Analiza strukture grafova primjenom programskog modula NetworkX

Matković, Darin

SVEUČILIŠTE U SPLITU

PRIRODOSLOVNO MATEMATIČKI FAKULTET

ZAVRŠNI RAD

Analiza strukture grafova primjenom programskog modula NetworkX

Darin Matković

Split, rujan 2022.

Temeljna dokumentacijska kartica

Završni rad

Sveučilište u Splitu

Prirodoslovno-matematički fakultet

Odjel za informatiku

Ruđera Boškovića 33, 21000 Split, Hrvatska

Analiza strukture grafova primjenom programskog Modula NetworkX

Darin Matković

SAŽETAK

Rad opisuje proces analize temeljnih koncepata teorije grafova kao i sučelja za rad s istim preko Pythonovog modula NetworkX. U početku rada čitatelj se upoznaje s konceptima teme. Nakon toga je definiran i objašnjen, u nekoliko osnovnih uvodnih primjera princip rada s modulom kao npr. stvaranje grafa, dodavanje i brisanje čvorova/bridova, instalacija potrebnih paketa te je dana na uvid analiza zašto nam je to korisno i kada je dobro a kad ne koristiti Pythonov paket.

Ključne riječi: NetworkX, Python, Teorija grafova, Algoritmi najkraćeg puta
Rad sadrži: 33 stranice, 21 grafičkih prikaza, 0 tablica i 14 literaturnih navoda. Izvornik je na hrvatskom jeziku

Mentor: **izv. prof. dr. sc. Ani Grubišić**, *Prirodoslovno-matematički fakultet, Sveučilišta u Splitu*

Ocjenjivači: **izv. prof. dr. sc. Ani Grubišić**, *Prirodoslovno-matematički fakultet, Sveučilišta u Splitu*

izv. prof. dr. mag. Branko Žitko, *Prirodoslovno-matematički fakultet, Sveučilišta u Splitu*

mag. ing. comp. Ines Šarić-Grgić, *asistent Prirodoslovno-matematičkog fakulteta, Sveučilišta u Splitu*

Basic documentation card

Thesis

University of Split

Faculty of Science

Department of Informatics

Ruđera Boškovića 33, 21000 Split, Croatia

Graph structure analysis using the NetworkX program module

Darin Matković

ABSTRACT

The paper describes the process of analyzing the basic concepts of graph theory as well as the interface for working with the same through Python's NetworkX module. At the beginning of the work, the reader is introduced to the concepts of the topic. After that, the principle of working with the module is defined and explained, in a few basic introductory examples, such as creating a graph, adding and deleting nodes/edges, installing the necessary packages, and an analysis of why it is useful for us and when it is good and when it is not. use the Python package.

Keywords: NetworkX, Python, Graph theory, Shortest path algorithms
Thesis consists of: 33 pages, 21 figures, 0 tables and 14 references.
Original language: Croatian

Mentor: **Ani Grubišić, Ph.D.** *Associate Professor of Faculty of Science, University of Split*

Reviewers: **Ani Grubišić, PhD**, *Associate Professor, Faculty of Science, University of Split*

Branko Žitko, PhD, *Associate Professor, Faculty of Science, University of Split*

Ines Šarić-Grgić, mag. ing. comp., *Assistant, Faculty of Science, University of Split*

IZJAVA

kojom izjavljujem s punom materijalnom i moralnom odgovornošću da sam završni rad s naslovom **Analiza strukture grafova primjenom programskog modula NetworkX** izradio samostalno pod voditeljstvom mentorice Ani Grubišić. U radu sam primijenio metodologiju analitičke obrade teme rada i koristio literaturu koja je navedena na kraju završnog rada. Tuđe izjave, zaključke, teorije i zakonitosti koje sam izravno ili parafrazirajući naveo u završnom radu na uobičajen, standardan način citirao sam i povezao s fusnotama s korištenim bibliografskim jedinicama.

Potvrđujem da je za pohranu dostavljena završna verzija obranjenog i dovršenog završnog rada. Ovom izjavom, kao autor dajem odobrenje i da se moj rad, bez naknade, trajno javno objavi i besplatno učini dostupnim studentima i djelatnicima ustanove.

Student:

Darin Matković

Tablica sadržaja

Ovim putem zahvaljujem svima koji su mi aktivno i direktno, a tako i indirektno pomogli na putu do kraja preddiplomskog studija. Najviše od svega obitelji i kolegama koji su bili dio tog puta, a tako i nastavnicima/cama na ukazanom vremenu i znanju koje su u proteklih par godina prenosili na nas. Posebno bih se htio zahvaliti mentorici na ukazanom poštovanju, razumjevanju, strpljenju i pomoći pri izradi ovog rada.

1. Uvod

U ovom radu je kroz poglavlja pojašnjena struktura grafova, analiza NetworkX modula s djelomično općenitim ali također i detaljnim objašnjenjem korištenja tog istog. Kroz rad čitatelj će se susresti s matematičkom analizom „Teorije grafova“ koja je obrađena i raspisana kao uvod i osnova za daljnju analizu NetworkX modula. Osim teorije grafova čije poznavanje je temelj za rad s Pythonovom bibliotekom tj. modulom NetworkX, navedeni su koraci instalacije potrebnih paketa, osnovne naredbe za „basic“ korištenje programskog modula i primjenu. Osim navedenih stavki još su analizirani neki od najosnovnijih ili bolje rečeno, u praksi najkorištenijih algoritama u ovom slučaju **algoritmi najkraćeg puta** a s tim i na temeljnim primjerima pokazana logika koja stoji iza istih.

2. Teorija grafova - općenito

Teorija grafova, grana je diskretne matematike koja se bavi analizom i svojstvima grafovima, vrstom matematičkih objekata koji su nam jako korisni jer njima možemo modelirati kompleksne probleme i zahtjeve na vrlo jednostavan način. Da bi razumjeli funkcionalnosti NetworkX paketa, prvo morate razumjeti grafove, matematička analiza je obrazložena u nastavku.

Rad s grafovima je funkcija navigacije rubovima i čvorovima kako bi se otkrili i razumjeli složeni odnosi i/ili optimizirali putovi između povezanih podataka u mreži.

U stvarnom svijetu čvorovi mogu biti ljudi, grupe, mjesta ili stvari kao što su kupci, proizvodi, članovi, gradovi, trgovine, zračne luke, luke, bankovni računi, uređaji, mobilni telefoni, u teoretskom načelu one mogu biti bilo što, kao na primjer u kemiji i fizici atomi i molekule ili u informatici web stranice, neuronske mreže , u medicini i farmakologiji za izračune raznih permutacija i istraživanje spojeva i sl.

Primjeri rubova ili odnosa između čvorova uključuju prijateljstva, mrežne veze, hiperveze, ceste, rute, žice, telefonske pozive, e-poštu, "lajkove", plaćanja, transakcije, telefonske pozive i poruke društvenih mreža. Rubovi mogu imati jednosmjernu strelicu smjera koja predstavlja vezu od jednog čvora do drugog.

Analiza grafova može se koristiti za određivanje jačine i smjera odnosa između objekata u grafu. Potražnja za alatima za analizu odnosa ima gotovo neograničen potencijal s obzirom na rastuću ulogu mreža u informacijskom sustavu. Utjecaj društvenih mreža na sve, od odluka o kupnji do nacionalnih izbora, potaknuo je interes za analizu grafova. Posebno je koristan u otkrivanju odnosa koji nisu očiti zbog složenosti mreže ili broja putova između čvorova.

Analiza grafova je u praksi korištena za postizanje nekih od sljedećih ciljeva:

- Otkrivanje financijskih zločina kao što je pranje novca.
- Identificirajte lažnih transakcija i aktivnosti.
- Analiza utjecaja u zajednicama društvenih mreža.
- Identificirajte slabosti sustava u raznim elektroenergetskim, vodovodnim i transportnim mrežama

- Optimizirajte rute u zračnim prijevoznicima, maloprodaji i proizvodnoj industriji
- Tijekom COVID-19, identificiranje ljudi koji su susreli zaražene osobe tijekom određenog vremenskog razdoblja, aplikacijska primjena koja je doslovno imala posljedice na život i smrt.
- Pružanje sadržaja društvenog marketinga koji se temelji na odnosima između korisnika - čak i ako se korisnici ne poznaju - mapiranjem sličnih interesa i zajedničkih veza.
- Dopuštanje tražilicama posluživanje rezultata na temelju preferencija koje proizlaze iz ponašanja ljudi sa sličnim zahtjevima za informacijama (eng. cookies).

