PREŠTEVANJA VPETIH DREVES

Nikolaj Zalokar

6. januar 2007

Obstaja $2^{\binom{n}{2}}$ enostavnih grafov z množico vozlišč $[n] = \{1, \ldots, n\}$, kjer lahko vsak par vozlišč tvori povezavo ali ne. Koliko od teh grafov je dreves? V tej seminarski nalogi se bom ukvarjal z preštevanjem vpetih dreves v poljubnem grafu.

1 Preštevanja dreves

Na grafih z enim ali dvema vozliščema lahko najdemo le eno vpeto drevo. Za [3] imamo en izomorfni razred, vendar je matrika sosednosti določena s tem, katero vozlišče je na sredini. Tako dobimo tri vpeta drevesa. Množica [4] nam da 16 dreves in množica [5] nam da 125 dreves. Opazimo, da nam množica $[n]$ da n^{n-2} dreves. To je **Cayley-eva formula**.

Označimo s S množico n števil. Obstaja natanko n^{n-2} načinov, da iz elementov S sestavimo seznam dolžine $n-2$. Množico seznamov označimo s S^{n-2} in elementi te množice bodo predstavljali drevesa z množico vozlišč S. Seznamu, ki pripada drevesu T, pravimo **Prüfer-jeva koda**.

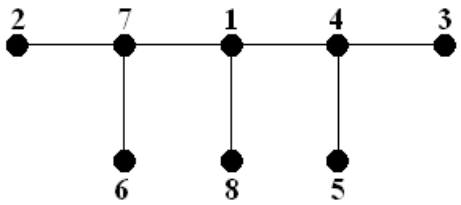
ALGORITEM 1 (Prüfer-jeva koda)
vhod: drevo T z množico vozlišč $S \subseteq \mathbb{N}$
izhod: koda $f(T) = (a_1, a_2, \ldots, a_{n-2})$
iteracija: Na i-tem koraku zbriši najmanjši preostali list in naj bo a_i njegov sosed.

Primer: Po $n-2$ ponovitvah ostane samo ena od začetnih $n-1$ povezav. Dobili pa smo seznam $f(T)$ dolžine $n-2$ s podatki iz S. V drevesu spodaj je najmanjši list 2, zbrišemo ga in zapišemo v seznam 7. Potem zbrišemo 3 pa 5 in vsakič zabeležimo 4. Najmanjši list, ki nam je ostal v drevesu na 5 točkah je 4 in tako nadaljujemo.

Koda, ki jo dobimo je $(7, 4, 4, 1, 7, 1)$ in točki, ki na koncu ostaneta sta 1 in 8. Po prvem koraku je ostanek Prüfer-jeve kode Prüfer-jeva koda poddrevesa T_0 z množico točk $V(T) - 2$.

Če poznamo množico točk S, lahko iz kode a nazaj dobimo drevo. Ideja je v tem, da najdeš vse povezave. Začnemo z množico izoliranih točk S. Na vsakem koraku dodamo eno povezavo in označimo eno točko. Ko razmišljamo o a_i, nam ostane $n - i + 1$

neoznačenih točk in $n-i-1$ znakov iz a (vključno z a_i). Tako se vsaj dve od neoznačenih točk ne pojavljata med ostalimi znaki v a. Naj bo x najmanjši od teh znakov. Dodaj povezavo xa_i in označi x. Po $n-2$ korakih, nam ostaneta dve neoznačeni točki; povežemo ju in s tem dodamo zadnjo povezavo.



Ogeljmo si zgornjo sliko. Najmanjši element množice S, ki ni v kodi a je 2. Tako bo prva povezava, ki jo bomo dodali povezovala 2 in 7, in označimo 2. Sedaj je najmanjši element, ki manjka v kodi 3. Zato povežemo 3 z 4, ki je a_2. Ko nadaljujemo, obnavljamo povezave po vrstnem redu, kot so bile izbrisane, ko smo iskali a iz T.

V tem procesu ima vsaka komponenta grafa, ki smo ga gradili eno neoznačeno točko. Če bi dodali povezavo med tema dvema neoznačenima točkama, bi med seboj povezali dve komponenti grafa. Po tem, ko označimo eno točko nove povezave, ima zopet vsaka komponenta eno neoznačeno točko. Po $n-2$ korakih, imamo dve neoznačeni točki in zato dve komponenti. S tem, ko dodamo zadnjo povezavo med tema dvema točkama, tvorimo povezan graf. Zgradili smo povezan graf z n točkami in $n-1$ povezavami. To je drevo, nismo pa še dokazali, da je njegova Prüfer-jeva koda enaka a.

