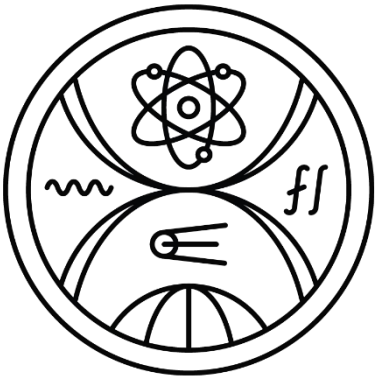
UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

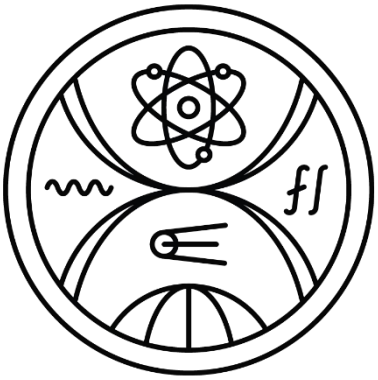


TOKEN GRAFY

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 Závacká

Bratislava, 2024

Timotea Chalupová

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 Závacká
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

Rada by som poďakovala mojej vedúcej práce Mgr. Dominike Závackej za odbornú pomoc, nadštandardnú ochotu, hodnotné rady a praktické pripomienky počas písania bakalárskej práce.

Abstrakt

Chalupová Timotea: Token grafy. Bakalárska práca. Univerzita Komenského v Bratislave, Fakulta matematiky, fyziky a informatiky. Vedúci záverečnej práce: Mgr. Dominika Závacká, Bratislava 2024.

Cieľom tejto práce je študovanie token grafov, ktoré vznikli z rôznych typov grafov, ako sú cesty a stromy. Na ukladanie štruktúr grafov sme použili knižnicu NetworkX, čo nám umožnilo testovať ich vlastnosti. V rámci analýzy sme sa zamerali na základné vlastnosti token grafov, ako sú hranová a vrcholová súvislosť, Hamiltonovské a Eulerovské cesty, a snažili sme sa nájsť efektívne algoritmy na riešenie týchto problémov. Vytvorili sme desktopovú aplikáciu, ktorá implementuje token grafy a príslušné algoritmy. Pre každý typ grafu sme pripravili testovacie údaje a výsledky testovania sme podrobne popísali v štvrtej kapitole tejto práce. Ciele našej práce boli dosiahnuté s využitím programovacieho jazyka Python a jeho knižníc NetworkX a Tkinter.

Kľúčové slová: Token graf, Desktopová aplikácia, Python, NetworkX, Tkinter

Abstract

Chalupová Timotea: Token graphs. Bachelor thesis. Comenius University in Bratislava, Faculty of Mathematics, Physics and Informatics. Bachelor thesis supervisor: Mgr. Dominika Závacká, Bratislava 2024.

The aim of this thesis is to study token graphs that have been created from different types of graphs such as paths and trees. We used the NetworkX library to store the graph structures, which allowed us to test their properties. As part of the analysis, we focused on basic properties of token graphs such as edge and vertex connectivity, Hamiltonian and Eulerian paths, and tried to find efficient algorithms to solve these problems. We developed a desktop application that implements token graphs and the corresponding algorithms. We prepared test data for each type of graph, and the results of the testing are described in detail in chapter four of this thesis. The goals of our work were achieved using the Python programming language and its libraries NetworkX and Tkinter.

Keywords: Token graph, Desktop application, Python, NetworkX, Tkinter

Obsah

Zoznam obrázkov

Zoznam tabuliek

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. Grafy sú základným nástrojom pre modelovanie komplexných systémov, ako sú počítačové siete, sociálne siete, dopravné systémy, biologické siete a mnohé ďalšie. 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 a zlepšenie výkonu rôznych systémov, čo je kľúčové pre vývoj inovatívnych technológií a aplikácií.

V prvej kapitole si objasníme základné pojmy z teórie grafov, ktoré sú nevyhnutné pre porozumenie danej problematiky. Spomenieme termíny ako jednoduchý graf, úplný graf, hranová a vrcholová súvislosť ale aj pojmy ako je cesta, cyklus ale aj strom. V tejto kapitole si taktiež popíšeme základné vlastnosti token grafov.

V druhej kapitole sa pozrieme na existujúce systémy, ale aj jazyky a knižnice ktoré by sme vedeli využiť v našej práci. Ďalej si popíšeme požiadavky kladené na našu aplikáciu.

Tretia kapitola sa zameriava na samotnú implementáciu, popíšeme a vysvetlíme si v nej triedy a ich jednotlivé metódy, ukážeme si entitno-relačný model našej aplikácie a na záver si ukážeme výsledné používateľské rozhranie.

A napokon v štvrtej kapitole si popíšeme výsledky z nášho experimentu na token grafoch.

Hlavným cieľom je bakalárskej práce je naštudovanie si vlastností token grafov, napísanie algoritmu na vytvorenie token grafu a následné implementovanie algoritmov na token grafoch.

1. ZÁKLADNÉ POJMY

V tejto kapitole vysvetlíme základné pojmy a definície, ktoré sú nevyhnutné pre vypracovanie našej práce. Tieto pojmy budeme neskôr používať pri implementácii našej aplikácie.

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 neusporiadaných dvojíc vrcholov. Každá hrana G môže byť vyjadrená ako $\{u, v\}$, kde u a v sú odlišné vrcholy, t. j. $u, v \in V, u \neq v$ [1, s. 497].

Jednoduchý graf je jedným zo základných pojmov v teórii grafov. Neformálne napísané, jednoduchý graf 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: Stupeň vrcholu v grafe G, označený $\deg(v)$, je počet hrán $|E(v)|$, ktoré končia vo vrchole v. Vrchol so stupňom 0 je izolovaný.

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 [1, s. 499].

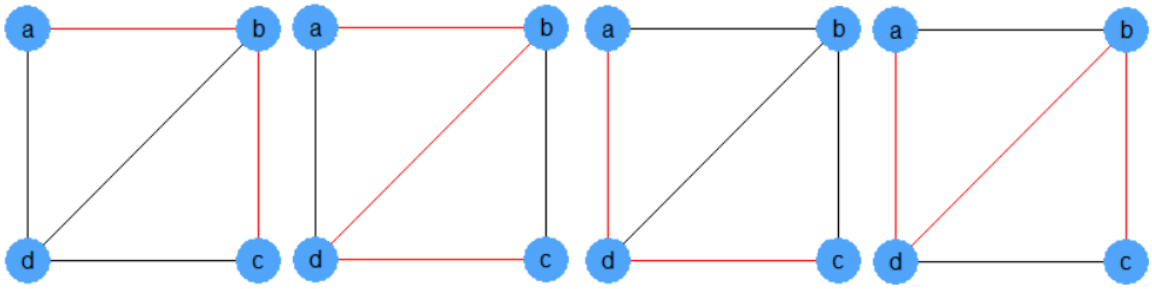
Pre jednoduchý graf, stupeň vrcholu je číslo vyjadrujúce počet susedov tohto vrcholu. To znamená že v k-pravidelnom grafe má každý vrchol presne k susedov, pričom k je z intervalu 0 až $|V(G)| - 1$.

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: $P = \langle v_0, e_1, v_1, e_2, v_2, \ldots, v_{k-1}, e_k, e_k \rangle$ pričom 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 $v_0, v_1, v_2, \ldots, v_{k-1}, v_k$ a prechádza hranami $e_1, e_2, \ldots, e_k$ a cesta má dĺžku k, nakoľko prechádza k hranami [1, 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. Ak cesta alebo cyklus neobsahuje žiadnu z hrán viac ako jeden raz, hovoríme o jednoduchej ceste respektíve o jednoduchom cykle [2, 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

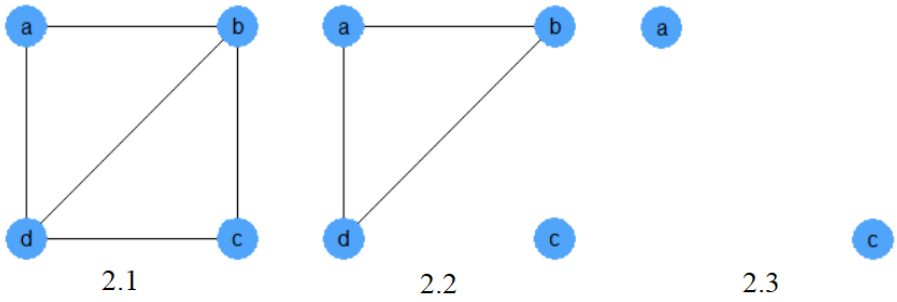
Definícia: Nech $G = (V, E)$ je súvislý graf. Množinu A:
$A \subseteq V$ nazývame vrcholovým rezom grafu G, ak graf $(V \setminus A, \{e | e \in E, e \cap A = \emptyset\})$ je nesúvislý.
$A \subseteq E$ nazývame hranovým rezom grafu G, ak graf$(V, E \setminus A)$ je nesúvislý.