2.1. Graf

Grafovi su matematičke strukture koje se koriste za modeliranje mnogih vrsta odnosa i procesa u fizičkim, biološkim, društvenim i informacijskim sustavima. Prva objavljena publikacija iz teorije grafova je bilo Eulerovo rješenje problema šetnje Koenigsberškim mostovima. Taj rad je službeno publiciran davne 1736. godine. Prva knjiga iz teorije grafova napisana je međutim tek 200 godina poslije, a njen autor je D. Koenig, a također taj događaj se smatra početkom razvoja teorije grafova kao zasebne matematičke cjeline. [10].

Graf je struktura koja se sastoji od čvorova ili vrhova (koji predstavljaju "entitete" u sustavu) koji su povezani bridovima (bridovi predstavljaju odnose tj. veze između tih "entiteta". U ranim 60-tim godinama dvadesetog stoljeća počinje uzlet razvoja istraživanja teorije grafova a samim tim i njenim primjenama, a taj trend traje i danas, u nastavku vidjeti ćemo i zašto [10].

2.1.1. Grafovi, vrhovi, bridovi i njihova povezanost

Graf je uređena trojka definirana preko relacije $G=(V,E,\varphi)$ gdje je $V=V(G)$ zapravo neprazan skup čije smo elemente nazivali vrhovima, E=E(G) skup je disjunktan s V čije elemente nazivamo bridovima i φ je funkcija koja će svakom bridu e iz E pridružiti

par $\{u,v\}$, najčešće ali nije uvjet, različitih vrhova iz V. Graf skraćeno označavamo $G=(V,E)$ ili samo G. [10]

Vrhove u i v koji su pridruženi bridu e nazivamo **krajevima** brida e. Bridove čiji se krajevi međusobno podudaraju po stupnju još zovemo **petljom**. Postoje dva tipa petlji a to su otvorene i zatvorene. **Višestruki bridovi** su dva ili više njih s istim parom krajeva. Za par vrhova u i v kažemo da su **susjedni** ako postoji brid e kojemu su oni krajevi. Pri tome kažemo da je brid e **incidentan** s vrhovima u i v, za što ćemo upotrijebiti oznaku $e=\{u,v\}$ ili $e=uv$ [10].

Stupanj vrha v u grafu G je broj bridova grafa G incidentnih s v, pri čemu se svaka petlja računa kao dva brida. Izolirani vrh je vrh stupnja 0. Posebno, zanimljivo je svakome grafu G pridruziti niz stupnjeva. Za graf s n vrhova to je n -torka koja se sastoji od rastućeg niza cijelih brojeva koji predstavljaju stupnjeve svih vrhova u grafu G (zajedno s kratnostima). Stupanj čvora (engl. degree) je isti kao i broj grana gdje je isti krajnja ili završna točka [10].

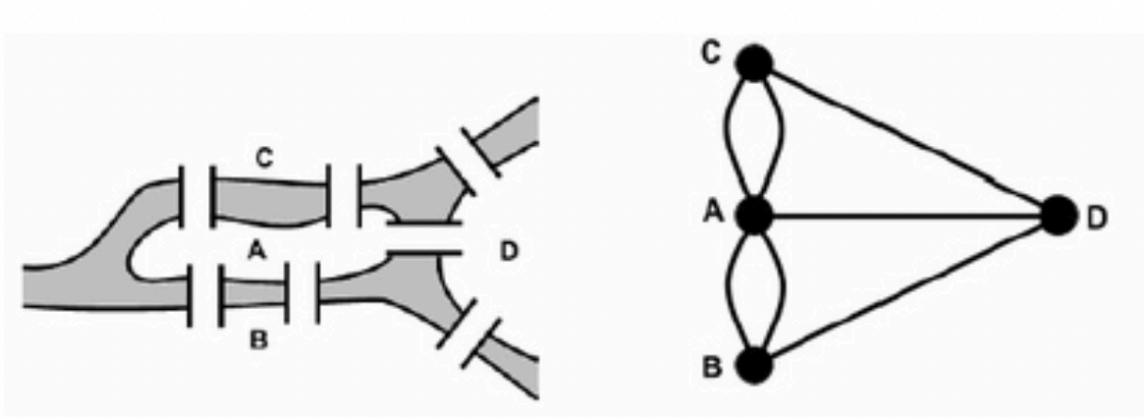
Šetnja, put i staza

Šetnja u grafu G je niz čiji članovi su naizmjenično vrhovi v_i i bridovi e_i, tako da su krajevi od e_i vrhovi v_{i-1} i v_i , i=1,…,ki=1,…,k. Kažemo da je v_0 početak a v_k kraj šetnje W, da je W šetnja od v0 do vk ili (v0,vk) - šetnja. Šetnja se, u različitim literaturama još može i skraćeno zapisati samo bridovima $W=e_1e_2...e_k$. Šetnja je zatvorena ako se početak i kraj podudaraju, tj. ako je $v_0=v_k$. Šetnju u kojoj se svi bridovi između sebe razlikuju nazivamo **stazom** [10].
Put je definiran kao staza čiji se vrhovi međusobno razlikuju u smislu da dva vrha u i v grafa G su povezana ako postoji (u,v) - put u G. Graf je povezan ako su svaka dva njegova vrha povezana. **Eulerova staza** je staza koja sadržava svaki brid grafa isključivo samo jednom (jedan posjet svakom čvoru). **Eulerova tura** u grafu je zatvorena Eulerova staza. Ako graf dopušta Eulerov put možemo kazati da je to **Eulerov graf** [10].

U literaturama se ponekad Eulerova staza, put i graf može navesti kao i Hamiltonova staza,put i graf iz razloga što su oba matematičara radili na istoj temi i uvelike pridonijeli razvoju teorije grafova kao matematičke discipline.

Povezani graf nazivamo još i Eulerov graf ukoliko su mu svi vrhovi parnog stupnja. Povezani graf sadrži Eulerovu stazu ukoliko ispunjava nužan uvjet da ima ni više ni manje nego 2 vrha stupnja koji je neparne potencije.

Graf sa *Slike 1*. koji prikazuje ilustracijski i grafički model jednog od najpopularnijih a isto tako i temeljnog problema, a to su Koenigsberški mostovi. Na njemu su sva četiri vrha stupnja koji je neparan. Na temelju toga i definicije **Eulerove staze** možemo spoznati da taj graf nema Eulerov put, a tako ni Eulerovu stazu. Dakle, šetnja gradom na način da se svakim mostom prođe samo i točno jednom nije ostvariva.



Slika 1. Shematski prikaz problema Koenigsberških mostova

Rješavajući probleme "obilazaka gradskih ulica" često treba uzeti u obzir neophodni argument, a to je "dopušteni smjer kretanja". Traženje matematičkog modela za sličnu problematiku nas vodi prema temi koju još nazivamo problemom usmjerenih grafova.

2.1.2. Usmjereni graf

Usmjereni graf *D* ili ga još nazivamo kraće **digraf** je uređena trojka $D=(V,A,\psi)$, gdje je $V=V(D)$ neprazan skup čije elemente nazivamo **vrhovima**, $A=A(D)$ skup koji je

disjunktan s V čije elemente još nazivamo i **lukovima.** Simbol ψ je zapravo funkcija koja svakom luku a iz A pridružuje uređeni par (u,v), pri čemu su $u,v \in V$ ali nisu nužno različiti. Kažemo da je u početak, a v kraj luka a, da je smjer ili orijentacija luka a od u prema v i koristimo oznaku $a=(u,v)$. Digraf skraćeno označavamo $D(V,A)$ ili samo D.

Pojmove koje smo imali kod grafova analogno definiramo i za digrafove. Dva ili više lukova s istim početkom i krajem su **višestruki lukovi.**

Usmjerena šetnja u digrafu D je niz čiji članovi su naizmjenično vrhovi i lukovi. Kažemo da je v_0 početak, a v_k kraj usmjerene šetnje W ili da je $W(v_0,v_k)$ usmjerena šetnja. Usmjerena šetnja je još i zatvorena šetnja ako je $v_0 = v_k$.

