ARCIBISKUPSKÉ GYMNÁZIUM

Korunní 586/2 | Vinohrady | 120 00 Praha 2

Maturitní práce z předmětu

IVT

SESTAVOVÁNÍ ROZVRHU SEMINÁŘŮ

Olga Cinková, 8.A

Vedoucí práce: Mgr. Lukáš Bernard
Oponent: Ing. Vladimír Nulíček, CSc.

Praha 2024

PROHLÁŠENÍ

Prohlašuji, že předložená práce je mým původním autorským dílem, které jsem vypracoval/a samostatně. Veškerou literaturu a další zdroje, z nichž jsem při zpracování čerpal/a, v práci řádně cituji a jsou uvedeny v seznamu použité literatury.
V Praze dne…….

…….
(podpis)

ABSTRAKT

Cílem mé práce bylo vyvinout software na sestavování rozvrhů seminářů na AG. Software dostává na vstupu od uživatele tabulky s daty o studentech, seminářích a profesorech školy. Výstupem je tabulka, kde je uvedeno, kterému semináři je přidělen který blok v rozvrhu.
Vstupní data ukládám jako graf, kde jsou semináře reprezentovány vrcholy. Vrcholy jsou spojeny hranou, pokud dané semináře sdílí studenty nebo profesora. Hodnota hrany stoupá s počtem sdílených studentů a profesorů.
Graf barvím tak, aby žádné vrcholy sdílející hranu neměly stejnou barvu. Barvy reprezentují seminářové bloky, protože semináře, které sdílejí příliš mnoho studentů nebo dokonce profesora, nesmějí skončit ve stejném bloku.
V původním zadání bylo i webové uživatelské rozhraní, které nebylo pro nedostatek času možné implementovat.

The goal of my work was to develop software for scheduling seminars at Archbishop Grammar School in Prague. The software receives as input from the user tables with data about students, seminars, and professors. The output is a table indicating which seminar has which block in the schedule.
I store the input data as a graph, where seminars are represented as vertices. Vertices are connected by an edge if the respective seminars share students or professors. The weight of the edge increases with the number of shared students and professors.
I color the graph so that no vertices sharing an edge have the same color. Colors represent seminar blocks because seminars that share too many students or even a professor must not end up in the same block.
Originally, the project was supposed to have a web-based user interface, which could not be implemented due to lack of time.

barvení grafu, Welshův-Powellův algoritmus, rozvrh, Python, automatizace
graph coloring, Welsh-Powell algorithm, schedule, Python, automation

OBSAH

1 ÚVOD

V této kapitole popisuji nejprve praktický problém hledání vhodného rozvrhu seminářů, který by můj program měl pomáhat řešit. Poté vysvětluji, jak problém souvisí s teorií grafů a jaké algoritmy lze při jeho řešení použít.

1.1 Problém na AG

Moje práce si klade za cíl zjednodušit tvorbu rozvrhů na AG. Vzhledem k tomu, kolik proměnných je třeba mít na vědomí, je obtížné tvořit rozvrh jen pomocí tužky a papíru. Proto jsem se rozhodla vytvořit program, který by měl tvorbu rozvrhu usnadnit.
Každý rok si studenti vyššího gymnázia vybírají z nabídky desítek seminářů, které je třeba nějakým způsobem začlenit do jejich rozvrhu. Přitom každý vyučující a každý ročník mají jiné časové možnosti a je tedy třeba najít pro každý seminář místo v rozvrhu tak, aby vyhovovalo vyučujícímu i studentům.
Je třeba dát pozor na to, aby žádný vyučující nemusel být na dvou místech zároveň a aby studenti mohli docházet na semináře, kam se přihlásili - např. aby se jim semináře z jejich výběru pokud možno navzájem nekryly.
Při sestavování je třeba především brát zřetel na požadavky učitelů. Proto má každý vyučující seznam bloků, kde se mohou objevit semináře, které učí.
Každý ročník, který se na semináře přihlašuje, má jiné požadavky ohledně rozvrhu. V následující tabulce jsou čísla bloků v rozvrhu.

	vyučovací hodina			
den	**7.**	**8.**	**9.**	**10.**
pondělí	1		2	
úterý	3		4	
středa	8			
čtvrtek	5		6	
pátek	7			

V následující tabulce je uvedeno, který ročník má mít semináře ve kterém bloku.

	vyučovací hodina			
den	**7.**	**8.**	**9.**	**10.**
pondělí	1: septima, oktáva		2: kvinta&sexta, septima, oktáva	
úterý	3: septima, oktáva		4: kvinta&sexta, septima, oktáva	
středa	8: oktáva (semináře mimo školu)			
čtvrtek	5: septima, oktáva		6: oktáva	
pátek	7: oktáva			

1.2 Jak s problémem souvisí barvení grafů

Problémy se sestavováním rozvrhu se dají řešit pomocí barvení grafů. O využití barvení grafů jsou k dispozici zajímavé články[1], [2], inspirovala jsem se zejména pracemi D. J. A. Welshe a M. B. Powella[3].

1.3 Grafová reprezentace

Jak tedy přesně souvisí barvení grafů s problémem s rozvrhy seminářů? Představme si neorientovaný graf ohodnocenými hranami. Každý vrchol grafu je ve skutečnosti jeden seminář. Pokud dva semináře sdílí studenta, zvýší se hodnota hrany mezi danými dvěma vrcholy o 1. Pokud semináře sdílí učitele, zvýší se hodnota hrany mezi nimi o 100.

1.4 Jak je třeba barvit

Cílem je, aby žádný učitel ani student nemusel být na dvou seminářích zároveň. To znamená, že musíme vrcholy grafu obarvit tak, aby spolu nesdílely hranu žádné vrcholy se stejnou barvou. Barva v našem případě reprezentuje časový blok v rozvrhu.