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 $k \leq k_V(G)$ [3, 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 vrcholy, 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 [2, 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 [4, s. 210].

Politické mapy sú nám poslúžia ako jednoduchý príklad využitia. Jednotlivé krajiny sú reprezentované ako vrcholy grafu a hrany medzi nimi naznačujú spoločné hranice. Použitím chromatického čísla zistíme počet farieb potrebných na zafarbenie krajín, tak aby žiadne dve susedné nemali rovnakú farbu. Vďaka tomuto prístupu sú jednotlivé krajiny jednoznačne rozlíšiteľné.

1.6. Obvod

Definícia: Obvod grafu G označený ako $g(G)$ je dĺžka najmenšieho cyklu v G. Ak neexistuje v G žiaden cyklus $g(G) = \infty$ [5].

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 [2, 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 [2, s. 698].

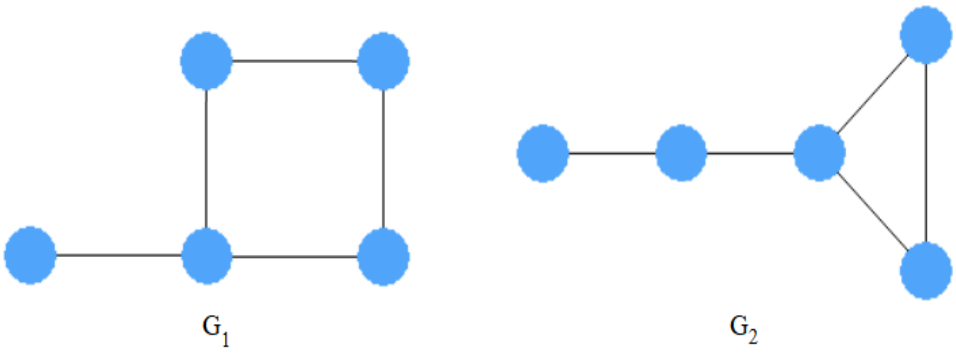
1.9. Izomorfizmus

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 vtedy a len vtedy 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 [2, 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 valenčnú postupnosť. Avšak ak majú dva grafy s rovnakú valenčnú postupnosť neznamená to, že tieto dva grafy sú izomorfné.

Definícia: Nech $G = (V, E)$ je graf a $n = |V|$. Valenčná postupnosť grafu G je n-prvková neklesajúca postupnosť, ktorá vznikne usporiadaním stupňov všetkých vrcholov grafu neklesajúco [6, s. 15].

Na obrázku 3 sú dva grafy G_1 a G_2, oba grafy majú rovnakú valenčnú postupnosť a to $(3, 2, 2, 2, 1)$, avšak na základe vyššie spomenutej definície nie sú izomorfné.



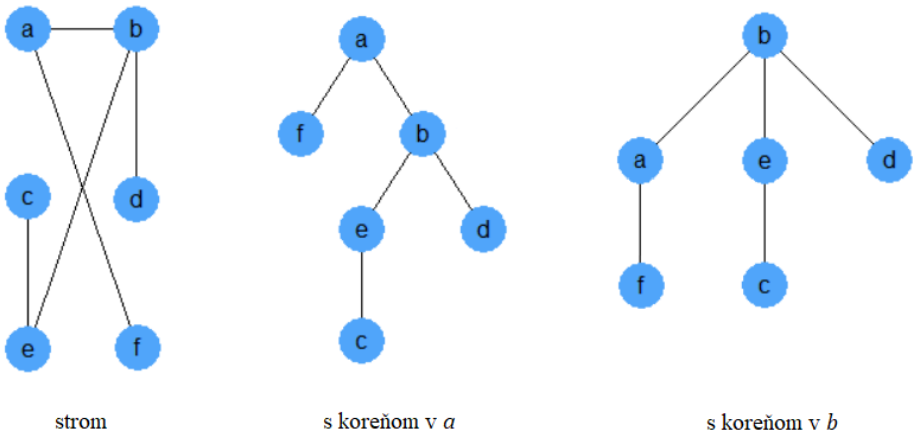
Obrázok 3: Grafy s rovnakou valenčnou postupnosťou

Najefektívnejší algoritmus na zistenie izomorfizmu sa nazýva quasipolynomiálny algoritmus. Jeho časová zložitosť je $\exp(C(\log n)^c)$ kde n je počet vrcholov a c a C sú konštanty. Bol predstavený v roku 2015 Lászlóm Babaiom [7].

1.10. Strom

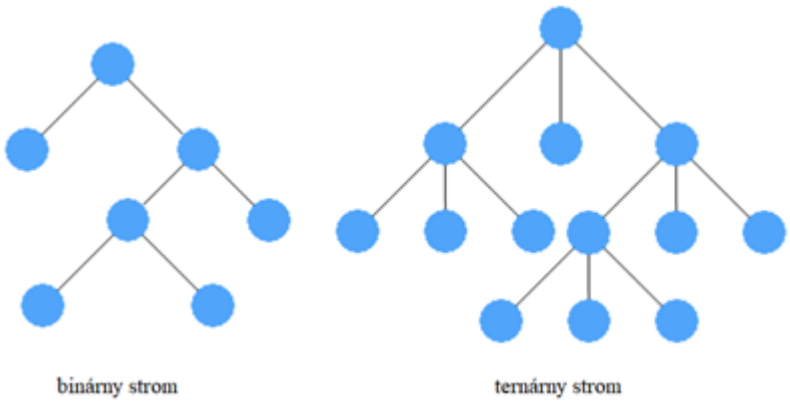
Definícia: Strom je spojený neorientovaný graf ktorý nemá žiadne jednoduché cykly [2, s. 746].

Strom so špeciálne vyznačeným vrcholom volaným koreň, nazývame koreňový (alebo zakorenený) strom. Výberom koreňa z jedného stromu môžeme vytvoriť rôzne koreňové stromy. Na obrázku 4 máme strom, z ktorého výberom koreňa, vytvoríme dva rôzne koreňové stromy [8].



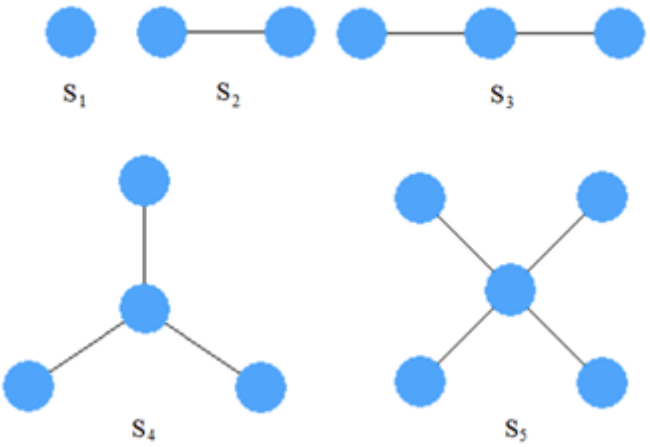
Obrázok 4: Ukážka rôznych koreňových stromov

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 5) [8].



Obrázok 5: 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. [9]. Na obrázku 6 môžeme vidieť príklad grafov typu hviezda kde $1 \leq n \leq 5$.

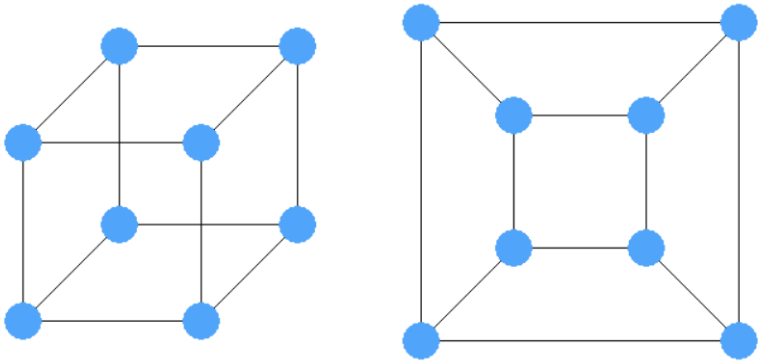


Obrázok 6: Grafy typu hviezda

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 [2, s. 719].

Na obrázku 7 sa na ľavej strane nachádza graf G, na pravej strane vidíme planárnu reprezentáciu grafu G.



