UNIVERZITA KOMENSKÉHO V BRATISLAVE

FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

Obrázok, na ktorom je nočná obloha

Automaticky generovaný popis

TOKEN GRAFY

Bakalárska práca

2024 Timotea Chalupová

UNIVERZITA KOMENSKÉHO V BRATISLAVE

FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

Obrázok, na ktorom je nočná obloha

Automaticky generovaný popis

TOKEN GRAFY

Bakalárska práca

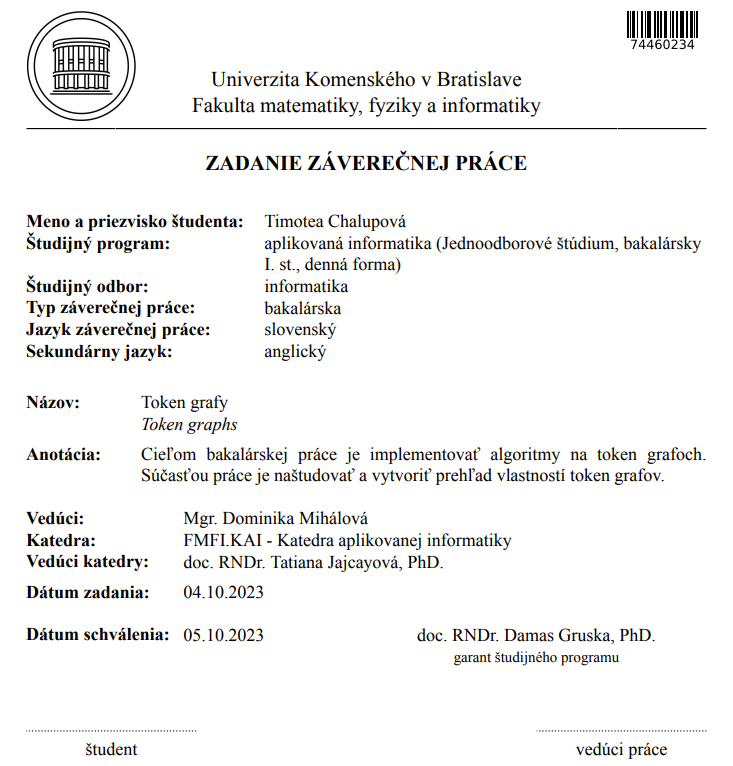
Študijný program: Aplikovaná informatika

Študijný odbor: Informatika

Školiace pracovisko: Katedra aplikovanej informatiky

Školiteľ: Mgr. Dominika Mihálová

Bratislava, 2024 Timotea Chalupová



**ČESTNÉ PREHLÁSENIE**

Čestne prehlasujem, že bakalársku prácu som vypracovala samostatne, len s použitím uvedenej literatúry a za pomoci konzultácií mojej školiteľky.

........................................................

Bratislava, 2024 Timotea Chalupová

**POĎAKOVANIE**

...

abstrakt

abstract

Obsah

[Úvod 1](#_Toc165923573)

[1. Základné pojmy 2](#_Toc165923574)

[1.1. Jednoduchý graf 2](#_Toc165923575)

[1.2. Pravidelný graf 2](#_Toc165923576)

[1.3. Cesty a cykly 2](#_Toc165923577)

[1.4. Hranová a vrcholová súvislosť 3](#_Toc165923578)

[1.5. Farbenie grafov 4](#_Toc165923579)

[1.6. Obvod 4](#_Toc165923580)

[1.7. Eulerova cesta a cyklus 4](#_Toc165923581)

[1.8. Hamiltonovská cesta 4](#_Toc165923582)

[1.9. Izomorfizmus 5](#_Toc165923583)

[1.10. Strom 5](#_Toc165923584)

[1.11. Planárny graf 5](#_Toc165923585)

[1.12. Úplný graf 6](#_Toc165923586)

[1.13. Johnsonov graf 6](#_Toc165923587)

[1.14. Token grafy 6](#_Toc165923588)

[1.2.1.Základné vlastnosti 7](#_Toc165923589)

[2. Prehľad technológií 8](#_Toc165923590)

[2.1. Existujúce systémy 8](#_Toc165923591)

[2.2. Knižnice 9](#_Toc165923592)

[2.3. Programovací jazyk 9](#_Toc165923593)

