UNIVERZITA PARDUBICE
Fakulta elektrotechniky a informatiky

Grafická aplikace pro vytvoření Hamiltonovského a Eulerovského tahu

Bc. Petra Erlebachová

Diplomová práce

2010

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: **2009/2010**

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Petra ERLEBACHOVÁ**
Studijní program:	**N2646 Informační technologie**
Studijní obor:	**Informační technologie**
Název tématu:	**Grafická aplikace pro vytvoření Hamiltovského a Eulerovského tahu**
Zadávající katedra:	**Katedra softwarových technologií**

Zásady pro vypracování:

V úvodní části práce je nutné provést přehled základních pojmů teorie grafů. Popis vytvoření hamiltovského a Eulerovského tahu na grafech. Popis problému barvení grafů a jeho spojitost teorií grafů. Popis Fleuryho algoritmu, Edmondsova algoritmu a Litlllova algoritmu. Cílem diplomové práce je vytvoření editoru, který umožní grafickou realizaci zadávaného grafu: Editor umožní vypsat jeho vlastnosti - skóre grafu, počet hran, počet vrcholů, zda se jedná o strom, zda jde o Eulerův graf, počet komponent. Průběh jednotlivých algoritmů bude zobrazen po krocích včetně zobrazení dalších hodnot v tabulce. Nedílnou součástí bude i Help a uživatelská příručka.

Rozsah grafických prací:

Rozsah pracovní zprávy:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

*** Demel, J., Grafy a jejich aplikace , Academia 2002 * Nešetřil, J., Teorie grafů , SNTL 1979 * Sedláček, J., Kombinatorika v teorii a praxi , Nakladatelství ČSAV 1964 * Nečas, J., Grafy a jejich použití , Polytechnická knižnice, SNTL 1978**

Vedoucí diplomové práce: **Ing. Soňa Neradová**
Katedra softwarových technologií

Datum zadání diplomové práce: **30. října 2009**

Termín odevzdání diplomové práce: **21. května 2010**

prof. Ing. Simeon Karamazov, Dr.
děkan

L.S.

doc. Ing. Antonín Kavička, Ph.D.
vedoucí katedry

V Pardubicích dne 10. listopadu 2009

Prohlášení autora

Prohlašuji, že jsem tuto práci vypracovala samostatně. Veškeré literární prameny a informace, které jsem v práci využila, jsou uvedeny v seznamu použité literatury.

Byla jsem seznámena s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorský zákon, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Souhlasím s prezenčním zpřístupněním své práce v Univerzitní knihovně.

Ve Vidochově dne 16. 08. 2010 Bc. Petra Erlebachová

Anotace

Teoretická část diplomové práce je zaměřena na základní pojmy z teorie grafů, reprezentaci grafů a vytvoření Hamiltonovského a Eulerovského tahu. Dále obsahuje popis Fleuryho, Edmondsova a Littlova algoritmu a také problematiku barvení grafů.

Implementační část práce je soustředěna na vytvoření editoru, který umožní grafickou realizaci zadávaného grafu. Editor mimo jiné umožňuje vypsat vlastnosti grafu a umí zobrazovat průběh jednotlivých algoritmů po krocích.

Klíčová slova

Eulerovský tah, Hamiltonovský tah, graf, barvení grafů, editor, Littlův algoritmus, Fleuryho algoritmus, Edmondsův algoritmus

Annotation

The theoretical part of the thesis is focuses on the basic concepts of graph theory, graph representation a Hamilton's and Euler's walk. It also contains a description of Fleury, Edmonds and Little algorithm and graph coloring problems. Implementation of the work is focused on creating an editor that allows graphical implementation, entered the chart. Editor, inter alia, to list properties, and can display a chart the course of the algorithm steps.

Keywords

Euler's walk, Hamilton's walk, graph, graph coloring, editor, Littl's algorithm, algorithm Fleury, Edmonds's algorithm

Obsah

Úvod

Cílem diplomové práce je vytvoření grafického editoru, který umožní grafickou realizaci zadávaného grafu. V první části práce jsou vysvětleny základní pojmy z teorie grafů. Na tuto část navazuje kapitola popisující rozdělení grafů podle orientace hran, četnosti hran, souvislosti, ohodnocení, existence kružnice v grafu a planárnosti. V další kapitole jsou vysvětleny ještě další typy grafů. Jsou to grafy pravidelný, úplný, bipartitní, Eulerovský a Hamiltonovský.

Následuje kapitola, v které je popsán Eulerovský tah. Mimo vysvětlení co je to Eulerovský tah a v jakých formách se vyskytuje, tu jsou zmíněny také podmínky, které graf musí splňovat, aby v něm existoval Eulerovský tah. Kapitola obsahuje také Fleuryho a Edmondsonův algoritmus, pomocí nichž je možné Eulerovský tah v grafu splňující dříve popsané podmínky najít. Další kapitolou je Hamiltnovský tah. Zde je popsán Hamiltonovský tah a následně také Littlův algoritmus.

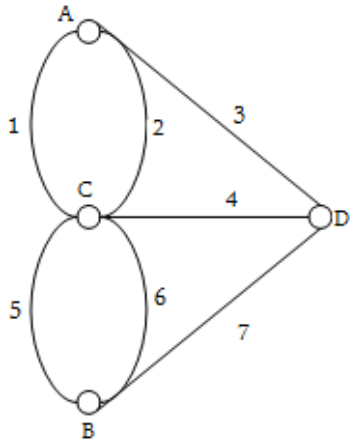
Další kapitola je o barvení grafů. Je v ní ukázáno dělení barvení grafů a popsány tři algoritmy pro barvení. Těmi je sekvenční barvení, paralelní barvení a LDF barvení. Poslední teoretickou kapitolou je kapitola, vysvětlující reprezentaci grafu. V této části je popsána především matice incidence, matice sousednosti a reprezentace datovou strukturou.

Další kapitoly se již věnují praktické části práce. Je zde kapitola zabývající se analýzou v ní jsou mimo jiné shrnuty požadavky na editor, výběr programovacího jazyka a popis datové struktury. V další části jsou popsány jednotlivé třídy, u kterých jsou vypsány a vysvětleny jejich datové složky a metody.

1 Základní pojmy teorie grafů

Teorie grafů je matematická věda zkoumající vlastnosti struktur, které se nazývají graf. První zmínky o grafech se spojují se jménem švýcarského matematika a fyzika Leonharda Eulera (1707 - 1783). Ten se v roce 1736 snažil vyřešit úlohu, která spočívala v projití přes sedm mostů v Königsbergu. Úkolem bylo projít každým z mostů právě jednou a vrátit se do výchozího místa.

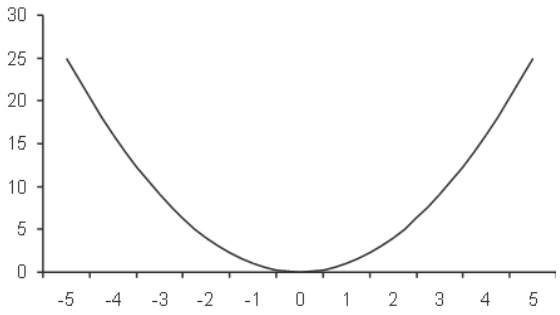
Jak vypadal graf k této úloze je ukázáno na obrázku 1. Čísla označují jednotlivé mosty, písmeny jsou pak označeny pevniny.



Obrázek 1- Graf k úloze sedmi mostů

1.1 Graf

Slovo graf je možné použít v různých významech. Většina lidí si pod pojmem graf představí graf nějaké funkce. Příkladem může být graf na obrázku 2. Grafem je však možné demonstrovat případy z nejrůznějších vědních oborů, techniky a běžného života. Nechá se jím zobrazit prodej výrobků za určité období, silniční síť, vzorec chemické sloučeniny či je pomocí něho možné vyřešit úlohu o převozníkovi, koze, vlkovi a zelí.



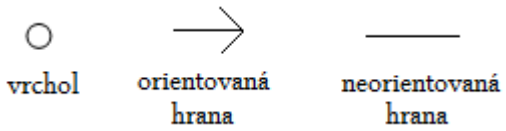
Obrázek 2 - Graf funkce $y=x^2$

V teorii grafů se za graf považuje množina vrcholů *V* a množina hran *E*. Prostým grafem se nazývá dvojice *(V,E)*. Hranou jsou spojeny vždy dva vrcholy, které mohou v určitých případech splynout. Tyto vrcholy se nazývají krajní vrcholy hrany.

1.2 Vrcholy a hrany

Jak již bylo zmíněno výše, graf je tvořen množinami vrcholů a hran. Vrcholy se někdy nazývají také jako uzly a jsou značeny kruhovými symboly. Vrcholy, které jsou spojeny hranou se nazývají sousedy.

Hrany grafu jsou spojnice mezi vrcholy. Vrcholy jsou tedy spojovány hranami a někdy se také říká, že uzly incidují s hranou. Podle toho, je-li graf orientovaný nebo neorientovaný, jsou hrany značeny jako přímka s šipkou – u orientovaného grafu, nebo jen jako přímka – u neorientovaného grafu.



Obrázek 3 - Značení vrcholů a hran

Hrany, které spojují vrchol se sebou samým jsou nazývány smyčkami. Pokud mezi dvěma vrcholy existuje více než jedna hrana, jedná se o multigraf. V opačném případě se hovoří o prostém grafu.

1.3 Sled, cesta a tah

Sled je posloupnost vrcholů a hran. Pokud sled obsahuje pouze jeden vrchol a žádnou hranu, nazývá se triviální sled. Cesta v grafu je sled, ve kterém se žádný vrchol neopakuje. Každý vrchol je tedy použit jen jednou. Sled, ve kterém se neopakují hrany se nazývá tah. Délka sledu je rovna počtu hran ve sledu. V ohodnoceném grafu je pak délka rovna součtu ohodnocených hran obsažených ve sledu. Tahu, který obsahuje všechny hrany grafu se říká Eulerovský tah.

1.4 Cyklus a kružnice

Cyklus nebo-li kružnice je taková cesta v grafu, která začíná a končí ve stejném vrcholu. Nejmenší možnou kružnicí je graf o třech vrcholech. Kružnice, která prochází všemi vrcholy grafu se nazývá Hamiltonovská kružnice.

1.5 Podgraf

Podgraf grafu je graf, který vznikne odebráním některých vrcholů a všech hran, které do odebraného vrcholu nebo z něho vedly, případně odebráním hran z původního grafu. Je to tedy část grafu, která vznikne vybráním podmnožiny hran a k nim všech vrcholů, které tyto hrany spojují.

Pokud z grafu byly odebrány jen hrany, které vedly z nebo do vrcholu, který byl také odebrán, je vzniklý podgraf nazýván indukovaným. Při odebrání i jiné hrany než té z odebraného vrcholu se jedná obecně o podgraf. Podgraf, který má stejný počet vrcholů jako původní graf, je označován jako faktor.

Obrázek 4 - Indukovaný podgraf a podgraf

1.6 Stupeň vrcholu

Stupeň vrcholu v neorientovaném grafu je číslo, které říká, kolik hran z vrcholu vychází. V orientovaném grafu je definován uspořádanou dvojicí, kde první hodnotou je počet hran vstupujících do vrcholu a druhou počet hran z vrcholu vycházejících.

O stupních vrcholu je známo, že

- součet stupňů všech vrcholů se vždy rovná dvojnásobku počtu hran a
- počet vrcholů lichého stupně je vždy sudé číslo.

1.7 Skóre grafu

Skóre grafu je libovolně uspořádaná posloupnost stupňů jeho vrcholů. Skóre je možné považovat za stejné, pokud přerovnáním pořadí jednotlivých vrcholů dostaneme stejná čísla. Pokud dva grafy nemají stejné skóre, můžeme o nich říci, že jsou neisomorfní. Nemůžeme o nich ale tvrdit, že pokud mají stejné skóre, jsou isomorfní.

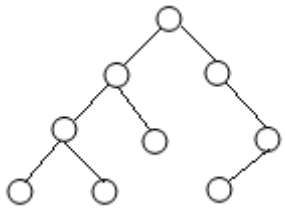
1.8 Komponenta

Komponenta grafu je jeho maximální souvislý podgraf. Maximální znamená, že se jedná o největší podgraf. Každý graf se dá jednoznačně rozložit do souvislých komponent, a tentýž graf je pak jejich disjunktním sjednocením. Souvislý graf má jedinou komponentu a to sebe samou. Komponenty grafu jsou navzájem disjunktní a každý z uzlů patří do některé z komponent.

1.9 Strom, list a les

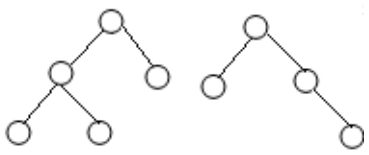
Strom je konečný, souvislý graf, který neobsahuje žádnou kružnici jako podgraf. Pokud z něho tedy odebereme nějakou hranu, stane se z něho nesouvislý graf. Pokud do stromu naopak hranu přidáme, vznikne v něm kružnice a tím pádem přestane být stromem. Speciálním případem stromu je hvězda. Strom se vyznačuje těmito vlastnostmi.

- Existuje v něm právě jeden sled z jednoho uzlu do druhého,
- po vynechání libovolné hrany ze stromu se graf stane nesouvislým,
- ve stromu existují alespoň dva různé vrcholy stupně jedna,
- počet hran ve stromě je o jednu nižší než počet vrcholů.



Obrázek 5 – Strom

Les je neorientovaný graf bez kružnic nebo-li množina navzájem nepropojených stromů. Každý strom, a tedy i každý les obsahuje listy. Těmi jsou vrcholy stupně jedna.



Obrázek 6 - Les

1.10 Kostra grafu

Kostra grafu je minimální souvislý podgraf grafu, který obsahuje všechny jeho vrcholy a libovolné hrany a zároveň neobsahuje žádnou kružnici a je souvislý. Je to tedy libovolný podgraf, který tvoří strom. Graf může obsahovat několik koster. Pokud je však graf stromem obsahuje pouze jednu kostru a tou je sám graf.

1.11 Most, artikulace

Most je hrana, jejímž odstraněním se zvýší počet komponent původního grafu o jedna. Každý koncový vrchol mostu, který má stupeň větší než jedna je artikulací.

Artikulace označuje vrchol v grafu, jehož vypuštěním se zvýší počet komponent původního grafu alespoň o jedna a maximálně o počet stupňů vrcholu snížený o jedna. Artikulací nemůže být vrchol, který má stupeň jedna nebo nula.

2 Rozdělení grafů

Grafy je možné dělit podle:

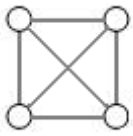
- orientace hran,
- četnosti hran,
- souvislosti,
- ohodnocení,
- existence kružnice v grafu,
- planárnosti.

2.1 Rozdělení grafu podle orientace hran

Graf se nechá rozdělit podle orientace hran na orientovaný a neorientovaný.

2.1.1 Neorientovaný graf

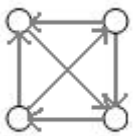
Neorientovaný graf je takový, který je určen množinou vrcholů a množinou hran, což jsou neuspořádané dvojice vrcholů. V neorientovaném grafu jsou hrany vedoucí z vrcholu *A* do vrcholu *B* a z vrcholu *B* do vrcholu *A* hranami totožnými.