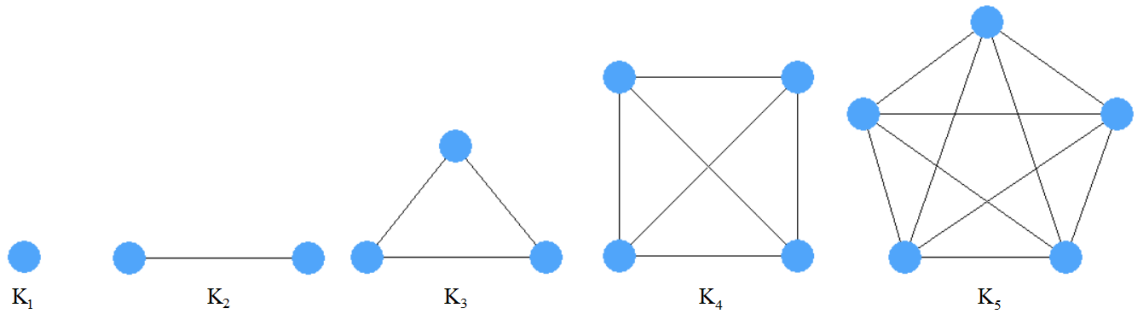
Obrázok 7: Planárny graf

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 [2, s. 655].

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

Na obrázku 8 môžeme vidieť príklad úplných grafov K_n pre $1 \leq n \leq 5$.



Obrázok 8: 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:

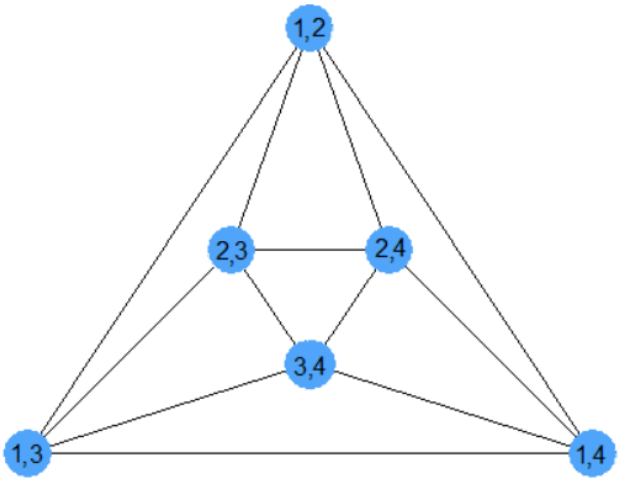
(1) Množina vrcholov je množina všetkých podmnožín $[n]$ s mohutnosťou presne m.

(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. $A \triangle B = (A - B) \cup (B - A)$ [2, 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 9.



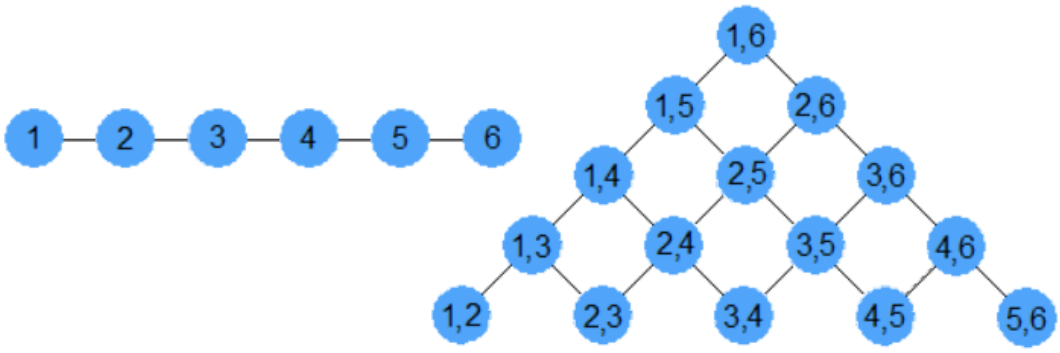
Obrázok 9: Johnsonov graf J(4,2)

1.14. Token grafy

V našej práci sa primárne venujeme token grafom, preto je nevyhnutné zadefinovať ich základné vlastnosti.

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 [10, s. 2].

Na obrázku 10 môžeme vidieť príklad 2-token grafu, ktorý vznikol zo šesť-vrcholovej cesty.



Obrázok 10: 2-token graf zo 6-vrcholového grafu typu cesta

1.14.1. *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

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

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

$$F_1(G) \simeq G \tag{1}$$

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 $F_{n-k}(G)$,

$$F_k(G) \simeq F_{n-k}(G) \tag{2}$$

Niekedy používame (2) na predpoklad že $k \leq \frac{n}{2}$. Je nutné poznamenať, že (1) a (2) implikujú dve známe vlastnosti Johnsonovho grafu: $J(n, 1) \simeq K_n$ a $J(n, k) \simeq J(n, n - k)$.

[10, s. 3]

Ak graf G obsahuje Hamiltonovskú cestu a n je párne a k je nepárne, potom aj $F_k(G)$ obsahuje Hamiltonovskú cestu. Navyše každý Johnsonov graf je aj Hamiltonovským takže obsahuje Hamiltonovskú cestu [10, s. 14].

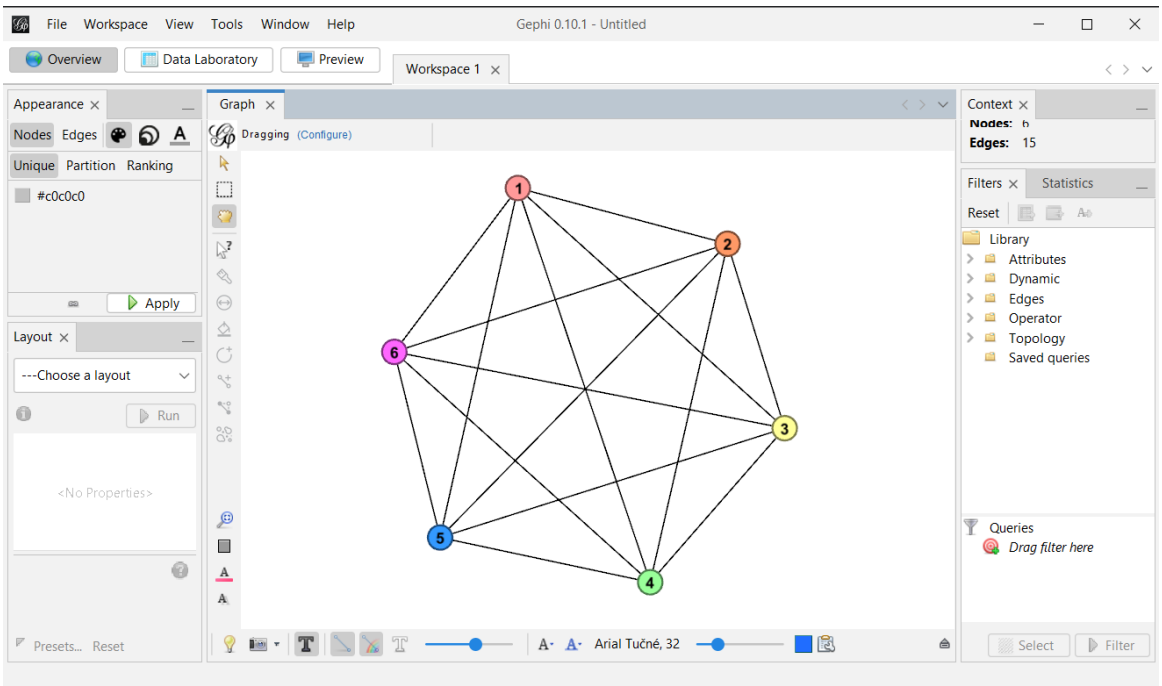
2. ANALÝZA

2.1. Prehľad technológií

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

2.1.1. *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 [11].



Obrázok 11: 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. (Pridať ďalší softvér)

2.1.2. 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 [12].

Aj napriek tomu, že knižnica NetworkX neposkytuje algoritmy na prácu s token grafmi, sme si ju vybrali ako štruktúru, v ktorej uložíme graf, na ktorom budeme vykonávať vlastné algoritmy.

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 [13].

Knižnicu Tkinter sme si vybrali na tvorbu používateľského rozhrania našej aplikácie, keďže s touto knižnicou už máme skúsenosti, je stabilná a dobre zdokumentovaná. S jej pomocou dokážeme rýchlo a efektívne implementovať požadovanú funkcionalitu.

2.1.3 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 [14].

2.2. Požiadavky

2.2.1. Funkcionálne požiadavky

Vytvorenie grafu: Používateľ zadá vrcholy a hrany grafu, z nich sa následne vygeneruje graf.

Vytvorenie token grafu: Používateľ zadá vrcholy grafu, hrany grafu a číslo označujúce počet tokenov. Z týchto údajov sa následne vygeneruje k-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 vo formáte PNG.