[2.4. Špecifické požiadavky 9](#_Toc165923594)

[2.4.1. Funkcionálne požiadavky 9](#_Toc165923595)

[2.4.2. Kvalitatívne požiadavky 10](#_Toc165923596)

[2.4.3. Nefunkcionálne požiadavky 10](#_Toc165923597)

[3. Návrh 11](#_Toc165923598)

[4. Implementácia 12](#_Toc165923599)

[4.1. Triedy 12](#_Toc165923600)

[4.1.1. Trieda App 12](#_Toc165923601)

[4.1.2. Trieda Graph 13](#_Toc165923602)

[4.1.3. Entitno-relačný model 14](#_Toc165923603)

[4.1.4. Screenshoty z aplikácie 14](#_Toc165923604)

[5. Testovanie 15](#_Toc165923605)

[Záver 16](#_Toc165923606)

[Použitá literatúra 17](#_Toc165923607)

Zoznam obrázkov

[Obrázok 1: Cesty v grafe 3](file:///C:\Users\Tima\Desktop\Bakalarka\Bakalarska%20praca.docx#_Toc165923608)

[Obrázok 2: Hranová a vrcholová súvislosť 4](file:///C:\Users\Tima\Desktop\Bakalarka\Bakalarska%20praca.docx#_Toc165923609)

[Obrázok 3: Planárny graf 5](file:///C:\Users\Tima\Desktop\Bakalarka\Bakalarska%20praca.docx#_Toc165923610)

[Obrázok 4: Úplné grafy 6](file:///C:\Users\Tima\Desktop\Bakalarka\Bakalarska%20praca.docx#_Toc165923611)

[Obrázok 5: 2-token graf zo 6-vrcholovej cesty 7](file:///C:\Users\Tima\Desktop\Bakalarka\Bakalarska%20praca.docx#_Toc165923612)

[Obrázok 6: Screenshot aplikácie Gephi 8](file:///C:\Users\Tima\Desktop\Bakalarka\Bakalarska%20praca.docx#_Toc165923613)

# Úvod

V dnešnom rýchlo vyvíjajúcom sa svete, plnom rôznych informačných technológií, je dôležité hľadať nové algoritmy a dátové štruktúry, ktoré môžu nájsť uplatnenie nielen v teoretickej informatike ale aj v praxi. V matematike, v informatike a rovnako aj v reálnom svete sa veľké množstvo problémov dá znázorniť pohybom objektov po vrcholoch grafu. Z toho dôvodu sú token grafy významnou matematickou štruktúrou, ktorá nachádza využitie v analýze grafov, grafovej teórii a distribuovaných systémoch. Ich výskum a analýza môžu poskytnúť užitočné poznatky pre optimalizáciu algoritmov.

...

V prvej kapitole si objasníme základné pojmy z teórie grafov, ktoré sú nevyhnutné pre porozumenie danej problematike. (Spomenieme termíny ako sú ...). Taktiež sa pozrieme na porovnanie technológií

V druhej kapitole si priblížime

V tretej....

Cieľom je...

# Základné pojmy

V tejto kapitole vysvetlíme základné pojmy a definície, ktoré sú nevyhnutné pre vypracovanie našej práce.

## Jednoduchý graf

**Definícia:** Jednoduchý graf je usporiadaná dvojica množín , kde  je neprázdna množina vrcholov , a , množina hrán , je množina dvojíc vrcholov. Každá hrana môže byť vyjadrená ako , kde  a  sú odlišné vrcholy, t. j. , [4, s. 497].

Jednoduchý graf je jedným zo základných pojmov v teórii grafov. Neformálne napísane jednoduchý predstavuje matematickú štruktúru, ktorá sa skladá z množiny vrcholov a množiny hrán. V tomto type grafu sa nenachádzajú žiadne zložitejšie prvky, ako sú slučky alebo viacnásobné hrany.

