

**УНИВЕРСИТЕТ ПО БИБЛИОТЕКОЗНАНИЕ**

**И ИНФОРМАЦИОННИ ТЕХНОЛОГИИ**

**КАТЕДРА „Информационни системи и технологии“**

**КУРСОВА РАБОТА**

**по**

**ДИСЦИПЛИНА:** **Структури от данни и алгоритми**

**на тема:**

**Определяне на най-кратък път между два града**

**чрез претеглен граф и алгоритъм на Dijkstra**

**(Приложение с Windows Forms GUI на C#)**

**СТУДЕНТ:**  **Преподавател:**.................................

*Стефан Кожухаров проф. д-р Иван Иванов*

Магистърска програма

ИСТ 2022/2023 – дистанционно

ф. номер 0463-имд

|  |
| --- |
|  |

# Съдържание

[**Увод** 1](#_Toc130403521)

[**1.** **Описание на реализираните функционалности** 2](#_Toc130403522)

[**2.** **Изходни данни за градовете** 4](#_Toc130403523)

[**3.** **Изобразяване на графа и най-късия път** 4](#_Toc130403524)

[**4.** **Решение на задачата за намиране на най-късия път** 4](#_Toc130403525)

[**5.** **Последователност на изпълнение** 5](#_Toc130403526)

[**Използвани източници** 6](#_Toc130403527)

[**Приложения** 6](#_Toc130403528)

# **Увод**

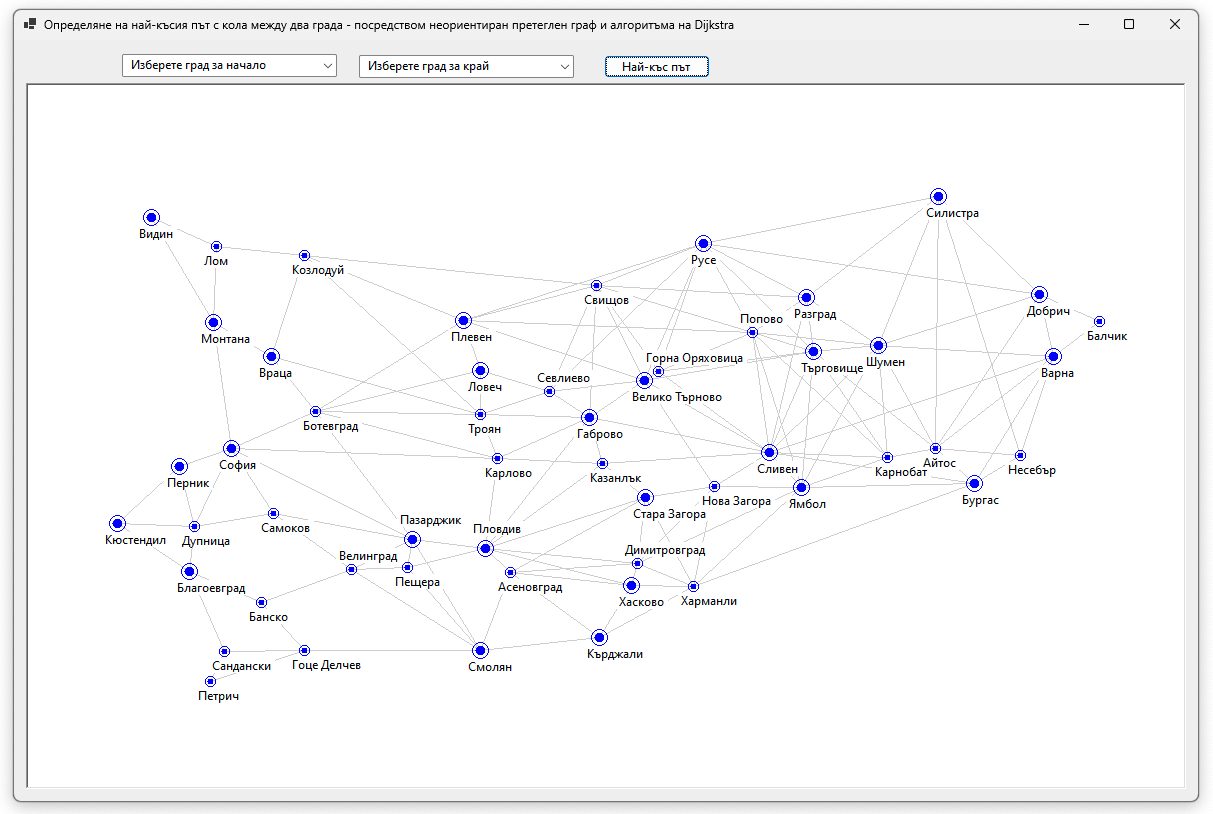
В курсовата работа е описано приложение за намиране на най-краткия път между два града (от списък с 53 от най-големите градове в България), изготвено на C#, .NET 6.0 (LTS) Framework, с Windows Forms GUI и Visual Studio 2022.