Optimální by bylo, aby byl graf tak obarven, aby počet barev byl stejný, jako počet bloků, které chceme mít v rozvrhu. Toho však není vždy možné dosáhnout vzhledem k počtu seminářů, žáků a vyučujících. Proto jsou hrany grafu ohodnocené.

Pokud nelze graf rovnou obarvit tak, aby počet použitých barev odpovídal počtu bloků v rozvrhu, je třeba začít zohledňovat hodnoty hran. Všimněme si, že vysoká hodnota hrany značí velký počet žáků (skóre 1 za každý překryv) nebo dokonce konkrétního učitele (skóre 100 za překryv), kterým se dané semináře budou krýt. To je nutné vzít v potaz. Máme omezený počet barev, protože máme jen určitý počet bloků v rozvrhu. Proto musíme porušit obvyklá barvicí pravidla: zde může vést hrana mezi dvěma stejně barevnými vrcholy, ale zároveň je nutné, aby takové vrcholy stejné barvy sdílely hranu co nejmenší hodnoty.

1.5 Můj algoritmus

Vzhledem k tomu, že jsem nenašla žádný článek, který by popisoval algoritmus pro barvení grafů s ohodnocenými hranami, musela jsem vyvinout vlastní postup. Můj algoritmus funguje tak, že dostane na vstupu graf, který chceme obarvit, a počet barev, které je dovoleno na barvení použít.

Nejdřív obarví graf tak, jak se to běžně dělá. Poté zkontroluje počet použitých barev. Pokud byl počet použitých barev stejný jako ten zadaný (nebo menší, což jsem v praxi nikdy nezaznamenala), skončí a vrátí obarvený graf. Pokud byl ale počet použitých barev větší než zadaný počet barev, odebere z grafu hranu, která má nejnižší ohodnocení a barvení se opakuje znovu. Celý proces se opakuje, dokud počet použitých barev není stejný jako zadaný počet bloků seminářů.

---

[1]Samarasekara, Wathsala. “Licensed under Creative Commons Attribution CC by an Application of Graph Coloring Model to Course Timetabling Problem.” International Journal of Science and Research (IJSR) ResearchGate Impact Factor, 2018, www.ijsr.net/archive/v8i12/ART20203698.pdf, https://doi.org/10.21275/ART20203698.

[2] Badoni, Rakesh P., and D.K. Gupta. “A Graph Edge Colouring Approach for School Timetabling Problems.” International Journal of Mathematics in Operational Research, vol. 6, no. 1, 2014, p. 123, https://doi.org/10.1504/ijmor.2014.057853. Accessed 7 June 2022.

[3] Welsh, D. J. A. “An Upper Bound for the Chromatic Number of a Graph and Its Application to Timetabling Problems.” *The Computer Journal*, vol. 10, no. 1, 1 Jan. 1967, pp. 85–86, https://doi.org/10.1093/comjnl/10.1.85.

1.6 Pro celou školu

Vzhledem k tomu, že má každý ročník jiné požadavky ohledně rozvrhu, musím nejprve udělat pro každý ročník samostatný graf a postupně tyto grafy barvit vyvinutým algoritmem pro barvení ohodnocených grafů.
Abych dosáhla fungujícího rozvrhu pro celou školu, musím barvit následujícím způsobem: nejdříve obarvím graf, který je pro kvintu a sextu (je to jeden graf, protože je beru jako jeden ročník). Poté vezmu ještě neobarvený graf pro septimu a do obarveného grafu kvinty a sexty přidám vrcholy z grafu septimy.
Tento nově vzniklý graf, složený z obarveného grafu pro kvintu a sextu a některých vrcholů z grafu pro septimu, znovu obarvím svým algoritmem pro barvení ohodnocených grafů. (algoritmus nepřebarvuje již obarvené vrcholy).
Do obarveného grafu přidám vrcholy z grafu pro oktávy, které v obarveném grafu ještě nejsou. Naposledy graf obarvím.

1.7 Výsledek a výstup

Výsledkem je obarvený graf, který zohlednil jak požadavky na bloky ročníků, tak požadavky žáků a učitelů. Každá barva v grafu symbolizuje jeden blok. Číslo barvy je tedy číslo bloku (vizte tabulku na začátku úvodu práce).
Výstupem programu je soubor ve formátu csv, kde jsou prvním sloupci semináře, druhý sloupec uvádí ke každému semináři číslo bloku, kam byl umístěn.

2 UŽIVATELSKÁ DOKUMENTACE

2.1 INSTALACE

2.1.1 Stažení repozitáře a instalace Pythonu

Následující postup by měl fungovat na Windows i různých distribucích Linuxu.
Stáhněte ZIP archiv repozitáře a následně ho rozbalte:
https://github.com/olgacinkova/rozvrh_seminaru/archive/master.zip
K používání programu je nutné mít nainstalovaný Python, který stáhnete zde:
https://www.python.org/downloads/
(Postupujte podle instrukcí v instalačním průvodci Pythonu. Ujistěte se, že zaškrtnete možnost "Add Python to PATH", aby byl Python přidán do systémových proměnných.)

