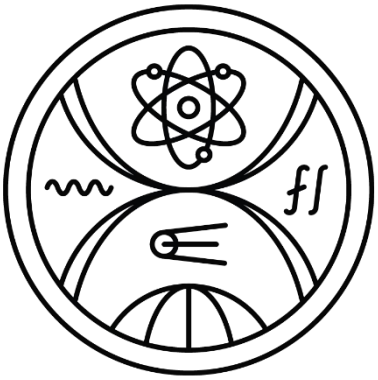
UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

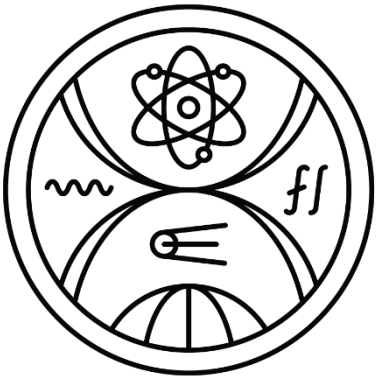


TOKEN GRAFY

Bakalárska práca

2024 Timotea Chalupová

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY



TOKEN GRAFY

Bakalárska práca

Študijný program: Aplikovaná informatika
Študijný odbor: Informatika
Školiace pracovisko: Katedra aplikovanej informatiky
Školiteľ: Mgr. Dominika Mihálová

Bratislava, 2024 Timotea Chalupová

74460234

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Timotea Chalupová
Študijný program: aplikovaná informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Token grafy
Token graphs

Anotácia: Cieľom bakalárskej práce je implementovať algoritmy na token grafoch. Súčasťou práce je naštudovať a vytvoriť prehľad vlastností token grafov.

Vedúci: Mgr. Dominika Mihálová
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: doc. RNDr. Tatiana Jajcayová, PhD.

Dátum zadania: 04.10.2023

Dátum schválenia: 05.10.2023

doc. RNDr. Damas Gruska, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

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

Zoznam obrázkov

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...

1. ZÁKLADNÉ POJMY

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

1.1. Jednoduchý graf

Definícia: Jednoduchý graf je usporiadaná dvojica množín $G = (V, E)$, kde V je neprázdna množina vrcholov G, a E, množina hrán G, je množina dvojíc vrcholov. Každá hrana G môže byť vyjadrená ako $\{u, v\}$, kde u a v sú odlišné vrcholy, t. j. $u, v \in V$, $u \neq v$ [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.

1.2. Pravidelný graf

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

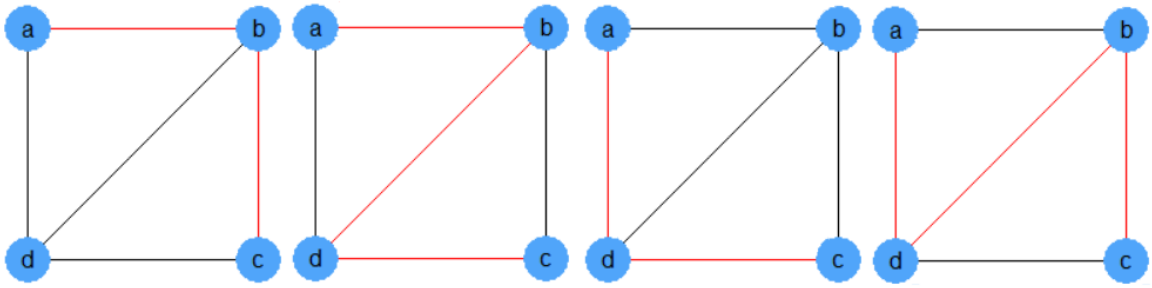
Pre jednoduchý graf, stupeň vrcholu je číslo vyjadrujúce počet susedov tohto vrcholu. To znamená že v k-pravidelnom grafe má každý vrchol presne k susedov, pričom k je z intervalu 0 až $|V(G)| - 1$.