## Pravidelný graf

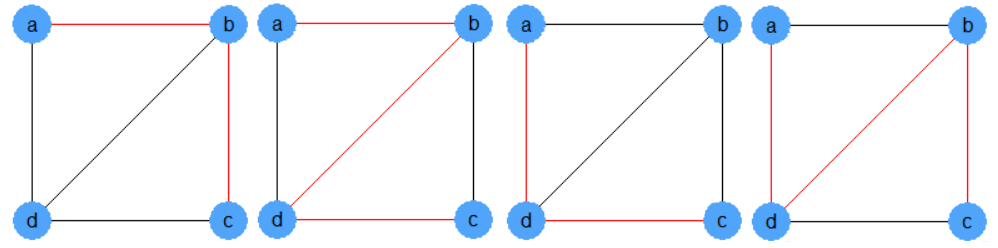
**Definícia:** Ak je vrcholom grafu , potom stupeň označený ako , je počet hrán pripadajúcich na , pričom každá slučka sa počíta dvakrát. Jednoduchý graf, v ktorom majú všetky vrcholy rovnaký stupeň sa nazýva pravidelný graf, presnejšie -pravidelný graf [4, s. 499].

Pre jednoduchý graf, stupeň vrcholu je číslo vyjadrujúce počet susedov tohto vrcholu. To znamená že v -pravidelnom grafe má každý vrchol presne susedov, pričom je z intervalu 0 až .

## Cesty a cykly

**Definícia:** Predpokladajme že je graf a sú dvojice vrcholov. Cesta v  z  do je striedavá postupnosť vrcholov a hrán: pri čom koncové body hrany sú vrcholy , pre , a . Hovoríme že cesta prechádza cez vrcholy a prechádza hranami a cesta má dĺžku , nakoľko prechádza  hranami [4, s. 540]. Cesta sa nazýva cyklus ak začína a končí v tom istom vrchole, čiže ak a jej dĺžka je väčšia ako nula, takže ak . Ak cesta alebo cyklus neobsahuje žiadnu z hrán viac ako jeden raz, hovoríme o jednoduchej ceste respektíve o jednoduchom cykle [5, s. 679].

Na obrázku 1 môžeme vidieť že v grafe môže z vrcholu  do vrcholu existovať viacero ciest. V našom prípade hľadáme cestu z vrcholu  do vrcholu . Takéto cesty existujú štyri a sú to: ; ; a cesta



Obrázok 1: Cesty v grafe

## Hranová a vrcholová súvislosť

**Definícia:** Nech je súvislý graf. Množinu A:

nazývame vrcholovým rezom grafu , ak graf je nesúvislý.

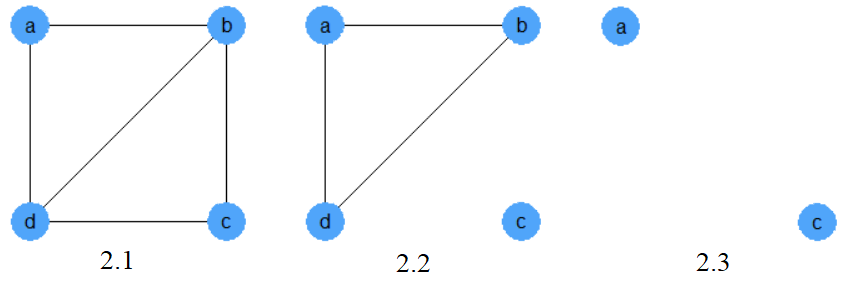
nazývame hranovým rezom grafu , ak graf je nesúvislý.

**Definícia:** Minimálna veľkosť hranového rezu sa nazýva hranová súvislosť grafu , označujeme . Graf sa nazýva k-hranovo súvislý, ak . Minimálna veľkosť vrcholového rezu sa nazýva vrcholová súvislosť grafu , označujeme . Graf sa nazýva k-vrcholovo súvislý, ak [6, s. 8].

Hranová súvislosť je teda minimálny počet hrán potrebných vymazať aby sme dostali neprepojené grafy.

Podobne vrcholová súvislosť predstavuje minimálny počet vrcholov, ktorých odstránením dostaneme neprepojené grafy.

Obrázok 2 sa skladá z troch častí, na obrázku 2.1 sa nachádza graf , ktorý rozdelíme na dva grafy pomocou hranového rezu (obrázok 2.2) a pomocou vrcholového rezu (obrázok 2.3). Pri hranovom reze sme odstránili dve hrany a to a , takže hranová súvislosť . Pri vrcholovom reze sme odstránili dva vrcholy a  a príslušné hrany , a , vrcholová súvislosť .