Poté se přesuňte do složky, kam jste si nainstalovali celý software. Zadejte tento příkaz, ale "*path*" nahraďte cestou ke složce, kam se chcete přesunout.

```
cd path
```

Nakonec je nutno importovat balíčky z requirements.txt. Spusťte v terminálu následující příkaz.

```
pip install -r requirements.txt
```

2.1.2 Spouštění

Program se spouští na Linuxu i na Windows stejně. Přesuňte se do složky, kam jste si nainstalovali celý software. Zadejte tento příkaz, ale "*path*" nahraďte cestou ke složce, kam se chcete přesunout.

```
cd path
cd rozvrh_seminaru
```

Pro spuštění skriptu zadejte tento příkaz.

```
python spousteni.py
```

2.2 POUŽÍVÁNÍ PROGRAMU

2.2.1 Vstup

Programu je třeba poskytnout následující soubory, podle kterých vytváří rozvrh.
seminare.csv: tabulka, kde je na každém řádku seminář a údaje o něm
seminare_kolize.csv: tabulka, kde je pro každý seminář zapsáno, ve kterých blocích se může objevit.
zaci.csv: tabulka, kde jsou údaje o všech žácích.
zapsani.csv: tabulka s jednotlivými zápisy do seminářů
Je nutné si všimnout, že všechny vstupní soubory musí být ve formátu csv a ve všech musí být semináře stejně seřazené.

2.2.1.1 Jak strukturovat vstupní tabulky

- **seminare.csv**: Zde můžou být uloženy libovolné informace o seminářích. Je však nezbytné, aby tabulka obsahovala sloupce s názvy "id", "ucitel", "pro5","pro6", "pro7" a "pro8". Každý seminář bude mít svůj řádek. Sloupec "id" obsahuje jedinečné číselné ID seminářů, "ucitel" jméno učitele daného semináře. Jeden seminář může samozřejmě učit více učitelů, s tím program počítá - zadejte všechny učitele daného semináře do kolonky "ucitel" a jejich jména oddělte čárkou.

  Ostatní kolonky určují, pro které ročníky seminář je. Pokud je v kolonce 1, znamená to, že seminář je určen pro daný ročník. Hodnota 0 znamená, že seminář pro daný ročník

není. Např. pokud je seminář pro kvintu, tedy pátý ročník, zadejte do kolonky “pro5” hodnotu 1, ale u ostatních kolonek hodnotu 0. Seminář může být pro více ročníků.

id	ucitel	pro5	pro6	pro7	pro8
1	Jana Jirková	0	0	1	1

- **seminare_kolize.csv**: V tabulce je třeba uvést sloupce s názvem “id” a “bloky”. Ve sloupci “id” jsou ID seminářů a ve sloupci “bloky” je každému semináři, v jakých blocích je možné jej učit. Čísla bloků oddělujte čárkami.

id	bloky
1	1, 2, 4

- **zaci.csv**: Jsou relevantní pouze první 3 sloupce, kde je uloženo ID žáka, jméno žáka a třída, do které chodí.

id	jmeno	trida	login	byl
1	Jan Novák	4.A	novakj	2023-02-15 20:00:39

- **zapsani.csv**: Důležité jsou pouze sloupce s názvy “zak” a “seminar”. Je v nich uvedené, který žák (jeho ID) se přihlásil kam (ID semináře).

id	zak	seminar	ro	date
1900	203	33	0	'2023-02-13 20:00:43'
1901	203	50	0	'2023-02-13 20:00:43'
1902	203	29	0	'2023-02-13 20:00:43'

2.2.2 Výstup

Výstup programu se ukládá do jednoho souboru s názvem **vystup.csv**. Tento csv soubor má pouze dva sloupce. První sloupec reprezentuje semináře (pomocí jejich id), druhý sloupec uvádí ke každému semináři číslo bloku, kam byl umístěn.

3 PROGRAMÁTORSKÁ DOKUMENTACE

V této kapitole se věnuji detailnějšímu popisu svého programu po odborné stránce. Nejprve zmiňuji používané knihovny, , poté popisuji jednotlivé "stavební bloky" algoritmu. Vysvětluji, jak se barví grafy, a podrobněji rozebírám fungování použitého barvicího algoritmu. Následně stručně popisuji, jak funguje celý algoritmus tvorby rozvrhu seminářů. Poté popisuji objektovou strukturu programu a zmíním důležité funkce. Pak se podrobněji věnuji jednotlivým fázím algoritmu. Nakonec podrobněji vysvětluji důležité funkce a proměnné programu.

3.1 Použité knihovny

Pandas: Pandas je knihovna pro analýzu a manipulaci s daty, která poskytuje vysokoúrovňové datové struktury a nástroje pro práci s tabulkovými daty.

CSV: Knihovna CSV poskytuje funkce pro čtení a zápis dat ve formátu CSV (Comma-Separated Values), což je často používaný formát pro ukládání tabulkových dat.

Matplotlib: Matplotlib je knihovna pro vizualizaci dat v Pythonu, která umožňuje tvorbu různých druhů grafů a obrázků. Zde je použita k vizualizaci obarvených grafů.

Networkx: Networkx je knihovna pro analýzu komplexních sítí a grafů, poskytující nástroje pro manipulaci, analýzu a vizualizaci sítí.

Copy: Knihovna Copy umožňuje vytvářet hluboké kopie objektů v Pythonu, což je užitečné pro manipulaci s daty, aniž by došlo k problémům spojeným s referencemi.

Collections: Knihovna Collections poskytuje alternativní datové struktury k vestavěným datovým typům Pythonu, jako jsou OrderedDict, defaultdict, namedtuple atd.

Itertools: Knihovna Itertools poskytuje funkce pro efektivní práci s iterátory a kombinacemi, což umožňuje efektivní zpracování a manipulaci s daty. Použita v barvicí funkci.

3.2 Barvicí algoritmus

3.2.1 Popis obecného algoritmu

K barvení grafů se obvykle používá Welshův-Powellův algoritmus.
Funguje takto:
- nejdřív seřadí vrcholy sestupně podle jejich stupně.[4]
  - Poté si vezme vrchol s největším stupněm. Pojmenujme ho například A. Vrchol A obarví nejnižší možnou barvou, tedy barvou 1.[5]
  - Pak zkontroluje každý neobarvený vrchol ze seznamu vrcholů, zda sousedí s A. Pokud daný vrchol nesousedí s A ani s jiným vrcholem, který je stejně barevný jako A, obarví ho stejnou barvou.
  - V dalším kroku najde v seznamu vrcholů neobarvený vrchol, který má nejnižší stupeň ze všech neobarvených vrcholů. Tento vrchol obarví barvou, která je o jedna větší než barva, kterou barvil naposledy.
  - Tyto kroky opakuje, dokud není v grafu ani jeden neobarvený vrchol.

