end, int distance)
    {
      this.Start = start;
```

```
            this.End = end;
            this.Distance = distance;
        }

        public override string ToString()
        {
            string line;

            line = String.Format($"| {this.Start,-11} " +
                  $"| {this.End,-11} | {this.Distance,8} km |");

            return line;
        }
    }
}



using System.Collections.Generic;

namespace LD22_kelias_tarp_vietoviu
{
    /// <summary>
    /// Register class in which the main
    /// information is stored
    /// </summary>
    public class Register
    {
        private List<Road> allRoads = new List<Road>();
        private List<string> cities { get; }
        public string start { get; }
        public string ending { get; }

        /// <summary>
        /// Gets cities, start and ending
        /// </summary>
        /// <param name="miestai"></param>
        /// <param name="start"></param>
        /// <param name="ending"></param>
        public Register(List<string> miestai,
            string start, string ending)
        {
            cities = new List<string>();

            foreach (string city in miestai)
            {
                cities.Add(city);
            }

            this.start = start;
            this.ending = ending;
        }

        /// <summary>
        /// Method adds road to the allRoads list
        /// </summary>
        /// <param name="road"></param>
        public void Add(Road road)
        {
            allRoads.Add(road);
        }

        /// <summary>
        /// Method returns the cities list
```

```
        /// </summary>
        /// <returns></returns>
        public List<string> GetCities()
        {
            return cities;
        }

        /// <summary>
        /// Method returns the roads list
        /// </summary>
        /// <returns></returns>
        public List<Road> GetRoads()
        {
            return allRoads;
        }
    }
}



using System.Collections.Generic;

namespace LD22_kelias_tarp_vietoviu
{
    /// <summary>
    /// Class contains calculations and recursion
    /// </summary>
    public class TaskUtils
    {
        /// <summary>
        /// Method with recursion to solve the task
        /// </summary>
        /// <param name="start"></param>
        /// <param name="end"></param>
        /// <param name="register"></param>
        /// <param name="length"></param>
        /// <returns></returns>
        public static List<Road> Travel(string start,
            string end, Register register, int length)
        {
            if (length > 5)
            {
                return null;
            }

            List<Road> path = null;
            int distance = -1;

            foreach (Road kelias in register.GetRoads())
            {
                List<Road> subpath;

                if (kelias.Start == start
                    && kelias.End == end)
                {
                    subpath = new List<Road>();
                    subpath.Add(kelias);
                }
                else if (kelias.Start == start)
                {
                    subpath = Travel(kelias.End, end,
                        register, length + 1);

                    if (subpath == null)
```

```
                {
                    continue;
                }

                subpath.Insert(0, kelias);
            }
            else
            {
                continue;
            }

            int subdistance = Distance(subpath);

            if (distance < 0 || distance
                > subdistance)
            {
                distance = subdistance;
                path = subpath;
            }
        }

        return path;
    }

    /// <summary>
    /// Method calculates the distance
    /// between all given roads
    /// </summary>
    /// <param name="path"></param>
    /// <returns></returns>
    public static int Distance(List<Road> path)
    {
        int distance = 0;

        foreach (Road road in path)
        {
            distance += road.Distance;
        }