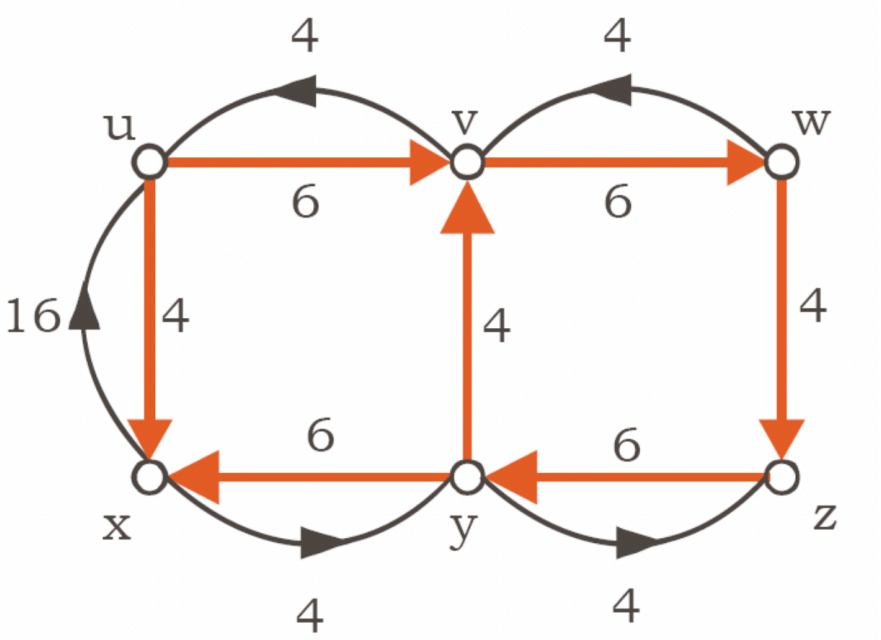
Usmjerenu šetnju kojoj su svi lukovi međusobno različiti još nazivamo **usmjerenom stazom** [11] .

Usmjereni put je staza koja s orjentacijom čiji su vrhovi međusobno različiti.

Usmjerena Eulerova tura je usmjerena zatvorena staza koja svaki luk digrafa sadržava točno jedanput.

Eulerov digraf je vrsta grafa onaj koji sadrži Eulerovu turu. Pripadajući graf G (D) digrafa D je graf dobiven zamjenom svakog luka a = (u , v) bridom e = {u , v} . Digraf D je slabo povezan ili kraće povezan ako je pripadajući graf povezan, a jako povezan ako za svaka dva vrha $u,v \in V(D)$ postoje (u,v) i (v,u) usmjereni putovi.

Ulazni (izlazni) stupanj vrha v digrafa je broj lukova kojima je v kraj (početak). Povezani digraf je Eulerov ako i samo ako je za svaki njegov vrh ulazni stupanj jednak izlaznom [11]**.**



Slika 2. Shema usmjerenog grafa

Smjer usmjerenog grafa je na skici grafa označen strelicom koja pokazuje od početka grafa prema kraju. Početnu točku možemo u praksi odabirati proizvoljno. Primjer jednog digrafa je prikazan na Slici 2. Vidimo da taj digraf nije Eulerov jer ne zadovoljava uvjet o stupnjevima a to je „*Povezani digraf je Eulerov ako i samo ako je za svaki njegov vrh ulazni stupanj jednak izlaznom.*"

Ukoliko mu dodamo nekakav luk (v,w) i još jedan luk (z,y), novi digraf će poprimiti parametre Eulerovog grafa.

Težinski graf je uređeni par $(\boldsymbol{G,\omega})$, gdje je $\boldsymbol{G}$ graf i $\boldsymbol{\omega:E(G) \rightarrow \mathbb{R}+0\omega:E(G)\rightarrow R0}$ funkcija koja svakom bridu $\boldsymbol{e}$ iz $\boldsymbol{G}$ pridružuje nenegativni broj $\boldsymbol{\omega(e)}$. Vrijednost $\boldsymbol{\omega(e)}$ nazivamo težinom brida e.

Težine bridova tog istog grafa možemo predstavljati raznim vrijednostima kao što su udaljenost, vrijeme, troškovi i sl.

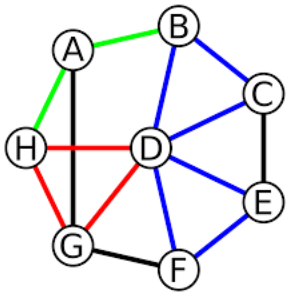
Graf je u gruboj definiciji skup objekata: vrhova, točaka ili čvorova koje povezuju bridovi odnosno crte (linije). Brid spaja dva čvora i to je odnos koji definira graf. Ako vrhove povezuje brid, grafove se prikazuje tako da se crtaju točke za svaki vrh i spajanjem odnosno povezivanjem lukom između ta dva vrha.

Da bi graf G', koji ima skup vrhova V(G') i skup bridova E(G') bio podgraf grafa G, grafa G koji ima skup vrhova V(G) i skupom bridova E(G), ako je V(G') podskup od V(G) i E(G′) podskup od E(G).
Ako podgraf G' ima isti skup vrhova kao i graf G', tad je G' je razapinjući podgraf grafa G.

2.1.3. Ciklus, šuma, stablo

Ciklus je zatvorena staza u kojoj su svi unutarnji vrhovi (tj. svi vrhovi osim krajeva) međusobno različiti. Ako je graf povezan i bez ciklusa, onda je taj graf stablo,a staza mu je pozitivne duljine.

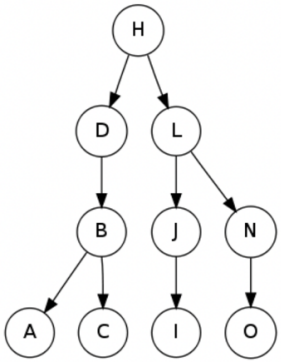
Ciklus C_k, duljine broja ciklusa koji je definiran s *k*, naziva se još *k*-ciklus. Ako je *k* paran onda je i *k*-ciklus paran, odnosno neparan ako je *k* neparan.



Slika 3. Prikaz ciklusa u grafu

U nekom grafu, ciklus predstavlja put u kojemu se prvi i posljednji vrh podudaraju u istoj točki. Duljinu najkraćeg ciklusa u grafu još nazivamo struk grafa [13].

Stablo je izraz za svaki graf u kojem su svaka njegova dva vrha povezana samo jednim jedinim putem. Svaki graf koji je povezan bez ijednog ciklusa jest po definiciji stablo. Ukoliko je graf koji promatramo povezan i neusmjeren, razapinjuće stablo u tom grafu je podgraf koji je ujedno i stablo i razapinje taj graf.

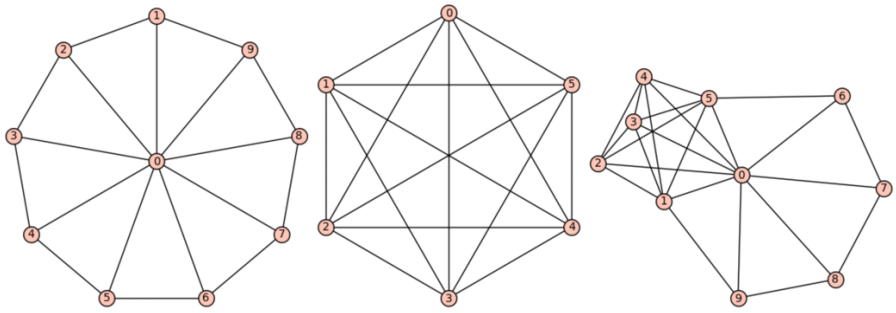


Slika 4. Primjer stabla

Šuma ili (eng. forest) je graf bilo usmjeren ili neusmjeren, težinski ili bestežinski) bez ciklusa, a povezana šuma zove se stablo (drvo; engl. tree). Ako je graf usmjeren, stablom smatramo svaku šumu koja je slabo povezana. Slika 13 - primjeri stabala U stablu može postojati poseban čvor koji se naziva korijenom (engl. root). Takvo se stablo naziva ukorijenjenim (engl. rooted). Od trenutka kada se ukorijeni stablo, svi bridovi postaju usmjereni. Njihov je smjer je po generalnom dogovoru unificiran a počinje sa korjenom - uvijek će biti usmjereni od čvora bližeg korijenu prema čvoru daljem od korijena. Ako postoji umreženost ili veza između dva čvora, onaj koji je bliži korijenu nazivamo roditeljem (eng. parent), a onaj dalji ili točnije zadnji u nizu dijetetom (eng. child) [13].

2.1.4. Unija grafova

Disjunktna unija ili direktna suma $G \cup H$ grafova G i H je graf sa skupom vrhova $V(G \cup H) = V(G) \cup V(H)$ i skupom bridova E(G∪H)=E(G)∪E(H) ili operacija koja kombinira dva ili više grafa da se stvori veći graf.