Zmena rozloženia grafu: Aby sa zabezpečila väčšia prehľadnosť grafu, bude možné meniť rozloženie grafu, presúvaním vrcholov po ploche aplikácie.

Spustenie algoritmov na grafe: Po vygenerovaní grafu sa budú dať spúšťať algoritmy ako napríklad hľadanie najmenšieho cyklu, hľadanie najkratšej cesty, Hamiltonovskej cesty, a ďalšie vlastnosti, ktoré sme podrobne rozobrali v kapitole 1.

2.2.2. 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.

2.2.3. Nefunkcionálne požiadavky

Desktopová aplikácia: Vyvíjaný softvér bude vo forme desktopovej aplikácie.

Backend aplikácie: Aplikácia bude napísaná v jazyku Python s využitím knižnice NetworkX.

Frontend aplikácie: Frontend aplikácie bude postavený na knižnici Tkinter.

3. 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 triedami a metódami, ktoré si podrobne vysvetlíme, a potom si ukážeme výsledné používateľské rozhranie.

3.1. Triedy

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

3.1.1. 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 zavolá metóda `start_screen`, ktorá vykreslí úvodnú obrazovku.

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 vo formáte textového súboru, načíta z neho hrany a vrcholy, ktoré je 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, tlačidlo späť, combobox na zvolenie algoritmu a tlačidlo na vyhľadanie zvoleného algoritmu v grafe a volá funkciu `draw`.

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

Metóda `button_save` otvorí okno na zvolenie mena súboru a jeho umiestnenia v zariadení. Graf sa uloží vo formáte obrázku do zvoleného priečinku. O úspešnom, respektíve neúspešnom uložení bude používateľ informovaný pomocou vyskakovacieho okna.

Metóda `button_save_graph` podobne ako pri ukladaní obrázku, aj tu sa používateľovi otvorí okno, kde zvolí meno a umiestnenie súboru. Metóda do zvoleného priečinku uloží konfiguráciu grafu, teda príslušné hrany a vrcholy, vo formáte textového súboru.

Metóda `button_back` vráti používateľa na predchádzajúcu plochu, bez toho, aby stratil zadané vrcholy a hrany, čo je užitočné v prípade, že pri vytváraní grafu urobil chybu.

Metóda `button_back_to_load` volá metódu `start_screen`, ktorá vráti používateľa na úvodnú plochu, teda na plochu kde si vyberie či chce nový graf alebo načítať existujúci.

Metóda `move` sa zavolá pri udalosti ťahania myšou. Ak používateľ chytil a potiahol niektorý z vrcholov, vypočítajú sa jeho nové súradnice a volá sa metóda `draw`, ktorá graf prekreslí.

Metóda `start_screen` na plátno umiestni dve tlačidlá. Jedno na vytvorenie nového grafu a druhé na načítanie existujúceho grafu.

Metóda `button_search` na základe hodnoty z comboboxu zavolá metódu na testovanie vlastnosti grafu. Výsledok testovania vypíše na plochu. V prípade že výsledok je číselná hodnota, a teda nie pravda alebo nepravda, zavolá metódu `draw_algorithm`, ktorá výsledok graficky znázorní na grafe.

Metóda `draw_algorithm` dostane parametre `nodes`, `edges` a `colors`, prvý z parametrov obsahuje zoznam vrcholov, druhý parameter je zoznam hrán a posledný parameter sú farby. Metóda prejde cez zoznamy vrcholov aj hrán a zafarbí ich na základe údajov zo zoznamu farieb.

3.1.2. Trieda Graph

Táto trieda rieši funkčnú časť, ako je ukladanie grafov, generovanie token grafov a vykonávanie algoritmov na grafoch, ako je napríklad hľadanie Eulerovej cesty alebo hľadanie najmenšieho cyklu.

Metóda `__init__` zadefinuje dva atribúty `self.g` a `self.token`, v ktorých budú uložene grafy.

Metóda `parse` dostane dva parametre `nodes` a `edges`. Prvý parameter obsahuje vrcholy vo formáte textového reťazca, ktorý rozdelíme podľa čiarok, čím dostaneme zoznam vrcholov. Parameter `edges` obsahuje hrany ako dvojice vrcholov, vo formáte textového reťazca. Hrany môžu byt zadané v jednoduchom formáte a to dva vrcholy oddelené čiarkou a jednotlivé hrany oddelené medzerou. Tento formát zabezpečuje jednoduché a rýchle zadávanie vrcholov. Druhý formát ktorý dokáže metóda `parse` spracovať sú vrcholy v zátvorkach oddelené čiarkou. Hrany sú oddelené čiarkou.

Metóda `to_token` zoberie údaje od používateľa ako sú vrcholy, hrany a počet tokenov a vygeneruje príslušný k-token graf. Vrcholy v k-token grafe, sú všetky podmnožiny vrcholov pôvodného grafu s mohutnosťou k, čiže kombinácie bez opakovania k-tej triedy na vrcholoch pôvodného grafu. Aby sme zistili aké hrany budú v k-token grafe zoberieme si všetky dvojice vrcholov a ak ich vzájomný prienik má mohutnosť $k-1$, a existuje hrana v pôvodnom grafe medzi vrcholmi symetrického rozdielu dvojice vrcholov, tak je hrana medzi týmito vrcholmi aj v k-token grafe.

Metóda `girth` vráti dĺžku najkratšieho cyklu. Pri volaní tejto metódy sa zároveň volá funkcia, ktorá vráti zoznam vrcholov, ktoré sa nachádzajú v tomto cykle. Robíme to preto aby sme tento cyklus vedeli zakresliť do grafu.

Metóda `shortest_path` vráti zoznam vrcholov najkratšej cesty z vrcholu v_1 do vrcholu v_2. Pred zavolaním tejto metódy používateľ označí dva vrcholy, medzi ktorými hľadá najkratšiu cestu.

Metóda `is_planar` vráti hodnotu pravda alebo nepravda, podľa toho či je graf planárny.

Metóda `is_tree` vráti hodnotu pravda alebo nepravda, na základe toho či graf patrí do rodiny stromov.

Metóda `is_complete` vytvorí kompletný graf s rovnakým počtom vrcholov ako má graf na ktorom skúmame vlastnosti. Ak je náš graf izomorfný s vytvoreným kompletným grafom, tak aj náš graf je kompletný a metóda vráti hodnotu pravda. V opačnom prípade vráti hodnotu nepravda.

Metóda `is_regular` vráti hodnotu pravda ak je graf pravidelný, teda či majú všetky vrcholy rovnaký stupeň, inak vráti hodnotu nepravda.

Metóda `edge_connectivity` vráti číslo zodpovedajúce minimálnemu počtu hrán, ktorých odstránením dostaneme dva neprepojené grafy. Pri tejto metóde používame funkciu, ktorá vráti zoznam hrán na odstránenie, vďaka čomu vieme tieto hrany vyznačiť na grafe.

Metóda `node_connectivity` je podobná predchádzajúcej metóde, avšak rozdiel je v tom, že nevráti počet hrán ale minimálny počet vrcholov, ktorých odstránením dostaneme dva neprepojené grafy. Rovnako používame funkciu, ktorá vráti zoznam vrcholov na odstránenie, aby sme ich mohli vyznačiť na grafe.

Metóda `coloring` vráti chromatické číslo grafu, teda minimálny počet farieb potrebných na vyfarbenie grafu tak aby žiadne dva susedné vrcholy nemali rovnakú farbu. Používame tu funkciu, ktorá každému vrcholu priradí určitú farbu, na základe čoho vieme vyfarbiť náš graf.

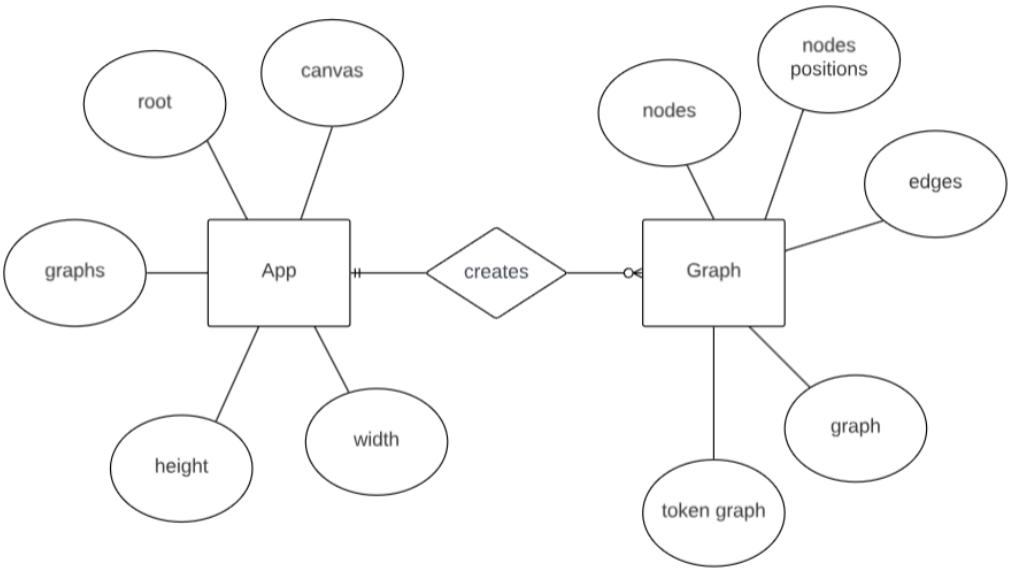
Metóda `has_eulerian_path` vráti hodnotu pravda alebo nepravda podľa toho, či sa v grafe nachádza Eulerovská cesta.