1.3. Cesty a cykly

Definícia: Predpokladajme že $G = (V, E)$ je graf a $v, w \in V$ sú dvojice vrcholov. Cesta v G z v do w je striedavá postupnosť vrcholov a hrán: $P = \langle v_0, e_1, v_1, e_2, v_2, \dots, v_{k-1}, e_k, e_k \rangle$ pri čom koncové body hrany e_i sú vrcholy $\{v_{i-1}, v_i\}$, pre $1 \leq i \leq k$, $v_0 = v$ a $v_k = w$. Hovoríme že cesta P prechádza cez vrcholy $v_0, v_1, v_2, \dots, v_{k-1}, v_k$ a prechádza hranami $e_1, e_2, \dots, e_k$ a cesta má dĺžku k, nakoľko prechádza k hranami [4, s. 540]. Cesta sa nazýva cyklus ak začína a končí v tom istom vrchole, čiže ak $v = w$ a jej dĺžka je väčšia ako nula, takže ak $k \geq 1$.

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 G môže z vrcholu v do vrcholu w existovať viacero ciest. V našom prípade hľadáme cestu z vrcholu a do vrcholu c. Takéto cesty existujú štyri a sú to: $a, \{a, b\}, b, \{b, c\}, c$; $a, \{b, c\}, b, \{b, d\}, d, \{d, c\}, c$; $a, \{a, d\}, d, \{d, c\}, c$ a cesta $a, \{a, d\}, d, \{d, b\}, b, \{b, c\}, c$



Obrázok 1: Cesty v grafe

1.4. Hranová a vrcholová súvislosť

Definícia: Nech $G = (V, E)$ je súvislý graf. Množinu A:
$A \subseteq V$ nazývame vrcholovým rezom grafu G, ak graf $(V \setminus A, \{e | e \in E, e \cap A = \emptyset\})$ je nesúvislý.
$A \subseteq E$ nazývame hranovým rezom grafu G, ak graf$(V, E \setminus A)$ je nesúvislý.

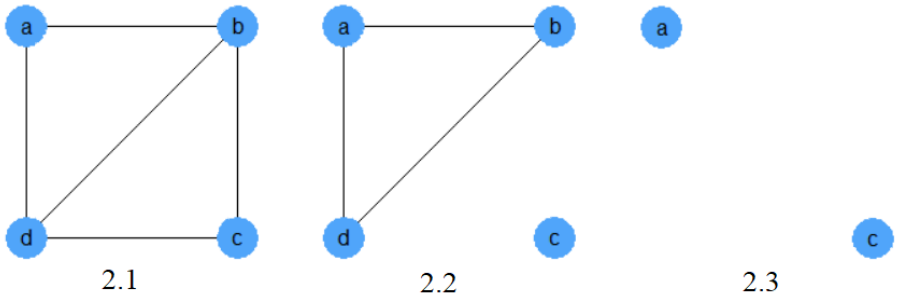
Definícia: Minimálna veľkosť hranového rezu sa nazýva hranová súvislosť grafu G, označujeme $k_E(G)$. Graf sa nazýva k-hranovo súvislý, ak $k \leq k_E(G)$. Minimálna veľkosť vrcholového rezu sa nazýva vrcholová súvislosť grafu G, označujeme $k_V(G)$. Graf sa nazýva k-vrcholovo súvislý, ak $k \leq k_V(G)$ [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 G. Tento graf sme rozdelili pomocou hranového rezu. Výsledok vidíme na obrázku 2.2, na ktorom sú dva izolované grafy. Graf G vieme rozdeliť aj pomocou vrcholového rezu, po ktorom dostaneme taktiež dva izolované, v našom prípade jednovrcholové grafy, viď obrázok 2.3. Pri hranovom reze sme odstránili dve hrany a to $\{b, c\}$ a $\{c, d\}$, takže

hranová súvislosť $k_E(G) = 2$. Pri vrcholovom reze sme odstránili dva vrcholy b a c a príslušné hrany $\{b, c\}$, $\{c, d\}$ a $\{b, d\}$, vrcholová súvislosť $k_V(G) = 2$.