Obrázok 2: Hranová a vrcholová súvislosť

## Farbenie grafov

**Definícia:** Pod pojmom vyfarbovanie jednoduchého grafu rozumieme priradenie farby každému vrcholu grafu tak, aby žiadne dva susedné vrcholy nemali priradenú rovnakú farbu [5, s. 727].

**Definícia:** Chromatické číslo grafu je najmenší počet farieb potrebných na zafarbenie vrcholov tak, aby žiadne dva susedné vrcholy nemali rovnakú farbu t.j. najmenšia hodnota , ktorú je možné dosiahnuť -farbením [8, s. 210].

## Obvod

**Definícia:** Obvod grafu označený ako je dĺžka najmenšieho cyklu v . Ak neexistuje v  žiaden cyklus [7].

## Eulerova cesta a cyklus

**Definícia:** Eulerov cyklus v grafe je jednoduchý cyklus obsahujúci každú hranu v . Eulerova cesta v  je jednoduchá cesta obsahujúca každú hranu v  [5, s. 694].

## Hamiltonovská cesta

**Definícia:** Jednoduchá cesta v , ktorá prechádza cez každý vrchol práve raz, sa nazýva Hamiltonovská cesta, a jednoduchý cyklus v  ktorý prechádza každým vrcholom práve raz sa nazýva Hamiltonovský cyklus alebo aj Hamiltonovská kružnica. Inak povedané, jednoduchá cesta v grafe je Hamiltonovská cesta ak a  pre , a jednoduchý cyklus (kde ) je Hamiltonovský cyklus ak je Hamiltonovská cesta [5, s. 698].

## Izomorfizmus

**Definícia:** Jednoduché grafy a  sú izomorfné ak existuje bijektívna funkcia z  do s vlastnosťou že a  sú susedné v  ak a iba ak a  sú susedné v , pre všetky a  vo . Takáto funkcia sa nazýva izomorfizmus [5, s. 672].

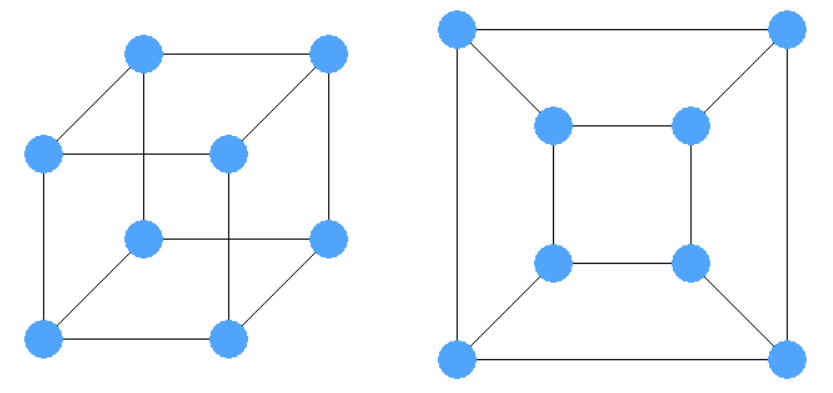
## Strom

**Definícia:** Strom je spojený neorientovaný graf ktorý nemá žiadne jednoduché cykly [5, s. 746].

## Planárny graf

**Definícia:** Planárny alebo inak nazývaný aj rovinný graf je taký graf ktorý vieme nakresliť v rovine bez prekrývania hrán. Nákres takéhoto grafu voláme planárna alebo rovinná reprezentácia grafu [5, s. 719].

Na obrázku 3 sa na ľavej strane nachádza graf , na pravej strane vidíme planárnu reprezentáciu grafu .