Izrek 1 (Cayley-eva formula, 1889)**.** *Za množico $S \subseteq \mathbb{N}$ moči n, obstaja n^{n-2} dreves, katerih množica vozlišč je S.*

Dokaz. Za $n = 1$ drži, zato predpostavimo $n \geq 2$. Dokažimo, da zgornji algoritem določa bijekcijo f iz množice dreves z množico točk S v množico S^{n-2}, ki je množica seznamov dolžine $n-2$ z elementi iz množice S. Za vsak $a = (a_1, a_2, \ldots, a_{n-2}) \in S^{n-2}$ moramo pokazati, da natanko eno drevo T z množico točk S zadošča $f(T) = a$. Dokažimo to z indukcijo po n. Baza indukcije: $n = 2$. Imamo eno drevo z dvema točkama. Prüferjevava koda je seznam dolžine 0 in je edini tak seznam.

Indukcijski korak: $n > 2$. Računanje $f(T)$ zniža stopnjo vsake točke na 1 in jo potem po možnosti izbriše. Zato se vsaka točka, ki ni list pojavi v $f(T)$. V $f(T)$ pa ni nobenega lista, ker bi to pomenilo, da je list sosed lista, kar pa je možno le v drevesu z eno samo točko, mi pa smo predpostavili, da imamo več kot 2 točki. Zato so listi T elementi S, niso pa nujno elementi $f(T)$. Če $f(T) = a$, potem je prvi list, ki ga izbrišemo, najmanjši element iz S, ki ni v a (imenumo ga x), in sosed od x je a_1.

Dan imamo $a \in S^{n-2}$ in iščemo vse rešitve $f(T) = a$. Pokazali smo že, da ima vsako tako drevo točko x kot najmanjši list in povezavo xa_1. Ko izbrišemo x nam ostane drevo

z množico točk $S' = S - x$. Njegova Prüfer-jeva koda je $a' = (a_2, \ldots, a_{n-2})$.

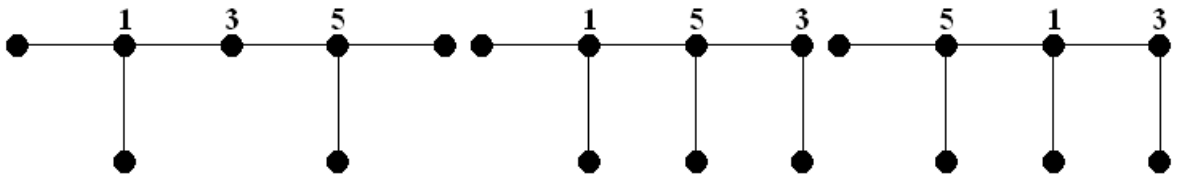
Po indukcijski predpostavki obstaja natanko eno drevo T' na množici točk S' in s Prüfer-jevo kodo a'. Ker vsako drevo s Prüfer-jevo kodo a dobimo tako, da takšnemu drevesu dodamo povezavo xa_1, obstaja največ ena rešitev za $f(T) = a$. Še več, če T' dodamo povezavo xa_1, dobimo drevo z množico točk S in Prüferjevo kodo a, torej obstaja vsaj ena rešitev. In zato obstaja natanko ena. □

Posledica 2. *Naj bodo $d_1, d_2, \ldots, d_n$ pozitivna cela števila, katerih vsota je $2n-2$. Potem obstaja natanko $\frac{(n-2)!}{\prod(d_i-1)!}$ dreves z množico točk $[n]$ takšnih, da ima točka i stopnjo d_i za vsak i.*

Dokaz. Med konstruiranjem Prüfer-jeve kode drevesa T, zapišemo x vsakič, ko izbrišemo soseda od x, dokler ne izbrišemo x samega ali pa ga pustimo med zadnjima dvema točkama. Tako se vsaka točka x pojavi $d_x - 1$ krat v Prüfer-jevi kodi.