Демонстрирани са умения за работа с контролите на Windows Forms GUI с App Properties / Resources (за csv файловете с данни за градовете), познания за подреждане и организиране на кода, както и за намиране на решения в oфициалната документация на Windows Forms [3] и StackOverflow [4].

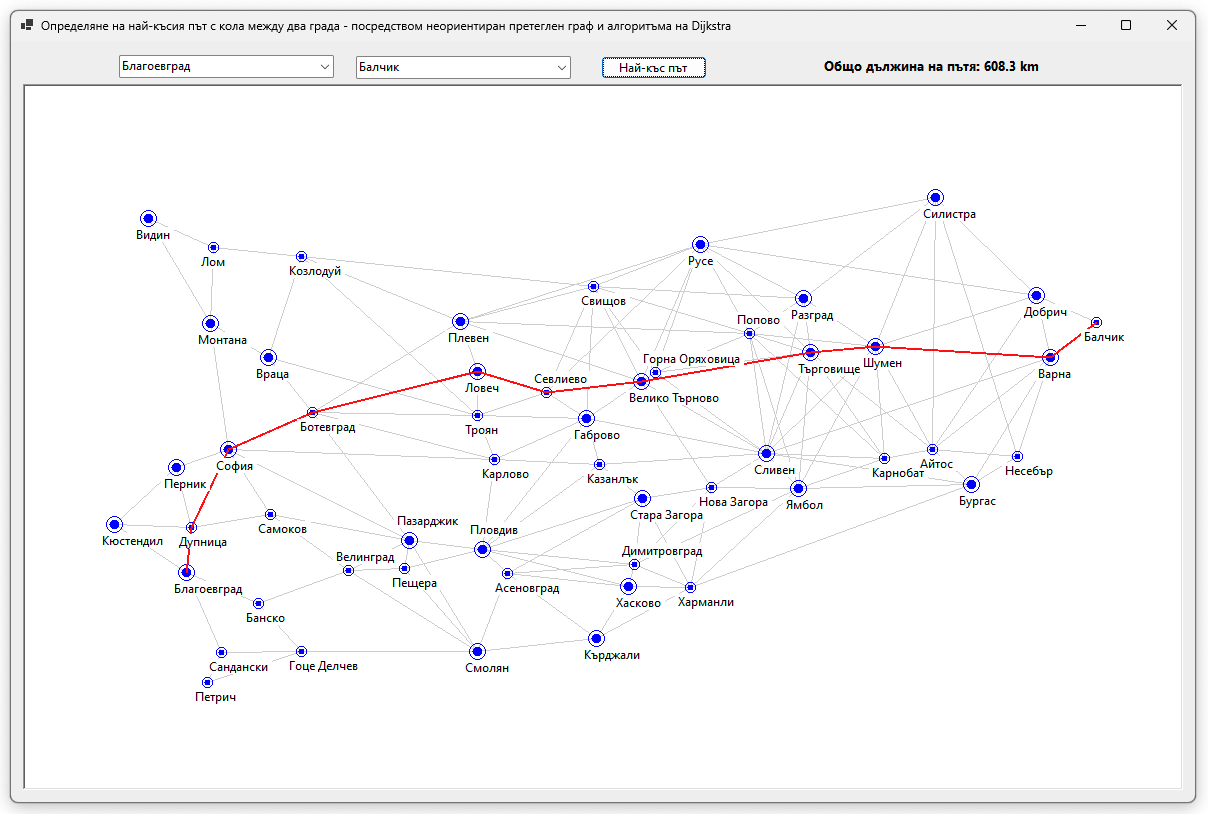
За решението на задачата е използван неориентиран претеглен граф и алгоритъма на Dijkstra (Структури от данни и алгоритми в С# [2], Глава 9), както и собствен (не наготово от интернет) алгоритъм filterPath() за филтриране на масива с ребра от решението на Dijkstra за намиране само на тези ребра, участващи в търсения най-къс път между два възела (града).