        return distance;
    }
  }
}
```

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Text;
using System.Text.RegularExpressions;

namespace LD22_kelias_tarp_vietoviu
{
  /// <summary>
  /// Reading and printing class
  /// </summary>
  public static class InOut
  {
    /// <summary>
    /// Reads the data from the given file
    /// </summary>
    /// <param name="path"></param>
```

```
/// <returns></returns>
public static Register ReadTxt(string path)
{
   string[] allLines = File.ReadAllLines(path);
   string pattern = "\\s+";

   string[] parts = allLines[0].Split(' ');
   int cityCount = int.Parse(parts[0]);
   int roadCount = int.Parse(parts[1]);

   List<string> cities = ReadCities(cityCount,
      allLines);

   string[] matches =
      Regex.Split(allLines[cityCount + 2],
      pattern);
   string begining = matches[0];
   string ending = matches[1];

   Register register = new Register(cities,
      begining, ending);

   for (int i = cityCount + 4;
      i < roadCount + cityCount + 4; i++)
   {
      string[] line = Regex.Split(allLines[i],
         pattern);

      Road road = new Road(line[0], line[1],
         int.Parse(line[2]));

      register.Add(road);
   }

   return register;
}

/// <summary>
/// Separately reads cities and returns the list
/// </summary>
/// <param name="cityCount"></param>
/// <param name="allLines"></param>
/// <returns></returns>
private static List<string> ReadCities(int cityCount,
   string[] allLines)
{
   List<string> cities = new List<string>();

   for (int i = 1; i < cityCount + 1; i++)
   {
      cities.Add(allLines[i]);
   }

   return cities;
}

/// <summary>
/// Prints a table to txt file with all the cities
/// </summary>
/// <param name="fileName"></param>
/// <param name="register"></param>
public static void PrintCitiesTxt(string fileName,
   Register register)
{
```

```
    File.AppendAllText(fileName,
        "Pradiniai duomenys:\r\n", Encoding.UTF8);

    List<string> lines = new List<string>();

    lines.Add(new string('-', 21));
    lines.Add(String.Format($"| " +
        $"{"Galimos vietovės",-17} |"));
    lines.Add(new string('-', 21));

    foreach (string city in register.GetCities())
    {
        lines.Add(String.Format($"| {city,-17} |"));
        lines.Add(new string('-', 21));
    }

    File.AppendAllLines(fileName, lines,
        Encoding.UTF8);
}

/// <summary>
/// Prints the table to txt file of all possible roads
/// </summary>
/// <param name="fileName"></param>
/// <param name="register"></param>
public static void PrintRoadsTxt(string fileName,
    Register register)
{
    List<string> lines = new List<string>();

    lines.Add("");
    lines.Add(new string('-', 43));
    lines.Add(String.Format($"| {"Pradžia",-11} " +
        $"| {"Pabaiga",-11} | {"Atstumas",-8} km |"));
    lines.Add(new string('-', 43));

    foreach (Road road in register.GetRoads())
    {
        lines.Add(road.ToString());
        lines.Add(new string('-', 43));
    }

    lines.Add("");

    File.AppendAllLines(fileName, lines, Encoding.UTF8);
}

/// <summary>
/// Prints the rezults to txt file in a table
/// </summary>
/// <param name="fileName"></param>
/// <param name="road"></param>
public static void PrintRezultTxt(string fileName,
    List<Road> road)
{
    File.AppendAllText(fileName, "Rezultatai:\r\n",
        Encoding.UTF8);

    List<string> lines = new List<string>();

    lines.Add(new string('-', 43));
    lines.Add(String.Format("| {0,-39} |", "Minimalus atstumas tarp vietovių"));
    lines.Add(new string('-', 43));
    lines.Add(String.Format($"| {"Pradžia",-11} " +
```

```
            $"| {"Pabaiga",-11} | {"Atstumas",-8} km |"));
        lines.Add(new string('-', 43));

        lines.Add(String.Format($"| {road[0].Start,-11} " +
            $"| {road[road.Count - 1].End,-11} | " +
            $"{TaskUtils.Distance(road),8} km |"));
        lines.Add(new string('-', 43));

        lines.Add(String.Format("| {0,-39} |", "Trasa eina per vietoves"));
        lines.Add(new string('-', 43));

        lines.Add(String.Format($"| {road[0].Start,-39} |"));
        lines.Add(new string('-', 43));

        foreach (Road r in road)
        {
            lines.Add(String.Format($"| {r.End,-39} |"));
            lines.Add(new string('-', 43));
        }

        File.AppendAllLines(fileName, lines, Encoding.UTF8);
    }
  }
}



using System;
using System.Collections.Generic;
using System.IO;
using System.Web.UI.WebControls;