Metóda `is_eulerian` vráti hodnotu pravda ak sa v grafe nachádza Eulerovský cyklus, inak vráti hodnotu nepravda.

Pri implementovaní metódy `has_hamiltonian_path` sme museli vyriešiť veľké množstvo problémov, keďže knižnica NetworkX má implementovanú funkciu na hľadanie Hamiltonovskej cesty iba v orientovanom grafe. Najskôr sme chceli prerobiť náš neorientovaný graf na orientovaný a to takým spôsobom, že by sme vytvorili nový orientovaný graf a každú hranu z neorientovaného grafu medzi dvomi vrcholmi v a w vložili do orientovaného grafu ako (v, w) a (w, v). Avšak tu sme narazili na ďalší problém a to že takýto graf môže prejsť po tej istej hrane dva krát, čo nám dáva nesprávne výsledky. Najjednoduchší spôsob na hľadanie cesty je použitie backtrackingu a ak prejde všetky vrcholy práve jeden krát vrátiť hodnotu pravda. Avšak tu by mohol nastať problém s maximálnou hĺbkou rekurzie. Preto si vytvoríme v metóde `has_hamiltonian_path` zásobník pomocou ktorého simulujeme rekurziu. Ak sme narazili na nový vrchol, ktorý v zásobníku nie je, pridáme ho a pokračujeme ďalej, ak sme narazili na vrchol ktorý už v zásobníku je, vieme že to nie je Hamiltonovská cesta, vrchol odstránime a hľadáme ďalšie cesty. Ak nájdeme Hamiltonovskú cestu metóda vráti hodnotu pravda, ak nenájde vráti hodnotu nepravda.

V metóde `is_johnson_graph` chceme zistiť či daný graf je Johnsonov graf. Zistíme to na základe toho, či graf spĺňa všetky požiadavky z definície pre Johnsonov graf, ktoré sme si uviedli v kapitole Základné pojmy. Preto si najskôr určíme n a m, kde n predstavuje počet vrcholov grafu a m počet tokenov. Ak je $m > \frac{n}{2}$ nespĺňa prvú požiadavku, preto vrátime hodnotu nepravda. V opačnom prípade porovnáme množinu vrcholov nášho grafu a množinu všetkých podmnožín $[n]$ s mohutnosťou presne m. Ak nie sú rovnaké vrátime hodnotu nepravda. Ďalej zistíme či všetky vrcholy v našom grafe majú dĺžku presne $m - 1$, následne skontrolujeme či nie je v našom grafe nejaká hrana navyše alebo naopak nejaká hrana nechýba v porovnaní s vypočítanými hranami ktoré by mal obsahovať Johnsonov graf. Ak sú všetky vyššie spomenuté podmienky splnené, ide o Johnsonov graf a vrátime hodnotu pravda.

3.1.3. Entitno-relačný model

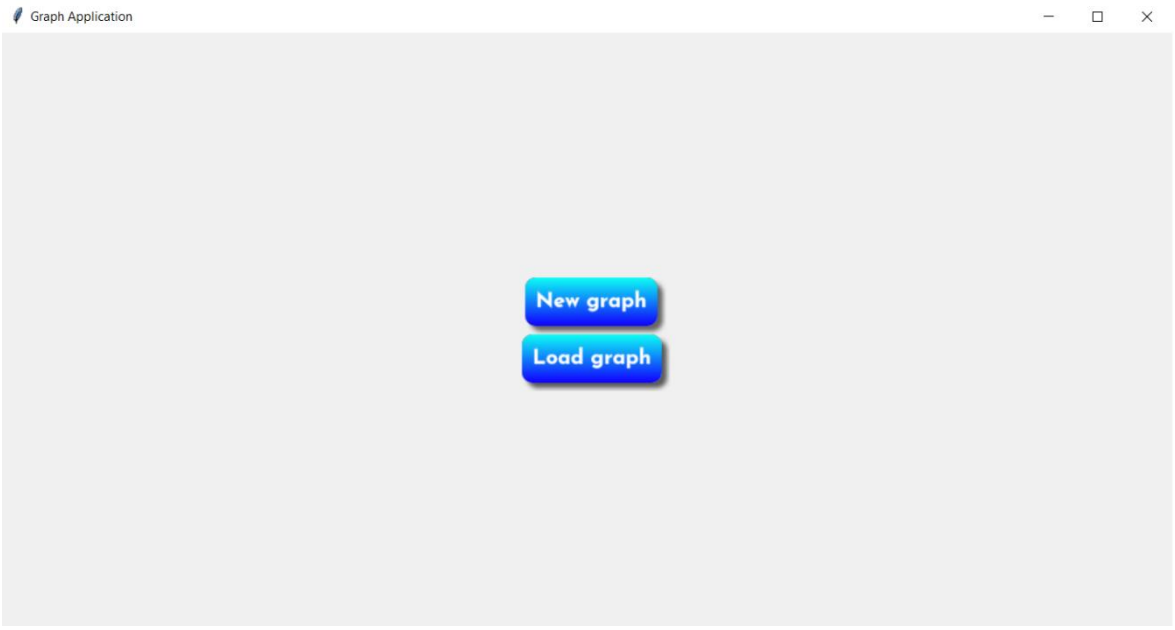


Obrázok 12: Entitno-relačný model

Na obrázku 12 vidíme entitno-relačný model našej aplikácie. Máme dve entity a to entitu App (aplikácia) a Graph (graf). Aplikácia môže vytvárať a spravovať viacero grafov ale každý graf patrí práve k jednej aplikácii, čo je znázornené vzťahom jedna k veľa. Entita aplikácia má niekoľko atribútov. Atribút canvas predstavuje plátno aplikácie na ktorom sa vykresľujú grafy. Atribút root je koreňový prvok aplikácie a predstavuje základnú štruktúru aplikácie. Ďalej máme dva atribúty height a width, v ktorých je uložená výška a šírka plátna v pixeloch. Údaje o veľkosti plátna sa vypočítavajú podľa veľkosti obrazovky používateľa. Posledným atribútom je graphs, v ktorom vytvárame, ukladáme a spravujeme grafy. Entita graf má dva dôležité atribúty a to graph a token graph, v ktorých sú uložené grafy. A atribúty nodes (vrcholy), edges (hrany) a nodes positions (pozície vrcholov), ktoré slúžia na vykresľovanie grafov.

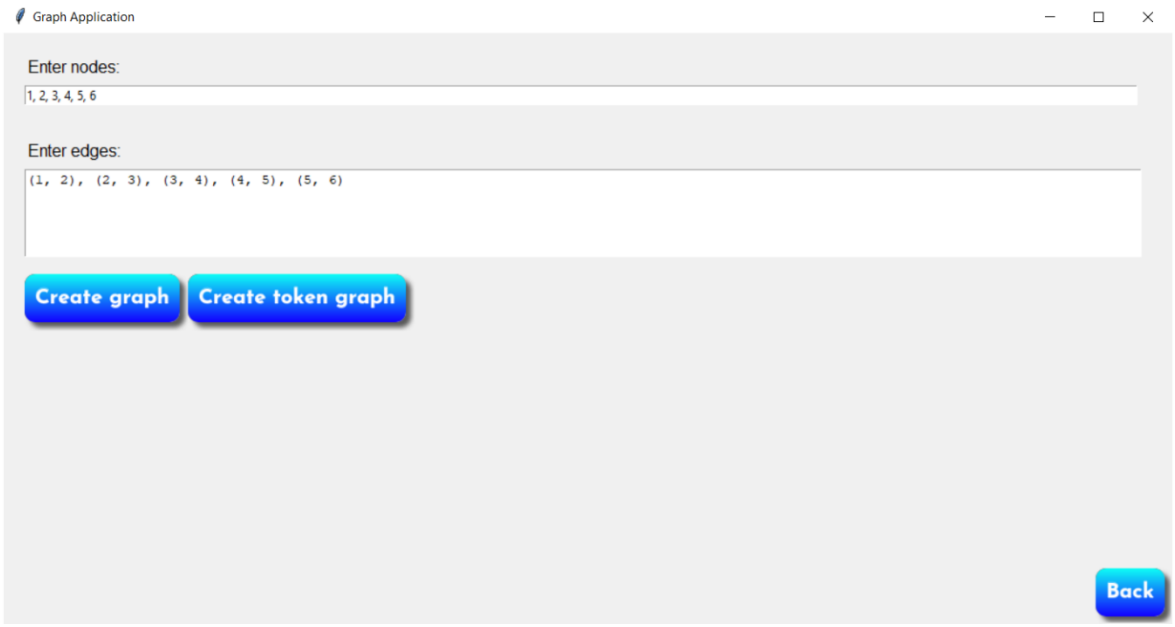
3.1.4. Výsledné používateľské rozhranie aplikácie

V tejto podkapitole si ukážeme a popíšeme výsledné používateľské rozhranie našej aplikácie. Ukážeme si len tie základné funkcie aplikácie.