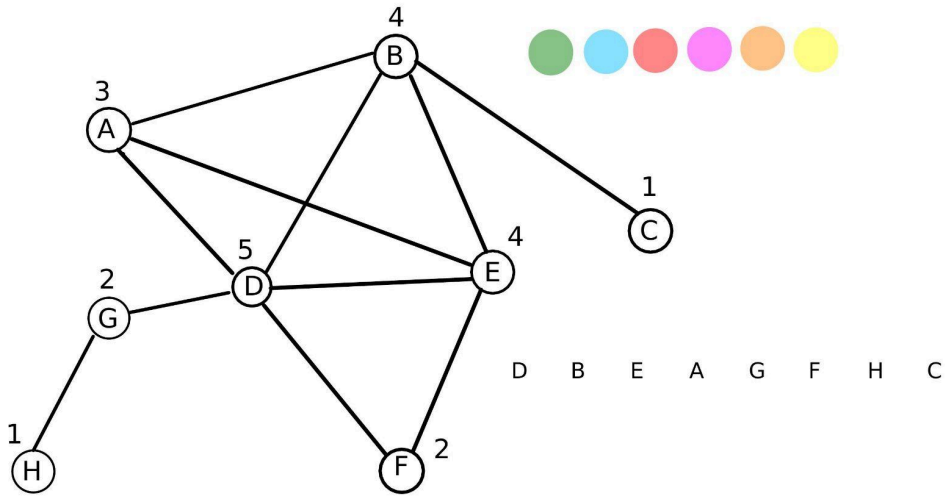
[4] Stupněm rozumějme počet hran, které zasahují do daného vrcholu.
[5] Je nutno brát v potaz, že tzv. barvy nejsou ty barvy, jaké je známe z běžného života, ale jsou to přirozená čísla.

3.2.2 Pseudokód

```
seřaď vrcholy grafu sestupně dle stupňů a ulož je do seznamu S
slovník barvy = {}, kam se budou ukládat vrcholy a jejich barvy
aktuální barva = 1
dokud nejsou obarveny všechny vrcholy v S:
      pro každý vrchol V ze S:
            pokud V ještě není obarven:
                  pokud V nesousedí s žádným vrcholem obarveným
                  aktuální barvou:
                        barvy[V] = aktuální barva
      aktuální barva += 1
vrať barvy
```

3.2.3 Příklad použití

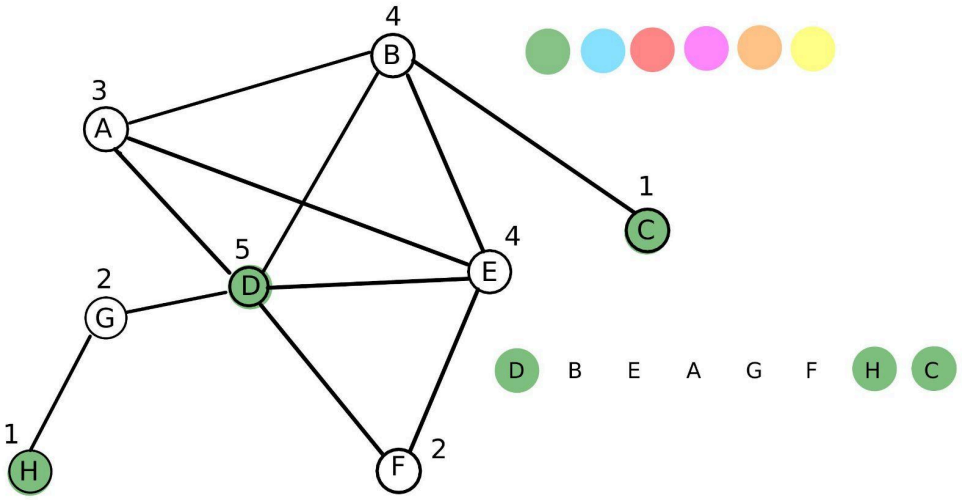
Pro lepší srozumitelnost popisovaného algoritmu uvádím příklad jeho použití. Máme následující neorientovaný graf s neohodnocenými hranami, který chceme obarvit pomocí Welshova-Powellova algoritmu.[6] Začneme tím, že seřadíme vrcholy dle jejich stupně a stanovíme si barvy,kterými budeme graf barvit.



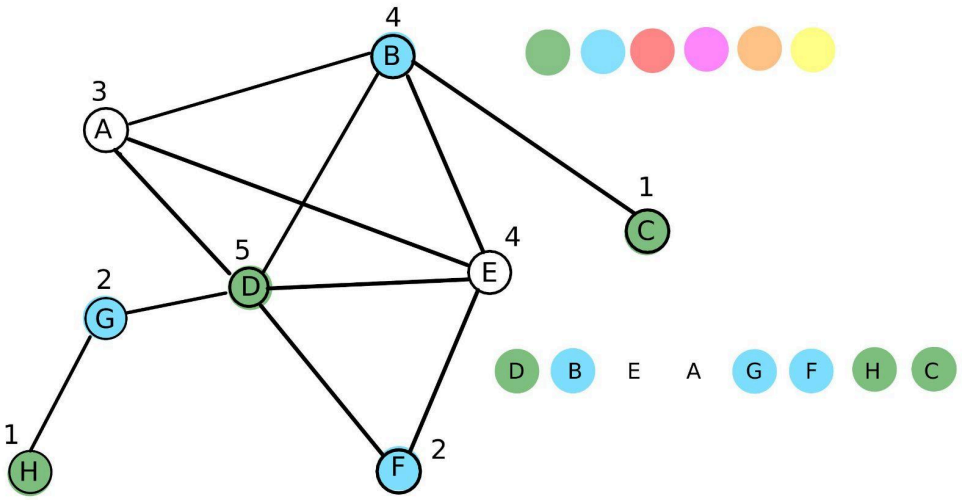
V následujícím kroku si vybereme první barvu ze seznamu, tedy zelenou. Touto barvou obarvíme první vrchol z našeho seřazeného seznamu vrcholů. První vrchol ze seznamu je D. Po obarvení vrcholu D posuneme se na další prvek ze seznamu vrcholů, který není obarvený, tedy vrchol B. Vrchol B sousedí s D, proto B nemůžeme obarvit zeleně. To stejné platí i pro E, A, G a