Obrázek 7 - Neorientovaný graf

2.1.2 Orientovaný graf

Orientovaný graf je graf obsahující hrany, které jsou tvořeny uspořádanými dvojicemi vrcholů. Tyto hrany se nazývají orientovanými hranami. U orientovaných hran je možné jednoznačně říci, z kterého a do kterého vrcholu hrana vede. V orientovaném grafu jsou hrany vedoucí z vrcholu *A* do vrcholu *B* a z vrcholu *B* do vrcholu *A* rozdílnými hranami.



Obrázek 8 - Orientovaný graf

2.2 Rozdělení grafu podle četnosti hran

Podle četnosti hran se graf nechá rozdělit na graf prostý a na multigraf.

2.2.1 Prostý graf

Prostý graf je takový graf, kde mezi všemi jeho vrcholy vede vždy nejvýše jedna hrana. Je to tedy graf, ve kterém je násobnost všech hran rovna jedné. Graf neobsahuje žádné rovnoběžné hrany. Pokud neobsahuje ani žádné smyčky hovoří se o něm jako o obyčejném grafu.

2.2.2 Multigraf

Multigraf je graf, mezi jehož libovolnými vrcholy existuje více než jedna hrana. V takovém grafu se můžou vyskytovat rovnoběžné hrany, ale ne smyčky. Multigrafem nikdy nemůže být strom.

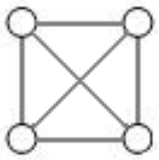
2.3 Rozdělení grafu podle souvislosti

Podle souvislosti je možné graf rozdělit na souvislý a nesouvislý.

2.3.1 Souvislý graf

Souvislý graf je graf, ve kterém se mezi každými dvěma jeho různými vrcholy nechá nalézt cesta. Graf je souvislý, pokud je tvořen právě jednou komponentou. Souvislý graf je možné dále rozdělit na k-hranově souvislý a k-vrcholově souvislý graf.

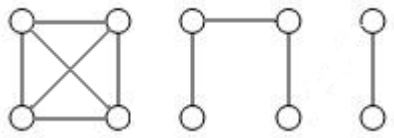
O souvislém grafu je možné říci, že je k-hranově souvislý, pokud po odebrání k-1 hrany zůstane stále souvislý. K-hranově souvislý graf je tedy takový graf, ve kterém je možné vést mezi dvěmi libovolnými vrcholy alespoň *k* hranově disjunktních cest. Obdobně je možné definovat k-vrcholově souvislý graf. To je takový graf, který po odebrání k-1 vrcholu zůstane souvislý. Je to tedy graf, ve kterém je možné vést mezi dvěmi libovolnými vrcholy alespoň *k* disjunktních cest. Příkladem může být úplný graf, který je zobrazen na obrázku 9.



Obrázek 9 - Úplný graf

2.3.2 Nesouvislý graf

Nesouvislý graf je takový, ve kterém se nenechá najít mezi každými dvěmi jeho různými vrcholy cesta. Takový graf je tvořen více než jednou komponentou. Jednotlivé komponenty, kterými je nesouvislý graf tvořen, jsou však souvislé.



Obrázek 10 - Nesouvislý graf

2.4 Rozdělení grafu podle ohodnocení hran

Podle ohodnocení hran je graf možné rozdělit na ohodnocený a neohodnocený. Ohodnocený i neohodnocený graf může být orientovaný nebo neorientovaný a souvislý nebo nesouvislý.

2.4.1 Ohodnocený graf

Ohodnocený graf je takový graf, jenž má k hranám přiřazeny nějaké hodnoty. Hranám, které mají přiřazeny hodnoty se říká ohodnocené hrany. Hodnoty hran můžou reprezentovat například vzdálenost, náklady na cestu nebo čas. Hodnotou může být libovolné kladné číslo. Hodnoty u hran jsou někdy nazývány jako cena hrany nebo délka hrany.

2.4.2 Neohodnocený graf

Neohodnocený graf je graf, v němž mají všechny hrany jednotkovou délku.

2.5 Rozdělení grafu podle existence kružnice

Pokud jsou grafy rozděleny podle toho, zda v nich existuje či neexistuje kružnice, říká se jim grafy acyklické nebo cyklické.

2.5.1 Cyklický graf

Cyklický graf je takový graf, ve kterém existuje alespoň jeden cyklus. Je to tedy graf, jehož podgraf obsahuje kružnici. Takovýmto typem grafů jsou Hamiltonovské grafy.

2.5.2 Acyklický graf

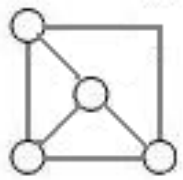
Acyklický graf je graf, jehož podgraf neobsahuje kružnici. Acyklickými grafy jsou všechny typy stromů.

2.6 Rozdělení grafu podle planárnosti

Podle planárnosti se grafy dělí na rovinné a na nerovinné grafy.

2.6.1 Rovinný graf

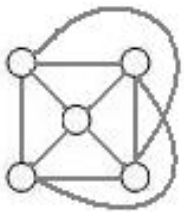
Rovinný graf je takový graf, který se nechá nakreslit v rovině bez křížení hran. Žádné dvě hrany se tedy nesmí protínat. Z toho vyplývá, že hrany nemají žádné společné body mimo vrcholů. Rovinným grafem je například úplný graf se čtyřmi vrcholy, který je zobrazen na obrázku 11. Rovinný graf je někdy nazýván také jako planární.



Obrázek 11 - Rovinný graf

2.6.2 Nerovinný graf

Nerovinný nebo-li neplanární graf je takový graf, který se nedá nakreslit jinak, než že se některé jeho hrany musí křížit.

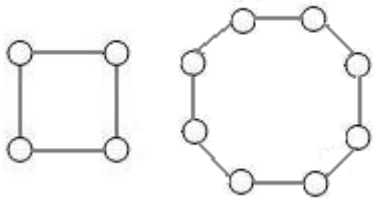


Obrázek 12 - Nerovinný graf

3 Další typy grafů

3.1 Pravidelný graf

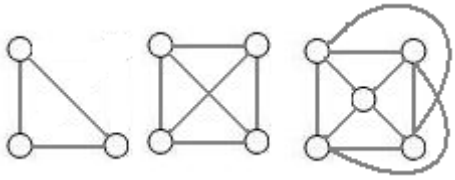
Pravidelný graf je takový graf, který obsahuje vrcholy, které mají všechny stejný stupeň. Pravidelný graf druhého stupně, je kružnice. Je to takový graf, který obsahuje vrcholy stupně dva. Ukázka takovýchto grafů je na obrázku 13.



Obrázek 13 - Pravidelný graf – kružnice

3.2 Úplný graf

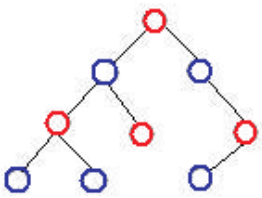
Úplný graf je takový graf, jehož každé dva vrcholy jsou propojeny hranou. Aby mohl být graf úplný musí být také souvislý.



Obrázek 14 - Úplné grafy

3.3 Bipartitní graf

Bipartitní graf je graf, jehož vrcholy je možné rozdělit na dvě disjunktní množiny tak, že žádné dva vrcholy ze stejné množiny nejsou spojeny hranou. Je to tedy takový graf, který má jeden koncový vrchol v jedné množině a druhý koncový vrchol v množině druhé. O grafu je možné říci, že je bipartitní právě tehdy, neobsahuje-li kružnici liché délky. Mezi bipartitní grafy tedy patří také stromy. Na obrázku 15 je bipartitní graf, při čemž vrcholy označené červeně patří do jedné množiny a vrcholy označené modře do druhé množiny.

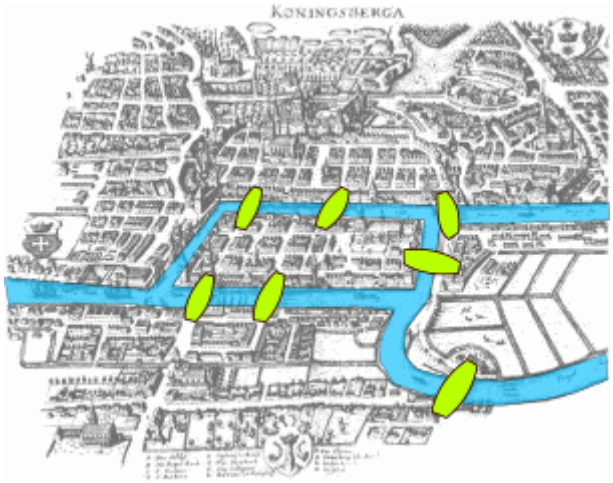


Obrázek 15 - Bipartitní graf

3.4 Eulerovský graf

Eulerovský graf je souvislý graf, který obsahuje uzavřený Eulerovský tah. Uzavřený Eulerovský tah je takový tah, který obsahuje všechny hrany v daném grafu právě jednou a končí ve vrcholu, ve kterém začal.

Eulerovský graf je pojmenován po švýcarském matematikovi Leonhardu Eulerovi, který se v roce 1736 snažil vyřešit úlohu o sedmi mostech v Königsbergu. Úloha byla založena na skutečné události. Městem Königsberg protéká řeka Pergole, která vytváří dva ostrovy, v době Eulera tyto ostrovy a břehy spojovalo sedm mostů tak jak je to naznačeno na obrázku 16. Úkolem bylo projít tyto mosty tak, aby se na každý z nich vstoupilo pouze jednou.



Obrázek 16 - Mapa města Königsberg [13]

Euler si mapu převedl na graf o čtyřech vrcholech a sedmi hranách tak, jako je ukázáno na obrázku 1, a zjistil, že vyřešit takovýto problém, je stejné jako nakreslit obrázek jedním tahem. To ovšem v takto zadaném grafu nelze, a tudíž nelze projít ani sedm těchto mostů tak, aby se na každý vstoupilo pouze jednou.

3.5 Hamiltonovský graf

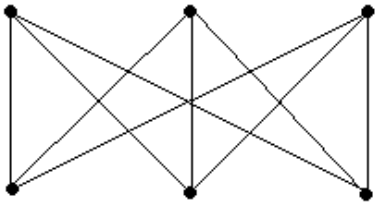
Hamiltonovský graf je souvislý graf, který obsahuje Hamiltonovskou kružnici. Hamiltonovská kružnice je kružnice, která prochází všemi vrcholy daného grafu právě jednou s výjimkou prvního vrcholu, který je zároveň koncovým vrcholem.

Hamiltonovský graf se vztahuje ke jménu irského matematika Williama Rowana Hamiltona. Ten v roce 1857 vymyslel hru, která se skládala z pravidelného 12-ti stěnu a 20-ti kolíků. Každý kolík byl umístěn do jednoho rohu 12-ti stěnu a bylo mu přiděleno jedno jméno některého hlavního města Evropy. Úkolem hráčů pak bylo najít cestu, která procházela přes všechny kolíky a vrátit se zpět do původního kolíku ze kterého vyšli.

Úkolem tedy bylo najít kružnici, která později získala přídavné jméno Hamiltonovská. Pro vytvoření Hamiltonovské kružnice není zatím znám žádný algoritmus, proto není lehké rozhodnout ani to, zda-li je graf Hamiltonovský. Neexistuje dosud totiž žádná jednoduchá nutná a postačující podmínka, která by řekla, jestli je graf Hamiltonovský. Existuje ale několik postačujících podmínek k hamiltonovskosti grafu.

- **Diracova podmínka** – každý vrchol grafu musí mít stupeň vrcholu rovný alespoň jedné polovině počtu vrcholů obsažených v grafu,
- **Oreho podmínka** – součet stupňů vrcholů, které nejsou spojeny hranou je roven alespoň počtu vrcholů v grafu,
- **Posova podmínka** – pro každé přirozené číslo k, které je menší než jedna polovina vrcholů obsažených v grafu, jejichž stupeň nepřevyšuje k, je menší než k.

Tyto podmínky jsou mezi sebou v určitém vztahu. Pokud je splněna Diracova podmínka, je automaticky splněna také podmínka Oreho a Posova. Při splnění Oreho podmínky je splněna také Posova podmínka, ale není automaticky splněna podmínka Diracova. Při splnění Posovy podmínky není zaručeno splnění Diracovy a Oreho podmínky.



Obrázek 17 - Hamiltonovský graf [14]

Nutné ale ne postačujícími podmínky pro nalezení Hamiltonovské kružnice.

- Graf musí být souvislý a obyčejný,
- musí obsahovat alespoň tři vrcholy,
- nesmí obsahovat mosty ani artikulace,
- každý vrchol musí mít stupeň rovný alespoň dvěma.

Dále je uvedeno několik pravidel pro hledání Hamiltonovské kružnice.

- Hamiltonovská kružnice má právě tolik hran, kolik vrcholů má graf, ve kterém je kružnice hledána.
- Pokud má daný vrchol stupeň *k*, pak Hamiltonovská kružnice musí obsahovat právě dvě hrany, které jsou incidentní s tímto vrcholem.
- Při konstrukci Hamiltonovské kružnice nemůže být vytvořena kružnice, která by neobsahovala všechny vrcholy grafu, v kterém je kružnice hledána.
- Pokud Hamiltonovská kružnice, která je konstruována, prochází daným vrcholem, pak je možné ostatní nepoužité hrany incidentní s tímto vrcholem vyloučit.

4 Eulerovský tah

Jak již bylo řečeno výše, Eulerovský tah zavedl švýcarský matematik Leonhard Euler, který se v roce 1736 snažil vyřešit úlohu o sedmi mostech v Königsbergu. Jedná se o tah, který prochází každou hranou v grafu právě jednou.

Eulerovský tah může být

- otevřený nebo
- uzavřený.

Otevřený Eulerovský tah je takový tah, který obsahuje všechny hrany grafu a začíná v jiném vrcholu, než v kterém končí.

Uzavřený Eulerovský tah je pak takový tah, který stejně jako otevřený Eulerovský tah obsahuje všechny hrany grafu, ale zároveň začíná i končí ve stejném vrcholu.

Aby bylo možné najít v grafu uzavřený Eulerovský tah, je potřeba, aby byly splněny následující podmínky.

- Graf musí být souvislý,
- pokud je graf neorientovaný, musí být stupeň všech vrcholů sudý,
- v orientovaném grafu pak musí být počet hran vstupujících do vrcholu rovný počtu hran z vrcholu vystupujících.

Pro vytvoření otevřeného Eulerovského tahu je nutné zajistit tyto podmínky.

- Graf musí být souvislý,
- v neorientovaném grafu musí obsahovat právě dva vrcholy lichého stupně a ostatní vrcholy stupně sudého,
- v orientovaném grafu musí existovat vrchol jehož počet vstupních hran se rovná počtu výstupních hran plus jedna a zároveň v něm musí existovat vrchol, jehož počet výstupních hran se rovná počtu vstupních hran mínus jedna. Ostatní vrcholy musejí mít počet vstupních hran rovný počtu výstupních hran.

4.1 Fleuryho algoritmus

Pro vytvoření uzavřeného Eulerovského tahu se používá Fleuryho algoritmus. Podmínkou pro sestavení algoritmu je, že každý stupeň musí být sudého stupně.

Fleuryho algoritmus

- **Krok 1** – Z libovolného vrcholu najdeme hranu z něho vycházející. Tu přidáme do tahu.
- **Krok 2** - Pokud už jsou v tahu zařazeny všechny hrany, skončí prohledávání.
- **Krok 3** – Do tahu zařadíme takovou hranu vycházející z posledního vrcholu, aby se jejím odebráním podgraf, který je tvořen z nevybraných hran, nerozpadl na dvě komponenty a nebo nezůstal prázdný a izolovaný začátek tahu. Pak se vrátíme na krok 2.