Obrázok 14: Úvodná obrazovka aplikácie

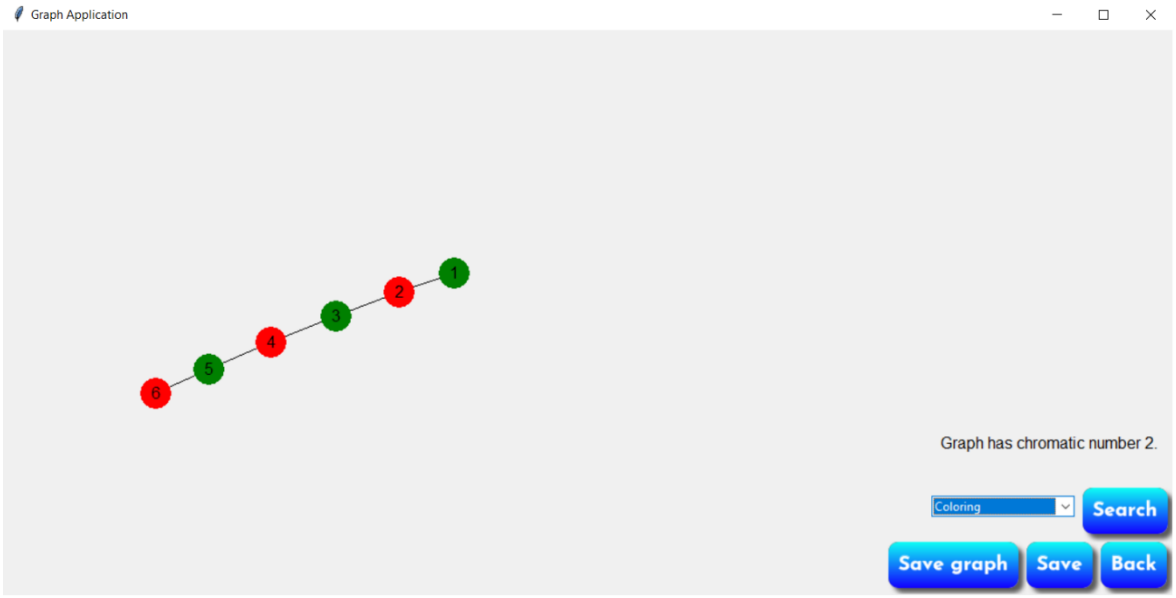
Na obrázku 13 vidíme úvodnú obrazovku, ktorá sa spustí hneď po otvorení našej aplikácie. Máme tu dve možnosti a to zadať nový graf alebo načítať existujúci. Pri zvolení možnosti načítania grafu sa nám otvorí okno, v ktorom vyberieme textový súbor zo zariadenia.



Obrázok 13: Zadávanie vrcholov a hrán v aplikácii

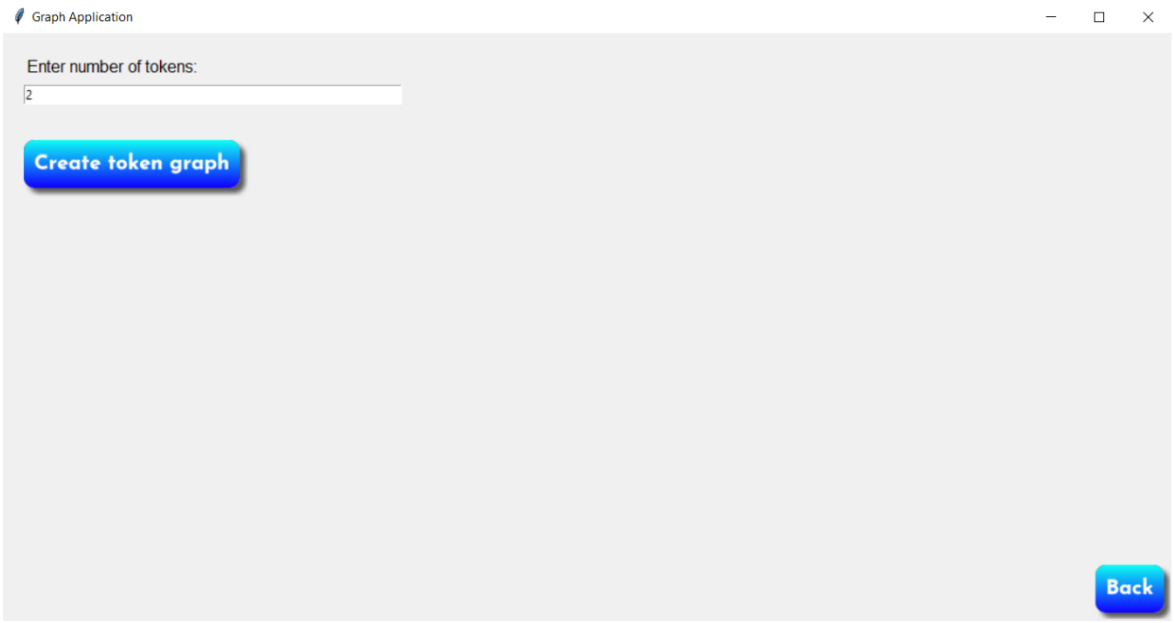
Po zvolení možnosti či chceme nový alebo existujúci graf, sa nám ukážu dve polia, kde je možné zadať vrcholy a hrany grafu. V prípade že sme zvolili možnosť načítania grafu polia budú vyplnené, no vieme ich ďalej upravovať.

Tak ako vidíme na obrázku 14, máme opäť dve možnosti a to či chceme vytvoriť graf alebo token graf. Ak zvolíme možnosť vytvoriť graf dostaneme sa na plochu, ktorá je na obrázku 15. Na tomto obrázku je okrem vykresleného grafu zakreslený aj jeden z implementovaných algoritmov, konkrétne vyfarbovanie grafu. V pravom dolnom roku môžeme vidieť, že tento konkrétny šesť-vrcholový graf typu cesta má chromatické číslo 2, čo znamená že nám stačia dve farby na vyfarbenie grafu tak, že žiadne dva susedné vrcholy nebudú mať rovnakú farbu, čo vidíme aj graficky znázornené červenou a zelenou farbou.



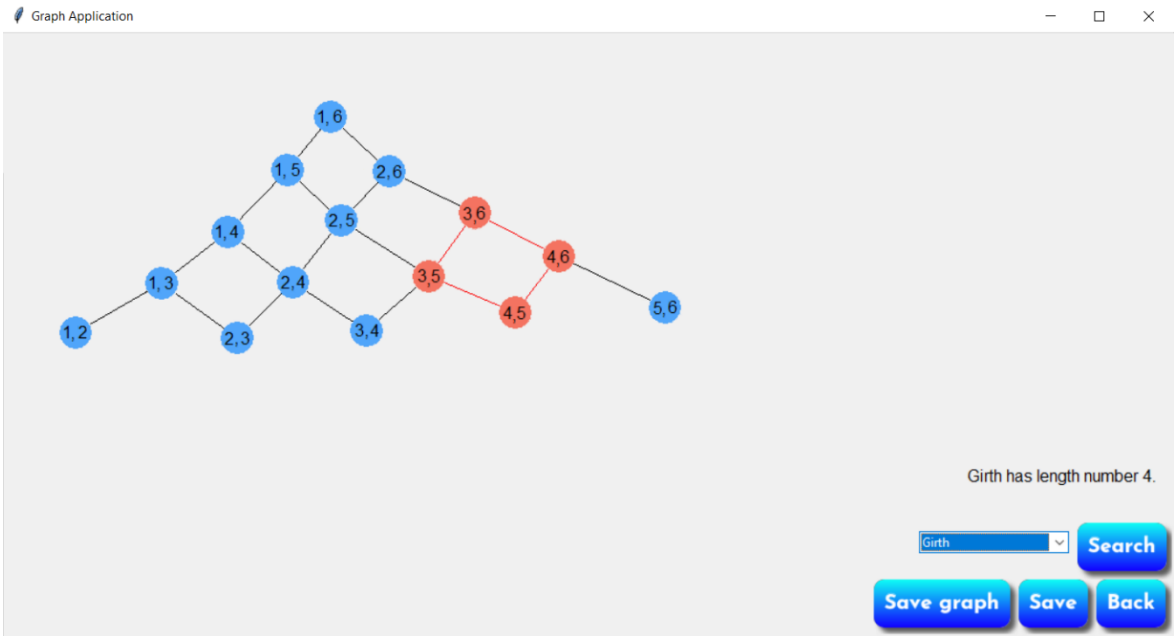
Obrázok 15: Graf s vypočítaním chromatickým číslom v aplikácii