[6] Pro lepší pochopení budu barvit opravdovými barvami, nikoli přirozenými čísly.

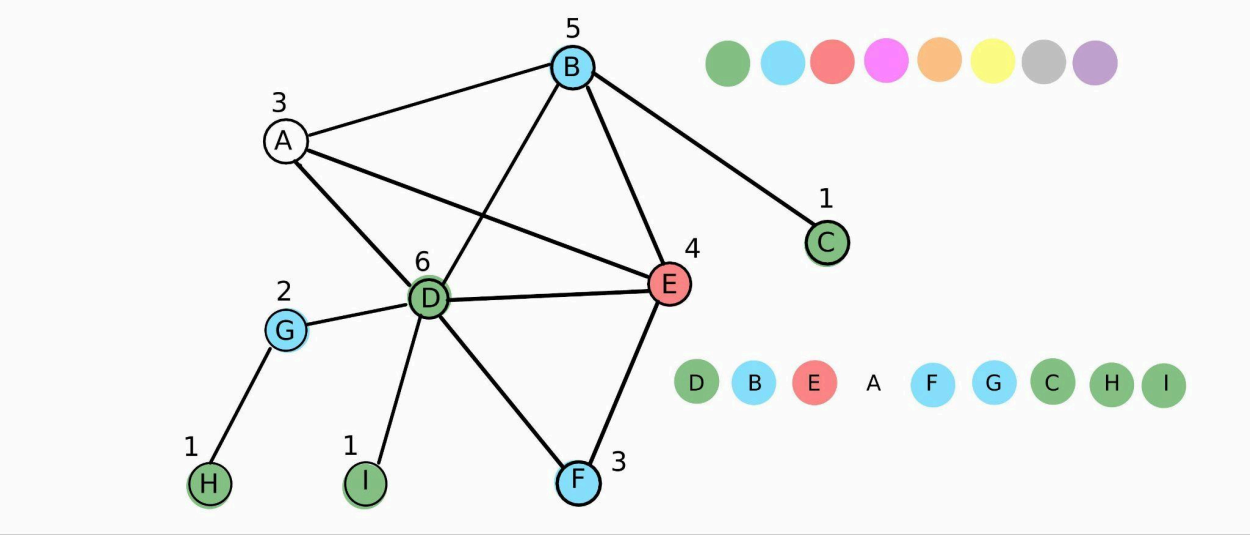
F, které jsou v seznamu dále. Vrcholy H a C však s D nesousedí, takže je obarvíme zeleně.



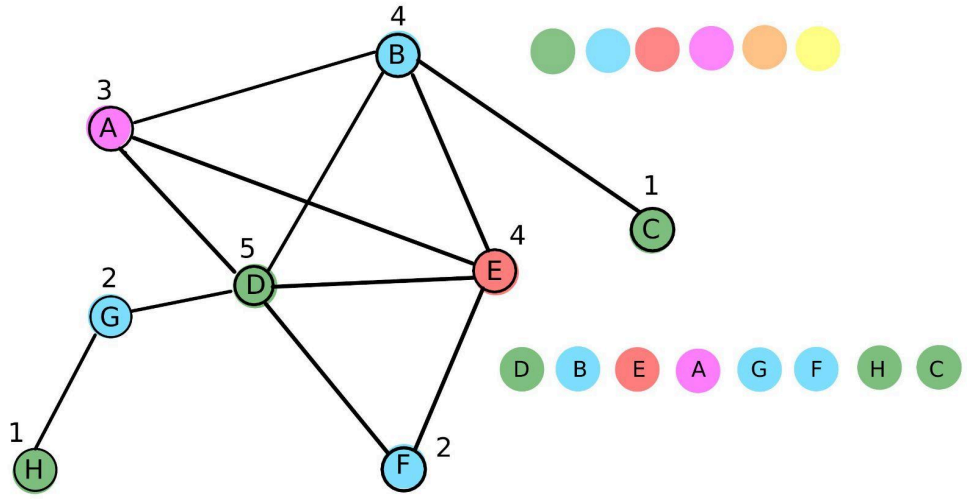
Nyní je čas si vzít ze seznamu barev další barvu a to modrou. Začneme vrcholem B, který obarvíme modře. E ani A modře obarvit nemůžeme, protože sousedí s B. Vrcholy G ani F však s B nesousedí, obarvíme je tedy modře. Všimněme si, že vrcholy C a H už jsou obarveny zeleně, takže se jimi nezabýváme.



Další barva, kterou budeme barvit, je červená. Nejdřív červeně obarvíme vrchol E, který je v seznamu první neobarvený vrchol. Poté zkontrolujeme vrchol A, zda sousedí s E. Vzhledem k tomu, že A s E sousedí, nemůžeme A obarvit červeně. Tím jsme vyčerpali všechny neobarvené vrcholy, takže se přesuneme k další barvě.



