Doprovodný soubor k zápočtovému programu Graphs

Uživatelská dokumentace

Interaktivní program slouží k tvorbě a následnému vykreslení orientovaného nebo neorientovaného grafu včetně možnosti na něm pustit vybrané grafové algoritmy.
Byl vyvinut pomocí platformy *Microsoft Visual Studio Community 2019* v jazyce C#.

Návod k použití

Po spuštění programu se před Vámi objeví okno rozdělené na dvě části.

Tmavá část, vlevo, pracuje pouze jako místo k vykreslování a tvorbě grafu. Pro vytvoření vrcholu klikněte levým tlačítkem myši. Po jeho vytvoření se z něho začne natahovat hrana, kterou je možno zakotvit v jiném vrcholu, nebo s ní vytvořit nový, znovu, kliknutím levého tlačítka myši. Pokud chcete táhnutí zastavit – klikněte pravým tlačítkem u myši.
Pro tvorbu hrany je nutné mít k dispozici alespoň jeden vrchol, na který je kliknuto levým tlačítkem.
Vprostřed vytvořené hrany je zobrazena její váha – tu lze změnit kliknutím levého tlačítka dostatečně do zmíněného středu hrany. Po kliknutí máte prostor pro psaní - můžete využít pouze čísel, tlačítka pro znaménko minus a tlačítka backspace pro mazání dosud napsaného čísla. Pro ukončení psaní kamkoliv klikněte, nebo stiskněte jiné než zmíněné povolené klávesy.
Orientovanost hran je možné zařídit kliknutím na jejich konce levým tlačítkem myši. To samé lze říci i pro odstranění orientovanosti. Je možnost tvořit pouze jednu orientovanou hranu mezi dvěma vrcholy.
Aby byl přehled o očíslování jednotlivých vrcholů – je proto možné jej přejetím myši po vrcholu zjistit.
K dispozici je i mazání dosud vytvořených objektů – lze ho docílit stisknutím pravého tlačítka u myši přímo na odstraňovaný objekt. S mazaným vrcholem se odstraní i hrany, které z něj vedou.

Světlá část, vpravo, funguje, zkráceně, jako okno pro práci s algoritmy a maticí sousednosti, dole, která zachycuje vykreslený graf. Vytvořený graf, tedy jeho matici sousednosti, lze tlačítkem *Copy* zkopírovat. Okno v dolní části panelu lze využít pro nahrání své vlastní matice sousednosti, která se po kliknutí na tlačítko *Insert* sama nahraje a v tmavé části vykreslí jako graf. Nahrávat lze pouze ve formátu s hranatými závorkami, kde jsou jednotlivé elementy odděleny čárkou - [[0, 1], [1, 0]].

V horní části je možné si tlačítky „◀ ▶“ vybrat algoritmus, který na grafu pustit – tlačítkem *Start*. Po spuštění se napravo od textu *Results:* objeví výsledky v různých formátech. Pro některé algoritmy je možné si vedle textu „Pick a source“ vybrat počáteční vrchol algoritmu – pokud není vybrán, je automaticky 1.

Každý algoritmus má graf pro příklad – nahrát ho lze kliknutím na tlačítko v pravém dolním rohu pod textem Import example.

Programátorská dokumentace

Program je rozdělen do pěti veřejných tříd – Form1, Vertex, Edge, Heap, Algorithms. Tvořený graf se ukládá do matice sousednosti, jednotlivé vrcholy do List<Vertex>, hrany do List<Edge>. Po kliknutí na tlačítko Start se vytvoří nový objekt třídy Algorithms, kde se v jeho konstruktoru zpracuje předaná matice sousednosti, a spustí se vybraný grafový algoritmus. Jeho výsledek se přenese jako string do jednoho z Labelů u textu *Results:*.

public partial class Form1 : Form

private void Clicked

public void AdjustMatrix

public void CheckMatrixSize

public string PrintMatrix

public void InsertMatrix

public string MatrixToString

public void InitEdge

public void LeftClickedVertex

public void LeftClickedEdge

public void DeleteVertex

public void DeleteEdge

public class Vertex

- Třída vrcholu. Udržuje si číslo, poloměr vrcholu (kvůli hover funkcím) a svoji pozici.

public class Edge

- Třída hrany. Udržuje si vrcholy, které spojuje, orientovanost, seznam tří Pointů, které udávají: na nultém indexu – pozici vrcholu kam (při orientovanosti) hrana vede a na zbylých dvou – pozice odkud vedou části šipky. Dále obsahuje svoji váhu jako integer a weighLabel, který váhu zobrazuje.

public class Heap

- Třída binární haldy s prvky tvaru immuntable tuple integerů. Nepoužívá nultý index a pro každý prvek v haldě je zaznamenána jeho pozice v *pos* poli integerů, které dostane z konstruktoru svoji délku (ta je určena maximální hodnotou klíče).

public void Insert – Přidá na poslední pozici v haldě, zaznamená si ji a nechá vybublat nahoru pomocí BubbleUp.

private void BubbleUp – Vybublá nahoru s tím, že pokaždé upraví pozici prvku. Pracuje v čase O(log n).

public int ExtractMin – Uloží si key prvku z prvního indexu, poslední prvek haldy vymění s prvním a zabublá dolů. Při této akci vždy upravuje pozice jednotlivých prvků v externím poli *pos*. Jakmile dobublá v čase O(log n), vrátí uložený klíč.

public void DecreaseKey – Dostane klíč prvku a hodnotu, na kterou ji má u klíče snížit. Podívá se do pole *pos*. Pokud se v haldě nevyskytuje vrátí se, pokud se zde vyskytuje, zavolá na něj funkci BubbleUp.

public bool Empty – vrací false, pokud je halda neprázdná (má více jak jeden prvek). Jinak true.

public class Algorithms

- Třída pro grafové algoritmy. V konstruktoru dostane matici sousednosti a kolik prvků z ní má použít – tu předělá s respektem k počtu vrcholů a také z ní vytvoří seznam následníků. Obojí se, společně s počtem vrcholů (*n*), uloží.

int[,] GetMatrix

int[][] seznamNasledniku

bool Relax

Tuple<string, int[], int[], List<int>, List<(int, int)>, List<(int, int)>, List<(int, int)>, List<(int, int)>> DFS

public string soloDFS

public string BFS

public Tuple<int[], int[]> Dijkstra

public string Components

public int[] SSK

public int[,] FW

public int[,] Jarnik

public string TopologicalSort