Zato štejemo drevesa s stopnjami teh točk tako, da štejemo sezname dolžine $n - 2$, ki ima za vsak i $d_i - 1$ kopij i. Če kopijam vsakega i dodamo indekse, da jih razlikujemo, potem permutiramo $n-2$ različnih predmetov in imamo $(n-2)!$ seznamov. Dokler nismo razlikovali kopij i, smo šteli vse željene razvrstitve $\prod(d_i - 1)!$ krat. □

Primer: *Drevesa s stalno stopnjo.* Razmislimo o drevesih s točkami $\{1, 2, 3, 4, 5, 6, 7\}$, ki so po vrsti stopnje $\{3, 1, 2, 1, 3, 1, 1\}$. Izračunajmo $\frac{(n-2)!}{\prod(d_i-1)!} = 30$; drevesa so predstaljvena spodaj. Le točke $\{1, 3, 5\}$ niso listi. Ko zbrišemo liste dobimo poddrevo na točkah $\{1, 3, 5\}$. Imamo tri taka poddrevesa, določena s tem, katera točka je na sredini.



Da dopolnimo vsako drevo, dodamo primerno število sosednjih listov vsaki točki, ki ni list, da dobi željeno stopnjo. Šest možnosti je, da dopolnimo prvo drevo (iz preostalih štirih točk izberemo dve, ki bosta sosednji točki 1) in dvanajst možnosti, da dopolnimo vsakega od ostalih dveh grafov (izberemo sosednjo točko točke 3 iz preostalih štirih točk in potem soseda centralne točke od preostalih treh).

2 Vpeta drevesa v grafih

Cayley-eva formula nam ne pove nič drugega kot to, koliko vpetih dreves je v polnem grafu z n vozlišči. Sedaj bomo problem vpetih dreves gledali na poljubnih grafih.

Če v grafu G odstranimo povezavo e, njeni krajišči pa identificiramo, potem temu pravimo **skrčitev povezave** e. Dobljeni graf označimo z G/e.

Trditev 3. *Naj bo $\tau(G)$ število vpetih dreves grafa G. Če $e \in E(G)$ ni zanka, potem je $\tau(G) = \tau(G-e) + \tau(G/e)$.*

Dokaz. Vpeta drevesa, ki ne vsebujejo e, so natanko vpeta drevesa grafa $G - e$. Da bi pokazali, da G ima $\tau(G/e)$ vpetih dreves, moramo pokazati, da skrčitev povezave e definira bijekcijo med množico vpetih dreves G-ja, ki vsebujejo e in množico vpetih dreves grafa G/e.

Ko skrčimo povezavo e v vpetem drevesu, ki vsebuje e, dobimo vpeto drevo grafa G/e. S skrčenjem se ostale povezave ohranijo, tako da nobeni dve drevesi dobljeni s skrčitvijo ne tvorita istega vpetega drevesa grafa G/e. Če naredimo obratno operacijo in novo vozlišče razširimo nazaj v e, dobimo vpeto drevo grafa G. Ker vsako vpeto drevo grafa G/e dobimo natanko enkrat, je funkcija bijekcija. □

$\tau(G)$ lahko računamo tudi z uporabo determinante. Ta tehnika je mnogo hitrejša, saj $n \times n$ determinante lahko izračunamo v manj kot n^3 operacijah.

Izrek 4. *Naj bo G graf brez zank, z $V(G) = \{v_1, \ldots, v_n\}$ in naj bo $a_{i,j}$ število povezav z krajiščema v_i in v_j. Naj bo Q matrika z $q_{ij} = -a_{i,j}$, če $i \neq j$ in $q_{ii} = d(v_i)$ za vsak i. Q^* naj bo matrika, ki jo dobimo, če v Q zbrišemo s-to vrstico in t-ti stolpec. Potem je $\tau(G) = (-1)^{s+t}\det(Q^*)$.*

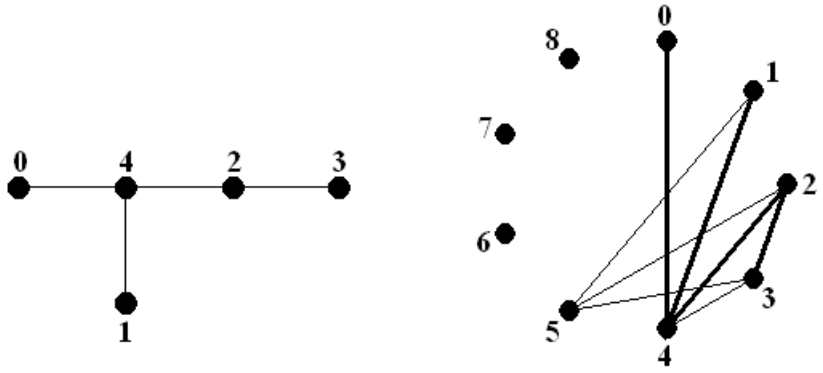
3 Dekompozicija in lepa označitev

Domneva 5 (Ringel, 1964)**.** *Če je T fiksno drevo z m vozlišči, potem lahko graf K_{2m+1} dekomponiramo na $2m+1$ kopij drevesa T.*

Lepa označitev grafa G z m povezavami je funkcija $f : V(G) \longrightarrow \{0, \ldots, m\}$, ki različna vozlišča različno oštevilči in velja $\{|f(u) - f(v)|; uv \in E(G)\} = \{1, \ldots, m\}$. Graf je **lep**, če ima lepo označitev.

