Kombinatorika, seminar

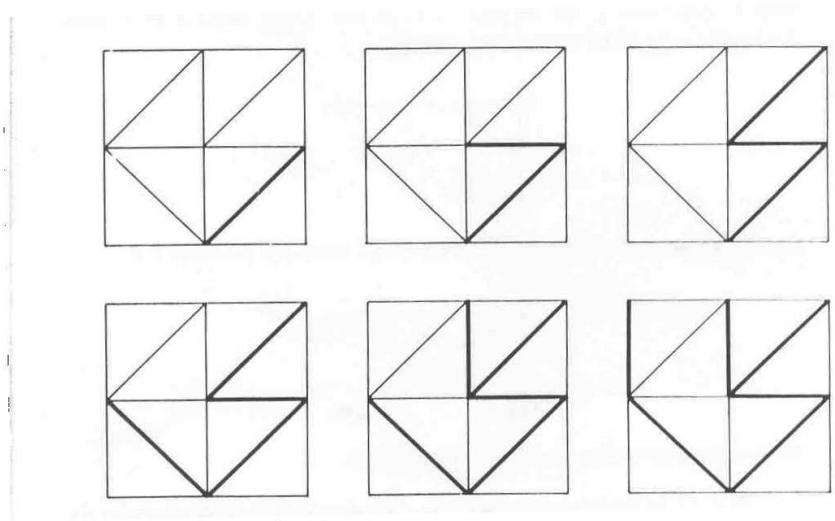
Andreja Tomažič, Alenka Šandor

11.maj, 2004

7 ALGORITMI IN UPORABA

7.1 Algoritem vpetega drevesa

Primer 1: Dan je povezan graf. Najdi vpeto drevo grafa.
Algoritem, ki ga bomo predstavili vsebuje tri korake.



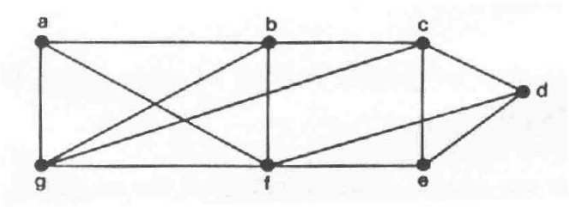
KORAK 1: Izberi katerikoli povezavo e iz grafa in označi končne točke povezave e z v_1 in v_2. Začni konstruirati drevo v tej povezavi. Vstavi $i = 1$ in $j = 3$.

KORAK 2: Preveri, če obstaja kakšen sosed od v_i v grafu, ki še ni v obstoječem drevesu.

KORAK 3: Če je sosed od v_i v grafu in ni v obstoječem drevesu, ga označi z v_j in dodaj povezavo v_iv_j k drevesu. Če je j enak številu točk v grafu se ustavi. Sedaj imamo vpeto drevo. Če ni, vstavi $j = j + 1$ in se vrni

k drugemu koraku. Če v_i nima več soseda, vstavi $i = i + 1$ in nadaljuj s korakom 2.

Primer 2: Dan je povezan graf. Najdi vpeto drevo.

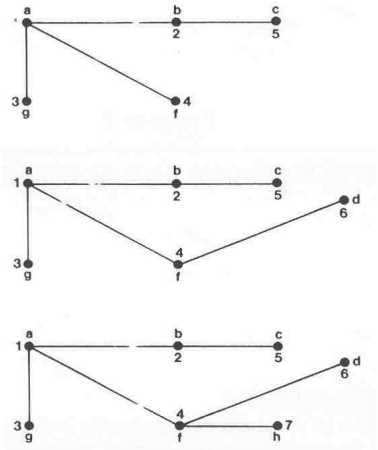


Korak 1: Izberemo poljubno povezavo ab za konstrukcijo drevesa. Označi končni točki izbrane povezave z v_1 in v_2.(glej sliko spodaj)

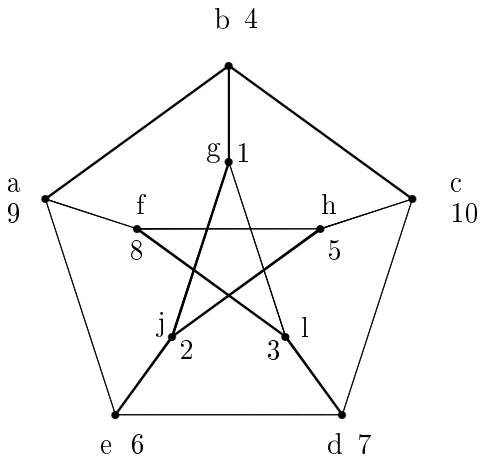
Korak 2: Ali obstaja v grafu kakšen sosed točke a, ki še ni v drevesu? Ja, g je tak sosed.