Nyní budeme barvit růžově. V seznamu nám zbývá poslední neobarvený vrchol a tím je A. Obarvíme A růžově. Máme obarvené všechny vrcholy grafu, takže je barvení dokončeno.



3.3 Zjednodušený popis průběhu programu

Algoritmus má tři části. První část zahrnuje načítání vstupu, kde program zpracovává data, poskytnutá uživatelem. Ve druhé části začíná samotné barvení, kdy program barví graf pro každý ročník zvlášť. V poslední, třetí, části se tyto obarvené grafy postupně spojují dohromady a dobarvují do finální podoby.

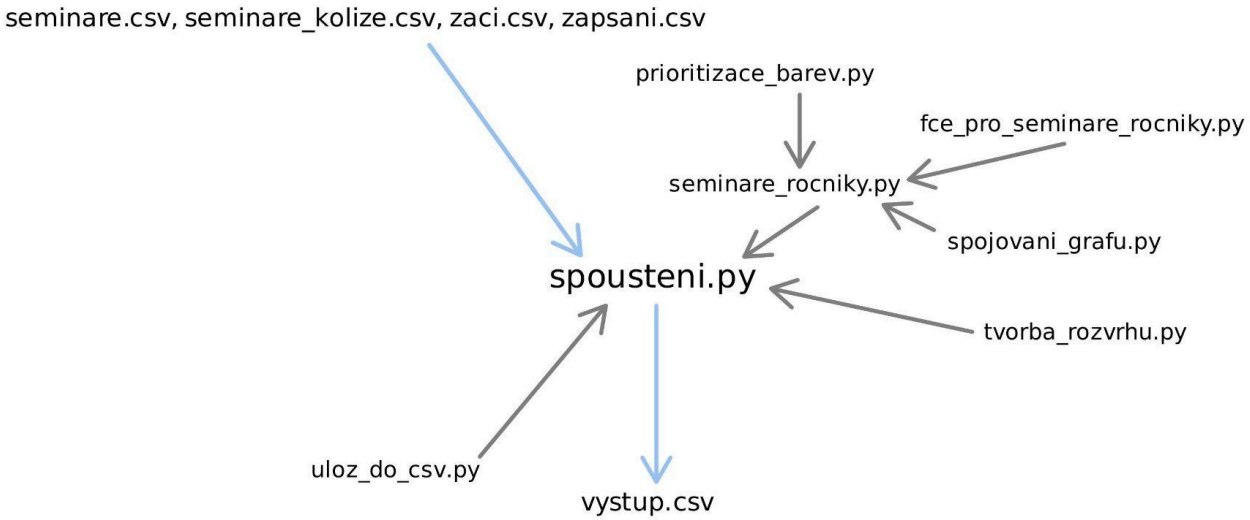
3.4 Členění programu do souborů, struktura objektů

Program je rozdělen do několika souborů, kde se nacházejí jednotlivé třídy a jejich metody.

- **spousteni.py**: spouští program
- **fce_pro_seminare_rocniky.py**: zde jsou definovány funkce, které jsou importovány do skriptu **seminare_rocniky.py**
- **seminare_rocniky.py**:
  - třída `Seminar()`: Obsahuje vše o jednotlivých seminářích.
  - třída `Rocnik()`: Obsahuje vše o jednotlivých ročnících.
- **prioritizace_barev.py**: definice funkce na prioritizované barvení
- **spojovani_grafu.py**: skript na sjednocování dvou grafů dohromady

- **tvorba_rozvrhu.py**:
  - třída `Rozvrh()`: Obsahuje informace o časovém rozložení bloků a o časových možnostech učitelů jednotlivých seminářů.
- **uloz_do_csv.py**: ukládá výstupní dictionary do csv tabulky
- **cislovani_seminaru.py**, **cislovani_seminaru_kolizi.py**: pokud nejsou id seminářů popořadě, použijte tyhle funkce, které přečíslují semináře na id od 1 do počtu seminářů

Následující schéma adresářové struktury znázorňuje, jak se soubory navzájem importují (šedé šipky) a jak proudí vstupní data (modré šipky).



3.5 Načítání vstupu

Načítání vstupu má tři fáze. V první fázi načteme ze vstupních csv tabulek data pro celou školu. Ve druhé fázi si uložíme data o jednotlivých seminářích do objektů. Ve třetí fázi se postupně data celé školy rozdělí do jednotlivých ročníků pomocí objektového členění.

3.5.1 Načtení vstupu pro celou školu

Vstupní data parsuji celá dohromady v modulu spousteni.py. Data pocházejí ze vstupních tabulek **seminare.csv**, **seminare_kolize.csv**, **zaci.csv** a **zapsani.csv**, které zadal uživatel.

3.5.2 Semináře do objektů

Vytvořím seznam, kam uložím objekty všech seminářů. Tento seznam slouží k tomu, abych později mohla snadno najít veškeré informace o jednotlivých seminářích. Objekt semináře má následující atributy:

- `id` (int): Identifikační číslo semináře.
- `pro_ktere_rocniky` (set): Množina ročníků, pro které seminář je.
- `kteri_zaci_tam_chodi` (set): Množina žáků, kteří na seminář chodí (jsou tam přihlášení).
- `kdo_seminar_uci` (set): Množina učitelů, kteří seminář učí. (Většinou jen jeden učitel na seminář, někdy dva.)

Metody objektu semináře slouží převážně k ukládání atributů, proto je nebudu detailněji rozebírat. Všechny metody objektu semináře čerpají vstupní data z dat pro celou školu.