Slika 5. Primjer unija grafova

3. NetworkX biblioteka

NetworkX je Pythonov paket za kreiranje, manipulaciju i proučavanje strukture, dinamike i funkcija složenih mreža. Postoje mnoge primjene analize mreže grafova, kao što je analiza odnosa na društvenim mrežama, otkrivanje kibernetičkih prijetnji i identificiranje ljudi koji će najvjerojatnije kupiti proizvod na temelju zajedničkih preferencija.
NetworkX ima kapacitet za rad na vrlo velikim grafovima s više od 10 milijuna čvorova i 100 milijuna rubova(bridova). Čvorovi mogu biti bilo koji objekt koji se može hashirati, što znači da se njegova vrijednost nikada ne mijenja. To mogu biti tekstualni nizovi, slike, XML objekti, cijeli grafikoni i prilagođeni čvorovi. Osnovni paket uključuje mnoge funkcije za generiranje, čitanje i pisanje grafova u više formata.
Isti taj, osnovni paket, koji je besplatni softver pod BSD licencom također uključuje strukture podataka za predstavljanje takvih stvari kao što su jednostavni grafovi, usmjereni grafovi i grafovi s paralelnim rubovima i samopetljama. BSD licence (Berkeley Software Distribution) su "obitelji" slobodnih licenci koje nam daju software na korištenje, također ne uvjetuju velike restrikcije u smislu korištenja i distribucije tog istog softvera. Za razliku od bazičnih „copyleft" licenci, koje imaju te restrikcije za korištenje i djeljenje pod određenim uvjetima korištenja. BSD je ujedno licenca, a tako i klasa različitih uporabnih licenci (nazivamo ih još BSD-like licences). Modificirana BSD licenca koja je danas u vrlo širokoj upotrebi, je dosta slična onima koje su se izvorno koristile za BSD-ovu verziju Unixa. Ona je jednostavna licenca koja zahtijeva samo da kod zadržava notifikacije o BSD-u u slučaju da se redistribuira u formatu izvornog koda ili da redistribuira obavijest ako se u binarnom obliku. Za razliku od nekih ostalih BSD ne zahtjeva distribuciju izvornog koda.

NetworkX-a se najčešće upotrebljava za:

- Proučavanje strukture i dinamike društvenih, bioloških i infrastrukturnih mreža
- Standardizirano programsko okruženje za grafove
- Brzi razvoj suradničkih, multidisciplinarnih projekata
- Integracija s algoritmima i kodom napisanim na C, C++ i FORTRAN-u

- Rad s velikim nestandardnim skupovima podataka

NetworkX pruža standardizirani način da podatkovni znanstvenici i drugi korisnici matematike grafova surađuju, grade, dizajniraju, analiziraju i dijele modele mreže grafova. Kao besplatni softver koji je poznat po svojoj skalabilnosti i prenosivosti, NetworkX je naširoko prihvaćen od strane Python entuzijasta. To je također najpopularniji grafički okvir koji koriste znanstvenici podataka (Data Scientists), koji doprinose "živom" sustavu Python paketa i koji proširuju NetworkX značajkama kao što su numerička linearna algebra i crtanje. Projekti velikih podataka kao što su strojno učenje i duboko učenje često zahtijevaju suradnju mnogih članova tima. Dostupnost standardiziranih alata i formata uvelike pojednostavljuje dijeljenje informacija. Sa svojim korijenima u Pythonu, jednom od najpopularnijih jezika znanosti o podacima, NetworkX pruža proširenje za analizu grafova za Python biblioteke koje zahtijeva minimalnu obuku za korisnike Pythona i može se primijeniti nad mnogo različitih problemima.

Python je moćan programski jezik koji omogućuje jednostavne i fleksibilne prikaze mreža, te jasne i sažete izraze mrežnih algoritama (i drugih algoritama također). Osim toga, Python je također izvrstan "ljepljivi" jezik za sastavljanje dijelova softvera iz drugih jezika koji omogućuje ponovnu upotrebu naslijeđenog koda i inženjering algoritama visokih performansi. Jednako važno, Python je besplatan, dobro podržan i rado ga je koristiti zbog jednostavnosti (user friendly). Kako biste maksimalno iskoristili NetworkX, morati ćete znati pisati osnovne programe u Pythonu. NetworkX je stvoren u svibnju 2002. Originalnu verziju dizajnirali su i napisali Aric Hagberg, Dan Schult i Pieter Swart 2002. i 2003. Prvo javno izdanje bilo je u travnju 2005. Godine.

Struktura NetworkX-a može se vidjeti po organizaciji njegovog izvornog koda. Paket pruža **klase** za objekte grafova, **generatore** za stvaranje standardnih grafova, **IO rutine** za čitanje postojećih skupova podataka, **algoritme** za analizu rezultirajućih mreža i neke osnovne alate za crtanje.

Većinu NetworkX API-ja osiguravaju funkcije koje uzimaju grafički objekt kao argument. Metode grafičkog objekta ograničene su na osnovnu manipulaciju i izvještavanje. To osigurava modularnost koda i dokumentacije. Također, novopridošlicama olakšava učenje o paketu u fazama. Izvorni kod bi za svaki modul trebao biti jednostavan za čitanje, te čitanje ovog Python koda je zapravo dobar način da naučite više o mrežnim algoritmima.

3.1. Instalacija i osnovne naredbe

NetworkX zahtijeva Python verziju 3.8, 3.9 ili 3.10. Ako još nemate konfigurirano okruženje Python na vašem računalu, pogledajte upute za instalaciju cijelog znanstvenog Python okruženja. U nastavku pretpostavljamo da imate zadano Python okruženje već konfigurirano na vašem računalu i namjeravate instalirati NetworkX unutar njega. Prvo provjerite imate li instaliranu najnoviju verziju "pip"-a (upravitelj paketa za Python okruženje). Ako nemate instaliran, pogledajte dokumentaciju i prvo instalirajte PIP.

3.2. Što je PIP i kako ga koristiti?

PIP ili "preferirani program za instalaciju" je sustav za instalaciju i upravljanje paketa za Python. Ukoliko se pitate što su paketi, to su elementi programa koji u sebi sadrže sve datoteke koje su nam potrebne za neki modul. Moduli su s druge strane ugrađene biblioteke Python kodova koje možete implementirati u svoj projekt.

Provjera da li je PIP instaliran i process instalacije (Navedeno vrijedi za Windows operacijske sustave).

1.Korak – Provjera instalacije pip-a

```
pip help
```

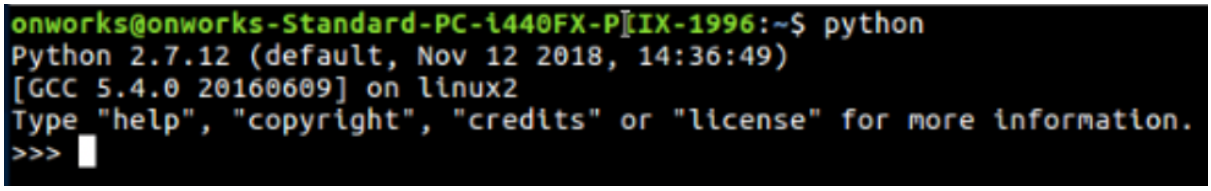
Slika 6. Naredba u Command promptu

U Naredbenom redku (Command prompt-u) upisemo naredbu: - pip help , zatim pritisnuti Enter.

Ukoliko je pip instaliran ispisati će se poruka koja objašnjava kako koristiti program, a ukoliko nije instaliran dobiti ćete poruku o pogrsšci u kojoj se navodi da program nije pronađen.

2.Korak - Potvrda instalacije Pythona

U naredbenom retku upisati: - python , zatim pritisnuti Enter.
Ukoliko je Python ispravno instaliran trebali bi vidjeti ispis sličan tekstu na sljedećoj slici.



Slika 7. Poruka potvrde instalacije Pythona

3.Korak – Instalacija PIP-a

Nakon što ste potvrdili da je Python ispravno instaliran, možemo nastaviti s instalacijom PIP-a. Preuzmite "get-pip.py" mapu na vase računalo, zatim otvorite naredbeni redak i idite do mape koja sadrži navedeni instalacijski program "get-pip.py".
Pokrenite sljedeću naredbu: - python get-pip.py, nakon toga bi se PIP trebao uspješno instalirati. Ponekad se dogodi greška s obavjesti "datoteka nije pronađena"

Nakon instalacije Pythona, uvezite NetworkX modul pomoću PIP-a u naredbenom retku na sljedeći način:

```
$ pip install networkx[default]
```

Slika 8. Naredba za instalaciju NetworkX modula

3.3. Pozitivne karakteristike NetworkX modula

Za razliku od mnogih drugih alata, NetworkX je dizajniran za rukovanje podacima koja su bitna za „moderne" probleme.