Korak 3: Točko g označimo z v_3 in dodamo povezavo v_1v_3 k drevesu. Ker je $3 < 7$, povečamo j = 4 in se vrnemo k drugemu koraku.

Nadaljujemo s postopkom dokler ne dobimo vpetega drevesa.



Primer 4: Dan je Petersenov graf. Najdi vpeto drevo.



Trditev 1 *Algoritem za vsak povezan graf vrne vpeto drevo.*
Dokaz. *Predpostavimo, da algoritem ne vrne vpetega drevesa, zato obstaja neka točka, ki je v grafu in ni v drevesu. Ker pa je originalen graf povezan, obstaja povezava do vsake točke.*
Povezave do točk dodajamo v tretjem koraku algoritma. Lahko predpostavimo, da je v sosednja z neko točko v dobljenem drevesu, zato postopek še ni končan. Dodamo jo v tretjem koraku. Torej postopek dejansko deluje in lahko mu rečemo algoritem. □

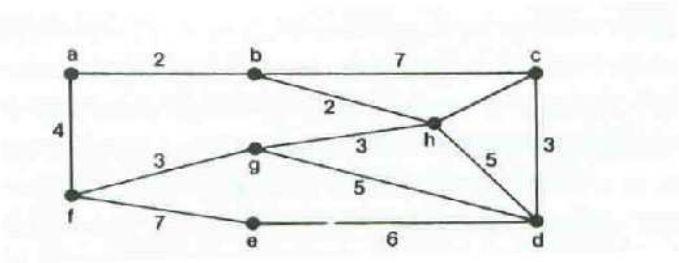
Primer 5: *Želimo zgraditi zabaviščni park. Tlakovati moramo poti do vsakega igrala. Pri tem bi radi minimizirali število tlakovcev in s tem znižali stroške. Točke v našem grafu predstavljajo igrala, povezave pa razdalje med igrali. Tam, kjer ni povezave, obstaja ovira. Najti torej želimo vpeto drevo z minimalno vsoto tež oz. najcenejše drevo. Točke, označene s številkami, imenujemo teže.*
Problem bomo rešili z algoritmom, ki se imenuje:

Kruskalov algoritem

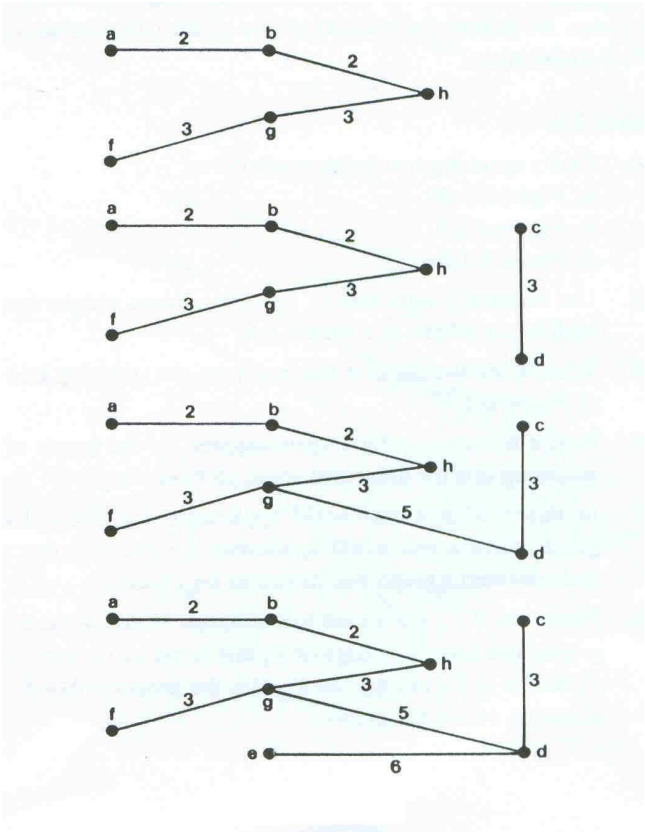
Korak 1: *Od vseh robov v grafu izberi tistega z najmanjšo težo (možno je, da je povezav z isto težo več zato je izbira poljubna). Začni z konstrukcijo gozda na tej povezavi.*

