Kosaraju-ov algoritam za utvrđivanje jakih komponenti povezanosti

Isidora Burmaz 1057/2023

Januar 2024.

Sadržaj

1 Opis problema

1.1 Jake komponente povezanosti kod usmerenog grafa

Za dat usmereni graf $G = (V, E)$ potrebno je odrediti njegove jake komponente povezanosti.

Definicija 1.1. *Definišemo relaciju $\sim$ na skupu čvorova V usmerenog grafa $G = (V, E)$ na sledeći način: Za čvorove u i v grafa G važi $u \sim v$ ako i samo ako u grafu G postoji put od čvora u do čvora v i od čvora v do čvora u.*

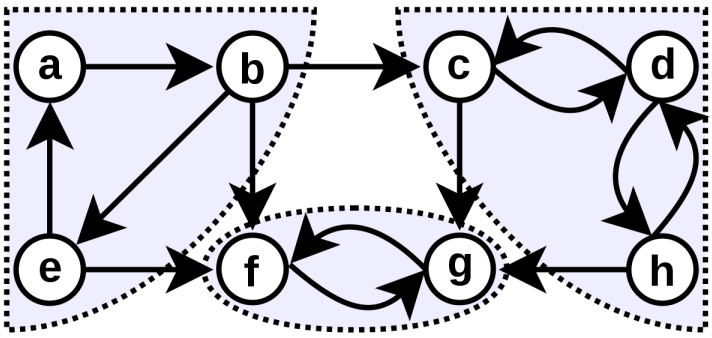
Teorema 1.1. *$\sim$ je relacija ekvivalencije.*

Dokaz. Refleksivnost relacije $\sim$ je očigledna, a simetričnost sledi direktno iz definicije. Dokažimo sada da važi i tranzitivnost. Neka su u, v i w čvorovi grafa $G = (V, E)$ takvi da važi $u \sim v$ i $v \sim w$. Dakle, u grafu G postoji put od čvora u do čvora v, put od čvora v do čvora u, kao i put od čvora v do čvora w i put od čvora w do čvora v. Put u grafu od u do w možemo dobiti nadovezivanjem puteva od u do v i od v do w. Slično, put od w do u možemo dobiti nadovezivanjem puteva od w do v i od v do u. Ovim je dokazana tranzitivnost relacije, i samim tim $\sim$ je relacija ekvivalencije. □

Definicija 1.2. *Jake komponente povezanosti grafa $G = (V, E)$ su klase ekvivalencije relacije $\sim$.*

Dakle, jake komponente povezanosti usmerenog grafa će biti međusobno disjunktni skupovi čvorova pri čemu za svaki takav skup važi da od proizvoljnog čvora tog skupa postoji put do svih ostalih čvorova iz skupa.

Na slici 1 je prikazan primer podele grafa na jake komponente povezanosti. U ovom grafu postoje tri jake komponente povezanosti. U jednoj komponenti su čvorovi a, b, e, u drugoj f i g, a u trećoj c, d i h. U svakoj od ovih jakih komponenati povezanosti postoji put od proizvoljnog čvora iz komponente do svih ostalih čvorova u komponenti. Sa druge strane, čvorovi a i f nisu u istoj komponenti povezanosti jer iako postoji put od a do f u grafu, ne postoji put od f do a pa ovi čvorovi ne mogu pripadati istoj jakoj komponenti povezanosti.



Slika 1: Podela grafa na jake komponente povezanosti

Radi jednostavnosti, pretpostavićemo da su vrednosti čvorova u grafu numerisani brojevima $\{0, \ldots V - 1\}$, gde je V broj čvorova u grafu.

1.1.1 Primene problema

Navedimo neke primene problema jakih komponenti povezanosti:

- **Web analiza** Jake komponente povezanosti pomažu u identifikovanju zajednica ili grupa povezanih web stranica ili sajtova.

- **Analiza društvenih mreža** Jake komponente povezanosti otkrivaju tesno povezane grupe pojedinaca ili zajednica unutar društvenih mreža.
- **Optimizacija kompajlera** Jake komponente povezanosti pomažu u optimizaciji generisanja koda identifikacijom nezavisnih blokova koda.