Fleuryho algoritmus je po menší modifikaci možné použít i pro otevřený Eulerovský tah. V takovém případě, je nutné začít v některém z vrcholu s lichým stupněm.

4.2 Edmondsův algoritmus

Edmondsův algoritmus se používá pro konstrukci uzavřeného Eulerovského sledu v grafu se sudým počtem vrcholů lichého stupně. V Eulerovském sledu se nevyskytuje každá hrana pouze jednou, ale některé hrany se v něm vyskytují dvakrát. Pokud je graf hranově ohodnocený, hovoří se o minimálním nebo o maximálním Eulerovském sledu.

Edmondsův algoritmus

- **Krok 1** – Určíme vrcholy lichého stupně v počtu 2krát počet dvojic lichého stupně. Při čemž počet dvojic lichého stupně musí být vetší nebo rovný jedné.
- **Krok 2** – Sestrojíme kompletní graf, jehož vrcholy jsou vrcholy lichého stupně původního grafu.
- **Krok 3** – Hrany kompletního grafu ohodnotíme vzdáleností příslušných vrcholů původního grafu.
- **Krok 4** – Určíme párování minimální délky.

- **Krok 5** – Hrany minimálního párování přidáme do původního grafu mezi příslušné vrcholy. Tím vznikne graf, který je Eulerovským grafem.

- **Krok 6** – V grafu, který vznikl v předešlém kroku sestrojíme uzavřený Eulerovský tah pomocí Fleuryho algoritmu. Tento tah je Eulerovským tahem minimální délky.

- **Krok 7** – V Eulerovském tahu nahradíme každou hranu párování odpovídající cestou minimální délky. Tím dostaneme sled, který je uzavřeným Eulerovským sledem pokrývající hrany grafu minimální délky.

5 Hamiltonovský tah

Hamiltonovský tah je tah, který obsahuje všechny vrcholy grafu, v kterém je Hamiltonovský tah hledán, právě jednou. Tento tah obsahuje každá Hamiltonovská kružnice v jejím případě je však výchozí vrchol zároveň vrcholem konečným.

5.1 Littlův algoritmus

Littlův algoritmus je algoritmus založený na metodě větví a hranic. Princip metody větví a hranic je založen na dělení množiny přípustných řešení na menší podmnožiny a výpočtu horního, resp. dolního odhadu hodnot účelové funkce na všech řešeních jednotlivých podmnožin.[3]

Odhad účelové funkce se během výpočtu používá pro vyloučení podmnožin, které nemohou obsahovat optimální řešení a pro určení nejnadějnější podmnožiny k dalšímu dělení, tedy k podmnožiny, ve které se předpokládá optimální řešení.

Algoritmy založené na metodě větví a hranic používají zpravidla jednoho ze dvou následujících způsobů vytváření a prohledávání stromu podmnožin přípustný řešení.

- Prohledávání do hloubky s případným použitím zpětného návratu k nejbližšímu vrcholu,
- usměrněného prohledávání.

Littlův algoritmus je založený na prvním z těchto způsobů a to na principu prohledávání do hloubky s případným použitím zpětného návratu k nejbližšímu vrcholu.

Littlův algoritmus

- **Krok 1** - V každém řádku matice vzdáleností vyhledáme minimální prvek a ten odečteme od všech prvků v uvedeném řádku.
- **Krok 2 -** V každém sloupci matice vzdáleností, v němž se nenachází žádná nula, vyhledáme minimální prvek a ten odečteme od všech prvků v příslušném sloupci.
- **Krok 3 -** Vypočítáme dolní odhad účelové funkce jako součet všech hodnot, o které jsme snižovali vzdálenosti v příslušných řádcích, resp. sloupcích. Dolní odhad účelové funkce ohraničuje hodnotu účelové funkce zdola, udává tedy hodnotu, pod kterou hodnota účelové funkce určitě neklesne.

- **Krok 4 -** Každou nulu nacházející se v matici vzdáleností ohodnotíme a to tak, že v příslušném řádku a sloupci vyhledáme minimální ze zbylých prvků a obě hodnoty sečteme. Pokud nelze ohodnotit nuly, pokračujeme krokem 10. Ohodnocení nuly představuje hodnotu, o kterou vzroste dolní odhad účelové funkce v případě, že uvedený prvek nebude vybrán, což odpovídá situaci, kdy příslušná relace nebude do Hamiltovnovy kružnice zařazena.

- **Krok 5 -** Z ohodnocených prvků (nul) vybereme prvek s maximálním ohodnocením. Máme - li v aktuální matici více prvků se stejným maximálním ohodnocením, volíme libovolný z nich. Výběr prvku znamená zařazení odpovídajícího úseku do vznikající Hamiltonovy kružnice, současně s tím začínáme pěstovat strom řešení (pokračujeme ve stromu řešení).

- **Krok 6 -** V aktuální řešící matici zrušíme ohodnocení zbylých nul.

- **Krok 7 -** Výběrem prvku se umožní redukce aktuální řešící matice o příslušný řádek a sloupec. Redukování aktuální řešící matice se provádí za účelem zpřehlednění výpočtu.

- **Krok 8 -** Provedeme zákaz prvků umožňujících vznik nepřípustného řešení (jedná se o prvky, které v důsledku prvku vybraného v kroku 5 umožňují uzavřít Hamiltonovu kružnici dříve, než dojde k návštěvě všech dosud nenavštívených uzlů).

- **Krok 9 -** Kontrola prvků v aktuální řešící matici (v rámci tohoto kroku se provádí kontrola, zda v každém řádku a sloupci po redukci ve smyslu kroku 6 zůstala alespoň jedna nula, pokud ne, postupujeme analogicky jako v kroku 1, resp. kroku 2. V případě dalšího odečítání se zároveň o stejnou hodnotu zvyšuje dolní odhad účelové funkce v příslušné větvi). Máme - li k dispozici hodnotu určitého řešení, porovnáme dosažené hodnoty dolních odhadů účelové funkce s tímto řešením. Dosáhne - li hodnota dolního odhadu účelové funkce hodnoty ve všech větvích již vytvořeného řešení, pokračujeme krokem 11, jinak se vracíme na krok 4.

- **Krok 10 -** Zbylé prvky v aktuální řešící matici určují úseky uzavírající Hamiltonovu kružnici, vyskytuje - li se v úloze větev s nižší hodnotou dolního odhadu účelové funkce než je hodnota dolního odhadu naposled vytvořeného řešení, vytvoříme novou výchozí řešící matici zohledňující podmínky v příslušné větvi řešícího stromu (zákazy některých prvků apod.) a vracíme se na krok 1.

- **Krok 11 -** Naposledy vytvořené řešení je optimálním řešením. [3]

6 Barvení grafů

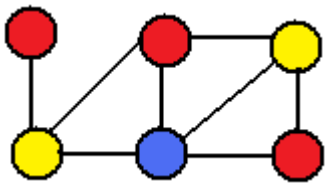
Barvení grafů je jednou z disciplín teorie grafů. Tato disciplína se zabývá přiřazováním barev různým objektům v grafu tak, aby žádné dva objekty, které spolu sousedí, neměly stejnou barvu. Barva je ve většině případů reprezentována číslem, ale je možné ji reprezentovat jakýmkoliv prvkem z libovolné množiny. Barvení grafů minimálním počtem barev patří k NP - úplným problémům.

Barvení grafů se dá rozdělit na

- vrcholové barvení grafů a
- hranové barvení grafů.

6.1 Vrcholové barvení grafu

Vrcholové barvení grafu je takové, které přiřadí každému vrcholu takovou barvu, aby žádné dva vrcholy, které jsou spojeny hranou neměly stejnou barvu. Nejmenší počet barev, kterými se nechají vrcholy v zadaném grafu obarvit, se pak nazývá barevností grafu nebo také chromatickým číslem grafu.

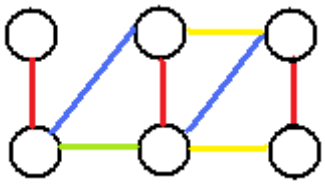


Obrázek 18 - Vrcholové barvení grafu

Většina úloh řešená pomocí barvení grafů se nechá převést na vrcholové barvení grafů a proto jsou dále uvedené algoritmy uvedeny pouze pro vrcholové barvení.

6.2 Hranové barvení grafu

Hranové barvení grafu je takové barvení, kdy žádné dvě hrany, které mají společný vrchol, nemají stejnou barvu. Nejmenší počet barev, kterými se nechají hrany v zadaném grafu obarvit, se pak nazývá chromatický index grafu.



Obrázek 19 - Hranové barvení grafu

6.3 Heuristiky pro barvení grafu

Jak již bylo zmíněno výše, barvení grafů minimálním počtem barev patří k NP - úplným problémům a proto je vhodné pro úlohy větších rozměrů použít heuristiky. Dále uvedené algoritmy jsou jednoduché a nemají žádné opravné kroky. Pokud je vrcholu jednou přidělena barva, je toto přidělení definitivní.

6.3.1 Sekvenční barvení grafu

Algoritmus pro sekvenční barvení grafu sice nedává optimální řešení, ale je vhodné ho použít pro rychlé zjištění horního odhadu chromatického čísla grafu.

- **Krok 1** – Vezmeme dosud neobarvený vrchol grafu a přiřadíme mu nejmenší možné číslo barvy tak, aby žádný obarvený sousední vrchol, neměl stejnou barvu jako právě obarvovaný vrchol.

- **Krok 2** – Pokud už jsou obarveny všechny vrcholy, tak skončíme s obarvováním. Jinak pokračujeme na krok 1.

6.3.2 Paralelní barvení grafu

Algoritmus pro paralelní barvení grafu pracuje tak, že nejdříve vezme jednu barvu a tou obarví co největší možné množství vrcholů. Pak vezme další barvu a obarví s ní další vrcholy, tak pokračuje dokud nejsou obarveny všechny vrcholy. Tento algoritmus je založený na domněnce, že nejdříve je potřeba obarvit vrcholy s největšími stupni.

- **Krok 1 –** Seřadíme vrcholy grafu do klesající posloupnosti podle stupňů vrcholů a číslem inicializujeme barvu, kterou budeme vrcholy barvit.
- **Krok 2 –** Postupně procházíme vrcholy posloupnosti a vrchol, který nemá obarveného souseda stejnou barvou, kterou právě barvíme, obarvíme.
- **Krok 3 –** Pokud jsou obarveny všechny vrcholy tak konec.
- **Krok 4 –** Zvýšíme číslo barvy o jedna a pokračujeme na krok 2.

6.3.3 Barvení grafu LDF (Largest Degrese First)

U tohoto algoritmu se definuje barevný stupeň vrcholu jako počet barev, kterými jsou obarveny sousední vrcholy obarvovaného vrcholu. Tento algoritmus je podobný sekvenčnímu algoritmu.

- **Krok 1 –** Ze všech neobarvených vrcholů s nejvyšším stupněm vybereme ten, který má největší barevný stupeň.
- **Krok 2 –** Vrcholu přiřadíme barvu nejnižšího možného čísla.
- **Krok 3 –** Pokud jsou všechny vrcholy obarvené tak skončíme. Jinak jdeme na krok 1.

7 Reprezentace grafu

Možností jak popsat strukturu grafu je několik a nelze jednoznačně říci, která z nich je nejlepší, protože každá se hodí pro jiné grafy a jiné účely. Grafy je možné popsat pomocí

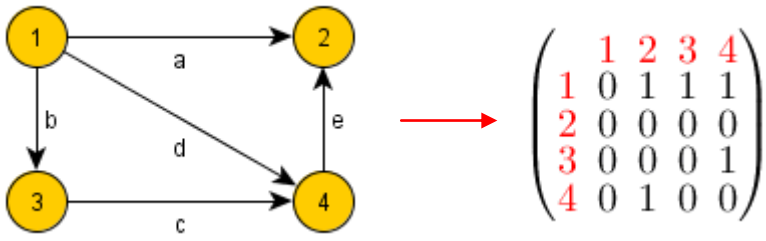
- diagramu,
- definice,
- matice a
- datové struktury.

V informatice se nejčastěji používá popis grafu za pomoci matic a datových struktur. Právě proto budou tyto způsoby dále popsány.

7.1 Reprezentace maticí sousednosti

Reprezentace maticí sousednosti se používá především pro husté grafy. Je to čtvercová matice o velikosti počtu vrcholů, ve které je souřadnice daná řádkem *m* a sloupcem *n* jednotková právě tehdy, pokud existuje hrana z vrcholu *m* do vrcholu *n*. Tuto matici je možné použít pro orientované i neorientované grafy. Přičemž pro neorientované grafy platí, že matice je symetrická podle hlavní diagonály. To vyplývá z definice neorientovaného grafu, která říká že pokud existuje hrana z vrcholu *m* do vrcholu *n*, pak také existuje hrana z vrcholu *n* do vrcholu *m*. Tato skutečnost, nám umožňuje snížit paměťové nároky na polovinu. Matici sousednosti ale není možné použít pro multigrafy.

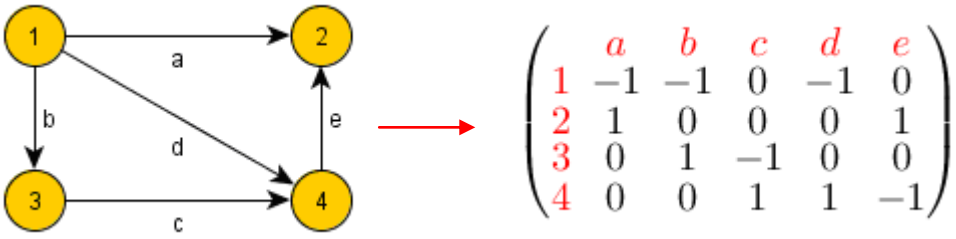
Následující obrázek ukazuje, jak vypadá matice sousednosti pro zadaný orientovaný graf.



Obrázek 20 - Matice sousednosti

7.2 Reprezentace maticí incidence

Matice incidence je taková matice, jejíž řádky reprezentují jednotlivé vrcholy grafu a sloupce jednotlivé hrany grafu. Pokud vede z vrcholu *m* hrana *n*, je tato skutečnost reprezentována hodnotou -1. V případě, že hrana *n* vede do vrcholu *m*, je souřadnice daná řádkem *m* a sloupcem *n* reprezentována hodnotou 1. Neinciduje-li vrchol *m* s hranou *n* je hodnota nastavena na 0. Neorientované grafy mají u obou vrcholů hrany hodnotu 1.



$$\begin{pmatrix} & a & b & c & d & e \\ 1 & -1 & -1 & 0 & -1 & 0 \\ 2 & 1 & 0 & 0 & 0 & 1 \\ 3 & 0 & 1 & -1 & 0 & 0 \\ 4 & 0 & 0 & 1 & 1 & -1 \end{pmatrix}$$

Obrázek 21 - Matice incidence

7.3 Reprezentace datovou strukturou

Ukládat graf, který má velký počet vrcholů, pomocí matice sousednosti není příliš efektivní. Efektivní není ani procházení takto uloženého grafu. Pro zefektivnění je lepší používat graf, který je reprezentovaný pomocí dvou polí. První pole se skládá z prvků, jejichž počet odpovídají počtu vrcholů v grafu. V nich jsou uloženy hodnoty indexů, od kterých v druhém poli začíná seznam sousedů, které s tímto vrcholem sousedí. Druhým způsobem, jak efektivně reprezentovat graf je požití dynamické datové struktury. V tomto případě, je každý vrchol reprezentován jako datový typ, který obsahuje seznam ukazatelů na sousední vrcholy. Tento způsob reprezentace je vhodné použít pro grafy, jejichž počet vrcholů a hran se v čase často mění.