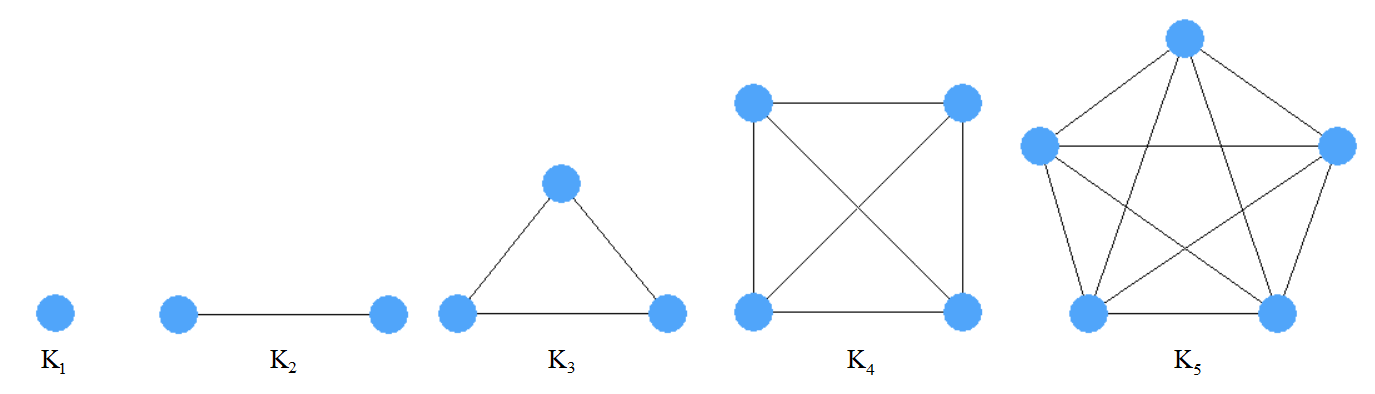


Obrázok 3: Planárny graf

## Úplný graf

**Definícia:** Úplný graf na vrcholoch, označovaný ako , je jednoduchý graf, ktorý obsahuje presne jednu hranu medzi každým párom rôznych vrcholov [5, s. 655].

Na obrázku 4 príklad úplných grafov pre .



Obrázok 4: Úplné grafy

## Johnsonov graf

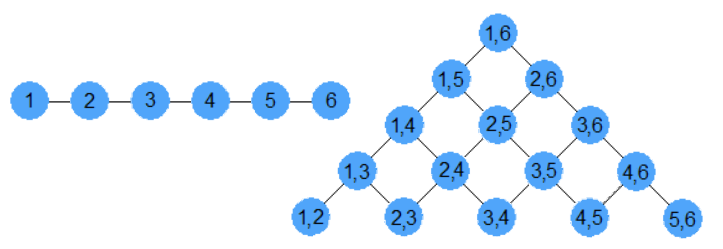
**Definícia:** Nech a , Johnsonov graf je definovaný ako:

1. Množina vrcholov je množina všetkých podmnožín s mohutnosťou presne .
2. Dva vrcholy sú susedné práve vtedy, keď symetrický rozdiel príslušných množín je dva.

## Token grafy

Pre graf a celé číslo , definujeme token graf ako graf s vrcholovou množinou , kde dva vrcholy a , grafu sú susedné, keď ich symetrický rozdiel je dvojica taká, že , a . Teda vrcholy v zodpovedajú konfiguráciám nerozlíšiteľných tokenov umiestnených na odlišných vrcholoch , pričom dve konfigurácie sú susedné, keď jedna konfigurácia môže byť dosiahnutá z druhej, posunutím jedného tokenu pozdĺž hrany z jeho súčasnej pozície na neobsadený vrchol. Preto nazývame ako -token graf grafu [9, s. 2].

Na obrázku 4 môžeme vidieť príklad 2-token grafu, ktorý vznikol zo šesť-vrcholovej cesty.



Obrázok 5: 2-token graf zo 6-vrcholovej cesty

### 1.2.1.Základné vlastnosti

Majme graf s  vrcholmi a  je kladné celé číslo. Aby sme sa vyhli triviálnym prípadom, budeme predpokladať že . Počet vrcholov je:

Na vypočítanie počtu hrán , každej hrane z  pridelíme hranu z , pre ktorú platí . Počet hrán v grafe , ktoré sú priradené k  je rovný . Preto

Susedia každého vrcholu z  sú

Takže stupeň vrcholu z  je rovný počtu hrán medzi a . Z toho vyplývajú ohraničenia na minimálny a maximálny stupeň .