Na tejto ploche môžeme stlačiť tlačidlo späť, vďaka ktorému sa dostaneme naspať k zadávaniu vrcholov, pričom vypísané vrcholy sa nám nestratia. Takto môžeme z toho istého grafu vytvoriť token graf bez opätovného zadávania parametrov. Po zvolení možnosti vytvorenia token grafu dostaneme sa na plochu kde zadáme počet token grafov, túto časť aplikácie môžeme vidieť na obrázku 16.



Obrázok 16: Zadávanie počtu token v aplikácii

Po zvolení počtu tokenov sa vykreslí graf na ktorom opäť môžeme testovať jednotlivé algoritmy. V našom prípade sme na 2-token grafe, ktorý je vytvorený zo šesť-vrcholového grafu typu cesta spustili algoritmus na hľadanie najmenšieho cyklu. Tento cyklus vidíme na grafe farebne odlíšený, aj graf obsahuje viac cyklov rovnakej dĺžky, zakreslí iba jeden z nich.



Obrázok 17: Token graf s vypočítanou dĺžkou najmenšieho cyklu v aplikácii

4. EXPERIMENT

V tejto kapitole si predstavíme a analyzujeme výsledky našich experimentov. Pokúsime sa odpovedať na otázku, či vždy keď zoberieme graf určitého typu, a pretransformujeme ho na token graf, či bude mať vždy nejaké z testovaných vlastností.

4.1. Graf typu cesta

Ako prvé sme testovali grafy typu cesta. Testovali sme niekoľko rôznych grafov, začali sme najmenším osem-vrcholovým grafom. Kvôli časovej zložitosti sme testovali iba grafy do osemnásť vrcholov, keďže najväčší token graf, ktorý vznikne z 18-vrcholového grafu, má 48620 vrcholov. Keby chceme testovať 20-vrcholový graf, náš algoritmus by musel prejsť cez 184756 vrcholov, čo je časovo omnoho náročnejšie.

V tabuľke číslo 1 vidíme výsledky z testovania grafu typu cesta, ktorý mal 14 vrcholov. Pre porovnanie sme uvideli aj 1-token graf, ktorý je totožný s našim pôvodným grafom. Ako môžeme vidieť v tejto tabuľke, token graf ktorý vznikol z grafu typu cesta, nie je ani stromom, nie je to ani úplný ani regulárny graf, nie je ani Johnsonov graf a nemá Eulerov cyklus. Každý graf typu cesta obsahuje Hamiltonovskú cestu. Podľa základných vlastností token grafov v kapitole Základné pojmy, vieme že ak graf s n vrcholmi obsahuje Hamiltonovskú cestu, n je párne číslo a k je nepárne číslo, tak aj v k-token grafe existuje Hamiltonovská cesta. Túto informáciu sa nám experimentom podarilo overiť. Zaujímavé sú aj číselné výsledky, pri ktorých môžeme vidieť že každý token graf ktorý vznikne z grafu typu cesta má najmenší cyklus dĺžky 4 a vrcholová a hranová súvislosť je 1.

Tabuľka 1: Graf typu cesta s 14 vrcholmi

	1	2	3	4	5	6	7
Je planárny	T	T	F	F	F	F	F
Je strom	T	F	F	F	F	F	F
Je úplný	F	F	F	F	F	F	F
Je regulárny	F	F	F	F	F	F	F
Vrcholová súvislosť	1	1	1	1	1	1	1

Hranová súvislosť	1	1	1	1	1	1	1
Obvod	∞	4	4	4	4	4	4
Eulerova cesta	T	F	F	F	F	F	F
Eulerov cyklus	F	F	F	F	F	F	F
Hamiltonovská cesta	T	F	T	F	T	F	T
Vyfarbovanie	2	2	4	4	4	4	4
Johnsonov graf	F	F	F	F	F	F	F

4.2. Graf typu hviezda

Ďalší typ grafu, ktorý sme testovali bola hviezda. Opäť sme testovali aj menšie aj väčšie grafy. V tabuľke 2 vidíme výsledky testovania 17 vrcholového grafu typu hviezda. Žiadny z k-token grafov neobsahuje Hamiltonovskú ani Eulerovu cestu. Každý graf typu hviezda, ktorý nie je zároveň typu cesta, to znamená že počet vrcholov je väčší ako 3, má v akomkoľvek k-token grafe najmenší cyklus dĺžky 6. Vrcholová a hranová súvislosť sa vždy rovná počtu tokenov v grafe, a chromatické číslo sa rovná dvom.

Tabuľka 2: Graf typu hviezda so 17 vrcholmi

	1	2	3	4	5	6	7	8
Je planárny	T	F	F	F	F	F	F	F
Je strom	T	F	F	F	F	F	F	F
Je úplný	F	F	F	F	F	F	F	F
Je regulárny	F	F	F	F	F	F	F	F
Vrcholová súvislosť	1	2	3	4	5	6	7	8
Hranová súvislosť	1	2	3	4	5	6	7	8
Obvod	∞	6	6	6	6	6	6	6
Eulerova cesta	F	F	F	F	F	F	F	F
Eulerov cyklus	F	F	F	F	F	F	F	F
Hamiltonovská cesta	F	F	F	F	F	F	F	F
Vyfarbovanie	2	2	2	2	2	2	2	2
Johnsonov graf	F	F	F	F	F	F	F	F

4.3. Graf typu strom

Ďalším testovaným typ grafu bol strom. Testovali sme grafy od 8 do 18 vrcholov. Rovnako sme testovali rôzne typy stromov, binárne, plne binárne, ternárne ale aj také, ktoré nemali špecifikované obmedzenie na počet detí. V tabuľke 3 sú výsledky testovania plne binárneho pätnásť-vrcholového stromu. Ako vidíme v tabuľke vo výsledných token grafoch sa nenachádzali ani Hamiltonovské ani Eulreove cesty, grafy vzniknuté zo stromov neboli stromy, neboli ani úplné, ani planárne, ani regulárny ani Johnsonove. Vytvorením token grafu zo stromu nedosiahneme žiadnu z našich testovaných vlastností.

Tabuľka 3: Plne binárny strom s 15 vrcholmi

	1	2	3	4	5	6	7
Je planárny	T	F	F	F	F	F	F
Je strom	T	F	F	F	F	F	F
Je úplný	F	F	F	F	F	F	F
Je regulárny	F	F	F	F	F	F	F
Vrcholová súvislosť	1	2	1	2	2	2	1
Hranová súvislosť	1	2	1	2	2	2	1
Obvod	∞	4	4	4	4	4	4
Eulerova cesta	F	F	F	F	F	F	F
Eulerov cyklus	F	F	F	F	F	F	F
Hamiltonovská cesta	F	F	F	F	F	F	F
Vyfarbovanie	2	2	3	2	5	4	5
Johnsonov graf	F	F	F	F	F	F	F

4.4. Úplný graf

V testovaní sme pokračovali úplnými grafmi. Vďaka testovaniu rôznych úplných grafov sme prakticky overili teoretické znalosti o Johnsonových grafoch, nakoľko z každý úplný graf transformovaný na token graf má tú vlastnosť že je Johnsonovým grafom. A nakoľko sú to Johnsonove grafy vieme s istotou povedať,

že sú aj regulárne a zároveň podľa základných vlastností z kapitoly Základné pojmy obsahujú Hamiltonovskú cestu čo môžeme vidieť aj v tabuľke číslo 4. Takisto v tabuľke vidíme aj to, že každý k-token graf obsahuje Eulerovskú cesta a aj cyklus.

Zaujímavé sú hodnoty pri vrcholovej a hranovej súvislosti. Podarilo sa nám zistiť, že čísla pri jednotlivých k-token grafoch vieme vypočítať kvadratickou postupnosťou $a_k = nk - k^2$. Kde k je počet tokenov a n je počet vrcholov v pôvodnom grafe.