1.2 Komponente povezanosti kod neusmerenog grafa

Komponente povezanosti u neusmerenom grafu se lako mogu naći korišćenjem `DFS` pretrage. Za svaki neposećeni čvor ćemo pokrenuti `DFS` pretragu iz tog čvora, i čvorovi koji se obiđu ovom pretragom predstavljaju jednu komponentu povezanosti.

Kod usmerenog grafa i jake povezanosti, situacija je komplikovanija i ne može se rešiti običnom `DFS` pretragom. Naime, ako u usmerenom grafu pokrenemo `DFS` pretragu iz nekog čvora v dobićemo sve čvorove do kojih postoji put iz v. Međutim, za razliku od neusmerenog grafa, nema garancije da postoji i put od ovih čvorova nazad do čvora v.

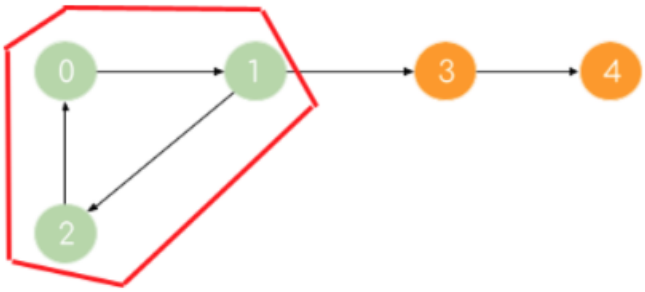
2 Kosaraju-ov algoritam

2.1 Algoritam

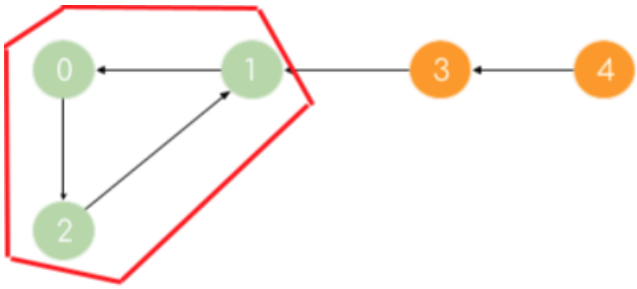
Kosaraju-ov algoritam se bazira na sledećem tvrđenju

Teorema 2.1. *Usmeren graf $G = (V, E)$ i njegov transponovani graf $G' = (V, E')$ imaju jednake jake komponente povezanosti (transponovani graf se dobija tako što svim granama u početnom grafu invertujemo smer).*

Na slikama 2 i 3 se mogu videti jake komponente povezanosti grafa $G = (V, E)$ i njegovog transponovanog grafa $G' = (V, E')$. *Teorema 2.1.* tvrdi da su jake komponente povezanosti ova dva grafa jednake.



Slika 2: Graf G



Slika 3: Transponovani graf G'

Ako pokrenemo `DFS` pretragu iz čvora v u grafu G, obići ćemo sve čvorove do kojih postoji put iz v u G. Ovaj skup čvorova ćemo obeležiti $F(v)$. Dodatno, ako pokrenemo `DFS` pretragu iz čvora v u transponovanom grafu G', obići ćemo sve čvorove do kojih postoji put iz v u G', odnosno obići ćemo one čvorove iz kojih se može doći do v u grafu G. Ovaj skup čvorova ćemo obeležiti $B(v)$. Dakle, skup čvorova koji su u istoj jakoj komponenti povezanosti kao i čvor v predstavlja presek ovako određenih skupova čvorova $B(v)$ i $F(v)$.

Kosaraju-ov algoritam funkcioniše u dve faze u kojima se izvršavaju `DFS` pretrage. Prva faza podrazumeva `DFS` pretragu na transponovanom grafu i u njoj se pamti redosled u kom čvorovi završavaju svoje rekurzivne pozive. U drugoj fazi se izvršava `DFS` pretraga na početnom grafu, pri čemu se sledeći čvor za obilazak grafa bira na osnovu redosleda ustanovljenog u prvoj fazi algoritma. Ovakav redosled obilaska obezbeđuje da `DFS` pokrenut iz nekog čvora može da obiđe samo čvorove u istoj jakoj komponenti povezanosti. Rezultat drugog prolaza je skup jakih komponenti povezanosti grafa.