S použitím iba jedného tokenu je výsledný token graf izomorfný s . Preto

Dva vrcholy a sú susedné v  práve vtedy, keď a  sú susedné v ,

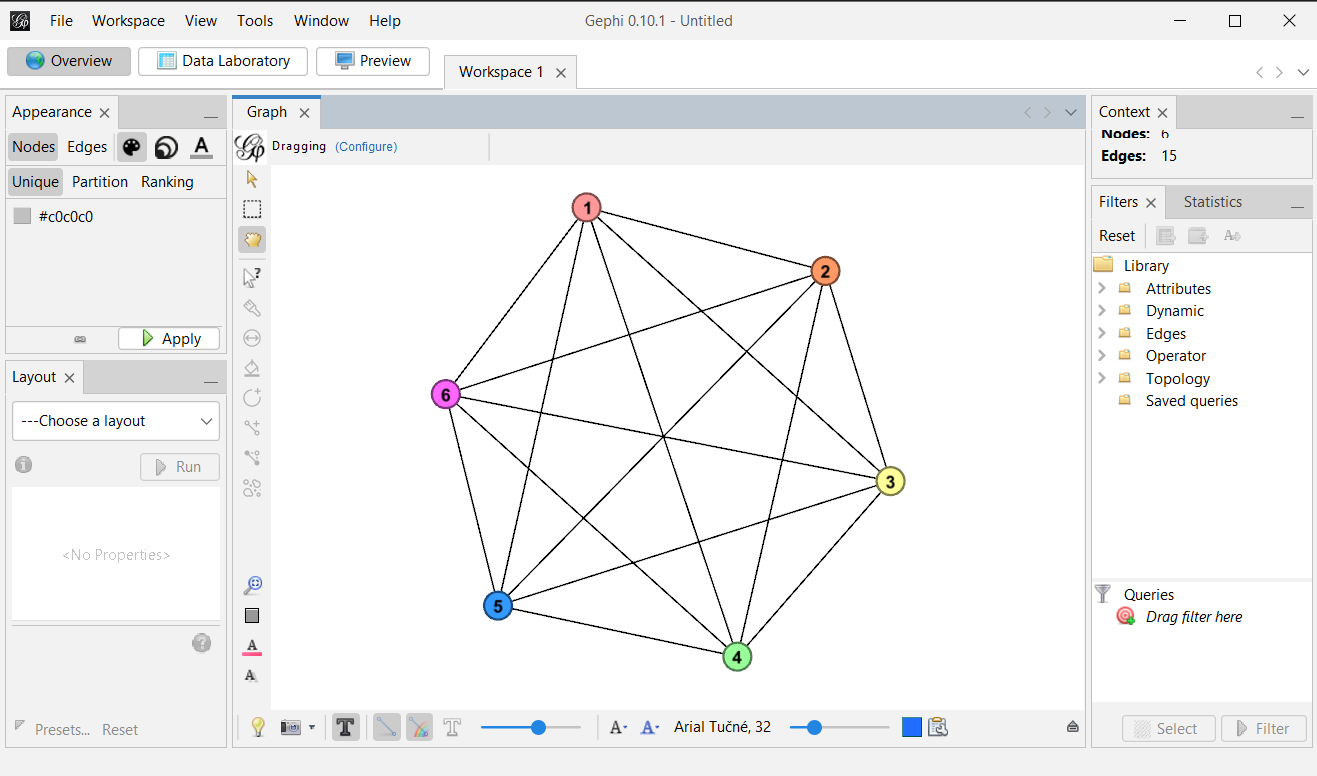
[9, s. 3]

# Prehľad technológií

V tejto kapitole sa zameriame na existujúce systémy, knižnice a programovacie jazyky, ktoré by sme vedeli využiť pri tvorbe našej aplikácie na analýzu token grafov. Ďalej si priblížime funkcionálne a nefunkcionálne požiadavky.

## Existujúce systémy

**Gephi** je open-source nástroj na analýzu grafov, ktorý ponúka široké možnosti manipulácie a vizualizácie dát. Zabezpečuje rýchlu a efektívnu vizualizáciu aj pre veľké siete s až 100 000 uzlami a 1 000 000 hranami. S využitím algoritmov rozloženia grafu dokáže Gephi efektívne a kvalitne tvarovať grafy, čo je kľúčové pre ich čitateľnosť a porozumenie. Rozsiahly rámec štatistík umožňuje používateľom hlbšie analyzovať siete a získavať relevantné informácie z dát. Okrem toho poskytuje možnosti dynamického filtrovania, čo umožňuje sledovať vývoj siete v čase a analyzovať zmeny [10].