Domneva 6 (Ringel, Kotzig, 1964)**.** *Vsako drevo ima lepo označitev.*

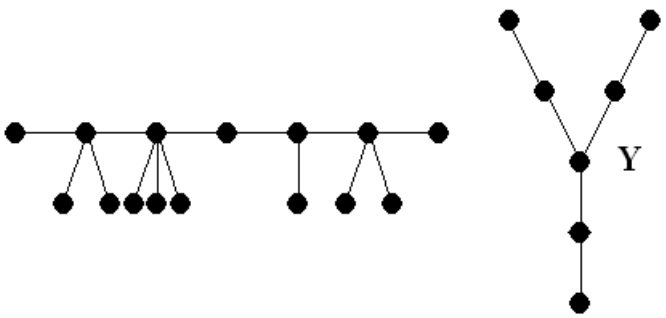
Izrek 7 (Rosa, 1967)**.** *Če ima drevo T z m povezavami lepo označitev, potem ima graf K_{2m+1} dekompozicijo na $2m+1$ kopij T.*



Dokaz. Poglejmo na točke grafa K_{2m+1} kot na kongruenčne razrede po modulu $2m+1$, urejene ciklično. Razlika med dvema kongruenčnima razredoma je 1, če sta zaporedna, 2, če je en razred med njima in tako naprej do razlike m. Povezave grafa K_{2m+1} razvrstimo v skupine glede na razliko med končnimi točkami. Za $1 \leq j \leq m$ obstaja $2m+1$ povezav z razliko j.

Glede na lepo označitev drevesa T definiramo kopije drevesa T v K_{2m+1}; te kopije so $T_0, T_1, \ldots, T_{2m}$. Točke drevesa T_k so $k, \ldots, k+m$ $(mod\ 2m+1)$ pri čemer je $k+i$ sosednja $k+j$ natanko takrat, ko je i soseden j v lepi označitvi drevesa T. Kopija T_0 izgleda kot lepa označitev in ima povezave z vsako razliko. V vsaki naslednji kopiji se vse povazave premaknejo v druge in razlike se ohranijo, če dodamo ena k imenu vsake končne točke. Vsi različni razredi povezav imajo eno povezavo v vsakem T_k in zato $T_0, \ldots, T_{2m}$ razčleni K_{2m+1}. □

Gosenica je drevo, v katerem je odlikovana pot (hrbtenica) sosednja z (ali vsebuje) vsako povezavo.



Grafa gosenica in Y, ki ni gosenica.

Izrek 8. *Drevo je gosenica natanko takrat, ko ne vsebuje drevesa Y na sliki.*

Dokaz. Naj G' označuje drevo, ki ga dobimo iz drevesa G, če izbrišemo vse liste drevesa G. Ker nobena od točk, ki ostanejo v G' niso listi v G, ima G' najvišjo točko stopnje vsaj 3 natanko takrat, ko se Y pojavi v G. Torej G nima kopij drevesa Y natanko takrat, ko je $\Delta(G') \leq 2$. To pa pomeni da je G' pot, kar je ekvivalentno, da je G gosenica. □

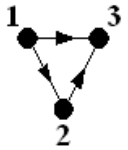
4 Navzven usmerjena drevesa in Eulerjevi digrafi

Z $d^+(v)$ oz. $d^-(v)$ označimo izhodno oz. vhodno stopnjo vozlišča v, to je število povezav iz oz. proti v.

Navzven usmerjeno drevo je drevo, v katerem obstaja med korenom in vsakim njegovim listom usmerjena pot od korena proti listu. Če obrnemo vse povezave dobimo **navznoter usmerjeno drevo**.