Kosaraju-ov algoritam se sastoji iz narednih koraka:

1. *Naći transponovan graf G' koji odgovara početnom grafu G.*

2. *Izvršiti `DFS` pretragu na grafu G' i dodavati indekse čvorova na stek u redosledu završetka njihovih rekurzivnih poziva.*

3. *Skidati čvorove sa vrha steka jedan po jedan i pokretati `DFS` na početnom grafu G počevši od svakog čvora koji još nije posećen. Svaki `DFS` prolazak će identifikovati jednu jaku komponentu povezanosti grafa.*

4. *Vratiti skup jakih komponenti povezanosti.*

2.2 Dokaz korektnosti

Algoritam je sam po sebi jednostavan, međutim potrebno je dokazati njegovu korektnost.

Teorema 2.2. *Posmatrajmo dve susedne jake komponente povezanosti C_1 i C_2 grafa G, tj. komponente takve da postoji grana (i, j) u G sa $i \in C_1$ i $j \in C_2$. Neka $f(v)$ označava vreme završetka obilaska čvora v u proizvoljnom izvršavanju `DFS` obilaska na transponovanom grafu G'.*

Tada važi:

$$\max_{v \in C_1} f(v) < \max_{v \in C_2} f(v)$$

Dokaz. Graf G i njegov transponovan graf G' imaju iste jake komponente povezanosti, stoga su C_1 i C_2 takođe jake komponente povezanosti grafa G' i postoji grana (j, i) u G' sa $i \in C_1$ i $j \in C_2$. Neka je v prvi čvor iz $C_1 \cup C_2$ koji će biti posećen tokom `DFS` pretrage grafa G'. Razlikujemo dva slučaja.

- Pretpostavimo da je $v \in C_1$. Ne postoji usmerena putanja od v do proizvoljnog čvora iz C_2 u G' (inače bi moglo da se od svakog čvora iz C_1 dođe do svakog čvora iz C_2 i obrnuto, pa bi C_1 i C_2 bile ista komponenta povezanosti). `DFS` iz čvora v će posetiti sve čvorove u C_1 bez dosezanja do bilo kojeg čvora u C_2. Dakle, vreme završetka `DFS` obilaska proizvoljnog čvora iz C_1 biće manje od vremena završetka obilaska proizvoljnog čvora iz C_2, što je čak jače od tvrđenja teoreme.

- Pretpostavimo da je $v \in C_2$. Poziv algoritma `DFS` na v će završiti rekurzivno obilazak svih čvorova u $C_1 \cup C_2$ pre nego što se završi. Dakle, vreme završetka `DFS` obilaska čvora v je najveće među čvorovima u $C_1 \cup C_2$, i posebno veće od vremena završetka obilaska svih čvorova iz C_1.

Time se kompletira dokaz. □

Teorema 2.2 tvrdi da prolazak sa jedne jake komponente povezanosti C_1 na drugu C_2 u originalnom grafu (pri čemu postoji grana (i, j) u G sa $i \in C_1$ i $j \in C_2$) strogo povećava maksimalnu f-vrednost trenutne jake komponente povezanosti. Intuitivno, kada se `DFS` pozove na G, obrađujući čvorove u opadajućem redosledu vremena završetka, sukcesivni pozivi na `DFS` odstranjuju jake komponente povezanosti grafa jedan po jedan.

Teorema 2.3. *Kosaraju-ov algoritam je korektan.*

Dokaz. Sada matematičkom indukcijom dokazujemo ispravnost Kosaraju-ovog algoritma za računanje jakih komponenti povezanosti u grafu G.

Induktivna hipoteza: Skup S koji se sastoji iz čvorova koji su već posećeni prethodnim pozivima `DFS` algoritma je unija nula ili više jakih komponenti povezanosti G.