Obrázok 2: Hranová a vrcholová súvislosť

1.5. 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 $\chi(G)$ grafu G je najmenší počet farieb potrebných na zafarbenie vrcholov G tak, aby žiadne dva susedné vrcholy nemali rovnakú farbu t. j. najmenšia hodnota k, ktorú je možné dosiahnuť k-farbením [8, s. 210].

1.6. Obvod

Definícia: Obvod grafu G označený ako $g(G)$ je dĺžka najmenšieho cyklu v G. Ak neexistuje v G žiaden cyklus $g(G) = \infty$ [7].

1.7. Eulerova cesta a cyklus

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

1.8. Hamiltonovská cesta a cyklus

Definícia: Jednoduchá cesta v G, ktorá prechádza cez každý vrchol práve raz, sa nazýva Hamiltonovská cesta, a jednoduchý cyklus v G ktorý prechádza

každým vrcholom práve raz sa nazýva Hamiltonovský cyklus alebo aj Hamiltonovská kružnica. Inak povedané, jednoduchá cesta $x_0, x_1, \dots, x_{n-1}, x_n$ v grafe $G = (V, E)$ je Hamiltonovská cesta ak $V = \{x_0, x_1, \dots, x_{n-1}, x_n\}$ a $x_i \neq x_j$ pre $0 \leq i < j \leq n$, a jednoduchý cyklus $x_0, x_1, \dots, x_{n-1}, x_n, x_0$ (kde $n > 0$) je Hamiltonovský cyklus ak $x_0, x_1, \dots, x_{n-1}, x_n$ je Hamiltonovská cesta [5, s. 698].

1.9. Izomorfizmus

Definícia: Jednoduché grafy $G_1 = (V_1, E_1)$ a $G_2 = (V_2, E_2)$ sú izomorfné ak existuje bijektívna funkcia f z V_1 do V_2 s vlastnosťou že a a b sú susedné v G_1 ak a iba ak $f(a)$ a $f(b)$ sú susedné v G_2, pre všetky a a b vo V_1. Takáto funkcia f sa nazýva izomorfizmus [5, s. 672].

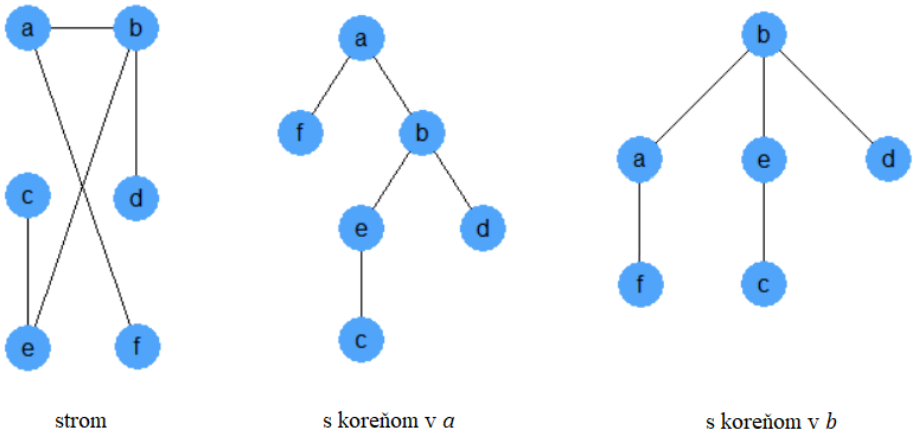
V podstate, dva grafy G_1 a G_2 sú izomorfné, ak majú rovnakú štruktúru. Ich množiny vrcholov a hrán môžu byť odlišné. Izomorfizmus je potom zobrazenie množiny vrcholov G_1 do množiny vrcholov G_2. Ak aplikujeme toto zobrazenie na vrcholy G_1, vytvárame G_1', pričom sa zabezpečuje že G_1' a G_2 majú rovnaké množiny vrcholov a hrán. Ak sú dva grafy izomorfné, musia mať rovnakú postupnosť stupňov. Avšak rovnaká postupnosť stupňov neznamená, že dva grafy musia byť izomorfné.