Korak 2: *Ponovno izberi povezavo z najmanjšo težo, ki še ni v že obstoječem gozdu in jo dodaj tako, da ne dobimo cikla.*

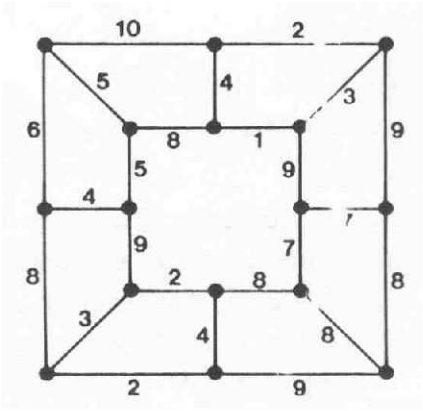
Korak 3: *Preveri, če imaš vpeto drevo. Če jo imaš se ustavi, če ne, ponovi korak 2.*



Imamo dve povezavi z najmanjšo težo 2. Izberimo povezavo ab. Od povezav, ki so ostale je povezava bh tista, ki ima najmanjšo težo. Lahko jo dodamo k obstoječemu gozdu, ker s to povezavo ne dobimo cikla. Povezava z najmanjšo težo, ki je ostala, je hg s težo 3. Spet jo lahko dodamo ne da bi ustvarili cikel. Postopek nadaljujemo dokler ne dobimo gozda, kar nam prikazuje spodnja slika.

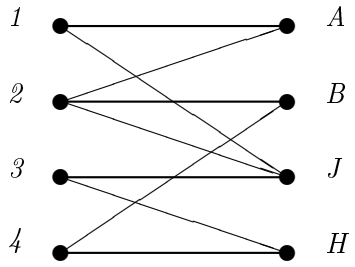


Primer 6: *S pomočjo Kruskalovega algoritma poišči vpeto drevo z minimalno težo na povezanem grafu.*



7.2 Prirejanje v grafih, problemi razporejanja

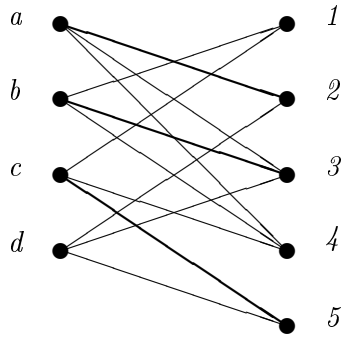
PRIMER 1: *Točke v dvodelnem grafu označimo z otroci in sadjem. Zastavimo si problem, kako naj otrokom razdelimo sadje tako, da bo vsak dobil sadež ki si ga želi. S tem smo definirali problem, kako najti maksimalno prirejanje v dvodelem grafu.*



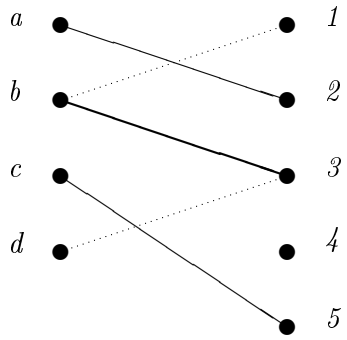
Točki 1 smo priredili točko A, torej sta 1 in H prirejeni točki. Tako naredimo za vsako točko označeno s številko. Našli smo prirejanje M, ki je hkrati maksimalno prirejanje.

Prirejanje *v (dvodelnem) grafu je podgraf, ki je 1-regularen graf. Torej je prirejanje množica povezav, v kateri ima vsaka točka eno povezavo.*
Maksimalno prirejanje *je prirejanje, ki ima največje možno število povezav.*

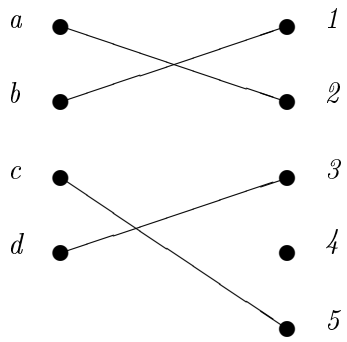
PRIMER 2: *Drug primer prirejanja je t.i. načrtovalni problem, s katerim želimo dodeliti vsem delavcem naloge. Predpostavimo, da ima delovodja na gradbišču ima 4 delavce in 5 del, ki morajo biti narejena. Naslednji graf ilustrira to situacijo. Točke označene s črkami predstavljajo delavce. Točke s številkami pa predstavljajo dela, ki jih je potrebno opraviti. Narisane so povezave med posameznim delavcem in nalogami, za katere je usposobljen.*



Maksimalno prirejanje v dvodelnem grafu bomo poiskali s pomočjo madžarskega algoritma.
Najprej poiščemo poljubno prirejanje in sicer tako, da vzamemo točko označeno s črko in ji priredimo točko označeno s številko. Vsaki črki priredimo primerno številko, ki še ni vključena v nobeno prirejanje.