3.5.3 Rozdělení dle ročníků

Každý ročník reprezentuji jako jednu třídu, ale kvintu a sextu slučuji.
Objekt ročníku má následující atributy:

- `__kolikaty` (int/list): Který ročník, to je. (Může být i pro dva ročníky zároveň, kvůli spojení kvinty a sexty.)
- `poradi` (list): Pořadí, ve kterém mám přiřazovat barvy vrcholům (bloky seminářům).
- `zaci` (set): Množina žáků, kteří chodí do daného ročníku.
- `ucitele` (set): Množina učitelů, kteří učí semináře daného ročníku.
- `id_seminaru_rocniku` (set): Množina ID všech seminářů ročníku.
- `ucitele_a_jejich_seminare` (dict): Dictionary, kde je vždy učitel a množina seminářů, které učí.
- `zaci_a_jejich_seminare` (dict): Dictionary, kde je vždy žák a množina seminářů, kam chodí.
- `graf` (networkx graph): Graf, kde vrcholy jsou semináře. Semináře které sdílí učitele a/nebo žáky jsou spojeny ohodnocenou hranou. Za každého sdíleného učitele se hodnota hrany zvyšuje o 100, za každého sdíleného žáka o 1.
- `graf_dict` (dict): Stejný graf jako self.graf, ale převedený do formátu dictionary dle nx.to_dict_of_dicts()
- `graf_colors` (dict): Dictionary, kde je vždy vrchol grafu a jeho barva. Jde o tzv. neobarvený graf, proto jsou barvy všech vrcholů stejné. Všechny vrcholy mají barvu 8.
- `obarveny_graf` (networkx graph): Graf self.graf po obarvení vrcholů, tak aby žádné vrcholy se stejnou barvou nebyly spojeny hranou.
- `obarveny_graf_dict` (dict): Obarvený graf převedený do formátu dictionary dle nx.to_dict_of_dicts()
- `obarveny_graf_colors` (dict): Dictionary, kde je vždy vrchol grafu a jeho barva.

Některé metody objektu ročníku slouží převážně k ukládání atributů z dat pro celou školu. Metody, které ale pracují nějakým způsobem s grafy ročníků, popíšu v následující kapitole.

3.6 Algoritmy použité při barvení

V této kapitole objasňuji svou implementaci barvení.

3.6.1 Jak barví můj program: prioritizované barvení

K důkladnému porozumění barvení je nutné pochopit funkci prioritizované barvení, kterou jsem vytvořila upravením funkce *greedy coloring* z knihovny Networkx.
Parametry funkce prioritizované barvení jsou následující:

- `G` (networkx graph): Graf, který chci barvit.
- `povolene_bloky_seminaru` (dict): Dictionary, kde je pro každý seminář, ve kterých blocích může být.
- `strategy` (str): Strategie barvení. Doporučuju nechat zde defaultní "largest_first" (je to stejné jako Welshův-Powellův algoritmus).
- `colors` (dict): Dictionary, kde je pro každý vrchol grafu jeho barva.
- `poradi` (list): Pořadí, ve kterém se při barvení přiřazují barvy.

Funkce prioritizované barvení vrací dictionary, kde je pro každý vrchol grafu jeho nová barva.
Moje prioritizované barvení se tedy od funkce greedy color z knihovny networkx podstatně neliší. Přidala jsem však povolené bloky seminářů a pořadí.

Povolené bloky seminářů umožňují při tvorbě rozvrhu zohledňovat požadavky učitelů. Povolené bloky seminářů pro jednotlivé semináře načítám ze vstupní tabulky **seminare_kolize.csv** pomocí metody `nacti_povolene_bloky_seminaru` třídy `Rozvrh()`.
Pořadí slouží k tomu, aby bylo barvení pro každý ročník jiné, protože každý ročník chce mít semináře v jiných blocích. Například septima by ideálně měla mít k dispozici 5 bloků. Ideální by bylo, kdyby se semináře septimy vešly do bloků s čísly 1, 3, 5, 2 a 4. Proto seznam s pořadím bloků pro septimu vypadá následovně: `[1,3,5,2,4,6,7,8,9,10,11,12]`. Algoritmus pro barvení tím pádem bude nejdříve barvit barvou 1, poté barvou 3 atd.
Pseudokód k algoritmu vypadá následovně:

```
vstup: graf G, povolene_bloky_seminaru,colors = {}, seznam
poradi

pro vrchol U v G:
     pokud U není v colors:
          neighbour_colors = {colors[V] for V in G[U
                              ] if V in colors}
          pro barva v itertools.cycle(poradi):
               pokud (barva není v neighbour_colors):
                    pokud barva je v povolene_bloky_seminaru[U]
                         break
          colors[U] = barva
return colors
```

3.6.2 Barvení ohodnoceného grafu

Můj algoritmus funguje tak, že dostane na vstupu graf, který chceme obarvit a počet barev, které je dovoleno na barvení použít.
Nejdřív obarví graf tak, jak se to běžně dělá. Poté zkontroluje počet použitých barev. Pokud byl počet použitých barev stejný jako ten zadaný (nebo menší, což jsem ještě nikdy nezaznamenala), skončí a vrátí obarvený graf. Pokud byl ale počet použitých barev větší než zadaný počet barev, odebere z grafu hranu, která má nejnižší ohodnocení a barvení se opakuje znovu. Celý proces se opakuje, dokud počet použitých barev není stejný jako zadaný počet.

Pseudokód vypadá takto:

```
B = počet barev, které může algoritmus použít (dostane na
vstupu)
obarvi graf
P = počet použitých barev
S = seznam hran grafu seřazených od nejmenší hodnoty po největší
dokud P > B
     odeber z S hranu s nejmenší hodnotou
     obarvi graf
     P = počet barev použitých při posledním barvení grafu
     smaž barvy z grafu - udělej z něj znovu neobarvený graf
vrať obarvený graf
```

V implementaci algoritmu na barvení ohodnoceného grafu používám výše popsané prioritizované barvení místo toho běžného.