Najefektívnejší algoritmus na zistenie izomorfizmu sa nazýva quasipolynomiálny algoritmus. Jeho časová zložitosť $\exp(C(\log n)^c)$

1.10. Strom

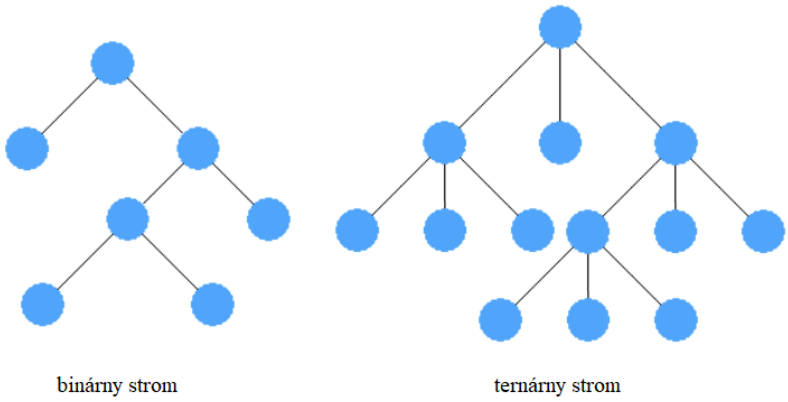
Definícia: Strom je spojený neorientovaný graf ktorý nemá žiadne jednoduché cykly [5, s. 746].

Strom so špeciálne vyznačeným vrcholom volaným koreň, nazývame koreňový (alebo zakorenený) strom. Výberom koreňa z jedného stromu môžeme vytvoriť rôzne koreňové stromy. Na obrázku 3 máme strom, z ktorého výberom koreňa vytvoríme dva rôzne koreňové stromy [11].



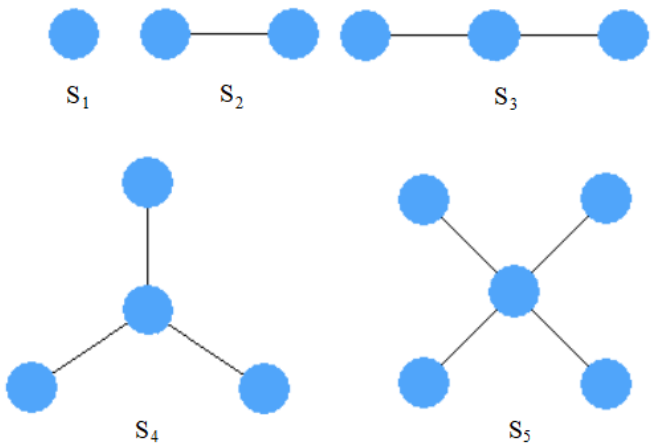
Obrázok 4: Ukážka rôznych koreňových stromov

Úroveň vrcholu je dĺžka cesty od vrcholu ku koreňu. Hĺbka stromu je maximálna úroveň vrcholov. Koreňový strom sa volá n-árny strom, keď každý vrchol má maximálne n detí. Koreňový strom je plne n-árny strom, keď každý vnútorný vrchol má práve n detí. Pre $n = 2$ sa n-árny strom volá binárny strom, pre $n = 3$ sa n-árny strom volá ternárny strom (pozri obrázok 4) [11].