8 Analýza

Pro vytvoření editoru, který umožní grafickou realizaci zadávaného grafu bylo zapotřebí zvolit vhodný způsob zadávání grafu. Tím byl, mimo načítání již uložených grafů, zvolen způsob „naklikávání“ grafu, kde uživatel pomocí myši zadá vrcholy a hrany, z kterých se graf bude skládat.

Dále bylo nutné zvolit vhodné algoritmy pro hledání Eulerovského a Hamiltonovského tahu. Pro hledání Eulerovského tahu byl jako vhodný algoritmus vybrán Fleuryho algoritmus, který byl lehce modifikován, aby pomocí něho mohl být nalezen také otevřený Eulerovský tah.

Pro hledání Hamiltonovského tahu nebyl nalezen žádný vhodný algoritmus, a proto byl sestaven vlastní algoritmus, pomocí něhož je možné najít Hamiltonovský tah. Bohužel hledání Hamiltonovského tahu patří k NP-úplným problémům a tudíž, se Hamiltonovský tah nepovede najít vždy. Při hledání Hamiltonovského tahu občas dojde k zacyklení algoritmu a proto je zapotřebí při jeho spouštění zadat počet kroků, které se mají vykonat.

8.1 Požadavky

Dle zadání diplomové práce, zadanými požadavky na editor byly

- grafická realizace zadávaného grafu,
- výpis vlastností grafu,
- zobrazování průběhu algoritmů po krocích včetně zobrazení dalších hodnot v tabulce.

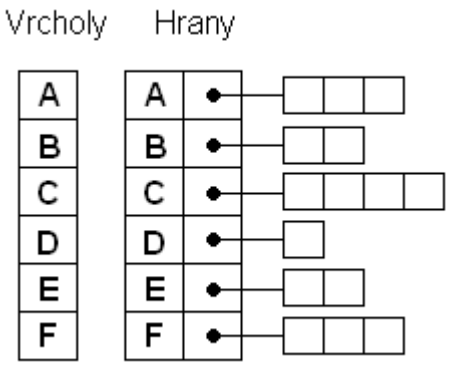
8.2 Programovací jazyk

K programování byl zvolen objektově orientovaný programovací jazyk C#, který zaručuje dostatečně rychlé vykonávání příkazů a poskytuje vhodné knihovny pro tvorbu uživatelského prostředí a následnou grafickou prezentaci. Použito bylo vývojové prostředí Microsoft Visual Studio 2008.

8.3 Datové struktury

Graf je tvořen z vrcholů a hran. Strukturou pro uložení vrcholů grafu je *pole*, pro které je využita generická kolekce *List<T>*.

Pro hrany je pak použita struktura *Tabulka(pole) – pole*, kde *Tabulka(pole)* je reprezentována generickou kolekcí *Dictionary<TKey, TValue>* a pro *TValue* je využita generická kolekce *List<T>*, která je opět implementována na *poli*.

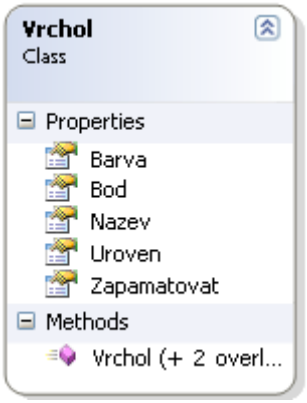


Obrázek 22 - Datová struktura vrcholů a hran

9 Implementace jednotlivých tříd

9.1 Třída Vrchol

Třída slouží pro reprezentaci vrcholu a neobsahuje žádné metody. Základní reprezentace vrcholu je pomocí bodu, který udává jeho souřadnice a názvu. Vrcholu je možné přiřadit také další vlastnosti jako je jeho barva, úroveň ve které se nachází při hledání Hamiltonovského tahu a to, máme-li si vrchol dále pamatovat jako navštívený.



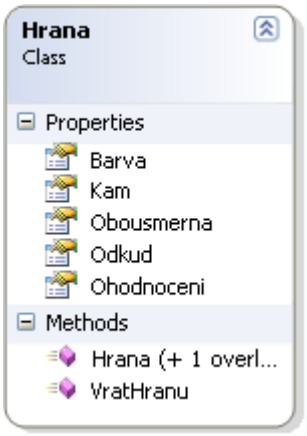
Obrázek 23 - Diagram třídy Vrchol

9.1.1 Datové složky třídy Vrchol

- *string Nazev* – vlastnost nastavující a poskytující název vrcholu.
- *Point Bod* – vlastnost nastavující a poskytující souřadnice vrcholu.
- *Color Barva* – vlastnost nastavující a poskytující barvu vrcholu.
- *bool Zapamatovat* – vlastnost nastavující a poskytující informaci o tom, máme-li si vrchol dále pamatovat.
- *int Uroven* – vlastnost nastavující a poskytující informaci o tom, v které úrovni prohledávání se vrchol nachází.

9.2 Třída Hrana

Třída *Hrana* slouží pro reprezentaci hrany grafu. Hrana se skládá z vrcholu, z kterého vychází, z vrcholu do kterého vede a dále pak z jejího ohodnocení a informace o tom, je-li obousměrná. Je jí ovšem možné přiřadit také informaci o její barvě. Třída *Hrana* obsahuje jednu metodu, a to metodu poskytující názvy vrcholů, z kterého a do kterého hrana vede. Názvy vrcholů jsou odděleny pomlčkou.



Obrázek 24 - Diagram třídy Hrana

9.2.1 Datové složky třídy Hrana

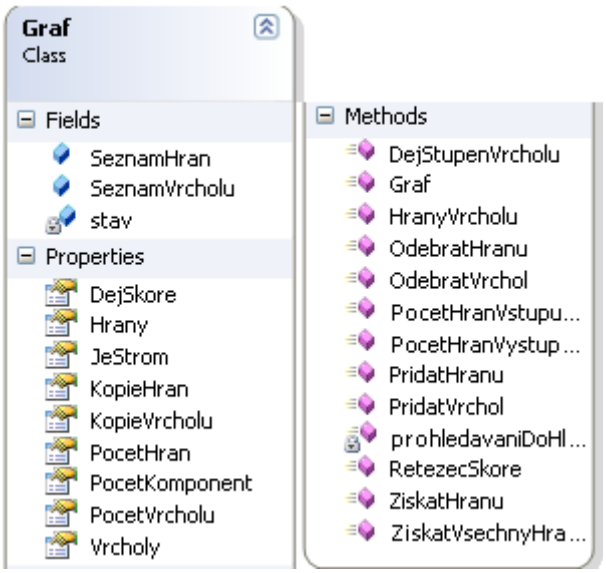
- *Vrchol Odkud* – vlastnost nastavující a poskytující vrchol, z kterého vede hrana.
- *Vrchol Kam* – vlastnost nastavující a poskytující vrchol, do kterého vede hrana.
- *double Ohodnoceni* – vlastnost nastavující a poskytující ohodnocení hrany.
- *bool Obousmerna* – vlastnost nastavující a poskytující údaj, zda je hrana obousměrná.

9.2.2 Metody třídy Hrana

- *string VratHranu()* – vrací řetězec obsahující názvy vrcholu z kterého a do kterého vede hrana. Názvy jsou oddělené pomlčkou.

9.3 Třída Graf

Třída *Graf* slouží pro práci s grafem. Mimo metod pro přidávání a odebírání hran a vrcholů obsahuje metody, sloužící pro zjišťování vstupních a výstupních hran zadaného vrcholu, skóre grafu a další metody pro manipulaci s grafem.



Obrázek 25 - Diagram třídy Graf

9.3.1 Datové složky třídy Graf

- *List<Vrchol> SeznamVrcholu* – seznam vrcholů z nichž se graf skládá.
- *Dictionary<Vrchol, List<Hrana>> SeznamHran* – seznam hran, které se nacházejí v grafu.
- *Dictionary<Vrchol, string> stav* – stav, ve kterém se nachází vrchol při prohledávání do hloubky.
- *List<Vrchol> KopieVrcholu* – vlastnost poskytující kopii seznamu vrcholů grafu.
- *Dictionary<Vrchol, List<Hrana>> KopieHran* – vlastnost poskytující kopii seznamu hran grafu.
- *int PocetVrcholu* – vlastnost poskytující počet vrcholů, z kterých je graf vytvořen.
- *int PocetHran* – vlastnost poskytující počet hran vyskytujících se v grafu.

- *IEnumerable<Vrchol> Vrcholy* - vlastnost poskytující vrcholy grafu.
- *IEnumerable<Hrana> Hrany* - vlastnost poskytující hrany grafu.
- *Dictionary<Vrchol, int> DejSkore* – vlastnost poskytující skóre grafu.
- *int PocetKomponent* – vlastnost poskytující počet komponent grafu.
- *bool JeStrom* – vlastnost poskytující informaci o tom, je-li graf stromem. Pokud graf je stromem vrací *true* v opačném případě vrací *false*.

9.3.2 Metody třídy Graf

- *void PridatVrchol(Vrchol vrchol)* – přidá vrchol, který je parametrem této metody do seznamu vrcholů.
- *void PridatHranu(Hrana hrana)* – přidá hranu, která je parametrem této metody do seznamu hran.
- *void OdebratVrchol(Vrchol vrchol)* – odebere zadaný vrchol ze seznamu vrcholů.
- *void OdebrattHranu(Hrana hrana)* – odebere hranu, která je parametrem této metody ze seznamu hran.
- *Hrana ZiskatHranu(Vrchol odkud, Vrchol kam)* – vrátí hranu vedoucí ze zadaného a do zadaného vrcholu, pokud hrana neexistuje, vrací *null.*
- *IEnumerable<Hrana> HranyVrchol(Vrchol vrchol)* – vrací hrany vedoucí ze zadaného vrcholu.
- *IEnumerable<Hrana> ZiskatVsechnyHranyVrcholu (Vrchol vrchol)* – vrací hrany, které vedou do nebo ze zadaného vrcholu.
- *int DejStupenVrcholu(Vrchol vrchol)* – vrací stupeň zadaného vrcholu.
- *int PocetHranVstupujicich(Vrchol vrchol)* – vrací počet hran vstupujících do zadaného vrcholu.
- *int PocetHranVystupujicich(Vrchol vrchol)* – vrací počet hran vystupujících ze zadaného vrcholu.

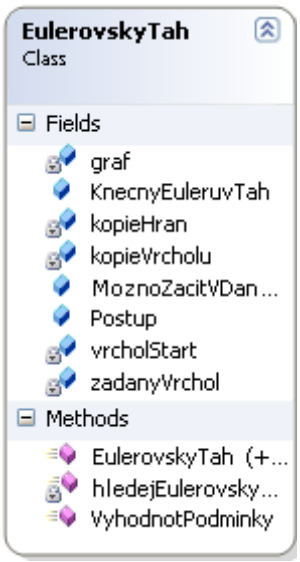
- *string RetezecSkore()* – převede skóre na řetězec, který začíná a končí složenou závorkou, jednotlivé stupně jsou odděleny čárkou.

- *void prohledavaniDoHloubky(Vrchol vrchol)* – od zadaného vrcholu prohledává graf do hloubky .

9.4 Třída EulerovskyTah

Třída EulerovskyTah slouží k hledání Eulerovského tahu. Třída mimo jiné obsahuje metodu *VyhodnotPodminky()*, která vyhodnocuje podmínky, na základě kterých je rozhodnuto, zda je graf, v kterém se má hledat Eulerovský tah, Eulerovský. Tato metoda také zjišťuje, zda-li je možné začít Eulerovský tah v zadaném vrcholu či nikoliv.

Samotné hledání Eulerovského tahu probíhá v metodě *hledejEulerovskyTah*().



Obrázek 26 - Diagram třídy EulerovskyTah

9.4.1 Datové složky třídy EulerovskyTah

- *List<Hrana> KnecnyEuleruvTah* - seznam obsahující vytvořený Eulerovský tah.

- *Dictionary<Vrchol, List<Hrana>> kopieHran* - kopie seznamu hran grafu.

- *List<Vrchol> kopieVrcholu* – kopie seznamu vrcholu grafu.

- *List<Hrana> Postup* – seznam hran, kterými prochází Eulerovský tah.

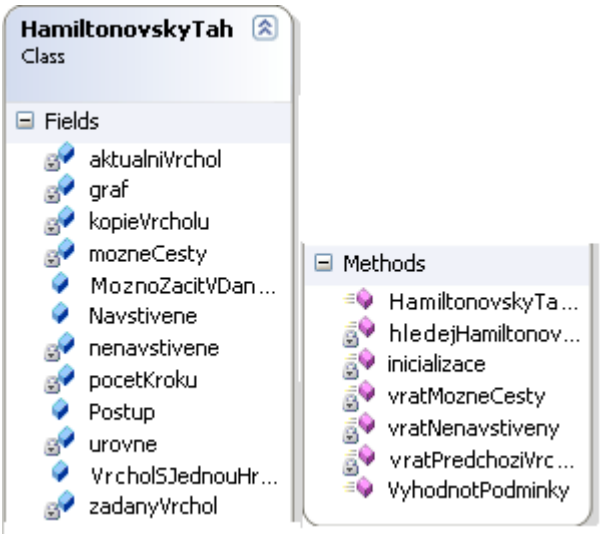
- *Vrchol zadanyVrchol* – vrchol, v kterém má začít Eulerovský tah.
- *Graf graf* – graf, v kterém je hledán Eulerovský tah.
- *Vrchol vrcholStart* – vrchol, z kterého vychází hrana.
- *bool MoznoZacitVDanemVrcholu* – hodnota udávající, zda-li je možné začít v zadaném vrcholu, pokud ano, je nastavena na *true* v opačném případě je nastavena na *false.*

9.4.2 Metody třídy EulerovskyTah

- *void hledejEulerovskyTah()* – hledá Eulerovský tah.
- *bool VyhodnotPodminky*() – vyhodnocuje podmínky, zda-li je možné vytvořit Eulerovský tah. Pokud ano, vrací *true*, pokud ne vrací *false*.

9.5 Třída HamiltonovskyTah

Třída slouží k hledání Hamiltonovského tahu. Mimo metod pro hledání možných a nenavštívených vrcholů obsahuje další metody mezi které patří také metoda *VyhodnotPodminky()*. Tato metoda vyhodnocuje podmínky, které je nutné splnit, pro existenci Hamiltonovského tahu. Dále zjišťuje, zda-li je možné začít hledat Hamiltonovský tah v zadaném vrcholu. Samotné hledání Hamiltonovského tahu pak probíhá v metodě *hledejHamiltonovskyTah*().



Obrázek 27 - Diagram třídy HamiltonovskyTah

9.5.1 Datové složky třídy HamiltonovskyTah

- *List<Vrchol> kopieVrcholu* – kopie seznamu vrcholů grafu.
- *Graf graf* – graf, v kterém se Hamiltonovský tah hledá.
- *List<Vrchol> nenavstivene* – seznam nenavštívených vrcholů.
- *List<Vrchol> Navstivene* – seznam navštívených vrcholů. Pokud se povede najít Hamiltonovský tah, reprezentuje ho tento seznam.
- *List<Vrchol> urovne* – seznam vrcholů, udávající, jaký vrchol, se vyskytuje v které úrovni prohledávání.
- *List<Vrchol> mozneCesty* – seznam vrcholů, do kterých je možné vstoupit.
- *Vrchol zadanyVrchol* – vrchol, v kterém má začít hledání Hamiltonovského tahu.
- *Vrchol aktualniVrchol* – vrchol, v kterém se právě nacházíme.
- *bool MoznoZacitVDanemVrcholu* – proměnná, udávající, zda-li je možné začít prohledávání v zadaném vrcholu.
- *int pocetKroku* – počet kroků, během kterých se pokusíme najít Hamiltonovský tah.
- *List<Vrchol> Postup* – seznam vrcholů, které byly navštíveny během hledání Hamiltonovského tahu.