Izrek 9. *Dan je digraf brez zank. Naj bo $Q^- = D^- - A'$ in $Q^+ = D^+ - A'$, kjer sta D^- in D^+ diagonalni matriki, kjer je $d^-_{ii} = d^-(v_i)$ in $d^+_{ii} = d^+(v_i)$, A' pa je matrika, kjer je a_{ij} število povezav od v_j proti v_i. Število vpetih navzven usmerjenih dreves (navznoter usmerjenih dreves) grafa G s korenom v_i je vrednost vsakega kofaktorja v i-ti vrstici matrike Q^- (i-tem stolpcu Q^+).*

Primer:



$$Q^- = \begin{pmatrix} 0 & 0 & 0 \\ -1 & 1 & 0 \\ -1 & -1 & 2 \end{pmatrix} \qquad Q^+ = \begin{pmatrix} 2 & 0 & 0 \\ -1 & 1 & 0 \\ -1 & -1 & 0 \end{pmatrix}$$

Digraf zgoraj ima dve navzven usmerjeni drevesi s korenom 1 in dve navznoter usmerjeni drevesi s korenom 3.

Digraf je **krepko povezan**, če obstaja usmerjena pot med poljubnim urejenim parom točk.

Lema 10. *V krepko povezanih digrafih je vsako vozlišče koren navzven (navznoter) usmerjenega drevesa.*

Dokaz. Premislimo za vozlišče v. Iterativno dodajamo povezave, ki so usmerjene iz v. Naj bo S_i množica vozlišč, ki jih dosežemo, ko smo dodali i-to povezavo, $S_0 = \{v\}$. Ker je digraf krepko povezan, obstaja povezava iz S_i. Ko jo dodamo dobimo S_{i+1}. To delamo, dokler ne dosežemo vseh vozlišč. Za navznoter usmerjena drevesa delamo podobno. □

ALGORITEM 2 (Euler-jev obhod v digrafu)
vhod: Euler-jev digraf G brez izoliranih vozlišč in navznoter usmerjeno vpeto drevo T s korenom v.
1. korak: Za vsak $u \in V(G)$ določi vrstni red povezav, ki zapuščajo u tako, da za $u \neq v$ povezava, ki zapušča u v drevesu T pride zadnja.
2. korak: Začenjši z v, konstruiraj Euler-jev obhod tako, da vedno zapustiš trenutno vozlišče u po naslednji neuporabljeni povezavi po vrstnem redu določenem za u.

Izrek 11. *Zgornji algoritem vedno vrne Euler-jev obhod.*

Dokaz. S ponočjo zgornje leme skonstruiramo navznoter usmerjeno drevo z korenom v. Z zgornjim algoritmom skonstruiramo sled. Zadostuje, da pokažemo, da se sled lahko konča le v v in le takrat, ko prepotuje vse povezave.

Ko smo v vozlišču $u \neq v$, povezava, ki zapušča u v drevesu T, še ni bila uporabljena, ker je $d^+(u) = d^-(u)$. Kadarkoli tako vstopimo v u, vedno obstaja povezava ven. Zato se sled lahko konča le v v.

Končamo, ko ne moremo več nadaljevati. Smo v v in smo porabili vse izhodne povezave. Ker je $d^+(u) = d^-(u)$, smo porabili tudi vse vhodne povezave v v. Ker ne moremo uporabiti povezave iz T dokler ni to zadnja možnost, ne moremo uporabiti vseh vhodnih povezav v v, dokler nismo obdelali vseh drugih vozlišč, saj T vsebuje pot iz kateregakoli vozlišča v v. □

Izrek 12 (van Aardenne-Ehrenfest, Bruijn, 1951)**.** *V Euler-jevem digrafu z $d_i = d^+(v_i) = d^-(v_i)$ je število Euler-jevih obhodov $c\prod_i (d_i - 1)!$, kjer c šteje navznoter (navzven) usmerjena drevesa pri kateremkoli vozlišču.*

Dokaz. Dokazati moramo le da prejšnji algoritem poišče vse Euler-jeve obhode.

Da najdemo drevo in vrstni red, ki generira Euler-jev obhod C, sledi C iz e in si zapomni vrstni red povezav, ki zapuščajo vsako vozlišče. Naj bo T usmerjen podgraf sestavljen iz zadnje povezave na C, ki izhaja iz kateregakoli vozlišča razen iz v. Ker se zadnja povezava iz vozlišča pojavi v C šele potem, ko so vse povezave vstopile vanj, se vsaka povezava v T razširi v pot v T, ki doseže v. T tako z $n-1$ povezavami tvori navznoter usmerjeno drevo s korenom v. C je obhod ki ga dobimo s prejšnjim algoritmom iz T in vrstnega reda povezav, ki smo ga zapisali. □

5 Literatura

B. West, Introduction To Graph Theory, 2001, Prentice Hall.