Naše začetno prirejanje dodeli trem delavcem tri dela, pri tem pa ostane delavec d brez dela.
Poskusili bomo izboljšati prirejanje, tako da bomo dobili še več prirejanj. Ker točki d in 1 ter d in 4 nista sosednji, ne moremo med njima prirediti prirejanja. Ker so vse točke, katere bi lahko priredili točki d že prirejene, bomo prirejanje izboljšali tako, da bomo odstranili prireditev med b in 3 ter dodali novi prireditvi med d in 3 ter b in 1. Dobili smo novo prirejanje, ki je hkrati maksimalno prirejanje. Seveda bo tudi v maksimalnem prirejanju eno delo ostalo prosto.



*Dano prireranje M v dvodelnem grafu je M-**alternirajoča pot**, katere povazave alternirajo (se izmenjujejo) med povezavami, ki so v M in ki niso v M.*
Pot med d in 1, prikazana v zgornjem grafu, je M-alternirajoča pot.

*M-alternirajočo pot, ki povezuje dve neprirejeni točki, imenujemo M-**nezasičena pot**.*

Madžarski algoritem

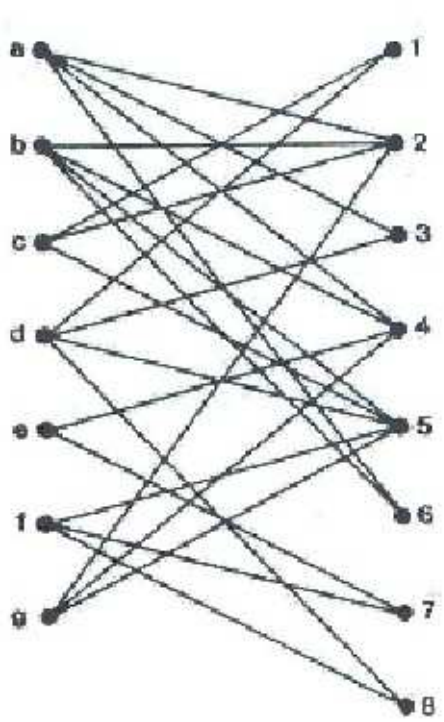
Naj bo G dvodelni graf z rdečimi in modrimi točkami. Pri tem naj bo m rdečih in n modrih točk. Naj velja $m \leq n$.

Korak 1. *Začni s poljubnim prirejanjem. Pri tem označimo vse rdeče točke kot primerne za prirejanje.*

Korak 2. *Če smo uporabili vse primerne rdeče točke za prirejane, končamo algoritem. Dobili smo maksimalno prirejanje.*
Če so ostale primerne točke, ki še niso prirejene, nadaljujemo s tretjim korakom.

Korak 3. *Naj bo a primera rdeča točka, ki še ni prirejena. Konstruiramo M-alternirajočo pot iz a.*
Če drevo vsebuje M-nezasičeno pot izbrišemo prirejene povezave in priredimo nove povezave. Označimo vsa rdeče točke kot prirejene in pojdimo k drugemu koraku.
Če drevo ne vsebuje M-nezasičene poti označimo a kot neprimerno za prirejanje in pojdimo k drugemu koraku.

***PRIMER 3:** 7 ljudi je prišlo pozno zvečer v restavracijo na večerjo, kjer so lahko izbirali le še med 8 jedmi. Graf prikazuje katero hrano si želi posameznik. Ali so se lahko vsi najedli po svoji želji?*



Korak 1. *Začnemo s poljubnim prirejanjem. Naše začetno prirejanje M dodeli 5 ljudem željene večerje. Dva pa ostaneta brez željene jedi.Primer je prikazan na sliki 1.*