Obrázok 3: Plne binárny a plne ternárny strom

Graf typu hviezda S_n, je strom s n vrcholmi, pričom jeden vrchol má stupeň $n - 1$ a ostatných $n - 1$ vrcholov má stupeň 1. [12]. Na obrázku 5 môžeme vidieť príklad grafov typu hviezda kde $1 \leq n \leq 5$.

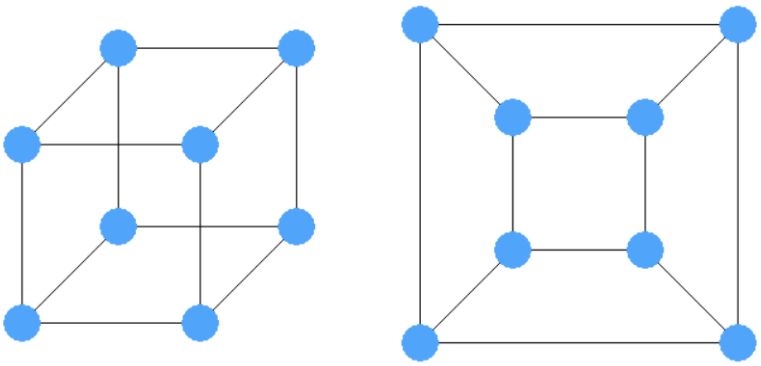


Obrázok 5: Grafy typu hviezda

1.11. 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 6 sa na ľavej strane nachádza graf G, na pravej strane vidíme planárnu reprezentáciu grafu G.



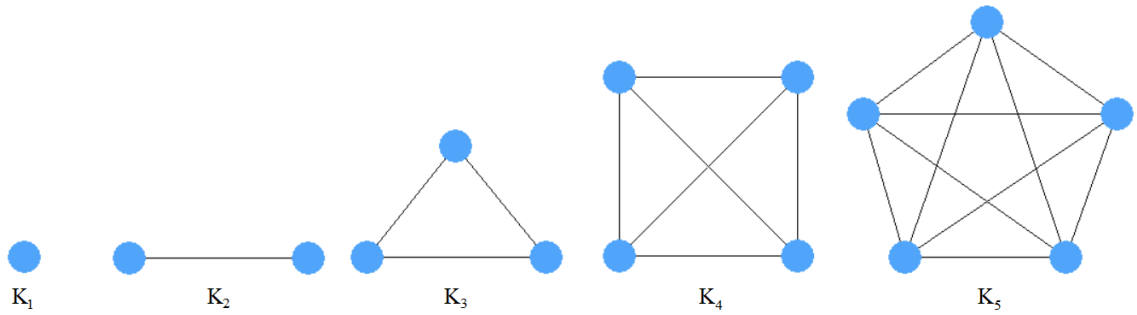
Obrázok 6: Planárny graf

1.12. Úplný graf

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

Počet hrán v úplnom grafe zistíme pomocou vzorca $\frac{n(n-1)}{2}$

Na obrázku 7 príklad úplných grafov K_n pre $1 \leq n \leq 5$.



Obrázok 7: Úplné grafy

1.13. Johnsonov graf

Definícia: Nech $n, m \in \mathbb{N}$ a $m \leq \frac{n}{2}$, Johnsonov graf $J(n, m)$ je definovaný ako:

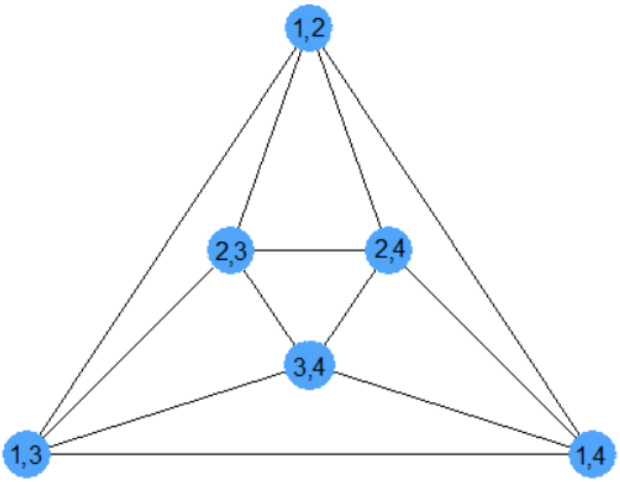
(1) Množina vrcholov je množina všetkých podmnožín $[n]$ s mohutnosťou presne m.