- Većina osnovnih algoritama se oslanja na iznimno brz nasljeđeni kod.
- Vrlo fleksibilne implementacije grafova (dodatno objašnjenje - graf/čvor može biti definiran kao bilo što tj. kao bilo kakav entitet)
- Opsežan skup izvornih formata za čitanje i pisanje.

- Iskorištava Pythonovu sposobnost da povlači podatke iz interneta i baza podataka.

NetworkX također ima veliku zajednicu programera koji održavaju osnovni paket i doprinose ekosustavu treće strane. Velika prednost je što iskorištava mnoge postojeće aplikacije u Pythonu i drugim jezicima i što omogućava izgradnju moćne platforme za analizu.

Korištenje Pythona smanjuje barijeru studentima i ne toliko stručnim osobama u učenju, korištenju i razvoju mrežnih algoritama. To je potaknulo doprinose u zajednici otvorenog koda (Open Source-a), a pogotovo u sveučilišnom obrazovanju. Jezgra NetworkX-a u potpunosti je napisana u Pythonu a to čini kod lakim za čitanje, pisanje i dokumentiranje.

3.4. Negativne karakteristike NetworkX modula - kada ga izbjegavati!

- Kada imamo probleme velikih razmjera koji zahtjevaju brže pristupe – bilo koja društvena mreža.
- Kada nam je potrebna bolja iskoristivost resursa (niti).
- Vizualizacija mreža je bolje upravljana drugim alatima.

4. Rad s grafovima u NetworkX-u

U ovom dijelu teksta će biti predstavljeno nekoliko ključnih koncepata o grafovima i implementacija istih.
Graf G(V,E) je struktura podataka koja je definirana skupom vrhova (V) i skupom bridova(E). Vrh (V) ili čvor je nedjeljiva točka predstavljena komponentama označenim slovima.
Brid je komponenta povezuje 2 ili više vrhova

Nakon instalacije potrebno je uvesti paket, tek nakon toga možemo provesti inicijalizaciju, a to radimo u sučelju naredbom:

```
import networkx as nx

G = nx.Graph()
```

Slika 9. Uvezivanje(importanje) networkX modula i
inicijalizacija praznog grafa

Vrste grafova: neusmjereni i usmjereni s petljama, neusmjereni i usmjereni s višestrukim bridovima

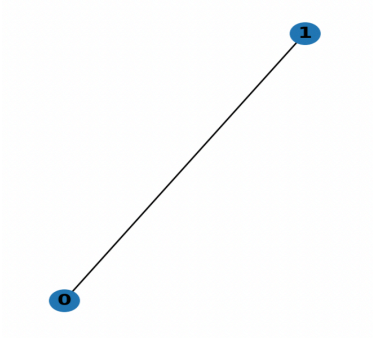
4.1. Dodavanje vrhova i bridova

Zatim možemo dodati vrhove i bridove po želji, a to radimo na sljedeći način:

```
G.add_node(1)
G.add_node(2)
G.add_edge(1, 2)
```

Slika 10. Dodavanje vrhova i bridova

Prema naredbama na *Slici 10.* dodali smo dva vrha i povezali ih jednim bridom. Rezultat naredbi je ilustriran na sljedećoj slici (*Slika 11.*).



Slika 11. Jednostavan graf s 2 vrha povezana jednim bridom

4.1.1. Dodavanje više čvorova i bridova istovremeno

Dodavanje vrhova jednog po jednog je poprilično sporo ali zato možemo napraviti liste vrhova i bridova gdje je svaki brid predstavljen n-torkama vrhova tj. uređenim parovima kao npr.

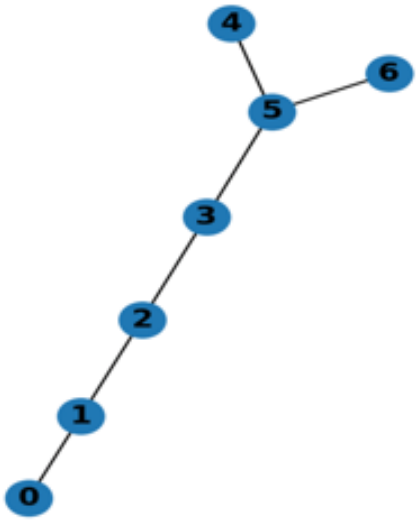
```
G.add_nodes_from([2,3,4,5,6])
G.add_edges_from([(1,2),(4,5),(3,5),(2,3),(5,6)])
```

Slika 12. Primjer stvaranja grafa preko lista i uređenih parova

Na postojeći graf koji smo u inicijalnom primjeru stvorili *Slika.11.* dodali smo pomoću lista još nekoliko inačica grafa, tj. 5 bridova i vrhova.

Na *Slici.12.* imamo primjer naredbi za dodavanje još 5 vrhova i 5 bridova koji na kraju operacije daju graf kao na *Slici.13* na sljedećoj stranici. Mogli smo povezati i bilo koje od ostalih vrhova, ovdje je samo ilustriran najjednostavniji primjer.

Čvorovi mogu biti bilo koji objekti koji se može hashirati, kao što su nizovi, brojevi, datoteke ili funkcije tj. svaki objekt koji ima **hash vrijednost** — cjelobrojni identifikator tog objekta koji se nikada ne mijenja tijekom svog životnog vijeka.



Slika 13. Ilustracija grafa nakon dodanih komponenti iz lista

NetworkX uključuje funkcije za računalne mreže, statistike i metrike kao što su promjer, distribucija stupnjeva, broj povezanih komponenti, grupiranje, koeficijent itd. Također, generatori za mnoge klasične grafove i slučajne grafove modela su dostupni. Ovi grafovi su korisni za modeliranje i analizu mrežnih podataka, te također za testiranje novih algoritama ili mrežnih metrika.

Za računalne analize mreža korištenjem tehnika iz algebarske teorije grafova, NetworkX koristi prikaze mreža matricom susjedstva s NumPyjem guste matrice i SciPy rijetke matrice.

NumPy i SciPy paketi također pružaju linearne sustave i koji imaju mogućnost rješavanja svojstvenih vrijednosti za statističke alate i mnoge druge korisne funkcije.

Za vizualizaciju i crtanje, NetworkX sadrži sučelja za Graphviz mrežu. Različiti standardni mrežni modeli uključeni su za realizaciju i stvaranje mrežnih modela i NetworkX može uvesti grafove podataka iz mnogih vanjskih formata kao što je na početku objašnjeno.

4.2. Brisanje čvora i brida

Uklanjanje elemenata iz grafa
Iz grafa se mogu ukloniti čvorovi i rubovi na sličan način kao što se dodaju.

Koristite metode:

- `Graph.remove_node()` – ukloni čvor
- `Graph.remove_nodes_from()` – uklanjanje čvora iz liste,hash tablice i sl.
- `Graph.remove_edge()` – ukloni brid
- `Graph.remove_edges_from()` – ukloni brid iz liste, hash tablice i sl.

```
>>> G.remove_node(2)
>>> G.remove_nodes_from("spam")
>>> list(G.nodes)
[1, 3, 'spam']
>>> G.remove_edge(1, 3)
```

Slika 14.Primjer uklanjanja bridova i čvorova u sučelju

4.3. Dodavanje atributa grafovima, čvorovima i rubovima

Atributi kao što su težine, oznake, boje ili bilo kakav Python objekt koji želite generirati, mogu se pridružiti u grafu, čvorovima ili rubovima.

Svaki graf, čvor i rub mogu sadržavati parove atributa tj. ključeve ili vrijednosti u pridruženom rječniku atributa, ključevi se obavezno moraju hashirati. Prema zadanim postavkama oni su prazni, ali atributi se mogu dodavati ili mijenjati pomoću :

`add_edge`

`add_node`

ili izravnom manipulacijom rječnika atributa za graf G s naredbama:

```
G.graph

G.nodes

G.edges
```

Imajte na umu da se dodavanjem čvora u `G.nodes` čvor ne dodaje na graf,već je bolje upotrijebiti `G.add_node()` za dodavanje novih čvorova. Slično za rubove.