Tabuľka 4: Úplný graf s 15 vrcholmi

	1	2	3	4	5	6	7
Je planárny	F	F	F	F	F	F	F
Je strom	F	F	F	F	F	F	F
Je úplný	T	F	F	F	F	F	F
Je regulárny	T	T	T	T	T	T	T
Vrcholová súvislosť	14	26	36	44	50	54	56
Hranová súvislosť	14	26	36	44	50	54	56
Obvod	3	3	3	3	3	3	3
Eulerova cesta	T	T	T	T	T	T	T
Eulerov cyklus	T	T	T	T	T	T	T
Hamiltonovská cesta	T	T	T	T	T	T	T
Vyfarbovanie	15	15	16	16	16	16	16
Johnsonov graf	T	T	T	T	T	T	T

4.5. Regulárny graf

Posledný z experimentov sme robili na regulárnych grafoch. V tabuľke číslo 5, v ktorej sa nachádzajú výsledky z testovania 3-regulárneho grafu s 12 vrcholmi, môžeme vidieť že k-token graf, kde k je párne číslo, obsahuje Eulerovu cestu aj cyklus, a kde k je nepárne neobsahuje ani cestu ani cyklus. To nás viedlo k bližšiemu preskúmaniu tohto úkazu. Testovali sme rôzne token grafy, ktoré vznikli z n-regulárnych grafov kde m je počet vrcholov. Testovali sme také grafy kde čísla n aj m boli párne aj nepárne čísla (tabuľka 5 a 6).

Tabuľka 5: 3-regulárny graf s 12 vrcholmi

	1	2	3	4	5	6
Je planárny	F	F	F	F	F	F
Je strom	F	F	F	F	F	F
Je úplný	F	F	F	F	F	F
Je regulárny	T	F	F	F	F	F
Vrcholová súvislosť	3	4	5	4	5	4
Hranová súvislosť	3	4	5	4	5	4
Obvod	4	4	4	4	4	4
Eulerova cesta	F	T	F	T	F	T
Eulerov cyklus	F	T	F	T	F	T
Hamiltonovská cesta	T	F	T	F	T	F
Vyfarbovanie	3	4	5	6	6	7
Johnsonov graf	F	F	F	F	F	F

Tabuľka 6: 4-regulárny graf s 15 vrcholmi

	1	2	3	4	5	6	7
Je planárny	F	F	F	F	F	F	F
Je strom	F	F	F	F	F	F	F
Je úplný	F	F	F	F	F	F	F
Je regulárny	T	F	F	F	F	F	F
Vrcholová súvislosť	4	6	6	6	6	4	6
Hranová súvislosť	4	6	6	6	6	4	6
Obvod	3	3	3	3	3	3	3
Eulerova cesta	T	T	T	T	T	T	T
Eulerov cyklus	T	T	T	T	T	T	T
Hamiltonovská cesta	T	F	F	F	F	F	F
Vyfarbovanie	4	6	7	8	9	10	9
Johnsonov graf	F	F	F	F	F	F	F

Existujú tri kombinácie ako sa dá vytvoriť k-token graf n-regulárneho grafu ktorý má m vrcholov. Prvá možnosť je že n aj m je párne, druhá možnosť je že n je párne a m nepárne. V tretej možnosti je n nepárne a m párne. Regulárny graf, kde by boli čísla n aj m nepárne, nie je možné zostrojiť, preto sú možnosti iba tri.

Predpokladali sme že všetky token grafy kde n a m je párne alebo n je párne a m je nepárne budú obsahovať Eulerovu cestu a cyklus vo všetkých k. A grafy kde n je nepárne a m je párne budú Eulerovu cestu a cyklus obsahovať len v k-token grafoch kde k je párne.

Oba naše predpoklady sme vyvrátili tým že sme našli kontrapríklad v ktorom bol náš predpoklad nesplnený. Preto nie je nič, čo by sme o token grafoch, vytvorených z regulárnych grafov, mohli tvrdiť so stopercentnou istotou.

Tak ako v predchádzajúcich experimentoch aj v tomto sme si znovu overili, že ak graf s n vrcholmi obsahuje Hamiltonovskú cestu, n je párne číslo a k je nepárne číslo, tak aj v k-token grafe existuje Hamiltonovská cesta.

ZÁVER

Hlavným cieľom bakalárskej práce bolo naštudovanie si teórie token grafov, a na základe získaných informácií vytvoriť jednoduchú a intuitívnu aplikáciu na ich analýzu. Po dôkladnom preštudovaní základných princípov a vlastností token grafov sme sa zamerali na praktickú implementáciu týchto poznatkov.

V prvej úvodnej kapitole sme si objasnili základné pojmy z teórie grafov, ktoré boli nevyhnutné pre správne pochopenie danej problematiky a následne sme si vysvetlili základné vlastnosti token grafov.

V druhej kapitole sme porovnávali existujúce technológie ako je napríklad open-source softvér Gephi, ktorý implementuje algoritmy na analýzu grafov, bližšie sme sa pozreli na knižnicu NetworkX, ktorú sme napokon v našej práci využili na uloženie štruktúry grafov, a knižnicu Tkinter pomocou, ktorej sme spravili používateľské rozhranie. Na záver tejto kapitoly sme si podrobne popísali špecifické požiadavky, ktoré boli kladené na našu aplikáciu.

V tretej kapitole sme sa zamerali na samotnú implementáciu. Oboznámili sme sa so základnou štruktúrou kódu, ktorá bola rozdelená do dvoch tried. Jedna trieda sa venovala správe grafových štruktúr pomocou knižnice NetworkX, zatiaľ čo druhá trieda sa sústredila na používateľské rozhranie ktoré sme implementovali pomocou knižnice Tkinter. Každá metóda bola podrobne vysvetlená, aby bolo jasné, ako prispieva k celkovej funkčnosti aplikácie. Vysvetlili sme, ako jednotlivé časti kódu spolupracujú na vytváraní efektívneho a užívateľsky prívetivého nástroja na analýzu token grafov.

Nakoniec sme aplikáciu v štvrtej záverečnej kapitole podrobili experimentálnemu overeniu. Pre toto overenie sme si vybrali päť typov grafov ako je cesta, strom, strom typu hviezda, regulárny graf a úplný graf. Aby sme mali dostatočne veľkú testovaciu vzorku, pre každý typ sme si pripravili niekoľko grafov, ktoré sme pretransformovali na rôzne k-token grafy. Na týchto grafoch sme testovali vybrané vlastnosti ako je napríklad dĺžka najmenšieho cyklu, aké je chromatické číslo alebo aj či je graf úplný. Ďalším krokom bolo vyhodnotenie údajov získaných z experimentálneho testovania. Zistili sme zaujímavé vlastnosti ako to, že token graf, ktorý vznikol z grafu typu cesta, bude mať vrcholovú a hranovú súvislosť vždy rovnú číslu 1 a graf typu hviezda má vrcholovú a hranovú súvislosť vždy rovnú počtu tokenov v tomto grafe. Podarilo sa nám overiť informácie ktoré sme si na začiatku práce

naštudovali a to napríklad to, že ak má graf Hamiltonovskú cestu, párny počet vrcholov a počet tokenov v token grafe je nepárny, tak sa aj v tomto token grafe nachádza Hamiltonovská cesta. Taktiež sme overili to, že ak je náš počiatočný graf úplný tak je aj Johnsonovým grafom a všetky token grafy sú tiež Johnsonové.

Napriek všetkým problémom, na ktoré sme pri práci narazili, ako napríklad efektivita algoritmov pri veľkých grafoch, sa nám podarilo splniť ciele našej práce. Naša aplikácia úspešne analyzuje a vizualizuje token grafy.

POUŽITÁ LITERATÚRA

[1] Stanoyevitch Alexander, Discrete Structures with Contemporary Applications. Chapman and Hall/CRC Press. ISBN 2011. 978-1-4398-1768-1.

[2] Kenneth H. Rosen, Discrete Mathematics and Its Applications, Seventh Edition. McGraw-Hill Press. 2012. ISBN 978-0-07-338309-5.

[3] https://edu.fmph.uniba.sk/~winczer/diskretna/pred8z03.pdf

[4] Steven Skiena, Implementing discrete mathematics: combinatorics and graph theory with Mathematica. 1990. ISBN 0-201-50943-1.

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

[6] https://frcatel.fri.uniza.sk/users/paluch/grkap1-nopics.pdf

[7] https://arxiv.org/pdf/1512.03547

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

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

[10] https://arxiv.org/pdf/0910.4774

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

[12] https://networkx.org/documentation/stable/

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

[14] https://www.python.org/doc/