# **Описание на реализираните функционалности**

Два ComboBox (падащи менюта) със сортиран списък на градовете – за избор на начало и край, Button за стартиране/показване на резултата, Label с данни за дължината на намерения най-кратък път и Panel, в който се визуализира графа и решението. Видът на приложението при стартиране и след намиране на резултата е даден на (*фиг.1)* и (*фиг.2)*.



*Фигура 1. Изглед на разработеното приложение при стартиране*

**

*Фигура 2. Изглед на разработеното приложение след намерено решение*

# **Изходни данни за градовете**

Данните за градовете са организирани в два файла (Cities.csv и Distances.csv) в App Properties/Resources.

Файл Cities.csv съдържа имената на градовете с население над 20 000 души (плюс няколко курортни селища), както и географските им координати, взети от *Списък на градовете в България по население* [5], а също и обозначение Big (за областен град) или Small (за не-областен град).

Файл Distances.csv съдържа двойки градове (имената им) и дължината на пътя между тях. За определяне на наличието на път между два града и съответната му дължината е ползвана функционалността Directions на Google Maps, като са включени само пътища от републиканската пътна мрежа (Автомагистрали и I-III клас), без общински пътища (IV клас).

В папката на проекта е записан и файл Градове.xlsx, в който предварително са генерирани данните за градовете и от който са експортирани двата csv файла.

# **Изобразяване на графа и най-късия път**

Визуализирането на графа (възлите и ребрата) се осъществява посредством добавяне към event panelMap.Paint на съответния метод – DrawGraph(PaintEventArgs e), DrawTown(Vertex vertex, PaintEventArgs e) и т.н., в който е капсулирано самото изчертаване посредством системните методи FillEllipse, DrawEllipse и DrawLine на e.Graphics.

За изобразяване на графа, географските координати на градовете са трансформирани съобразно размера на панела Panel и разликата в ориентацията на координатните системи. Методите за четене от csv и трансформация на координатите са капсулирани в struct Cities и class Vertex.

В зависимост от вида на града (Big или Small) се изобразява голям или малък кръгов знак от метода DrawTown() на формата. Надписването на градовете е капсулирано в метод LabelTown() на формата, като за целите на курсовата е hardcode-нато позициониране на надписа над града за няколко случая, поради застъпване с близък град (реално би трябвало да се проверява разстоянието между близки градове и да се генерализира разместването на надписи).

Теглата на ребрата между възлите не е просто геометрично разстояние между две точки по координати, а се чете от Distances.csv (методът за четене е капсулиран в самия граф).

За изобразяване на най-късия път се изчертават нови по-дебели червени линии между съответните градове.

# **Решение на задачата за намиране на най-късия път**

За целта се ползва неориентиран претеглен граф и един от най-известните алгоритми в компютърните науки – алгоритъма на Dijkstra.

Графът е нелинейна, динамична структура от данни, изградена от възли (върхове) и връзки между тях наречени ребра.