Dodavanje i mijenjanje atributa rubova radi se pomoću naredbi:

```
add_edge()

add_edges_from()
```

4.4. Izračunavanje broja čvorova i bridova

U networkX paketu nam je dana ugrađena funkcija koja nam sama računa broj bridova i čvorova na naš zahtjev ovisno o strukturi zadanog grafa. To izvodimo pozivanjem funkcije

```
Graph.number_of_edges(u,v)
```

a ona vraća broj bridova između dva čvora. Ako su navedeni u i v, vraća broj bridova između u i v, u suprotnom vraća ukupan broj svih bridova. Broj bridova u grafu. Ako su navedeni čvorovi u i v, vraća broj bridova između tih čvorova. Ako je graf usmjeren, ovo vraća samo broj bridova od u do v.

4.5. Ekscentričnost čvora

Ekscentričnost čvora v je maksimum najkraćih udaljenosti od svih čvorova grafa G do čvora v.

Matematički se definira kao ecc(v) = max{W[i, v] za svaki čvor i}. Središte grafa je čvor v grafa koji ima najmanju ekscentričnost.

Ideja: Primenom algoritma najkraćeg puta određuje se matrica W najkraćih udaljenosti izmedju svaka dva čvora. Zatim se mogu naći ekscentričnosti svih čvorova u grafu kao maksimumi po kolonama matrice W, a zatim i minimum ovih maksimuma. Čvor kojem odgovara ta minimalna vrijednost se postavlja za središte(centar) grafa G.

4.6. Erdős–Rényijev model

ER model jedan je od najpopularnijih i najjednostavnijih metoda generiranja grafova. Glavna ideja ovog modela je postaviti jedinstveni prag vjerojatnosti za rub između dva čvora. Sve u svemu, postoje dvije varijante ovog modela, prvi i formalni definiramo na sljedeći nači: [10]

$$P(\mathbf{A}[u, v] = 1) = r, \forall u, v \in \mathcal{V}, u \neq v,$$

Slika 15. Matematička definicija ER modela

Kao što je navedeno na Slici 15. **A** je matrica susjedstva, **u** i **v** su čvorovi u grafu V.

Može se interpretirati kao vjerojatnost **r** ruba između dva proizvoljna i različita čvora.

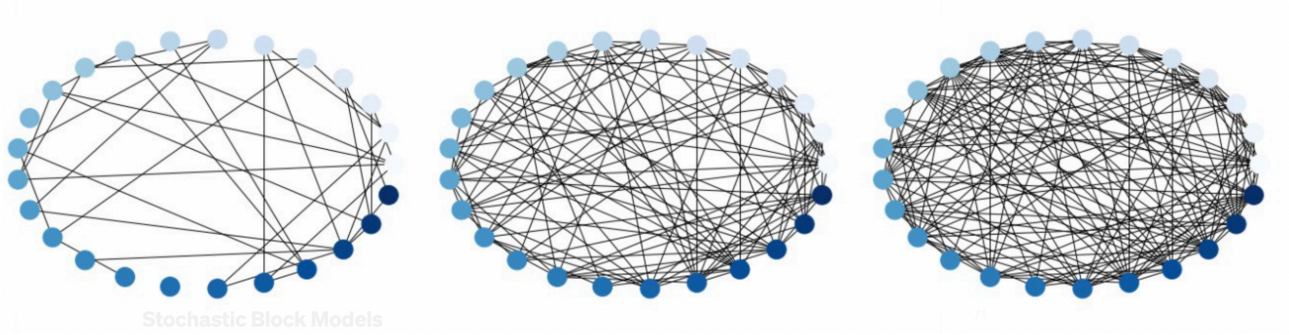
Gornja jednadžba znači: postavite vjerojatnost **r** za rub koji se pojavljuje između dva proizvoljna i različita čvora. Što je **r** veći, to će graf biti gušće povezan.

Prilično je jednostavno generirati ER graf s funkcijom *nx.erdos_renyi_graph* iz paketa NetworkX, a to možemo napraviti na način na koji je prikazano u primjeru.[10]

Stvaranje G{n,m} slučajnog grafa s *n* čvorova i *m* rubova.

Ovaj se graf ponekad naziva Erdős-Rényi graf, ali se razlikuje od G{n,p} ili binomnog_grafa koji se također ponekad naziva Erdős-Rényi graf.

U nastavku su iscrtana 3 nasumično generirana grafikona od lijeve prema desnoj strani s vjerojatnostima od 0,1, 0,3 i 0,5, dok je na sljedećoj stranici prikazan kod koji omogućuje plotiranje i crtanje istih.



Slika 16. Ilustracija grafova različitih vjerovatnosti koristeći ER model

Dok je ER generirani graf jednostavan i brz za izračunavanje, ovom metodom ne možemo modelirati složenije strukture grafa. S ER-om nije moguće modelirati motive, distribuciju stupnjeva ili zajednice pomoću grafikona.

5. Primjeri korištenja NetworkX modula

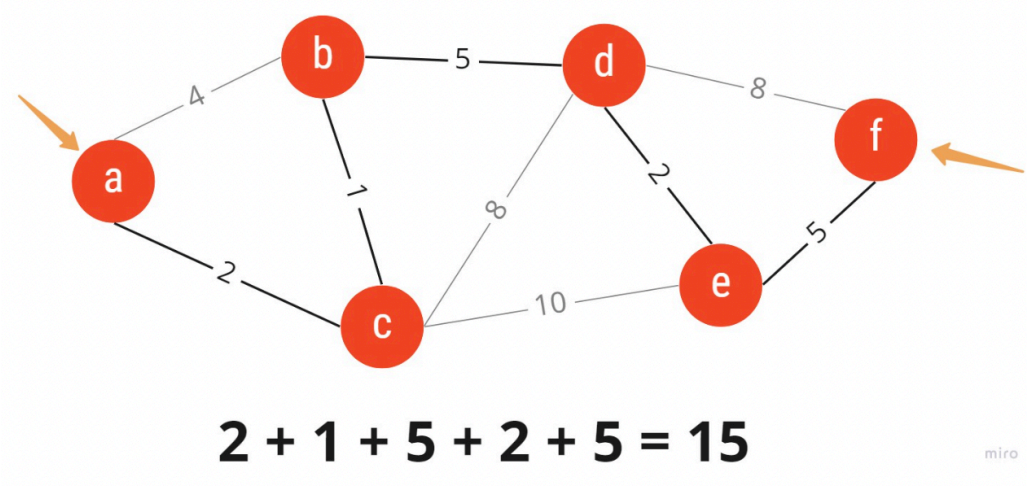
5.1. Algoritmi najkraćeg puta

Za najosnovniji primer, kada planiramo putovanje, nastojimo minimizirati naše troškove u mnogim različitim područjima - gorivo, vrijeme, noćenja, troškovi prometa itd. Izračun ovih troškova može oduzeti puno truda i vremena, ali što ako postoji elegantniji način koji bi vam mogao riješiti problem?

Algoritmi pronalaženja puta jedan su od klasičnih problema s grafovima i istražuju se od 19. stoljeća. Algoritam najkraćeg puta je algoritam koji izračunava put između dva čvora u grafu kao što je zbroj vrijednosti na rubovima koji čine put minimiziran.

5.1.1. Kako radi?

Polazeći od izvornog čvora, algoritam traži težine na (odlaznim) rubovima (u grafovima). Bira rub koji, zbrojen s prethodnim ukupnim zbrojem, daje najniži rezultat. Algoritam prolazi kroz svaki čvor do odredišne točke. Rezultati su put i ukupni zbroj najkraćeg puta[5].