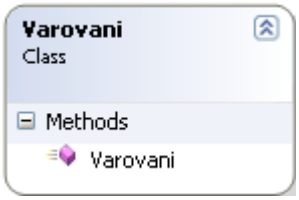
9.5.2 Metody třídy HamiltonovskyTah

- *void hledejHamiltonovskyTah()* – hledá Hamiltonovský tah.
- *Vrchol vratNenavstiveny(List<Vrchol> mozneCesty, Vrchol aktualniVrchol)* - vrací ještě nenavštívený vrchol z těch, do kterých je možné jít z aktuálního vrcholu.
- *Vrchol vratPredchoziVrchol(Vrchol aktualniVrchol)* – vrací předchozí vrchol, který jsme navštívili před vstupem do vrcholu, v kterém se nyní nacházíme.

- *List<Vrchol> vratMozneCesty(Vrchol aktualniVrchol)* - vrací seznam vrcholů, do kterých je možné jít z právě navštíveného vrcholu.
- *void inicializace()* – naplní seznam nenavštívených vrcholů.
- *bool VyhodnotPodminky()* – vyhodnocuje podmínky, zda-li v grafu existuje Hamiltonovský tah.

9.6 Třída Varovani

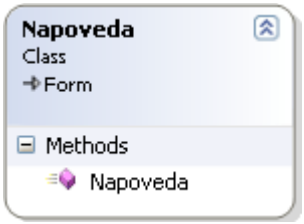
Tato třída slouží pro zobrazování různých hlášení. Vyvolává okno se zadaným textem, titulkem *Pozor*, tlačítkem *OK* a ikonou varování. Neobsahuje žádné datové složky ani metody.



Obrázek 28 - Diagram třídy Varovani

9.7 Třída Napoveda: Form

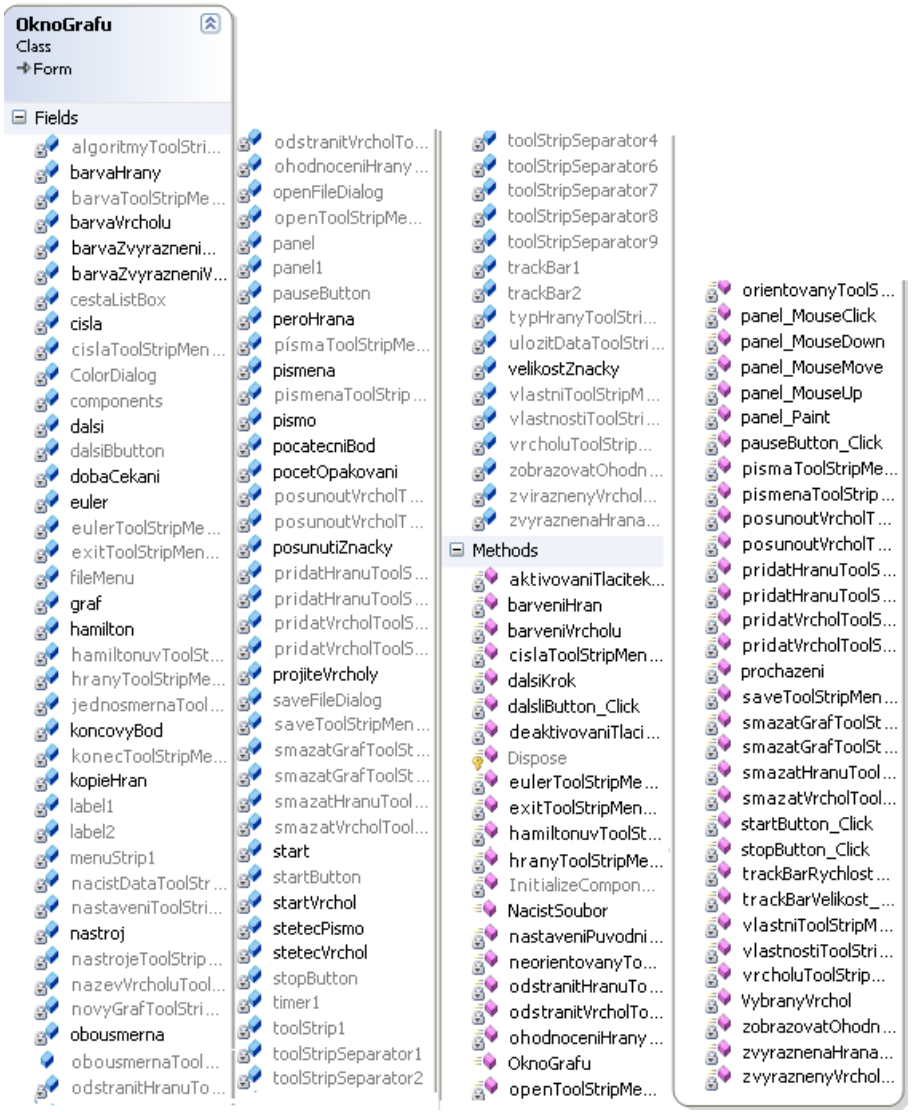
Třída *Napoveda* obsahuje *textBox*, ve kterém se zobrazuje nápověda k programu. Nápověda v sobě zahrnuje základní instrukce k vytváření, editování, načítání a ukládání grafu.



Obrázek 29 - Diagram třídy Napoveda

9.8 Třída OknoGrafu: Form

Tato třída slouží jako uživatelské rozhraní. Mimo vykreslování, vytváření, editování, načítání a ukládání grafu umožňuje spouštět zobrazování hledání vybraného tahu. Pomocí metod této třídy je možné měnit barvy hran a vrcholů a provádět další nastavení grafu jako je například jeho orientace a zadávání názvů vrcholů.



Obrázek 30 - Diagram třídy OknoGrafu

9.8.1 Datové složky třídy OknoGrafu

- *Nastroje nastroj* – uchovává hodnoty typu *Nastroje*, které jsou výčtového typu a mohou nabývat těchto hodnot:
  - PridatVrchol,

- PridatHranu,
- OdebratVrchol,
- OdebratHranu,
- PosunoutVrchol.

- *HamiltonovskyTah hamilton* – Hamiltonovský tah.
- *Graf graf* – graf, který je právě zobrazován v panelu pro kreslení grafu.
- *EulerovskyTah euler* – Eulerovský tah.
- *Point pocatecniBod* – počáteční bod z kterého je kreslená pomocná čára při přidávání a mazání hran a také při posunu vrcholu.
- *Point koncovyBod* - koncový bod do kterého je kreslená pomocná čára při přidávání a mazání hran a také při posunu vrcholu.
- *Vrchol startVrchol* – vrchol, který byl označen jako počáteční vrchol při přidávání a mazání hrany a také vrchol, který má být posunut.
- *bool obousmerna* – udává hodnotu, zda je hrana obousměrná, pokud ano, je nastavena ta *true*, pokud je jednosměrná je hodnota nastavena na *false*.
- *List<Vrchol> projiteVrcholy* – obsahuje vrcholy, které byly projity v průběhu hledání Hamiltonovského tahu.
- *Dictionary<Vrchol, List<Hrana>> kopieHran* – obsahuje kopii hran grafu.
- *bool start* – obsahuje údaj, zda-li již bylo stisknuto tlačítko *Start*. Pokud ano, je nastaveno na *true* jinak má hodnotu *false*.
- *bool dalsi* – obsahuje údaj, zda-li již byl proveden další krok zobrazování. Pokud ano, je nastaveno na *true* jinak má hodnotu *false*.
- *int pocetOpakovani* – počet kroků při hledání Hamiltonovského tahu.
- *int dobaCekani* – doba, mezi jednotlivými kroky při zobrazování vybraného tahu.

- *Size velikostZnacky* – velikost vykreslované značky.
- *Size posunutiZnacky* – velikost, o kterou má být posunut začátek vykreslování značky.
- *Pen peroHrana* – pero, kterým budou vykreslovány hrany.
- *Brush stetecVrchol* – barva štětce, kterým je vykreslována značka vrcholu.
- *Font pismo* – obsahuje font písma použitý pro čísla a písmena při vykreslování grafu.
- *Brush stetecPismo* – barva štětce, kterým budou vypisovány názvy vrcholů a ohodnocení hran.
- *Color barvaHrany* – barva hrany při vykreslování grafu.
- *Color barvaVrcholu* – barva vrcholu při vykreslování grafu.
- *Color barvaZvyrazneniVrcholu* – barva vrcholu, zobrazovaného jako projitý při hledání Hamiltonovského tahu.
- *Color barvaZvyrazneniHrany* – barva hrany při zobrazování vybraného tahu.
- *char pismena* – znak, reprezentující písmeno, které bude použito, v případě používání písmena jako automatického názvu vrcholu, pro další vrchol.
- *int cisla* – číslo, které bude použito, v případě používání čísla jako automatického názvu vrcholu, pro další vrchol.

9.8.2 Metody třídy OknoGrafu

- *void panel_Paint(object sender, PaintEventArgs e)* – metoda sloužící k vykreslení prvků grafu.

- *void panel_MouseClick(object sender, MouseEventArgs e)* – pro případ kliknutí levým tlačítkem myši na panelu pro vykreslování grafu jsou připraveny dvě akce.

  - *PridatVrchol* – byl-li vybrán nástroj *PridatVrchol*, a klikli jsme do panelu pro vykreslování grafu a v zadaném místě ještě neexistuje žádný vrchol, bude do tohoto místa přidán nový vrchol.

  - *OdebratVrchol* – pokud jsme klikli na již existující vrchol, a měli jsme vybrán nástroj pro odebírání vrcholů, bude tento vrchol odstraněn.

- *void panel_MouseDown(object sender, MouseEventArgs e)* – pro případ stisknutí levého tlačítka myši a vybrání některého z nástrojů

  - *PridatHranu*,

  - *OdebratHranu*,

  - *PosunoutVrchol,*

  je zjišťováno, zda bylo kliknuto na některý z vrcholů.

- *void panel_MouseMove(object sender, MouseEventArgs e)* – při výběru některého z nástrojů

  - *PridatHranu*,

  - *OdebratHranu*,

  - *PosunoutVrchol*

  a při tažení myší po panelu pro vykreslování grafu, s předcházejícím stisknutím levého tlačítka na některém z vrcholů, je vykreslována čára, od vybraného vrcholu, k současné pozici myši.

- *void panel_MouseUp(object sender, MouseEventArgs e)* – po uvolnění tlačítka myši je podle vybraného nástroje provedena příslušná akce.
    - *PridatHranu* – pokud byl vybrán počáteční a koncový vrchol, a neexistuje mezi nimi ještě hrana, je vytvořena a do grafu přidána nová hrana.
    - *OdebratHranu* – po výběru nástroje *OdebratHranu*, a následném vybrání počátečního a koncového vrcholu, je odstraněna hrana mezi vybranými vrcholy.
    - *PosunoutVrchol* – po zvolení nástroje pro posunutí vrcholu a výběru posouvaného vrcholu, s následným uvolněním tlačítka na pozici, na kterou chceme původní vrchol posunout, je původní vrchol posunut.
- *Vrchol VybranyVrchol(Point bod)* – metoda vrací vrchol, který byl vybrán, pokud nebyl vybrán žádný vrchol, vrací *null*.
- *void trackBarVelikost_Scroll(object sender, EventArgs e)* – při změně velikosti grafu, nastaví tloušťky hran a velikost vrcholu a písma.
- *void trackBarRychlost_Scroll(object sender, EventArgs e)* – při změně rychlosti zobrazování procházení grafu, nastaví dobu čekání, mezi jednotlivými kroky zobrazování.
- *bool NacistSoubor()* – otevře dialogové okno pro výběr otevíraného souboru. Pokud byl vybrán soubor, pokusí se soubor otevřít a načíst. Pokud se soubor povede načíst, vrací *true*, v opačném případě vrací *false*.
- *void openToolStripMenuItem_Click(object sender, EventArgs e)* – volá metodu *NacistSoubor()*.
- *void saveToolStripMenuItem_Click(object sender, EventArgs e)* – otevře dialogové okno pro výběr umístění ukládaného grafu. Po výběru, graf uloží pod zadaným názvem.
- *void pridatVrcholToolStripMenuItem_Click(object sender, EventArgs e)* – nastaví nástroj na *PridatVrchol*, zaškrtne položky *Přidat vrchol* ze záložky *Nástroje* a *panelu nástrojů*. Ostatní položky z těchto skupin budou nezaškrtnuty.

- *void smazatVrcholToolStripMenuItem_Click(object sender, EventArgs e)* – nastaví nástroj na *SmazatVrchol*, zaškrtne položky *Odebrat vrchol* ze záložky *Nástroje* a *panelu nástrojů*. Ostatní položky z těchto skupin budou nezaškrtnuty.

- *void pridatHranuToolStripMenuItem_Click(object sender, EventArgs e)* – nastaví nástroj na *PridatHranu*, zaškrtne položky *Přidat hranu* ze záložky *Nástroje* a *panelu nástrojů*. Ostatní položky z těchto skupin budou nezaškrtnuty.

- *void smazatHranuToolStripMenuItem_Click(object sender, EventArgs e)* – nastaví nástroj na *SmazatHranu*, zaškrtne položky *Odebrat hranu* ze záložky *Nástroje* a *panelu nástrojů*. Ostatní položky z těchto skupin budou nezaškrtnuty.

- *void smazatGrafToolStripMenuItem_Click(object sender, EventArgs e)* – vytvoří nový prázdný graf. Nástroj nastaví na *PridatVrchol* zaškrtne položky *Přidat vrchol* ze záložky *Nástroje* a *panelu nástrojů* Ostatní položky z těchto skupin budou nezaškrtnuty. Dále nastaví číselné i písmenné názvy vrcholů na počáteční hodnoty.

- *void posunoutVrcholToolStripMenuItem_Click(object sender, EventArgs e)* – nastaví nástroj na *PosunoutVrchol*, zaškrtne položky *Posunout vrchol* ze záložky *Nástroje* a *panelu nástrojů*. Ostatní položky z těchto skupin budou nezaškrtnuty.

- *void pridatVrcholToolStripButton_Click(object sender, EventArgs e)* – nastaví nástroj na *PridatVrchol*, zaškrtne položky *Přidat vrchol* ze záložky *Nástroje* a *panelu nástrojů*. Ostatní položky z těchto skupin budou nezaškrtnuty.

- *void odstranitVrcholToolStripButton_Click(object sender, EventArgs e)* – nastaví nástroj na *SmazatVrchol*, zaškrtne položky *Odebrat vrchol* ze záložky *Nástroje* a *panelu nástrojů*. Ostatní položky z těchto skupin budou nezaškrtnuty.

- *void pridatHranuToolStripButton_Click(object sender, EventArgs e)* – nastaví nástroj na *PridatHranu*, zaškrtne položky *Přidat hranu* ze záložky *Nástroje* a *panelu nástrojů*. Ostatní položky z těchto skupin budou nezaškrtnuty.