Korak 2. *Točka g je primerna za prirejanje in še ni primerejena.*

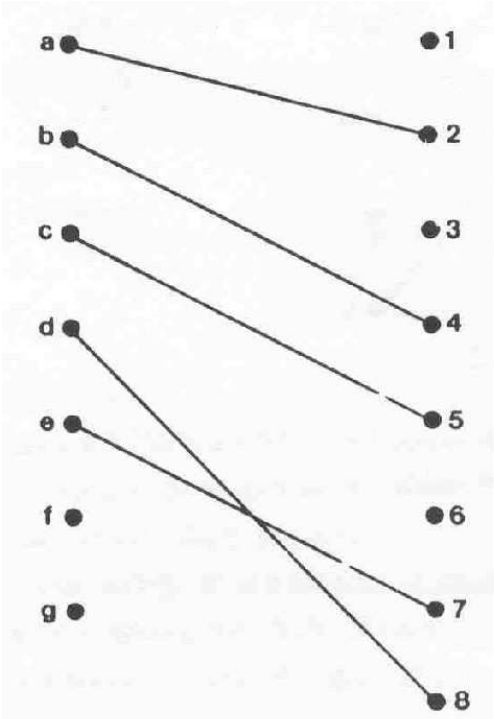
Korak 3. *Konstruiramo M-alternirajočo pot, ki poteka med g in 1 (torej od g do 5, naprej do c in 1). Odstranimo prirejeno povezavo med c in 5 ter priredimo povezavi med g in 5 ter c in 1, kot je vidno na sliki 2.*

Korak 2. *Točka f je primerna točka za prirejena in še ni prirejana.*

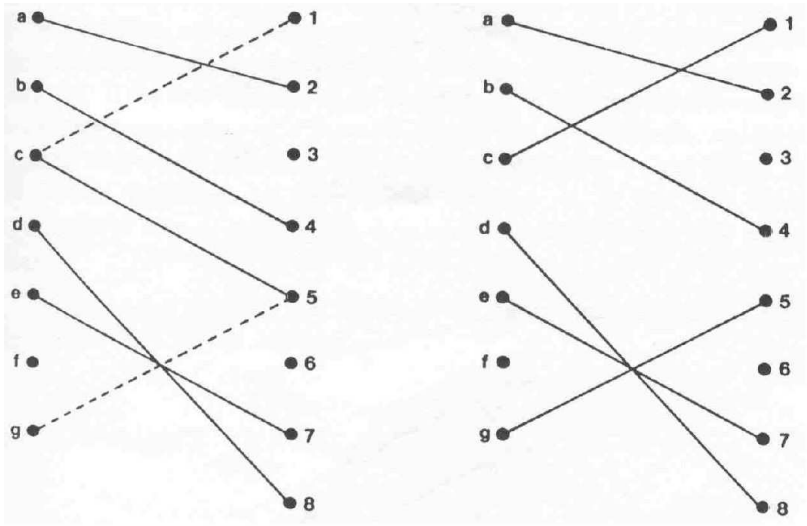
Korak 3. *Konstruiramo M-alternirajočo pot, ki poteka med f in 3 (torej od f do 5, naprej do g in 2, nato do a in 3). Odstranimo že prirejeni*

povezavi med g in 5 ter mad a in 2. Priredimo nove povezave med f in 5, g in 2 ter a in 3.

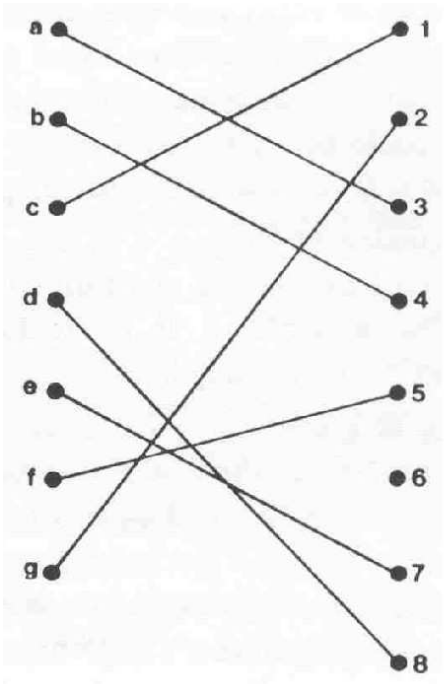
Korak 2. *Uporabili smo vse primerne točke za prirejanje. Dobili smo maksimalno prirejanje.Rezultat je na sliki 3. Končamo algoritem.*



Slika 1: Slika poljubnega prirejanja (korak 1)



Slika 2: Konstrukcija M-alternirajoče poti (korak 3)



Slika 3: Maksimalno prirejanje