(2) Dva vrcholy sú susedné práve vtedy, keď symetrický rozdiel príslušných množín je dva.

Symetrický rozdiel A a B označený ako $A \triangle B$, je množina obsahujúca prvky ktoré sa nachádzajú v A alebo v B no nie v A aj B. $A \triangle B = (A - B) \cup (B - A)$ [5, s. 137].

Ukážeme si príklad $J(4,2)$ nazývaný aj ako oktahedrálny graf.

Podľa definície dostaneme množinu vrcholov V rovnú $\{\{1,2\},\{1,3\},\{1,4\},\{2,3\},\{2,4\},\{3,4\}\}$. Pre vrcholy $\{1,2\},\{1,3\}$ je symetrický rozdiel $\{2,3\}$, čo je množina s mohutnosťou dva takže tieto dva vrcholy sú susedné. Keď zoberieme vrcholy $\{1,2\}$ a $\{3,4\}$, symetrický rozdiel je $\{1,2,3,4\}$, takže vieme že tieto vrcholy nie sú susedné. Rovnako to spravíme pre všetky zvyšné dvojice množín. Množina hrán $E = \{(\{1,2\},\{1,3\}),(\{1,2\},\{1,4\}),(\{1,2\},\{2,3\}),(\{1,2\},\{2,4\}),(\{1,3\},\{1,4\}),(\{1,3\},\{2,3\}),(\{1,3\},\{3,4\}),(\{1,4\},\{2,4\}),(\{1,4\},\{3,4\}),(\{2,3\},\{2,4\}),(\{2,3\},\{3,4\}),(\{2,4\},\{3,4\})\}$. Johnsonov graf $J(4,2)$ môžeme vidieť na obrázku ...

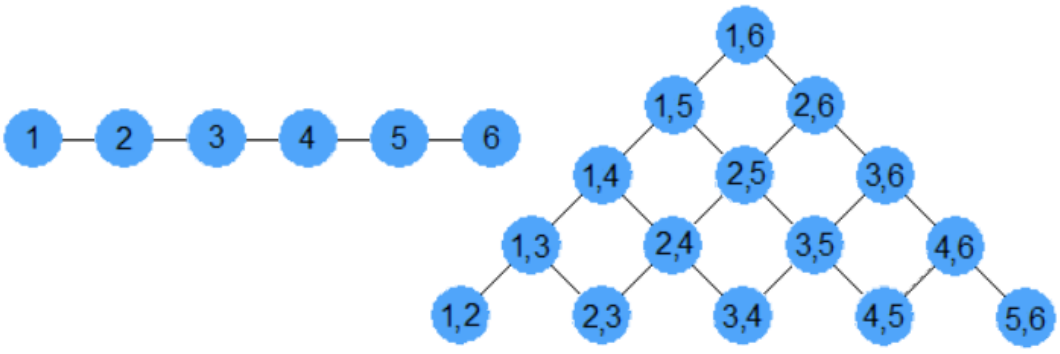


Obrázok 8: Johnsonov graf J(4,2)

1.14. Token grafy

Pre graf G a celé číslo $k \geq 1$, definujeme token graf $F_k(G)$ ako graf s vrcholovou množinou $\binom{V(G)}{k}$, kde dva vrcholy A a B, grafu $F_k(G)$ sú susedné, keď ich symetrický rozdiel $A \triangle B$ je dvojica $\{a, b\}$ taká, že $a \in A$, $b \in B$ a $ab \in E(G)$. Teda vrcholy v $F_k(G)$ zodpovedajú konfiguráciám k nerozlíšiteľných tokenov umiestnených na odlišných vrcholoch G, 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 $F_k(G)$ ako k-token graf grafu G [9, s. 2].