V mém skriptu tento algoritmus používá funkce `obarvi_graf_lip()` z modulu **seminare_rocniky.py**.

3.7 Finální obarvení a zasazení do rozvrhu

3.7.1 Spojování grafů

Při finálním obarvení postupně spojuji různé grafy do jednoho. Používám k tomu vlastní funkce `merge_weighted_graphs()`, kterou popisuje následující pseudokód.

```
graf A a graf B, které dostane na vstupu
pro graf A slovník barvy_A, kde je pro každý vrchol z A jeho barva
pro graf B slovník barvy_B, kde je pro každý vrchol z B jeho barva
spojeny_graf = kopie A
spojene_barvy = kopie barvy_A

pro vrchol, sousedé v B:
     pokud je vrchol v spojeny_graf:
          pokud spojene_barvy[vrchol] == neobarvená:
               pokud vrchol in barvy_B:
                    pokud barvy_B[vrchol] == obarvená:
                         spojeny_graf[vrchol] = B[vrchol]
                         spojene_barvy[vrchol]=barvy_B[
                                                  vrchol]
     jinak:
          pokud je vrchol v barvy_B:
               pokud barvy_B[vrchol] = obarvená:
                    spojeny_graf[vrchol] = B[vrchol]
                         spojene_barvy[vrchol]=barvy_B[
                                                  vrchol]
# přidání hran do spojeného grafu
pokud vrchol, sousedé jsou v B:
     pokud vrchol není v spojeny_graf:
          pokračuj
     pro soused, atributy v sousedé:
          pokud soused není v spojeny_graf[vrchol]:
               spojeny_graf[vrchol][soused] = atributy
          jinak:
               spojeny_graf[vrchol][soused][váha]=atributy[
                                                  váha]
vrať spojeny_graf, spojene_barvy
```

3.7.2 Postupné obarvení celé školy

Poslední fází programu je obarvení celé školy, aby byl výstupem rozvrh, který bude vyhovovat celé škole.

Nejdříve obarvím graf pro kvintu a sextu zároveň. Poté do obarveného grafu kvinty a sexty přidám vrcholy z neobarveného grafu pro septimu. (Spojování grafů popisuji důkladněji výše.) Tento nově vzniklý graf obarvím. Poté do obarveného grafu přidám vrcholy z grafu pro oktávu a

znovu obarvím. Na konci mám tedy jeden obarvený graf pro celou školu. Výstupem programu je dictionary, kde je pro každý vrchol z tohoto grafu jeho barva. Dictionary nakonec převedu na soubor csv se dvěma sloupci pomocí funkce `uloz_do_csv()`

4 ZÁVĚRY

4.1 Cíle

Cílem práce bylo vytvořit software, který by usnadnil proces sestavování rozvrhu seminářů na AG. V plánu bylo vymyslet a naprogramovat algoritmus, který by rozvrh sestavoval, a následně navrhnout webové uživatelské rozhraní systému.

4.2 Výsledky

Podařilo se mi naprogramovat systém, který odpovídá zadání. Nestihla jsem však vytvořit webové uživatelské rozhraní, protože samotné programování systému se ukázalo časově mimořádně náročné.
Program byl zatím úspěšně testován na datech ze školního roku 2022/2023. Při tomto testování mě překvapilo, že se nikdy neodstranila žádná hrana a rozvrh se podařilo vměstnat do šesti bloků. To mě přivádí na myšlenku, že je buď v mém programu chyba, kterou jsem neobjevila, nebo jsou data z roku 2022/2023 něčím výjimečná. Další testování provede v budoucnu zadavatel.
Napadl mě testovací scénář, který by mohl pomoci odhalit, proč program při barvení neodstraňuje hrany. Do dat z roku 2022/2023 by byl přidán "umělý" profesor, který by vyučoval mnoho seminářů, ale mohl jen v jednom bloku. V takové případě by program měl nějaké hrany odstraňovat.

4.3 Přínos

Doufám, že můj projekt škole usnadní tvorbu rozvrhu seminářů. Věřím, že můj program má potenciál ušetřit učitelům každý rok alespoň zlomek času stráveného nad vymýšlením rozvrhů.
Mně osobně toho práce přinesla mnoho. Naučila jsem se programovat větší projekt s objektovou strukturou. Zjistila jsem, jak se správně píše dokumentace k programu a jak důležité je, aby byl kód dobře čitelný. Dozvěděla jsem se toho spoustu o teorii grafů a rozšířila si obzory četbou literatury na dané téma. Při psaní uživatelské i programátorské dokumentace jsem se učila popisovat srozumitelně algoritmy, které jsem programovala.

4.4 Zhodnocení práce

Cíle zadání práce byly splněny s výjimkou webového rozhraní. Výsledkem práce je program, který má potenciál praktického uplatnění v provozu školy.

5 SEZNAM POUŽITÉ LITERATURY

[1] Samarasekara, Wathsala. “An Application of Graph Coloring Model to Course Timetabling Problem.” International Journal of Science and Research (IJSR) ResearchGate Impact Factor, 2018, www.ijsr.net/archive/v8i12/ART20203698.pdf, https://doi.org/10.21275/ART20203698.

[2] Badoni, Rakesh P., and D.K. Gupta. “A Graph Edge Colouring Approach for School Timetabling Problems.” International Journal of Mathematics in Operational Research, vol. 6, no. 1, 2014, p. 123, https://doi.org/10.1504/ijmor.2014.057853. Accessed 7 June 2022.

[3] Welsh, D. J. A. “An Upper Bound for the Chromatic Number of a Graph and Its Application to Timetabling Problems.” *The Computer Journal*, vol. 10, no. 1, 1 Jan. 1967, pp. 85–86, https://doi.org/10.1093/comjnl/10.1.85.