namespace LD22_kelias_tarp_vietoviu
{
  public partial class Forma1 : System.Web.UI.Page
  {
    protected void Page_Load(object sender, EventArgs e)
    {
      Table1.Visible = false;
      Table2.Visible = false;
      Button2.Visible = false;
    }

    protected void Button1_Click(object sender, EventArgs e)
    {
      Register register =
        InOut.ReadTxt(Server.MapPath("App_Data/U3.txt"));

      Table1Header();
      FillTable1(register);
      Button1.Visible = false;
      Table1.Visible = true;
      Button2.Visible = true;

      List<Road> road = TaskUtils.Travel(register.start,
        register.ending, register, 0);

      File.Delete(Server.MapPath("Rezultatai.txt"));
      InOut.PrintCitiesTxt(Server.MapPath("Rezultatai.txt"), register);
      InOut.PrintRoadsTxt(Server.MapPath("Rezultatai.txt"), register);
      InOut.PrintRezultTxt(Server.MapPath("Rezultatai.txt"), road);

      Session["keliai"] = road;
```

```
}

protected void Button2_Click(object sender, EventArgs e)
{
   Button1.Visible = false;

   List<Road> path = (List<Road>)Session["keliai"];

   if (path.Count == 0)
   {
      TableCell cell = new TableCell();
      cell.Text = "Nėra trumpiausio kelio";

      TableRow row = new TableRow();
      row.Cells.Add(cell);

      Table2.Rows.Add(row);
   }
   else
   {
      Table2Header();
      FillTable2(path);
      Table2.Visible = true;
   }
}

/// <summary>
/// Makes a table1 header
/// </summary>
private void Table1Header()
{
   TableCell cell = new TableCell();
   cell.Text = "Duomenys";
   TableCell cellOne = new TableCell();
   cellOne.Text = "Miestai";
   TableCell cellTwo = new TableCell();
   cellTwo.Text = "Keliai";

   TableRow rowZero = new TableRow();
   rowZero.Cells.Add(cell);
   TableRow row = new TableRow();
   row.Cells.Add(cellOne);
   row.Cells.Add(cellTwo);

   Table1.Rows.Add(rowZero);
   Table1.Rows.Add(row);
}

/// <summary>
/// Makes a table2 header
/// </summary>
private void Table2Header()
{
   TableCell cellOne = new TableCell();
   cellOne.Text = "Rezultatai";

   TableRow row = new TableRow();
   row.Cells.Add(cellOne);

   Table2.Rows.Add(row);
}

/// <summary>
/// Fills table1 with given parameters
```

```
/// </summary>
/// <param name="register"></param>
private void FillTable1(Register register)
{
   TableCell cellOne = new TableCell();
   cellOne = FillCellOne(register);

   TableCell cellTwo = new TableCell();
   cellTwo = FillCellTwo(register);

   TableRow row = new TableRow();
   row.Cells.Add(cellOne);
   row.Cells.Add(cellTwo);

   Table1.Rows.Add(row);
}

/// <summary>
/// Table1 cell is filled with cities
/// </summary>
/// <param name="register"></param>
/// <returns></returns>
private TableCell FillCellOne(Register register)
{
   TableCell cellOne = new TableCell();

   foreach (string city in register.GetCities())
   {
      cellOne.Text += city + "<br />";
   }

   return cellOne;
}

/// <summary>
/// Table1 second cell is filled with posible roads
/// </summary>
/// <param name="register"></param>
/// <returns></returns>
private TableCell FillCellTwo(Register register)
{
   TableCell cellTwo = new TableCell();

   foreach (Road road in register.GetRoads())
   {
      cellTwo.Text += road.Start + " -> "
         + road.End + " " + road.Distance + " km" + "<br />";
   }

   return cellTwo;
}

/// <summary>
/// Fills table2 with calculated results
/// </summary>
/// <param name="path"></param>
private void FillTable2(List<Road> path)
{
   TableCell cellOne = new TableCell();
   cellOne = FillCellOne2(path);

   TableRow row = new TableRow();
   row.Cells.Add(cellOne);
```

```
            Table2.Rows.Add(row);
        }

        /// <summary>
        /// Table2 cell is filled with results
        /// </summary>
        /// <param name="path"></param>
        /// <returns></returns>
        private TableCell FillCellOne2(List<Road> path)
        {
            TableCell cellOne = new TableCell();

            cellOne.Text = "Minimalus atstumas tarp vietovių" + "<br />";
            cellOne.Text += path[0].Start + " ir "
                + path[path.Count - 1].End + " "
                + TaskUtils.Distance(path) + " km" + "<br />";
            cellOne.Text += "Trasa eina per vietoves:" + "<br />";

            foreach(Road kelias in path)
            {
                cellOne.Text += kelias.Start + "<br />";
            }

            cellOne.Text += path[path.Count - 1].End;

            return cellOne;
        }
    }
}
```