Na obrázku 8 môžeme vidieť príklad 2-token grafu, ktorý vznikol zo šesť-vrcholovej cesty.



Obrázok 9: 2-token graf zo 6-vrcholového grafu typu cesta

1.2.1.Základné vlastnosti

Majme graf G s n vrcholmi a k je kladné celé číslo. Aby sme sa vyhli triviálnym prípadom, budeme predpokladať že $n \geq k + 1$. Počet vrcholov $F_k(G)$ je:

$$\left|V\left(F_k(G)\right)\right| = \binom{n}{k}$$

Na vypočítanie počtu hrán $F_k(G)$, každej hrane AB z $F_k(G)$ pridelíme hranu ab z G, pre ktorú platí $a \triangle b = \{a, b\}$. Počet hrán v grafe $F_k(G)$, ktoré sú priradené k ab je rovný $\binom{n-1}{k-1}$. Preto

$$\left|E\left(F_k(G)\right)\right| = \binom{n-1}{k-1}|E(G)|$$

Susedia každého vrcholu A z $F_k(G)$ sú

$$\{A \setminus \{v\} \cup \{w\} : v \in A, w \in V(G) \setminus v w \in E(G)\}$$

Takže stupeň vrcholu A z $F_k(G)$ je rovný počtu hrán medzi A a $V(G) \setminus A$. Z toho vyplývajú ohraničenia na minimálny a maximálny stupeň $F_k(G)$.

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

$$F_1(G) \simeq G$$

Dva vrcholy A a B sú susedné v $F_k(G)$ práve vtedy, keď $V(G) \setminus A$ a $V(G) \setminus B$ sú susedné v $F_{n-k}(G)$,

$$F_k(G) \simeq F_{n-k}(G)$$

[9, s. 3]

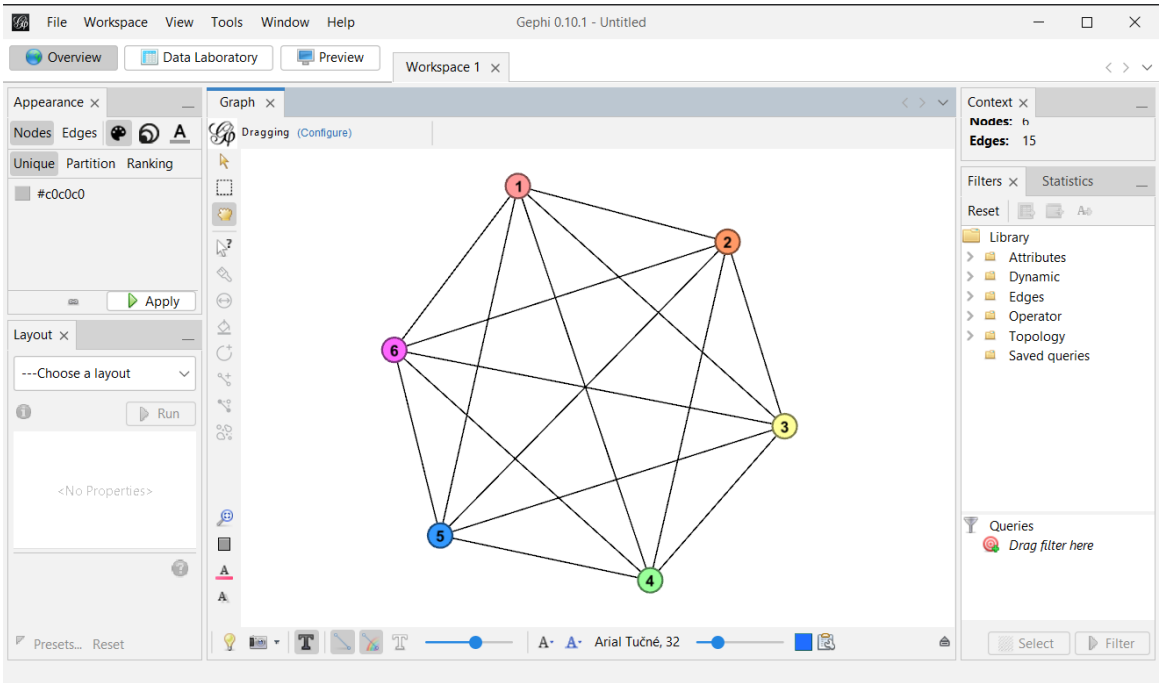
Ďalšie vlastnosti token grafu (pol strana až strana navyše)