Slika 17. Ilustrirani primjer funkcionalnosti algoritma najkraćeg puta

5.1.2. Praktične aplikacije algoritma najkraćeg puta

Najčešće korišten je Dijkstrin algoritam, kao na primjer u uslugama digitalnog mapiranja u Google kartama. Svaki put kada tražimo upute, dobivamo vrijeme za putovanje optimalnom rutom. Obje te informacije su rezultati Dijkstrinog algoritma.
Dok koristimo društvene mreže, mogli bi vidjeti prijedloge poput "Ljudi koje možda poznajete" ili "Ljudi koje vaši prijatelji slijede" i sl. Ako je društveni graf jako malen, možete upotrijebiti algoritme za pronalaženje najkraćeg puta između korisnika mjereći veze među njima.
U telekomunikacijama svaka linija ima propusnost, mjeru koja nam govori koliko podataka može proći kroz liniju. Možemo koristiti algoritme za određivanje najkraćih putova između točaka slanja i primanja u mreži prilikom prijenosa podataka, na primjer slanje e-pošte svim vašim prijateljima. [6]
Neki od najpopularnijih algoritama najkraćeg puta:

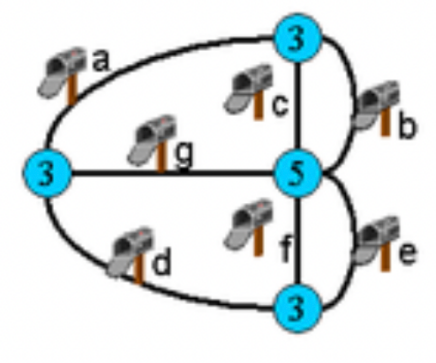
- **Dijkstrin algoritam** - Pronalazi najkraći put od izvornog čvora do ciljnog čvora ako su težine na rubovima nenegativne vrijednosti. Pozadina ideje ovog algoritma leži u tome da stablo koje se sastoji od bridova primjeni na način da se uzmu minimalni putovi od odabranog(početnog) vrha do svih ostalih.Od izbora svih bridova koje možemo uzeti u obzir za početni vrh, npr. kao Primov algoritam. Kod Primovog algoritma u prvom koraku biramo rub koji je spojnica vrha koji nije dio stabla , ali pri tome računamo udaljenost od početnog vrha za svaki taj vrh. Od svih bridova koje se u prvom koraku može izabrati, uzima se najčešće onaj za kjoj je pripadni vrh, koji nije u stablu, najkraće udaljen od vrha koji smo odabrali za početni.
- **Bellman-Ford** algoritam - Pronalazi najkraći put od izvornog čvora do ciljnog čvora kada rubne težine mogu biti negativne. Bellman Fordov algoritam radi na principu da procjenjuje duljinu puta od početnog vrha do svih ostalih vrhova. Zatim iterativno ublažava i umanjuje te procjene pronalaženjem novih putanja koje su kraće od prethodno precijenjenih staza. Ponavljajući radnju više puta za sve vrhove, možemo jamčiti da je rezultat optimiziran.

- **A* (A - star) algoritam pretraživanja** - Pronalazi najkraći put između para čvorova pomoću heurističkih metoda. A* Algoritam pretraživanja jedna je od najboljih i najpopularnijih tehnika koje se koriste u pronalaženju putanja i obilaženju grafova. Neslužbeno govoreći, algoritmi A* pretraživanja, za razliku od drugih tehnika prelaska, imaju "mozak". To znači da je to stvarno pametan algoritam i to ga razlikuje od ostalih konvencionalnih algoritama. Također je vrijedno spomenuti da mnoge igre i karte temeljene na webu koriste ovaj algoritam za vrlo učinkovito pronalaženje najkraćeg puta. Zato ga se još ponekad zove optimalnim algoritmom.

- **Floyd-Warshall algoritam** - Pronalazi sve najkraće putove između svakog para čvorova u grafu. Floydov algoritam koristimo kako bi našli najkraći put tj. udaljenost između svih kombinacija vrhova u grafu. Temeljna ideja ovog algoritma je testiranje svih mogućnosti među vrhovima tj. putova u grafu, ali se koristi činjenica da svako takvo ispitivanje za ovakav problem ima optimalnu podstrukturu te da se do ukupnog najkraćeg puta može doći spajanjem minimuma problema. Floyd-Warshall-ov algoritm je bazičan primjer dinamičkog programiranja, a dokaz da je rezultat izvođenja algoritma ispravan je veoma kompleksan.
- **Johnsonov algoritam** - Pronalazi sve najkraće putove između svakog para čvorova u usmjerenom grafu kombinirajući Dijkstrine i Bellman-Fordove algoritme.

5.2. Problem kineskog poštara

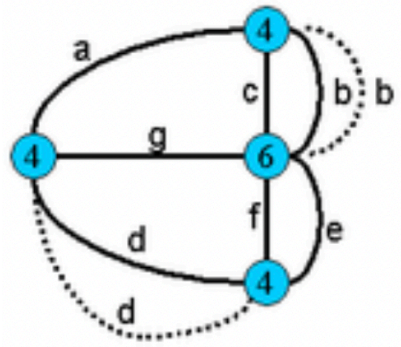
Kod problema kineskog poštara uzimamo pretpostavku da je svaki brid u grafu nekakva ulica s raznim adresama gdje se treba efektivno dostaviti pošta, a vrhovi bi bile točke na grafu (bridovi), gdje poštar ima mogućnost promijene smjera i modulacije rute, također postoji više različitih ruta koje mogu dati jednak rezultat, to su faktori koji ovise o parametrima i smjeru.



Slika 18. Ilustracija modela problema kineskog poštara

Poštar u ovom problemu mora posjetiti svaku ulicu da bi dostavio poštu i želi to obaviti u najmanjem broju ponovljenih prolazaka istom ulicom,u slučaju kineskog poštara duple ture se ne mogu izbjeći jer graf nije Eulerov graf.

Ponovljeni prolazak ulicom na grafu ilustriranom na slikama prikazujemo kao ponovljeni brid kao u slučaju na Slici 19. to su bridovi **b** i **d.** Nastali višestruki bridovi nam omogućavaju da svaki vrh novog grafa bude stupnja parne potencije, odnosno dodavanjem bridova je izvršena "eulerizacija" grafa. U tako prilagođenom grafu se može bez problema odrediti Eulerov put ili Eulerova tura.



Slika 19. Ilustracija grafičkog pristupa
rješenju problema

Primjer problema, traženi poštarov obilazak za Eulerov graf sa Slike 19. može biti naprimjer a -> b -> c -> **b** -> e -> f -> g -> d -> **d**. U njemu se ponavljaju samo 2 brida a to su u ovom slučaju podebljani bridovi ***b*** i ***d***. Za pronalazak najkraće moguće poštareve

rute kojim će obići sve ulice na zadanom putu treba u jednadžbu ubaciti parametre kao što su vrijeme koje mu treba za proći rutom i duljine ulica koje posjećuje [10].

5.3. Problem trgovačkog putnika

Problem trgovačkog putnika (eng. Traveling Salesman Problem – TSP) je definiran kao problem u kojem tražimo najbolju tj. optimalnu rutu kroz skup raznih lokacija,a to radimo na način da se svaku lokaciju posjetimo isključivo samo jedanom, a zatim se vraćamo na polazište ili lokaciju odakle smo krenuli.
Problem je definiran teoremom iz teorije grafova, a glasi:
Neka je V određeni skup n + 1 vrhova (u ovom slučaju lokacija) gdje V brojimo od polazišta do n na način koji sljedi V={0,1,2,..,n}. Glavni problem je pronalazak rute (drugim riječima - ciklusa) koji započinje i završava u početnom vrhu 0, a na putu prolazi kroz sve vrhove od 1 do n točno i samo jednom, uz to da mu je ukupna udaljenost minimalan broj od svih mogućnosti. Ruta koja posjećuje sve vrhove grafa i završava u polazišnom vrhu još se naziva Eulerova tura.
Dakle, problem trgovačkog putnika je pronalazak Eulerove ture najmanje moguće duljine. Kao što je definirano u teoremu, Eulerova tura je put u neusmjerenom grafu koji prolazi kroz sve vrhove grafa isključivo jedanput. Eulerova tura je također definirana kao ciklus u grafu koji je neusmjeren, a isto tako posjećuje sve vrhove grafa isključivo jedanput, osim vrha koji je definiran kao početak i kraj ciklusa.
Graf u suštini nerjetko sadrži više od jedne Eulerove ture, i onda kad nastojimo naći najkraći put to nazivamo **problemom trgovačkog putnika**. Za kraj, da za graf postoji rješenje problema trgovačkog putnika, nužan uvjet je da ima najmanje jednu Eulerovu turu, odnosno mora zadovoljavati parametre Eulerovog grafa.

U potpunim grafovima s *n* vrhova broj Eulerovih tura racunamo preko firmule (n-1)! dok se u neusmjerenim grafovima računa (n-1)/2, zbog dvosmjernog puta grafa koji uzimamo u obzir prepolovimo broj putova.