Алгоритъмът на Dijkstra служи за намиране на най-краткия път от даден връх до всички останали върхове в претеглен граф с неотрицателни тегла на ребрата. При този алгоритъм се създава таблица за съхранение на разстоянията от началния връх до другите върхове в графа. Първоначално всеки връх има безкрайна дължина, която се намалява при изпълнение на алгоритъма. Всеки съседен връх на началния се посещава и в таблицата се записва теглото на съседното ребро. Алгоритъмът работи, като поддържа дължината на намерения до момента най-кратък път от началния връх до всеки посетен връх. При следващата итерация се посещават съседните върхове на последно обходените. Така постепенно множеството от върховете, за които тази дължина със сигурност е оптимална, се разширява. На всяка итерация алгоритъмът избира такъв връх, за който разстоянието от началния връх до него е минимално. Алгоритъмът приключва, когато се обходят всички върхове в графа.

За основа е ползван графът от „Структури от данни и алгоритми в С# [2], Глава 9“, с основни методи AddVertex(), AddEdge(), PathsToAllVertices() и помощни GetMin(), CalculateShortPath().

За целите на задачата графът е допълнен с методи LoadEdges() за четене на ребрата от csv файл, FindPathFromTo() за генериране на списък с ребрата от алгоритъма на Dijkstra и filterPath() с рекурсивен алгоритъм за филтриране на ребрата по търсения път (тъй като алгоритъмът на Dijkstra генерира най-малките разстояния от началния връх до всички в графа). За целите на метод FindPathFromTo() код базата е допълнена с class Edge. Принципът за капсулация е реализиран с публичен метод generatePath(), в който се извикват горе изброените методи. За целите на графичното изобразяване на графа, клас Vertex е допълнен със свойства за координатите x и y и размера на града (enum TownSize).

# **Последователност на изпълнение**

* В конструктора на формата:
* В свойството List<Vertex> cities се зареждат градовете от Cities.csv, като на метода loadVertices() на struct Cities се подават размерите на панела panelMap (за трансформация на координатите);
* инициализира се граф Graph g с размер равен на броя на заредените градове и върхове с данни от самите градове;
* в конструктора на графа се зареждат ребрата от Distances.csv;
* В метода Form1\_Load() на формата:
* Двата ComboBox се зареждат с имената на градовете от cities;
* Добавя се метод DrawGraph() към event Paint на панела panelMap и се извиква неговия метод Invalidate();
* Графът се изчертава от метода DrawGraph(), чрез итериране на записаните ребра в свойство AllEgdes на графа и с помощните методи DrawTown() и LabelTown();
* След избиране на начален и краен град и натискане на бутона buttonDijkstra:
* Прави се валидация на въведените данни в двата ComboBox и при липса на избран град се извеждат съответните съобщения;
* Извиква се метод generatePath() на графа, който реализира вътрешните методи с алгоритми за намиране на най-краткия път. Пътят се записва в свойство Path на графа;
* Публичният метод generatePath() връща общата дължина на намерения път, която се изобразява в Label;
* Добавя се метод DrawPath() към event Paint на панела panelMap и се извиква неговия метод Invalidate();
* Методът DrawPath() изчертава линиите на най-късият път, като итерира генерираните данни в свойство Path на графа;
* Методът DrawPath() приключва с извикване на метода CleartPaths() на графа, който изчиства/занулява свойствата на графа, съхраняващи решенията от изпълнение на алгоритмите. Така е възможно коректно изпълнение на задачата при селектиране на нови градове от падащите списъци, без да се налага рестартиране на приложението.

**this.Close()**

# **Използвани източници**

Литература:

1. Иван Иванов, Петър Стойков, Структури от данни и алгоритми, София, 2014
2. Иван Иванов, Петър Стойков, Структури от данни и алгоритми в С#, София, 2015

Уеб ресурси:

1. https://learn.microsoft.com/en-us/dotnet/desktop/winforms/?view=netdesktop-6.0
2. https://stackoverflow.com
3. https://bg.wikipedia.org/wiki/Списък\_на\_градовете\_в\_България\_по\_население

# **Приложения**

1. Solutions със C# кода от Visual Studio на Задачи 1 ÷ 9 от книгите „Структури от данни и алгоритми“ [1] и „Структури от данни и алгоритми в С#“ [2] в директории SDA\_0463\_imd\_01 ÷ 09;
2. Solution със C# кода от Visual Studio на приложението по настоящата курсова работа в директория SDA \_0463\_imd\_MyProject.