- *void odstranittHranuToolStripButton_Click(object sender, EventArgs e)* – nastaví nástroj na *SmazatHranu*, zaškrtne položky *Odebrat hranu* ze záložky *Nástroje* a *panelu nástrojů*. Ostatní položky z těchto skupin budou nezaškrtnuty.

- *void smazatGrafToolStripButton_Click(object sender, EventArgs e)* – vytvoří nový prázdný graf. Nástroj nastaví na *PridatVrchol* zaškrtne položky *Přidat vrchol* ze záložky *Nástroje* a *panelu nástrojů*. Ostatní položky z těchto skupin budou nezaškrtnuty. Dále nastaví číselné i písmenné názvy vrcholů na počáteční hodnoty.

- *void posunoutVrcholToolStripButton_Click(object sender, EventArgs e)* – nastaví nástroj na *PosunoutVrchol*, zaškrtne položky *Posunout vrchol* ze záložky *Nástroje* a *panelu nástrojů*. Ostatní položky z těchto skupin budou nezaškrtnuty.

- *void ohodnoceniHranyToolStripMenuItem_Click(object sender, EventArgs e)* – nastaví ohodnocení hrany na *true*.

- *void vlastniToolStripMenuItem_Click(object sender, EventArgs e)* – nastaví vlastní zadávání názvu vrcholu na *true*. Automatické zadávání názvu vrcholu písmeny a čísly bude nastaveno na *false*.

- *void cislaToolStripMenuItem_Click(object sender, EventArgs e)* – nastaví automatické zadávání názvu vrcholu jako čísel na *true*. Automatické zadávání názvu pomocí písmen bude nastaveno na *false* stejně jako zadávání vlastních názvů vrcholů.

- *void pismenaToolStripMenuItem_Click(object sender, EventArgs e)* – nastaví automatické zadávání názvu vrcholu jako písmena na *true*. Zadávání vlastních názvů vrcholů a automatické zadávání pomocí čísel bude nastaveno na *false*.

- *void hranyToolStripMenuItem_Click(object sender, EventArgs e)* – otevře dialogové okno pro výběr barvy a nastaví barvu hrany, při vykreslování grafu, na vybranou barvu.

- *void vrcholuToolStripMenuItem_Click(object sender, EventArgs e)* - otevře dialogové okno pro výběr barvy a nastaví barvu vrcholu, při vykreslování grafu, na vybranou barvu.

- *void pismaToolStripMenuItem_Click(object sender, EventArgs e)* - otevře dialogové okno pro výběr barvy a vybranou barvu nastaví jako barvu písma.

- *void zvyraznenyVrcholToolStripMenuItem_Click(object sender, EventArgs e)* - otevře dialogové okno pro výběr barvy a vybranou barvu nastaví pro zobrazování projitého vrcholu.

- *void zvyraznenaHranaToolStripMenuItem_Click(object sender, EventArgs e)* - otevře dialogové okno pro výběr barvy a vybranou barvu nastaví pro zobrazování projité hrany.

- *void zobrazovatOhodnoceniHranToolStripMenuItem_Click(object sender, EventArgs e)* - nastavuje, zda mají být zobrazeny hodnoty u ohodnocené hrany.

- *void neorientovanyToolStripMenuItem_Click(object sender, EventArgs e)* – nastavuje, zda má být graf neorientovaný.

- *void orientovanyToolStripMenuItem_Click(object sender, EventArgs e)* – nastavuje, zda má být graf orientovaný.

- *void eulerToolStripMenuItem_Click(object sender, EventArgs e)* – nastavuje, aby se po stisknutí tlačítka *Start*, začal hledat a následně zobrazovat Eulerovský tah.

- *void hamiltonuvToolStripMenuItem_Click(object sender, EventArgs e)* – nastavuje, aby se po stisknutí tlačítka *Start*, začal hledat a následně zobrazovat Hamiltonovský tah.

- *void exitToolStripMenuItem_Click(object sender, EventArgs e)* – zavře okno aktuálního grafu.

- *void vlastnostiToolStripMenuItem_Click(object sender, EventArgs e)* – otevře okno a v něm vypíše vlastnosti grafu.

- *void startButton_Click(object sender, EventArgs e)* – spustí zobrazování hledání vybraného tahu.

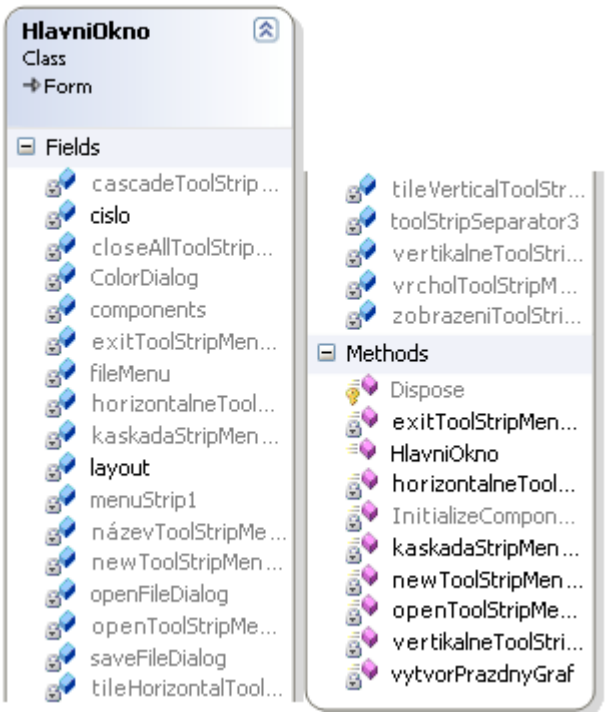
- *void pauseButton_Click(object sender, EventArgs e)* – pozastaví zobrazování hledání vybraného tahu.

- *void stopButton_Click(object sender, EventArgs e)* – ukončí zobrazování hledání vybraného tahu.

- *void dalsiButton_Click(object sender, EventArgs e)* – volá metodu *dalsiKrok()*.
- *void dalsiKrok()* – provede zobrazení dalšího kroku při zobrazování hledání vybraného tahu.
- *void barveniVrcholu(List<Vrchol> vrcholy)* – provádí nastavení barvy procházeného vrcholu a k němu vedoucí hrany při zobrazování hledání vybraného tahu.
- *void barveniHran()* – provádí nastavení barvy projitých hran při zobrazování hledání vybraného tahu.
- *void nastaveniPuvodniBarvy()* – nastaví původní barvy vrcholů a hran.
- *void aktivovaniTlacitekZobrazovani()* – aktivuje zbylá tlačítka pro zobrazování hledání vybraného tahu. Zobrazí *listBox* pro vypisování jednotlivých kroků hledání.
- *void deaktivovaniTlacitekZobrazovani()* – deaktivuje tlačítka pro zobrazování hledání vybraného tahu mimo tlačítka *Start*. Skryje *listBox*, do kterého se provádí vypisování jednotlivých kroků hledání.
- *void prochazeni(object sender, EventArgs e)* – metoda volána z handleru událostí.

9.9 Třída HlavniOkno:Form

Tato třída slouží především jako kontejner pro okna jednotlivých grafů. Uživatel z ní může vytvářet nová okna pro grafy, otvírat již uložené grafy, ukončit celou aplikaci a nastavovat, jak mají být jednotlivá okna grafu rozmístěna.



Obrázek 31 - Diagram třídy HlavniOkno

9.9.1 Datové složky třídy HlavniOkno

- *MdiLayout layout* – nastavení layoutu.
- *int cislo* – číslo okna grafu.

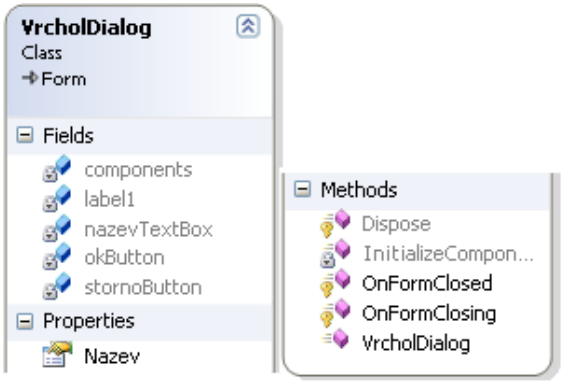
9.9.2 Metody třídy HlavniOkno

- *void newToolStripMenuItem_Click(object sender, EventArgs e)* – volá metodu *vytvorPrazdnyGraf()*.
- *void openToolStripMenuItem_Click(object sender, EventArgs e)* – vytvoří nové okno grafu a zavolá metodu z *OknaGrafu NacistSoubor()*, která načte a zobrazí dříve uložený graf.

- *void exitToolStripMenuItem_Click(object sender, EventArgs e)* – ukončí celou aplikaci.
- *void vytvorPrazdnyGraf()* – vytvoří a zobrazí nové *OknoGrafu*, které umístí na pozici dle nastaveného layoutu.
- *void kaskadaStripMenuItem1_Click(object sender, EventArgs e)* – nastaví layout na kaskádu.
- *void vertikalneToolStripMenuItem_Click(object sender, EventArgs e)* – nastaví layout na vertikální dlaždice.
- *void horizontalneToolStripMenuItem_Click(object sender, EventArgs e)* - nastaví layout na horizontální dlaždice.
- void *napovedaToolStripMenuItem_Click(object sender, EventArgs e)* – zobrazí okno s nápovědou.

9.10 Třída VrcholDialog: Form

Třída slouží pro zadávání názvu vrcholu. Používá se v případě, že si uživatel zvolil vlastní název vrcholu.



Obrázek 32 - Diagram třídy VrcholDialog

9.10.1 Datové složky třídy VrcholDialog

- *string Nazev* - vlastnost nastavující a poskytující název přidávaného vrcholu.

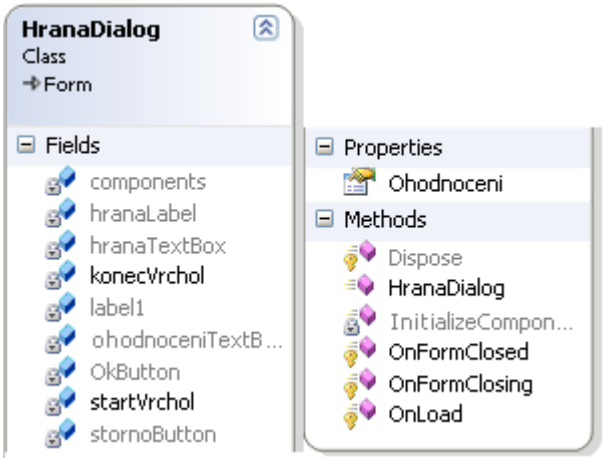
9.10.2 Metody třídy VrcholDialog

- *void OnFormClosing(FormClosingEventArgs e)* – metoda po stisku tlačítka *OK* kontroluje, zda-li je vyplněné pole pro název vrcholu, pokud ne, otevře okno s varováním.

- *void OnFormClosed(FormClosedEventArgs e)* – metoda kontroluje, nebylo-li stisknuté tlačítko *Storno*, pokud ne, do vlastnosti *Nazev* uloží zadané jméno vrcholu.

9.11 Třída HranaDialog: Form

Formulář sloužící pro zadávání ohodnocení hrany. Je používán v případě ohodnoceného grafu. Uživatel do něho zadá ohodnocení právě vytvářené hrany.



Obrázek 33 - Diagram třídy HranaDialog

9.11.1 Datové složky třídy HranaDialog

- *int Ohodnoceni* – vlastnost nastavující a poskytující ohodnocení přidávané hrany.

- *Vrchol startVrchol* – vrchol, v kterém hrana začíná.

- *Vrchol konecVrchol* – vrchol, v kterém hrana končí.

9.11.2 Metody třídy HranaDialog

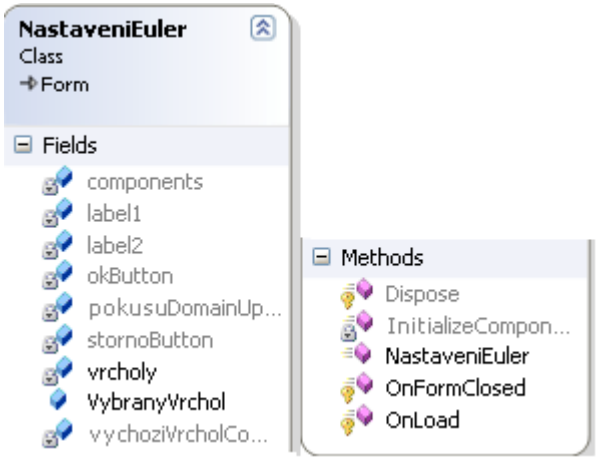
- *void OnLoad(EventArgs e)* – metoda vypíše do textBoxu název vrcholu v kterém hrana začíná, pomlčku a název vrcholu v kterém končí.

- *void OnFormClosing(FormClosingEventArgs e)* – metoda po stisku tlačítka *OK* kontroluje, zda-li je správně vyplněné pole pro ohodnocení hrany, pokud ne, otevře okno s varováním.

- *void OnFormClosed(FormClosedEventArgs e)* – metoda kontroluje, nebylo-li stisknuté tlačítko *Storno*, pokud ne, do vlastnosti *Ohodnoceni* uloží zadanou hodnotu reprezentující ohodnocení hrany.

9.12 Třída NastaveniEuler: Form

Formulář *NastaveniEuler*, slouží k zadání vrcholu, v kterém má začít Eulerovský tah.



Obrázek 34 - Diagram třídy NastaveniEuler

9.12.1 Datové složky třídy NastaveniEuler

- *Vrchol VybranyVrchol* – vrchol, který byl vybrán jako výchozí pro Eulerovský tah.

- *List<Vrchol> vrcholy* – seznam vrcholů vyskytujících se v grafu.

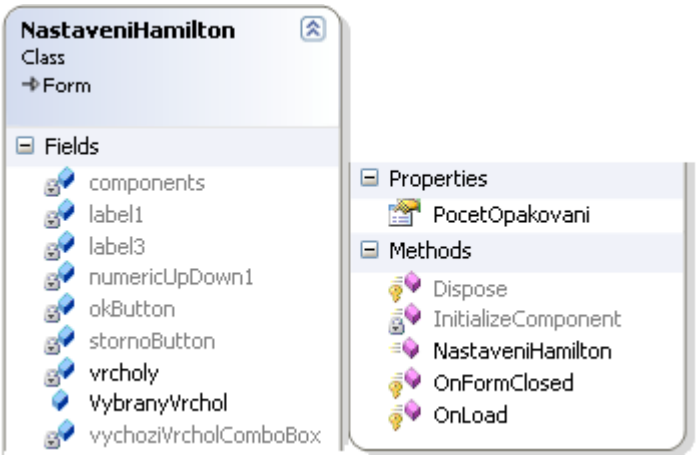
9.12.2 Metody třídy NastaveniEuler

- *void OnLoad(EventArgs e)* – metoda naplní comboBox názvy vrcholů, které jsou v grafu.

- *void OnFormClosed(FormClosedEventArgs e)* – metoda přiřadí do *VybranehoVrcholu* vrchol, který byl vybrán v *comboBoxu*, pokud nebyl vybrán žádný vrchol, je hodnota *VybranehoVrcholu* nastavena na *null*.

9.13 Třída NastaveniHamilton: Form

Formulář NastaveniHamilton slouží pro zadání výchozích hodnot při hledání Hamiltonovského tahu. Těmito hodnotami je výchozí vrchol a počet kroků.