Baza indukcije: U trenutku prvog poziva `DFS` algoritma, skup S je prazan pa važi da je unija nula ili više jakih komponenti povezanosti.

Induktivni korak: Pretpostavimo da je `DFS` pretraga pokrenuta iz čvora v i neka C označava jaku komponentu povezanosti čvora v u G. S obzirom da su jake komponente povezanosti grafa disjunktni

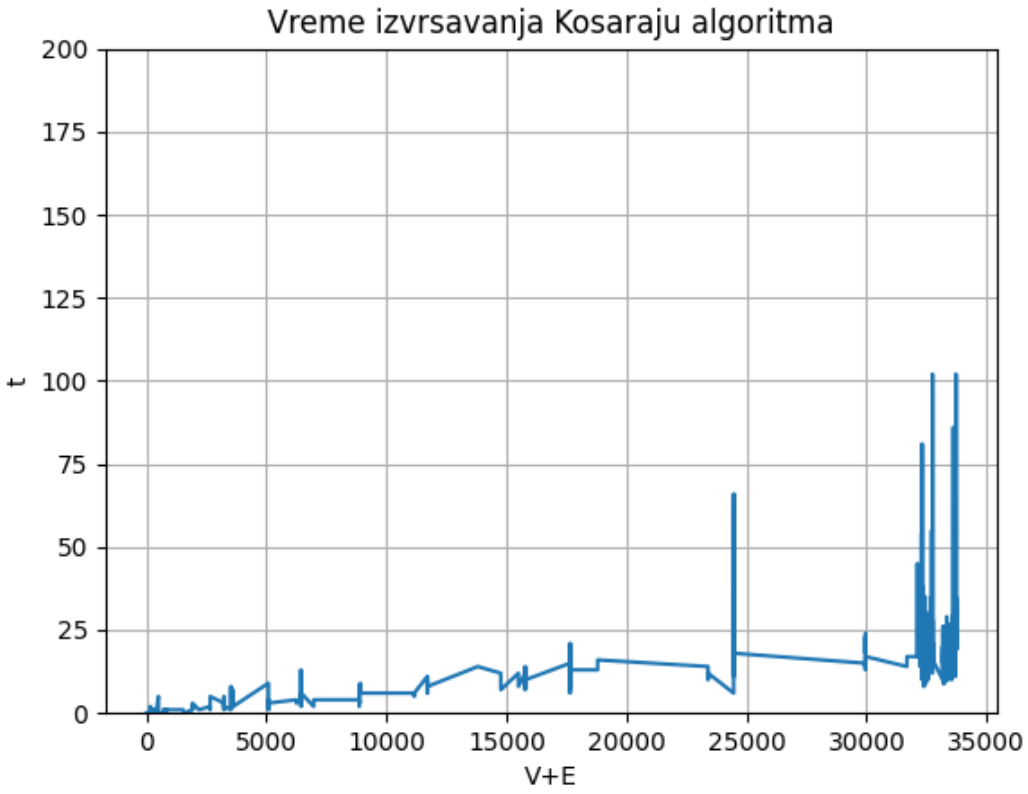
skupovi, da je S unija nekih jakih komponenti povezanosti grafa G i $v \notin S$, može se zaključiti da nijedan čvor iz C ne pripada S. Dakle, ovaj poziv `DFS` će posetiti, barem, sve čvorove komponente C. Prema *Teoremi 2.2*, svaka grana (i, j) takva da $i \in C$ i $j \notin C$ vodi do neke jake komponente povezanosti C' koja sadrži čvor w sa vremenom završetka većim od $f(v)$. Pošto su čvorovi obrađeni u opadajućem redosledu vremena završetka, w je već istražen i pripada S. Kako je po induktivnoj hipotezi S unija jakih komponenti povezanosti, mora sadržati sve čvorove komponente C'. Sumirano, svaka grana (i, j) takva da $i \in C$ i $j \notin C$ vodi direktno do čvora koji je već posećen. Dakle, ovaj poziv `DFS` posećuje čvorove komponente C i ništa više. Ovo kompletira induktivni korak.