Obrázok 6: Screenshot aplikácie Gephi

Jednou z nevýhod ktoré aplikácia Gephi má, je to že vytvorenú sieť nie je možné exportovať ako HTML alebo obrázok. Avšak pre našu prácu je najpodstatnejšie to, či ponúka algoritmy pre token grafy, na čo existuje jednoduchá odpoveď, nie neponúka.

## Knižnice

**NetworkX** je open-source knižnica pre jazyk Python, používaná najmä na vytváranie, manipuláciu a študovanie štruktúry, dynamiky a funkcií grafových štruktúr. Poskytuje veľké množstvo algoritmov na analýzu, ako sú vzdialenosti medzi uzlami, hľadanie najkratšej cesty, hľadanie najmenšieho cyklu a mnoho ďalších. Zaujímavosťou je, že vrcholom grafu môže byť čokoľvek, od textového reťazca až po obrázky [2].

**Tkinter** je open-source knižnica pre jazyk Python, určená predovšetkým na tvorbu používateľského rozhrania pre desktopové aplikácie. Vývojárom poskytuje množstvo nástrojov na vytváranie, manipuláciu a správu grafických komponentov, ako sú napríklad tlačidlá alebo polia na zadávanie textu. Tkinter je schopný práce s viacvláknovým prostredím, čo umožňuje efektívne riadenie viacerých úloh súčasne. Je obľúbený hlavne vďaka jednoduchej syntaxi a intuitívnemu používaniu [3].

## Programovací jazyk

**Python** je vysokoúrovňový interpretovaný jazyk. Medzi jeho základné vlastnosti patrí jednoduchá syntax, ktorá zlepšuje čitateľnosť. Výhodou je veľké množstvo knižníc slúžiace na prácu s webovými aplikáciami, s vývojom hier ale aj databázami a mnoho ďalšími. Taktiež je multiplatformový, takže aplikácia naprogramovaná v tomto jazyku môže byť spustená na zariadeniach s rôznymi operačnými systémami bez potreby upravovať kód. Python je na rozdiel od staticky typovaných jazykov, kde je potrebné vopred deklarovať typy všetkých dát, typovaný dynamicky [1].

## Špecifické požiadavky

### Funkcionálne požiadavky

**Vytvorenie grafu:** Používateľ zadá vrcholy a hrany grafu, z týchto údajov sa následne vygeneruje graf.

**Vytvorenie token grafu:** Používateľ zadá vrcholy grafu, hrany grafu a číslo označujúce počet tokenov. Z týchto údajov sa následne vygeneruje token graf.

**Export grafu:** Exportovanie údajov potrebných na vygenerovanie grafu, čiže vrcholy a hrany grafu.

**Načítanie grafu:** Aby sa predišlo potrebe opätovne zadávať vrcholy a hrany grafu, ktorý už si používateľ raz vygeneroval a exportoval, vyberie súbor zo zariadenia v ktorom sa nachádzajú potrebné údaje na vygenerovanie grafu.

**Export grafu ako obrázok:** Graf bude možné uložiť ako obrázok v rôznych formátoch.

**Zmena rozloženia grafu:** Aby sa zabezpečila väčšia prehľadnosť grafu, bude možné meniť rozloženie grafu.

**Spustenie algoritmov na grafe:** Po vygenerovaní grafu sa budú dať spúšťať algoritmy ako hľadanie cyklu, hľadanie najkratšej cesty, Hamiltonovskej cesty, a mnohé ďalšie.

### Kvalitatívne požiadavky

**Generovanie grafov:** Keďže algoritmy na grafoch môžu byť časovo náročné, je potrebné zvoliť si správnu štruktúru na ukladanie grafov a čo najefektívnejšie algoritmy.

### Nefunkcionálne požiadavky

Desktopová aplikácia

Backend postavený na jazyku Python a knižnici NetworkX

Frontend postavený na knižnici Tkinter

# Návrh

...

# Implementácia

V tejto kapitole sa zameriame na použité algoritmy a na samotný program. Programovací jazyk a knižnice, ktoré sú zadané v požiadavkách sú popísané v predchádzajúcej kapitole. Začneme teda s vybranými algoritmami, pokračovať budeme triedami a metódami, a nakoniec si ukážeme výsledné používateľské rozhranie.