Obrázek 35 - Diagram třídy NastaveniHamilton

9.13.1 Datové složky třídy NastaveniHamilton

- *List<Vrchol> vrcholy* – seznam vrcholů vyskytujících se v grafu.
- *Vrchol VybranyVrchol* – vrchol, který byl vybrán jako výchozí pro Hamiltonovský tah.
- *int PocetOpakovani* – kolik kroků se má vykonat při hledání Hamiltonovského tahu.

9.13.2 Metody třídy NastaveniHamilton

- *void OnLoad(EventArgs e)* – metoda naplní comboBox názvy vrcholů, které jsou v grafu.
- *void OnFormClosed(FormClosedEventArgs e)* – metoda přiřadí do *VybranehoVrcholu* vrchol, který byl vybrán v *comboBoxu* a zároveň do *PocetOpakovani* přiřadí hodnotu nastavenou pro počet kroků, které se mají vykonat.

Závěr

V teoretické části práce byly vysvětleny základní pojmy z teorie grafů, bylo naznačeno dělení grafů podle různých kritérii a také popsány další typy grafů. Následovalo ujasnění, co je to Eulerovský a Hamiltonovský tah a jak fungují Fleuryho, Edmondsův a Littlův algoritmus.

Hlavním cílem práce pak bylo vytvoření aplikace pro zobrazování Eulerovského a Hamiltonovského tahu. Tento program byl vytvořen v prostředí Microsoft Visual Studio 2008 za použití programovacího jazyka C#. Uživatel může v programu vytvářet vlastní grafy, které může dále editovat nebo ukládat. Na grafech je možné zobrazovat Eulerovský a Hamiltonovský tah. Bohužel se mi nepovedlo najít ani vymyslet takový algoritmus, který by zaručil, aby při hledání Hamiltonovského tahu nedošlo k jeho zacyklení. Aby bylo možné dokázat, že mnou vy myšlený algoritmus je schopný Hamiltonovský tah za určitých podmínek nalézt, byl problém se zacyklením vyřešen pomocí zadání počtu kroků, které se mají při hledání provést.

Editor je tedy schopný znázorňovat jednotlivé kroky, během nichž prochází hrany a vrcholy, a zároveň tyto kroky vypisovat. Dále umožňuje zobrazit vlastnosti grafu a obsahuje také nápovědu.

Součástí diplomové práce je i uživatelská příručka, která se nachází na konci práce jako příloha.

Použitá literatura a další zdroje

[1] Algoritmy.net, *Graf*, [online]. 2008 - 2010 [cit. 2010-08-11]. Dostupný z WWW:<*http://www.algoritmy.net/article/1369/Graf*>

[2] Doc. RNDr. Ing. Miloš Šeda, Ph.D., Teorie grafů [online], 2003 [cit. 2010-08-09]. Dostupný z WWW: <*http://www.uai.fme.vutbr.cz/~mseda/TG03_MS.pdf*>

[3] Ing. Michal Dorda, Ph.D, *Littlův algoritmus* [online], ? [cit. 2010-08-11], Dostupný z WWW: <*homel.vsb.cz/~dor028/Littluv_algoritmus.doc*>

[4] J. Nečas, *Grafy a jejich použití*, Praha, SNTL, 1978. 191 s. ISBN 04-337-78

[5] Jaroslav Nešetřil, *Teorie grafů*, Praha, SNTL, 1979. 316 s. ISBN 04-017-79

[6] Jiří Sedláček, *Úvod do teorie grafů*, Praha, Academia, 1981. 272 s. ISBN 509-21-826

[7] Josef Volek. *Operační výzkum I*, Pardubice : Univerzita Pardubice, 2005. 111 s. ISBN 80-7194-410-6.

[8] Juraj Bosák, *Grafy a ich aplikácie*, Bratislava, ALFA, 1980. 176 s. ISBN 63-159-80

[9] Lukáš Jirkovský, *Teorie grafů* [online]. 28. 5. 2008 [cit. 2010-08-10]. Dostupný z WWW: <*http://teorie-grafu.elfineer.cz/*>

[10] Marie Demlová, *Logika a grafy* [online]. 20. 4. 2010 [cit. 2010-08-10]. Dostupný z WWW: <*http://math.feld.cvut.cz/demlova/teaching/lgr/pred10.pdf*>

[11] Prof. RNDr. Ivan Mezník, CSc., *Diskrétní matematika*, Brno, Akademické nakladatelství CERM, s.r.o. Brno, 2004. 74 s. ISBN 80-214-2754-X

[12] Stanislav Palúch, *Algoritmická teória grafov* [online]. 2008 [cit. 2010-08-10]. Dostupný z WWW: <*http://frcatel.fri.utc.sk/users/paluch/grafy.pdf*>

[13] Wikipedia, *Eurelovský tah* [online]. 29. 6. 2010 [cit. 2009-08-10]. Dostupný z WWW: <*http://cs.wikipedia.org/wiki/Eulerovsk%C3%BD_tah*>

[14] Wikipedia, *Hamiltonovský graf* [online]. 20. 5. 2010 [cit. 2010-08-10]. Dostupný z WWW: <*http://cs.wikipedia.org/wiki/Hamiltonovsk%C3%BD_graf*>

Seznam použitých obrázků

Seznam příloh

Uživatelská příručka

CD s aplikací

Přílohy

Uživatelská příručka

Obsah

1 Hlavní okno

Program se skládá z hlavního okna a oken grafů. Po spuštění programu se v hlavním okně automaticky otevře prázdné okno grafu.

1.1 Menu

Hlavní okno má v horní části okna *menu*, ve kterém se nacházejí tři položky a to *Soubor*, *Zobrazení oken* a *Nápověda*.

1.1.1 Soubor

Pod položkou *Soubor* jsou položky, které umožní vytvářet nová okna grafů, otvírat již uložené grafy a nebo celou aplikaci zavřít. K těmto účelům slouží položky *Nový*, *Otevřít* a *Konec*.

Po stisknutí položky *Nový* se vytvoří a otevře nové okno grafu. V tomto okně je možné vytvářet nové grafy. Nové okno je možné vytvořit také pomocí klávesové zkratky *Ctrl + N*.

Pokud z nabídky zvolíte položku *Otevřít*, otevře se dialogové okno, ve kterém vyberte soubor, který si přejete otevřít. Po výběru souboru, a následném potvrzení tlačítkem *Otevřít*, se vytvoří nové okno grafu, stejně jako po vyvolání položky *Nový*, a načte se a zobrazí otevíraný graf. Nebude-li možné graf z nějakých důvodů otevřít, budete o tom informováni pomocí dialogového okna. Položku *Otevřít* je možné vyvolat také pomocí klávesové zkratky *Ctrl + O*.

Položka *Konec* slouží k ukončení celé aplikace. Stisknete-li tedy tuto položku, zavřete celou aplikaci.

1.1.2 Zobrazení oken

Tato položka menu obsahuje další tři položky těmi jsou *Kaskáda*, *Dlaždice vertikálně* a *Dlaždice horizontálně*. Tyto položky slouží k rozvržení otevřených oken grafu.

Po zaškrtnutí položky *Kaskáda*, budou otvíraná okna skládány do kaskády.

Položka *Dlaždice vertikálně*, po jejím zaškrtnutí, poskládá okna vertikálně vedle sebe.

Podobně jako *Dlaždice vertikálně*, funguje *Dlaždice horizontálně*, v tomto případě jsou okna skládány horizontálně a následně jako dlaždice.

1.1.3 Nápověda

Kliknutím na tuto položku se zobrazí okno *Nápověda*, které v sobě obsahuje popis, jak ovládat aplikaci.

2 Okno grafu

Toto okno obsahuje veškeré potřebné nástroje pro ovládání. V horní části okna se nachází *menu*, které je složeno z šesti položek, kterými jsou *Soubor*, *Nástroje*, *Nastavení*, *Algoritmy*, *Vlastnosti* a *Nápověda.* Pod ním vlevo, je *panel nástrojů* ten má stejnou funkci jako položka nástroje v *menu*. Vpravo od *panelu nástrojů* se *nachází panel pro ovládání zobrazování algoritmů*. Uprostřed okna se nachází místo pro kreslení grafu. Pod ním dole je *lišta* jejíž posunem je možné zvětšovat a zmenšovat graf. Vedle se nachází *lišta pro ovládání rychlosti běhu zobrazování algoritmu*.

2.1 Menu

Menu se nachází v horní části okna grafu. Je složeno z pěti položek těmi jsou *Soubor*, *Nástroje*, *Nastavení*, *Algoritmy* a *Vlastnosti.* Obsah a funkce jednotlivých položek je popsána níže.

2.1.1 Soubor

Po kliknutí na položku *Soubor* v okně grafu, se rozbalí nabídka, která obsahuje položky *Otevřít*, *Uložit* a *Konec*.

Pokud z této nabídky zvolíte položku *Otevřít*, otevře se dialogové okno, v kterém vyberte soubor, který si přejete otevřít. Po výběru souboru, a následném potvrzení tlačítkem *Otevřít*, se načte a zobrazí otevíraný graf. Nebude-li možné graf z nějakých důvodů otevřít, budete o tom informováni pomocí dialogového okna. Položku *Otevřít* je možné vyvolat také pomocí klávesové zkratky *Ctrl + O*.

Položka *Uložit* slouží k uložení grafu. Po jejím zvolení, se otevře dialogové okno, ve kterém zadáte název souboru a zvolíte umístění, kam chcete graf uložit. Pokud výběr potvrdíte tlačítkem *Uložit*, uloží se graf pod zadaným názvem na určené místo. Tuto položku je podobně jako položku *Otevřít* možné vyvolat stisknutím kláves *Ctrl + S*.

Kliknutím na položku *Konec*, zavřete okno daného grafu.

2.1.2 Nástroje

Pod položkou *Nástroje* jsou ukryty nástroje pro editaci grafu. Těchto nástrojů je zde umístěno šest a jejich funkce se skrývají pod položkami *Přidat vrchol*, *Přidat hranu*, *Odebrat vrchol*, *Odebrat hranu*, *Smazat graf* a *Posunout vrchol*. Výběr aktuálního nástroje je označen zaškrtnutím u jeho názvu.

Po výběru položky *Přidat vrchol* můžete do grafu přidat nový vrchol. To uděláte klinutím levým tlačítkem myši na plochu určenou pro kreslení grafu. Pokud jste zvolili, že název vrcholu chcete zadávat vlastní, zobrazí se dialogové okno, ve kterém je potřeba zadat název vrcholu a následně ho potvrdit tlačítkem *OK*. Poté se v zadaném místě objeví vrchol se zadaným názvem. Pokud nemáte zvolenou volbu vlastního názvu vrcholu, dialogové okno se nezobrazí a vrchol bude vytvořen okamžitě po kliknutí myší. Tato položka je automaticky zaškrtnuta při spuštění programu a při vytvoření nového okna grafu.

Položka *Přidat hranu* umožňuje do grafu přidávat nové hrany. Hrany se přidávají vždy mezi dva vrcholy a mezi každými dvěma vrcholy může existovat jenom jedna hrana. Hrana se do grafu přidá tak, že stisknete levé tlačítko myši na vrcholu, z kterého chcete hranu vést a táhnete myší na vrchol, do kterého má vytvářená hrana vstupovat. Ve chvíli, kdy stojíte nad vrcholem do kterého má hrana vést, pusťte tlačítko myši a pokud jste si zvolili, že chcete mít hrany ohodnocené, zobrazí se dialogové okno. Do něho musíte zadat číslo, které bude udávat ohodnocení dané hrany. Po potvrzení dialogového okna se vytvoří mezi zadanými vrcholy hrana. Pokud jste ohodnocení hran nezaškrtli, vytvoří se hrana okamžitě po uvolnění tlačítka myši.

Odebrat vrchol z grafu umožní položka s názvem *Odebrat vrchol*. Vrchol pak odeberete tak, že levým tlačítkem myši kliknete na vrchol, který chcete odebrat. Poté bude vrchol z grafu odebrán.

Položka *Odebrat hranu* umožňuje odebrat hranu z grafu. Po jejím vybrání, stiskněte levé tlačítko myši na vrcholu, ze kterého vede hrana, kterou chcete odebrat a pak táhněte myší na vrchol, do kterého vede odebíraná hrana. Po uvolnění tlačítka myši nad tímto vrcholem, dojde k odebrání hrany.

Po kliknutí levým tlačítkem na položku *Smazat graf*, dojde ke smazání grafu, a nastavení nástroje na přidávání vrcholu.

Výběrem položky *Posunout vrchol*, je umožněno posouvat vrcholy grafu na nové pozice. Učiníte tak stisknutím levého tlačítka myši na vrchol, který chcete posunout a následným tažením myši na požadovanou pozici. Po uvolnění tlačítka dojde k posunu vrcholu na novou pozici.

2.1.3 Nastavení

Pod položkou *menu Nastavení* je možné nastavit některé vlastnosti grafu a jeho částí. K tomu slouží položky *Ohodnotit hrany*, *Zobrazit ohodnoceni hran*, *Název vrcholu*, *Typ grafu* a *Barva*.

Položku *Ohodnotit hrany* je možné zvolit pouze v případě, že do grafu nebyla vložena ještě žádná hrana. Pokud tuto položku zvolíte, budete vždy při přidání hrany vyzváni, dialogovým oknem k tomu, abyste zadali ohodnocení hrany. Ohodnocení hrany musí být celé kladné číslo. Protože graf musí být buď ohodnocený nebo neohodnocený, není možné tuto položku měnit pokud se v grafu vyskytují již nějaké hrany.

Pokud chcete zobrazovat ohodnocení hran, musíte zvolit položku *Zobrazit ohodnocení hran*. Zobrazovat ohodnocení hran je možné jen v případě, bylo-li zvoleno ohodnocování hran. Nechcete-li již dále zobrazovat ohodnocení hran, stačí znovu kliknout na položku *Zobrazit ohodnocení hran* a zrušit tak její zaškrtnutí.

Položka *Název vrcholu* umožňuje zvolit název vrcholu. Na výběr je název reprezentovaný písmeny abecedy, čísly nebo vlastní název. Název vrcholu je možné měnit kdykoliv během vytváření grafu. Pokud tedy chcete, aby vrcholy měly automatický název, který bude reprezentován písmeny abecedy, zaškrtněte pod položkou *Název vrcholu* položku *Písmena*. Pokud chcete, aby vrcholy měly jako název čísla, zaškrtneme položku *Čísla*. Pro vlastní názvy vrcholů je pak potřeba zvolit položku *Vlastní*. V tomto případě budete vždy při vytváření nového vrcholu vyzváni dialogovým oknem, abyste zadali název vrcholu.

Pod položkou *Barva* je možné měnit jednotlivé barvy vrcholů, hran a písma. Po kliknutí na *Barva* a *Hrany* se zobrazí dialogové okno, které umožňuje vybrat si barvu, kterou chcete aby se zobrazovaly vytvořené hrany. Obdobně položka *Vrcholy*. Pomocí ní nastavíte, jakou barvu mají mít vrcholy, které vytváříte v grafu. Po vybrání barvy v dialogovém okně, které se zobrazí po kliknutí na položku *Písma*, se změní barva písma. Tedy barva ohodnocení hran a názvů vrcholů. Kliknutím na položku *Barva* a následně na *Zvýrazněné hrany* bude umožněno

vybrat barvu, kterou se budou zobrazovat hrany při hledání Eulerovského a Hamiltonovského tahu v grafu. Obdobně tomu bude po kliknutí na položku *Zvýrazněného vrcholu*. V tomto případě volíte barvu pro zobrazování projitých vrcholů při zobrazování hledání Hamiltonovského tahu.