2. 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.

2.1. 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 10: Screenshot aplikácie Gephi

Jednou z nevýhod ktoré aplikácia Gephi má, je to že vytvorenú sieť nie je možné exportovať ako HTML alebo obrázok. Avšak najdôležitejšie pre našu prácu je to, že Gephi neposkytuje algoritmy na prácu s token grafmi.

2.2. 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].

2.3. 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].

3. ANALÝZA

3.1. Špecifické požiadavky

3.2.1. 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.

3.2.2. 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.

3.2.3. Nefunkcionálne požiadavky

Desktopová aplikácia

Backend postavený na jazyku Python a knižnici NetworkX

Frontend postavený na knižnici Tkinter

Sem by sa patrilo niečo pridať, ešte neviem čo

4. 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.

4.1. Triedy

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

4.1.1. 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`.

Pridať ďalšie metódy, + upraviť text k existujúcim (cca 1 strana navyše)

4.1.2. 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` ...

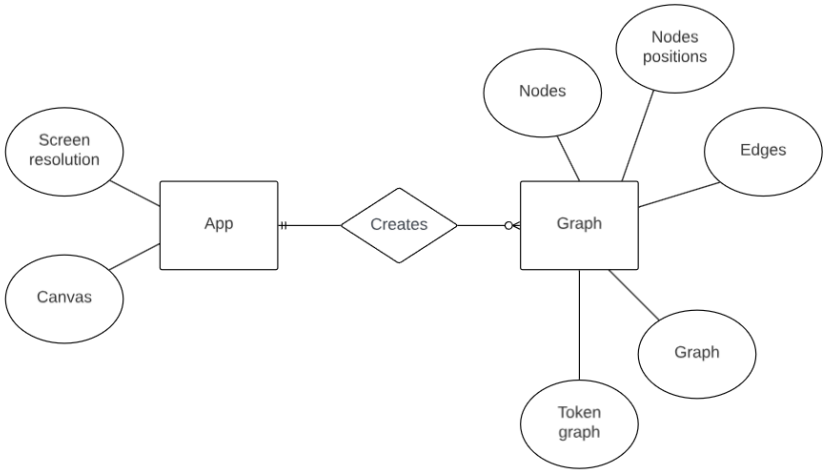
Metóda `is_tree` ...

Metóda `is_regular` ...

... (ostatné algoritmy cca 1 strana navyše)

4.1.3. *Entitno-relačný model*

!!! Graf potrebuje ešte upraviť + popis k nemu cca strana



Obrázok 11: Entitno-relačný model

4.1.4. *Screenshoty z aplikácie*

...

+ cca 2 – 3 strany

5. EXPERIMENT

5 – 6 strán

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] http://www2.fiit.stuba.sk/~kvasnicka/DiskretnaMatematika/Chapter_12/kapitola12.pdf

[12] https://mathworld.wolfram.com/StarGraph.html

[13]

[14]

[15]

[16]

[17]

[18]

[19]

[20]