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Forma1.aspx.cs"
Inherits="LD22_kelias_tarp_vietoviu.Forma1" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:Button ID="Button1" runat="server" OnClick="Button1_Click" Text="Pradėti" Width="103px" />
            <br />
            <br />
            <asp:Table ID="Table1" runat="server" BorderColor="Black" BorderStyle="Solid" BorderWidth="1px" Width="390px">
            </asp:Table>
            <br />
            <asp:Button ID="Button2" runat="server" Text="Apskaičiuoti" OnClick="Button2_Click" Width="107px" />
            <br />
            <br />
            <asp:Table ID="Table2" runat="server" BorderColor="Black" BorderStyle="Solid" BorderWidth="1px" Width="390px">
            </asp:Table>
        </div>
    </form>
</body>
</html>
```

1.7. Pradiniai duomenys ir rezultatai

1.7.1 Pradiniai duomenys ir rezultatai 1

Pradiniai duomenys:

```
5 8
Preivai
Saukai
Salai
Rekai
Balkai

Preivai    Balkai

Preivai    Saukai     2
Preivai    Salai      8
Preivai    Balkai     10
Saukai     Balkai     7
Saukai     Rekai      2
Salai      Rekai      4
Salai      Balkai     6
Rekai      Balkai     2
```

Šiais duomenimis tikrinama ar veikia programa ar teisingai yra apdorojami duomenys pritaikant rekursiją.

Pradiniai duomenys web:

Duomenys

Miestai	Keliai
Preivai Saukai Salai Rekai Balkai	Preivai -> Saukai 2 km Preivai -> Salai 8 km Preivai -> Balkai 10 km Saukai -> Balkai 7 km Saukai -> Rekai 2 km Salai -> Rekai 4 km Salai -> Balkai 6 km Rekai -> Balkai 2 km

Rezultatai web:

Rezultatai

Minimalus atstumas tarp vietovių
Preivai ir Balkai 6 km
Trasa eina per vietoves:
Preivai
Saukai
Rekai
Balkai

Duomenys ir rezultatai txt faile:

```
Pradiniai duomenys:
---------------------
| Galimos vietovės  |
---------------------
| Preivai           |
---------------------
| Saukai            |
---------------------
| Salai             |
---------------------
| Rekai             |
---------------------
| Balkai            |
---------------------

-------------------------------------------
| Pradžia     | Pabaiga     | Atstumas km |
-------------------------------------------
| Preivai     | Saukai      |        2 km |
-------------------------------------------
| Preivai     | Salai       |        8 km |
-------------------------------------------
| Preivai     | Balkai      |       10 km |
-------------------------------------------
| Saukai      | Balkai      |        7 km |
-------------------------------------------
| Saukai      | Rekai       |        2 km |
-------------------------------------------
| Salai       | Rekai       |        4 km |
-------------------------------------------
| Salai       | Balkai      |        6 km |
-------------------------------------------
| Rekai       | Balkai      |        2 km |
-------------------------------------------
```

```
Rezultatai:
-------------------------------------------
| Minimalus atstumas tarp vietovių        |
-------------------------------------------
| Pradžia     | Pabaiga     | Atstumas km |
-------------------------------------------
| Preivai     | Balkai      |        6 km |
-------------------------------------------
| Trasa eina per vietoves                 |
-------------------------------------------
| Preivai                                 |
-------------------------------------------
| Saukai                                  |
-------------------------------------------
| Rekai                                   |
-------------------------------------------
| Balkai                                  |
-------------------------------------------
```

1.7.2 Pradiniai duomenys ir rezultatai 2

Pradiniai duomenys:

```
5 8
Preivai
Saukai
Salai
Rekai
Balkai

Preivai    Balkai

Preivai    Saukai     2
Preivai    Salai      2
Preivai    Balkai     10
Saukai     Balkai     7
Saukai     Rekai      2
Salai      Rekai      4
Salai      Balkai     6
Rekai      Balkai     5
```

Šiais duomenimis tikrinama ar rekursija pereina per kitus variantus, o ne tik per pirmąjį.

Pradiniai duomenys web:

Duomenys

Miestai	Keliai
Preivai Saukai Salai Rekai Balkai	Preivai -> Saukai 2 km Preivai -> Salai 2 km Preivai -> Balkai 10 km Saukai -> Balkai 7 km Saukai -> Rekai 2 km Salai -> Rekai 4 km Salai -> Balkai 6 km Rekai -> Balkai 5 km

Rezultatai web:

Rezultatai

Minimalus atstumas tarp vietovių
Preivai ir Balkai 8 km
Trasa eina per vietoves:
Preivai
Salai
Balkai

Duomenys ir rezultatai txt faile:

```
Pradiniai duomenys:
---------------------
| Galimos vietovės  |
---------------------
| Preivai           |
---------------------
| Saukai            |
---------------------
| Salai             |
---------------------
| Rekai             |
---------------------
| Balkai            |
---------------------

-----------------------------------------
| Pradžia     | Pabaiga     | Atstumas km |
-----------------------------------------
| Preivai     | Saukai      |        2 km |
-----------------------------------------
| Preivai     | Salai       |        2 km |
-----------------------------------------
| Preivai     | Balkai      |       10 km |
-----------------------------------------
| Saukai      | Balkai      |        7 km |
-----------------------------------------
| Saukai      | Rekai       |        2 km |
-----------------------------------------
| Salai       | Rekai       |        4 km |
-----------------------------------------
| Salai       | Balkai      |        6 km |
-----------------------------------------
| Rekai       | Balkai      |        5 km |
-----------------------------------------
```

```
Rezultatai:
-------------------------------------------
| Minimalus atstumas tarp vietovių        |
-------------------------------------------
| Pradžia     | Pabaiga     | Atstumas km |
-------------------------------------------
| Preivai     | Balkai      |        8 km |
-------------------------------------------
| Trasa eina per vietoves                 |
-------------------------------------------
| Preivai                                 |
-------------------------------------------
| Salai                                   |
-------------------------------------------
| Balkai                                  |
-------------------------------------------
```

1.8. Dėstytojo pastabos

2. Dinaminis atminties valdymas (L2)

2.1. Darbo užduotis

2.2. Grafinės vartotojo sąsajos schema

2.3. Sąsajoje panaudotų komponentų keičiamos savybės

Komponentas	Savybė	Reikšmė

2.4. Klasių diagrama

2.5. Programos vartotojo vadovas

2.6. Programos tekstas

2.7. Pradiniai duomenys ir rezultatai

2.8. Dėstytojo pastabos

3. Bendrinės klasės ir testavimas (L3)

3.1. Darbo užduotis

3.2. Grafinės vartotojo sąsajos schema

3.3. Sąsajoje panaudotų komponentų keičiamos savybės

Komponentas	Savybė	Reikšmė

3.4. Klasių diagrama

3.5. Programos vartotojo vadovas

3.6. Programos tekstas

3.7. Pradiniai duomenys ir rezultatai

3.8. Dėstytojo pastabos

4. Polimorfizmas ir išimčių valdymas (L4)

4.1. Darbo užduotis

4.2. Grafinės vartotojo sąsajos schema

4.3. Sąsajoje panaudotų komponentų keičiamos savybės

Komponentas	Savybė	Reikšmė

4.4. Klasių diagrama

4.5. Programos vartotojo vadovas

4.6. Programos tekstas

4.7. Pradiniai duomenys ir rezultatai

4.8. Dėstytojo pastabos

5. Deklaratyvusis programavimas (L5)

5.1. Darbo užduotis

5.2. Grafinės vartotojo sąsajos schema

5.3. Sąsajoje panaudotų komponentų keičiamos savybės

Komponentas	Savybė	Reikšmė

5.4. Klasių diagrama

5.5. Programos vartotojo vadovas

5.6. Programos tekstas

5.7. Pradiniai duomenys ir rezultatai

5.8. Dėstytojo pastabos