2.1.4 Algoritmy

Algoritmy jsou další položkou *menu* okna grafu. Je zde možné, zvolit si, zda-li se má po spuštění algoritmu provádět Eulerovský nebo Hamiltonovský tah. K tomu slouží položky *Eulerovský tah* a *Hamiltonovský tah*.

Po vybrání položky *Eulerovský tah* a následném spuštění pomocí tlačítka *Start*. Budete vyzváni, abyste vybrali vrchol, v kterém chcete začít. Pokud v tomto vrcholu nebude možné začít, budete na to upozorněni dialogovým oknem. V opačném případě bude možné zobrazovat kroky při hledání Eulerovského tahu.

Zaškrtnutím položky *Hamiltonovský tah*, se po stisknutí tlačítka *Start* bude vykonávat algoritmus pro nalezení Hamiltonovského tahu. Proto, aby mohl být algoritmus hledání zobrazován po krocích bude třeba do dialogového okna zadat vrchol, ve kterém chcete začít a také počet kroků, které se mají provést. Při zadání vrcholu, ze kterého není možné začít, budete na tuto skutečnost opět upozorněni dialogovým oknem.

2.1.5 Vlastnosti

Po stisknutí položky *Vlastnosti*, se otevře nové okno, ve kterém budou vypsány základní vlastnosti grafu. Vypsanými vlastnostmi bude skóre grafu, počet jeho hran, počet vrcholů, počet komponent grafu, to zda-li je graf stromem a jedná-li se o Eulerovský graf.

2.2 Panel nástrojů

Panel nástrojů je umístěn vlevo pod *menu*. Obsahuje stejná tlačítka jako položka *Nástroje* v *menu*. Jsou jimi tedy *Přidat vrchol*, *Přidat hranu*, *Odebrat vrchol*, *Odebrat hranu*, *Smazat graf* a *Posunout vrchol*. Všechny tyto tlačítka plní stejnou funkci, jako již výše zmíněné položky z *menu Nástroje*.

2.3 Panel ovládání algoritmu

Tento panel je umístěn vpravo pod *menu*. Obsahuje tlačítka *Start*, *Pauza*, *Stop* a *Další* a slouží k ovládání zobrazování algoritmů hledání Eulerovského a Hamiltonovského tahu.

2.3.1 Start

Aby mohlo být spuštěno zobrazování vybraného algoritmu je třeba stisknout tlačítko *Start*. Po jeho stisknutí se otevře dialogové okno. V němž jste, u zobrazování Eulerovského tahu, vyzváni, abyste vybrali vrchol, ve kterém si přejete začít. Při spouštění zobrazování Hamiltonovského tahu pak mimo výběru počátečního vrcholu zadáváte také počet kroků, během kterých se algoritmus pokusí najít Hamiltonovský tah. Po potvrzení dialogového okna se aktivují i zbylá tlačítka tedy *Pauza*, *Stop* a *Další* a pod *panelem ovládání zobrazování algoritmů* se objeví tabulka, ve které budou vypisovány jednotlivé kroky algoritmu. Pokud jste vybrali vrchol, ve kterém není možné začít, budete o tom informováni a je nutné akci zopakovat.

Pokud stisknete tlačítko *Start* znovu, začne se zobrazovat, jak algoritmus postupoval při hledání vybraného tahu. Procházení můžete přerušit nebo pozastavit. K tomu slouží tlačítka *Pauza* a *Stop*.

2.3.2 Stop

Stisknutím tlačítka *Stop* dojde k ukončení zobrazování vybraného algoritmu. Po tomto úkonu již není možné pokračovat ve zobrazování.

2.3.3 Pauza

Po stisknutí tlačítka *Pauza* dojde k přerušení zobrazování vybraného algoritmu. V algoritmu je ale možné pokračovat a to buď stisknutím tlačítka *Start* nebo *Další*.

2.3.4 Další

Tlačítko *Další* slouží ke zobrazení dalšího kroku algoritmu. Poté je zobrazování opět pozastaveno a pro jeho pokračování je třeba stisknout tlačítko *Start* nebo *Další*.

2.4 Panel pro kreslení grafu

Panel pro kreslení grafu se nachází uprostřed okna grafu. Je na něm možné vytvářet a editovat libovolné grafy. Grafy se vytvářejí pomocí nástrojů na přidávání a odebírání vrcholů a hran.

2.5 Zvětšování grafu

Pro zvětšování a zmenšování grafu, slouží posuvná lišta, která se nachází v levém dolním rohu okna grafu. Při posunu ukazatele na liště směrem doprava, se graf zvětšuje. Tažením ukazatele posunu na levou stranu se pak graf bude zmenšovat.

2.6 Rychlost zobrazování algoritmu

Ovládání rychlosti zobrazování jednotlivých kroků algoritmu se nachází na spodním okraji okna grafu, vedle *lišty pro zvětšování grafu*. Ovládá se tažením ukazatele. Při tažení ukazatele vpravo, se rychlost zobrazování jednotlivých kroků bude zrychlovat. Naopak tažením ukazatele vlevo, bude interval zobrazování jednotlivých kroků algoritmu delší.

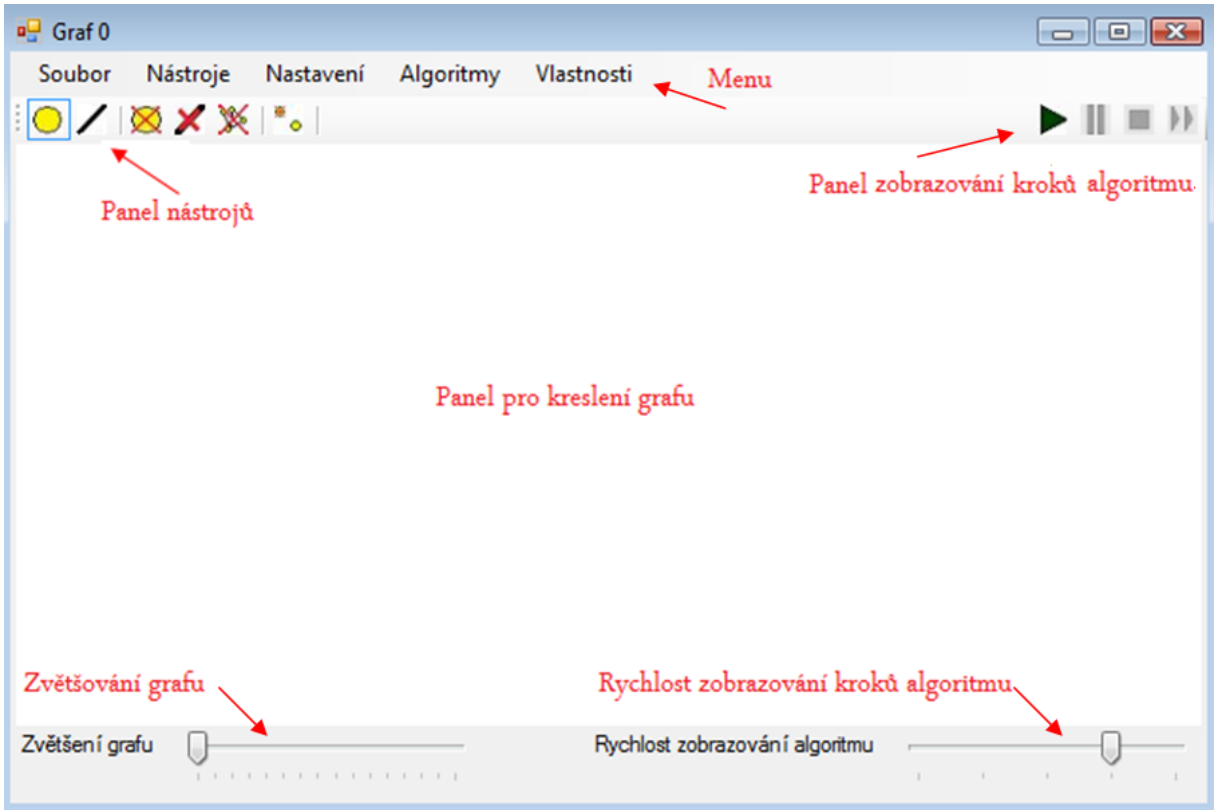
3 Práce s programem

3.1 Otevření okna grafu

Po spuštění aplikace se v hlavním okně programu otevře prázdné okno grafu. Oken grafů je možno otevřít několik najednou a to několika způsoby.

Prvním způsobem je otevření prázdného okna grafu. To lze udělat po kliknutí na složku *Soubor* v *menu* v hlavním okně a následným vybráním položky *Nový* nebo stisknutím kláves *Ctrl + N*. Dalším způsobem je otevření okna grafu s již vytvořeným grafem. To uděláte opět přes položku *Soubor* v *menu* hlavního okna a dále vybráním položky *Otevřít*.

Okno grafu se skládá z *menu*, *panelu nástrojů*, *panelu ovládání zobrazování algoritmu*, *panelu pro zobrazování grafu*, *místa pro kreslení grafu*, *lišty pro zvětšování grafu* a *lišty pro rychlost zobrazování jednotlivých kroků algoritmu*.



Obrázek 36 - Okno grafu

V tomto okně je možné vytvořit si nový graf, nebo je do něho možné načíst data, která reprezentují graf již dříve uložený.

3.2 Otevření uloženého grafu

Pokud si načtete již dříve uložená data, je možné je dále upravovat pomocí nástrojů pro editaci grafu. Graf otevřete po kliknutí na *Soubor* a vybrání položky *Otevřít*, nebo stisknutím *Ctrl + O*. Dále stačí vybrat soubor, ve kterém je graf, který chcete otevřít uložen a stisknout tlačítko *Otevřít*.

3.3 Vytváření a editování grafu

Graf se vytváří a edituje pomocí nástrojů na editaci grafu. Ty naleznete v *panelu nástrojů* nebo v *menu*, pod položkou *Nástroje*. Graf je tvořen z vrcholů a hran, a proto, pokud chcete vytvořit nový graf, je třeba do něho vložit nějaké vrcholy a hrany.

3.3.1 Přidávání vrcholů a hran

Vrchol se přidává vybráním nástroje *Přidat vrchol* a následným kliknutím levým tlačítkem myši do prostoru pro kreslení grafu. Pokud máte vytvořeny alespoň dva vrcholy, můžete začít vytvářet hrany. Hrany je možné vytvořit po výběru nástroje *Přidat hranu*. Proto vyberte tento nástroj a následně stiskněte levé tlačítko myši na vrcholu, ze kterého chcete aby hrana vycházela. Dále táhněte myší až do vrcholu, do kterého má vytvářená hrana vést. V tomto vrcholu tlačítko myši uvolněte přičemž dojde k vytvoření hrany.

3.3.2 Odebíraní vrcholů a hran

Podobně jako přidávání vrcholů a hran funguje jejich odebírání. Vrchol odeberete po vybrání nástroje *Odebrat vrchol* a následném kliknutí na vrcholu, který chcete odebrat. Odebrání hrany pak provedete pokud zvolíte nástroj *Odebrat hranu* a následně stisknete levé tlačítko myši na vrcholu, z kterého vede odebíraná hrana a táhnete myší do vrcholu v němž hrana, kterou chcete odebrat končí. V tomto vrcholu tlačítko myši pusťte. Tím dojde k odstranění vybrané hrany.

3.3.3 Posouvání vrcholů

Další možností, jak editovat graf, je posouvat jeho vrcholy. To je možné po výběru nástroje *Posunout vrchol*. Poté stiskněte levé tlačítko myši na vrcholu, který chcete posunout a táhněte myší na místo, na které chcete posouvaný vrchol umístit. Zde tlačítko myši uvolněte. Tím jste posunuli vrchol.

3.3.4 Mazání grafu

Pokud chcete vymazat celý graf, nemusíte zavírat okno grafu a následně vytvořit nové, ale můžete tak učinit kliknutím na položku *Nástroje* a následně pak na *Smazat graf*. Po kliknutí na tento nástroj dojde ke smazání celého grafu.

3.4 Zobrazování kroků algoritmu

Hlavní činností programu je zobrazovat jednotlivé kroky algoritmů, které hledají Eulerovský a Hamiltonovský tah. Jaký graf se má zobrazovat určíte zaškrtnutím příslušného políčka algoritmu. Tato volba se provádí v *menu* pod položkou *Algoritmy*.

3.4.1 Spuštění zobrazování kroků algoritmu

Samotné ovládání krokování algoritmu se pak provádí za pomocí *panelu pro zobrazování kroků algoritmu*. Algoritmus spustíte stisknutím tlačítka *Start*. Nyní jste vyzvání, abyste vybrali vrchol, ve kterém chcete tah začít. U Hamiltonovského tahu pak zadáváte ještě počet kroků během nichž se algoritmus pokusí najít Hamiltonovský tah. Po potvrzení zadaných informací tlačítkem *OK*, se zpřístupní také ostatní tlačítka panelu.

3.4.2 Zobrazování jednoho následujícího kroku

Nyní máte několik možností, jak zobrazovat jednotlivé kroky. Pokud chcete zobrazit pouze jeden následující krok, stiskněte tlačítko *Další*. Toto tlačítko je možné stisknout několikrát po sobě a projít jím třeba i celý tah.

3.4.3 Postupné zobrazování kroků

Další možností jak zobrazovat kroky algoritmu je stisknutí tlačítka *Start* v tomto případě se budou zobrazovat jednotlivé kroky algoritmu automaticky a to tak, že mezi jednotlivými kroky bude časová prodleva, kterou je možné nastavit pomocí *panelu pro rychlost zobrazování algoritmu*.

3.4.4 Pozastavení zobrazování

Běh zobrazování algoritmu je možné pozastavit pomocí tlačítka *Pauza*. V dalším zobrazování je možné pokračovat stisknutím tlačítka *Další* nebo *Start*. V tomto případě se opět začne zobrazovat další krok.

3.4.5 Zastavení zobrazování

Zobrazování, které ještě není dokončeno je možné ukončit tlačítkem *Stop*. Tím je ukončeno celé zobrazování, v kterém již není možno pokračovat.

3.5 Zvětšování a zmenšování grafu

Graf je možné zvětšovat a zmenšovat pomocí *lišty pro zvětšování grafu*. Ta se nachází v levém dolním rohu okna. Pokud chcete graf zvětšit, posuňte ukazatel na liště směrem doprava. Při posunu do leva, se bude graf zmenšovat.

3.6 Zrychlování a zpomalování jednotlivých kroků algoritmu

Zrychlit nebo zpomalit zobrazování jednotlivých kroků algoritmu je možné pomocí *lišty pro rychlost zobrazování kroků*. Pokud popotáhnete ukazatel na lište doprava, doba mezi jednotlivými kroky se zkrátí a tím se zobrazování jednotlivých kroků zrychlí. Pokud ukazatel posunete doleva, interval čekání mezi jednotlivými kroky se prodlouží a zobrazování tedy bude probíhat pomaleji.

3.7 Ukládání grafu

Graf je možné uložit po kliknutí na *Soubor* a následně na položku *Uložit*. Otevře se okno, ve kterém vyberte kam a pod jakým názvem chcete soubor s grafem uložit. Pak již stačí kliknout jen na tlačítko *Uložit* a graf se uloží.