## Triedy

Aplikáciu sme si rozdelili na dve triedy a to trieda App a trieda Graph.

### Trieda App

Táto trieda rieši grafickú časť, ako je vykresľovanie grafov, tlačidlá a udalosti.

Metóda \_\_init\_\_ vytvorí plátno podľa veľkosti obrazovky na ktorej je aplikácia spustená. Zadefinuje sa v nej atribút self.graphs, ktorý je inštanciou triedy Graph, v ktorom bude uložený graf a token graf. Zároveň sa na plátno umiestnia dve tlačidlá, na vytvorenie nového grafu a načítanie existujúceho grafu.

Metóda new\_graph sa zavolá po stlačení tlačidla na vytvorenie nového grafu. Používateľovi sa zobrazia polia na zadanie vrcholov a hrán, a klikne na jednu z možností: vytvoriť graf alebo vytvoriť token graf.

Metóda load\_graph otvorí okno v ktorom si používateľ vyberie súbor zo zariadenia, načíta z neho hrany a vrcholy, ktoré možné dodatočne upraviť a rovnako ako pri vytváraní nového grafu používateľ vyberie či chce vytvoriť zo zadaných údajov graf alebo token graf.

Metóda button\_click skontroluje či používateľ zadal nejaké vrcholy a volá metódu z triedy Graph, ktorá ďalej vytvorí graf, a volá metódu prepare\_to\_draw.

Metóda button\_click\_token rovnako ako predchádzajúca metóda skontroluje či používateľ zadal nejaké vrcholy. Ďalej sa opýta na počet tokenov a volá metódu z triedy Graph, ktorá ďalej vytvorí token graf, a volá metódu prepare\_to\_draw.

Metóda prepare\_to\_draw vyčistí plochu, vypočíta pozície vrcholov grafu a na plochu pridá tlačidlá na uloženie grafu, na uloženie obrázku grafu a tlačidlo späť a volá funkciu draw.

Metóda draw sa stará o samotné vykreslenie grafu. Vrcholy umiestni na predom vypočítané pozície.

Metóda button\_save uloží graf ako vo formáte obrázku do zvoleného priečinku.

Metóda button\_save\_graph uloží konfiguráciu grafu do zvoleného priečinku.

Metóda button\_back vráti používateľa na predchádzajúcu plochu, bez toho aby stratil vyplnené vrcholy a hrany.

Metóda move sa zavolá pri udalosti ťahania myšou. Ak používateľ chytil niektorý z vrcholov, prepočítajú sa jeho súradnice a zavolá sa metóda draw.

(+ )

### Trieda Graph

Táto trieda rieši funkčnú časť, ako je ukladanie grafov a vykonávanie algoritmov na grafoch.

Metóda \_\_init\_\_ zadefinuje dva atribúty self.g a self.token v ktorých budú uložene grafy.

Metóda parse spracuje vstup od používateľa.

Metóda to\_token zoberie údaje od používateľa ako sú vrcholy, hrany a počet tokenov a vygeneruje príslušný token graf.

Metóda is\_eulerian ...

Metóda is\_tree ...

Metóda is\_regular ...

... (ostatné algoritmy)

### Entitno-relačný model

!!! Graf potrebuje ešte upraviť

Obrázok, na ktorom je náčrt, diagram, kresba, obrysy

Automaticky generovaný popis

Obrázok 7: Entitno-relačný model

### Screenshoty z aplikácie

...

# Testovanie

# Záver

# Použitá literatúra

[1] <https://www.python.org/doc/>

[2] <https://networkx.org/documentation/stable/>

[3] <https://docs.python.org/3/library/tkinter.html>

[4] kniha

[5] Kenneth H. Rosen, Discrete Mathematics and Its Applications

[6] <https://edu.fmph.uniba.sk/~winczer/diskretna/pred8z03.pdf>

[7] <http://people.qc.cuny.edu/faculty/christopher.hanusa/courses/634sp12/Documents/634sp12ch1-4.pdf>

[7] Implementing discrete mathematics: combinatorics and graph theory with Mathematica, Steven Skiena. Pp 334. 1990. ISBN 0-201-50943-1 (Addison-Wesley)

[9] 0910.4774

[10] https://gephi.org/features/

[11]

[12]

[13]

[14]

[15]

[16]

[17]

[18]

[19]

[20]

[21]