Dakle, svaki put kada se `DFS` algoritam pozove iz čvora v, čvorovi obiđeni ovim pozivom su tačno oni u jakoj komponenti povezanosti v u G.

□

2.3 Opis implementacije i evaluacija algoritma

Algoritam je napisan u programskom jeziku C++. Implementacija ima oko dvesta linija koda i dostupna je na sledećoj adresi Kosaraju's algoritm.



Slika 4: Vreme izvršavanja Kosaraju-ovog algoritma

Glavne komponente ovog algoritma su dve `DFS` pretrage i računanje transponovanog grafa. Graf je reprezentovan kao kolekcija nizova, tako da se lako može izračunati transponovan graf u vremenu $O(E)$. Kako se `DFS` pretraga može izvršiti u vremenu $O(V + E)$ (gde je V broj čvorova u grafu, a E broj grana u grafu), ukupno vreme izvršavanja algoritma je $O(V + E)$.

Za evaluaciju vremena izvršavanja algoritma korišćena je funkcija *clock()* iz standardne C++ biblioteke. Ova funkcija meri procesorsko vreme potrošeno na izvršavanje programa do momenta poziva funkcije. Vraćena vrednost je izražena u taktovima procesorskog sata. Ova funkcija se poziva neposredno pre i posle funkcije koja izračunava jake komponente povezanosti Kosaraju-ovim algoritmom, i razlika dobijenih

vremena predstavlja procesorsko vreme utrošeno na izvršavanje algoritma. Dodatno, kako bi se izračunalo vreme utrošeno na izvršavanje Kosaraju-ovog algoritma na ulazima različitih dimenzija, napisana je funkcija za generisanje nasumičnog grafa sa datim brojem čvorova i grana. Ova funkcija generiše jedan nasumični graf datih dimenzija koji će biti korišćen za evaluaciju vremena izvršavanja algoritma. Generisan graf može biti i gust i redak jer se za broj čvorova V prosleđuje broj grana E koji može biti bilo koji broj između 0 i maksimalnog broja grana u grafu $\frac{V(V+1)}{2}$. Princip generisanja grana je drugačiji u zavisnosti od toga da li je graf redak ili gust. U slučaju da je redak, generisaćemo grane koje će biti uključene u graf dok ne izgenerišemo traženi broj različitih grana. U slučaju gustog grafa, ovakav pristup bi bio sporiji jer bismo posle nekog vremena često generisali grane koje smo već dodali grafu, i time ne bismo napredovali u broju generisanih grana. Efikasnije je u ovom slučaju da se generišu grane koje ne pripadaju grafu, jer je njih dosta manje.

Za grafički prikaz vremena izvršavanja algoritma korišćeni su podaci o vremenu izvršavanja izračunati u C++ programu i *python* sa bibliotekom *matplotlib*. Na x osi grafika je vrednost $V + E$ grafa, a na y osi vreme izvršavanja Kosaraju-ovog algoritma za graf date veličine izraženo u otkucajima procesorskog sata.

Sa grafika na slici 4 se može videti da je trend vremena izvršavanja algoritma linerna funkcija (sa šumovima u pojedinim tačkama).

2.3.1 Poboljšavanje vremena izvršavanja

Postoji bolji pristup za rešavanje problema jakih komponenti povezanosti od onog koji koristi Kosaraju-ov algoritam. Naime, Tarjan-ov algoritam za detekciju jakih komponenti povezanosti je efikasniji. Iako je asimptotsko vreme jednako, Tarjan-ov algoritam je brži za konstantan faktor jer izvršava jedan `DFS` prolazak umesto dva.

Literatura

[1] https://stanford-cs161.github.io/winter2022/assets/files/lecture10-notes.pdf

[2] https://pratikbarjatya.medium.com/exploring-strongly-connected-components-and-the-kosaraju-algorithm-f36d70ec7c5d

[3] https://web.stanford.edu/class/archive/cs/cs161/cs161.1138/lectures/04/Small04.pdf

[4] https://www.topcoder.com/thrive/articles/kosarajus-algorithm-for-strongly-connected-components

[5] https://www.scaler.com/topics/kosaraju-algorithm/

.