Za pronalazak rješenja ovog problema potrebna nam je odgovarajuća metoda koju drugim riječima nazivamo algoritam, algoritmi pronalaska rješenja dijelimo na:

- Egzaktne metode
- Heuristički algoritmi

5.3.1. Egzaktne metode rješenja problema trgovačkog putnika

Egzaktne metode nam nude najoptimalnije rješenje problema, ali su u praksi dosta ograničene na probleme u kojima uzimamo u obzir one stavke manjih broja vrhova i bridova te olakšano računanje istih, razlog tome je kompleksnosnost problema koji nam je predočen. Problem trgovačkog putnika u literaturi kategoriziramo pod teške probleme, kod kojih je vrijeme računanja bitan faktor, a potreba za rješenjem raste gradijalno s veličinom informacija datog problema. Zato je često dobro pokušati prvo riješiti problem nekom približno sličnom metodom koja će biti dovoljno brža i točna za svrhu rješavanja iz čega dalje možemo modelirat problem na kompleksnijim primjerima.
Danas se raadi na razvoju raznih konstrukcijskih heurističkih algoritm za generiranje nekog rješenja ali isto tako se radi na algoritmima lokalnog pretraživanja koji nam služe za poboljšavanje određenih generiranih rješenja. Sve te metode u načelu mogu pronaći više rješenja i za velike probleme. Neki od tih problema su povezanost među gradovima, letovi, pomorski promet i sl.

5.3.2. Heuristički algoritmi problema trgovačkog putnika

Heuristički agoritmi koje koristimo za rješavanje TSP-problema tj. problema trgovačkog putnika rade na principu da pokušavaju brzo i na ako je to moguće jednostavan način pronaći rješenje koje zadovoljava odrteđene uvjete postavljene u algoritmu, a to je da je rješenje dovoljno dobro po zadanim parametrima. Analiza heuristike se temelji na usporedbama s nekim već poznatim riješenjima koji su unaprijed procjenjeni i imaju gotove modele i obrasce. Načelno, izrada rute se bazira na procedurama dodavanja vrhova pri svakom koraku. U nastavku su navedenio i objašnjeni neki od najpoznatijih heurističkih algoritama. Jako su pogodni za analizu i rješavanje i problema velikih broja informacija.

5.3.3. Algoritam najbližeg susjeda

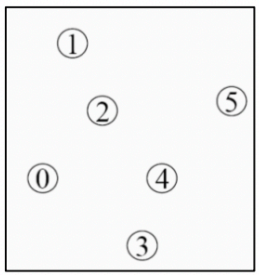
Algoritam najbližeg susjeda (eng. nearest neighbour algorithm) je heuristički algoritam za generiranje rješenja kojemu je temelj postavke problema pohlepni algoritam (eng. *Greedy*

algorithm) a radi tako da dodaje najbliži vrh već dodanim vrhovima u zadanoj ruti ruti [12].

Algoritam se po koracima opisuje na sljedeći način:

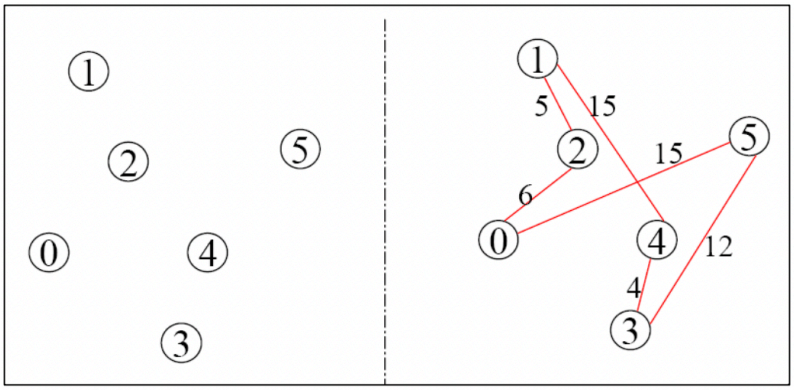
1. Korak – Odabrati početni vrh
2. Korak – iz slobodnih vrhova odrediti onaj koji je najbliži posljednjem koji je dodan i tako ga spojiti s rutom
3. Korak – Ponavljati korak 2 dok se svi vrhovi u ruti ne povežu, zatim za kraj povezati prvi i zadnji vrh

	0	1	2	3	4	5
0	-	10	6	8	7	15
1		-	5	20	15	16
2			-	14	7	8
3				-	4	12
4					-	6



Slika 20. Primjer algoritma najbližeg susjeda

Na Slici 20. je pokazan ilustrirani primjer heurističkog algoritma najbližeg susjeda. Za početnu točku uzimamo vrh 0 (možemo vrh koji želimo kao početni) zatim u tablici on traži najbliži vrh početnom vrhu 0. U ovom primjeru najbliži vrh vrhu 0 je vrh 2 čija je udaljenost 7. Zatim tražimo vrh najbliži vrhu 2, a u ovom slučaju je to vrh 1. Postupak ponavljamo sve dok ne povežemo sve vrhove. Na Slici 21. je pokazan način na koji doći do rješenja navedenog primjera.



Slika 21. Rješenje algoritma najbližeg susjeda

Kao što je prikazano na Slici 21. dobivena je ruta 0,2,1,4,3,5,0 koja ima ukupnu duljinu 6+5+15+4+12+15=57.

Kompleksnost ovog algoritma je definirana relacijom O (n^2), što znači ukupan broj varijacija nije viši od n^2 . Prilagođena primjena ovog algoritma je ako uzmemo sve n vrhove kao početne točke,a time kompleksnost raste za čitavu potenciju na O(n^3), ali zato daje bolje i konkretnije rješenje.

Zaključak

Python nudi mnoge alate za olakšavanje istraživanja znanstvenih problema. Jedna od njegovih najvećih snaga i prednosti je sposobnost povezivanja postojećeg koda i biblioteke na prirodan način što olakšava integraciju mnogih alata. U radu je prikazano kako se NetworkX, u kombinaciji s Pythonovim paketima SciPy, NumPy, Matplotlib i sl. veže na druge alate napisane u npr. FORTRAN-u i C-u, a pogotovo u modernim jezicima i također pružaju moćan alat za analizu računalnih mreža. Iz svega navedenog i opisanog dolazimo do spoznaje da je NetworkX biblioteka itekako korisna i pristupačna za korištenje te isto tako možda i najbolji „basic“ paket za učenje mreža, strukture grafova i sl. te bilo kakve problematike vezane za tu temu zbog izuzetno dobre i opširne dokumentacije. Ne samo da je prihvaćen od strane laika, već i od strane profesionalaca koji razumiju da je to za mnoge početni korak u smjeru te grane znanosti koja je temelj neuronskih mreža, umjetne inteligencije itd. na čemu se danas uz pomoć moderne i napredne tehnologije izvode revolucionarni scenariji i analize

Literatura

[1] NetworkX Homepage, [Mrežno] Available: https://networkx.org/documentation/

[2] Towards Data Science, [Mrežno] Available: https://towardsdatascience.com/intro-to-graphs-in-python-using-networkx-cfc84d1df31f

[3] pypi.org, [Mrežno] Available: https://pypi.org/project/networkx/

[4] www.cl.cam.ac.uk, [Mrežno] Available: https://www.cl.cam.ac.uk/~cm542/teaching/2010/stna-pdfs/stna-lecture8.pdf

[5] NetworkX: Network Analysis with Python, [Mrežno] Available: https://networkx.guide/algorithms/shortest-path

[6] Shortest path problem, [Mrežno] Available: https://en.wikipedia.org/wiki/Shortest_path_problem

[7] Understanding Community Detection Algorithms with Python NetworkX, [Mrežno] Available: https://memgraph.com/blog/community-detection-algorithms-with-python-networkx

[8] Benchmark of popular graph/network packages, [Mrežno] Available: https://www.timlrx.com/blog/benchmark-of-popular-graph-network-packages

[9] What is R?, [Mrežno] Available: https://www.r-project.org/about.html

[10] Generative Graph Models with NetworkX, [Mrežno] Available: https://towardsdatascience.com/generative-graph-models-with-networkx-727b154ceda4

[11] Problem kineskog poštara, [Mrežno] Available: http://e.math.hr/category/klju-ne-rije-i/problem-kineskog-po-tara

[12] ANA JELIĆ, Analiza primjene algoritama problema trgovačkog putnika kod komisioniranja unutar visokoregalnog skladišta s vrlo uskim prolazima

[13] Grafovi - X.FER, [Mrežno] Available: https://www.fer.unizg.hr/_download/repository/Osnovni_pojmovi-teorija_grafova.pdf

[14] ANKA GOLEMAC, ANA MIMICA I TANJA VUČIČIĆ, Od koenigsberških mostova do kineskog poštara