Primer velikega omrežja: povezave med besedami v slovarju

Iz besed slovarja lahko ustvarimo veliko omrežje, tako da sta dve besedi povezani (z neusmerjeno povezavo), če lahko iz ene besede dobimo drugo besedo

- z zamenjavo ene črke (npr. kosa – koza)
- z dodajanjem oziroma brisanjem ene črke (npr. kos – kosa, kos – kaos).

Za prikaz smo uporabili Knuthov slovar angleških besed, ki vsebuje 52.652 besed z dolžinami od 2 do 8 črk. Na opisani način dobimo neusmerjeno omrežje z 89.038 povezavami. Omrežje je redko: gostota je 0.0000642.

Datoteka: dic28.net.

Poti v grafu

V *usmerjenem grafu* velja:

- Zaporedje točk grafa $(v_1, v_2, \ldots, v_k)$ imenujemo ***sprehod***, če velja

  $$(v_i, v_{i+1}) \in A, i = 1 \ldots k-1$$

  (zaporedne točke morajo biti povezane z usmerjenimi povezavami).

- Zaporedje točk grafa $(v_1, v_2, \ldots, v_k)$ imenujemo ***veriga***, če velja

  $$(v_i, v_{i+1}) \in A \text{ ali } (v_{i+1}, v_i) \in A, i = 1 \ldots k-1$$

  (zaporedne točke morajo biti povezane, smer povezav ni pomembna).

- Sprehod je ***osnoven***, če so vse točke, razen morda začetne in končne, različne. Osnoven sprehod bomo imenovali tudi *pot*.

- Sprehod je ***enostaven***, če so vse povezave različne.

- Če je $v_1 = v_k$, se sprehod imenuje ***cikel*** (sprehod se začne in konča v isti točki).

- Veriga, ki se začne in konča v isti točki, se imenuje

sklenjena veriga.

- Če je $v_1 = v_k$ in $k = 1$, se cikel imenuje ***zanka*** (točka je povezana sama s sabo).
- ***Dolžina poti*** je $k - 1$ (število povezav na poti med dvema točkama).
- Med dvema točkama lahko obstaja več poti. Posebej zanimiva je ***najkrajša pot*** (oziroma vse najkrajše poti).

  V primeru relacije *sporočanje novic* predstavlja najkrajša pot potovanje informacije med osebami – npr. čimkrajše so najkrajše poti od vseh oseb do izbrane osebe, prej bo oseba seznanjena z novicami.

  V primeru relacije *je (bil) v stiku z* predstavlja najkrajša pot najverjetnejšo možnost prenosa okužbe. Za razliko od prejšnjega primera pa v tem primeru za neko osebo ni ugodno, da so poti čimkrajše.
- Dolžina najdaljše najkrajše poti v grafu se imenuje ***premer*** (*diameter*) grafa.

Nekaj zanimivih rezultatov

Omrežja z zelo velikim številom točk imajo velikokrat kratke najkrajše poti med točkami.

Primer:

- Povprečna dolžina najkrajše poti omrežja WWW, z več kot 800 milijoni točk, je okrog 19.

  Albert, R., Jeong, H., and Barabasi, A.-L. (1999): Diameter of the World-Wide Web. *Nature*, **401**, 130-131.

  http://www.nd.edu/ alb/Publication06/062
  Diameter of the world wide web/
  Diameter of the world wide web.pdf

- Ocenjuje se, da je v socialnih omrežjih (koga poznaš) s preko 6 milijardami posameznikov, povprečna dolžina najkrajše poti med dvema posameznikoma okrog 6.

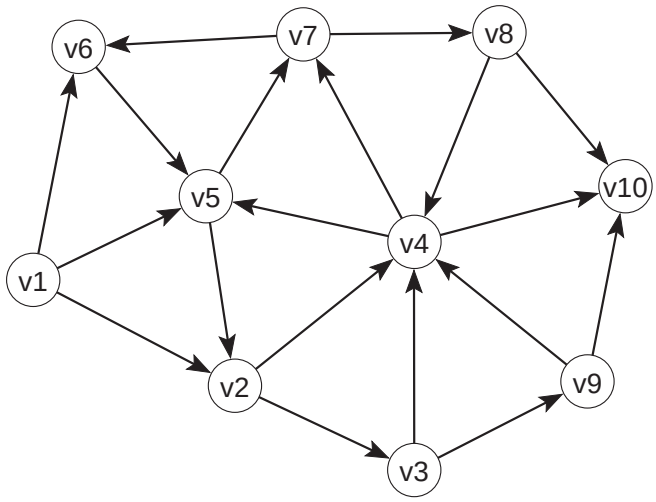
  Milgram, S. (1967): The small-world problem. *Psychol. Today*, **2**, 60-67.

Vredno ogleda:

Albert, R., Jeong, H., and Barabasi, A.-L. (2000):
Error and attack tolerance of complex networks.
Nature, **406**, 378-382.
http://www.nd.edu/~alb/Publication06/072 Error and attack tolerance of complex networks/Error and attack tolerance of complex networks.pdf

in

Barabasi, A.-L., A., and Jeong, H. (2000):
Scale-free characteristics of random networks: The topology of the World Wide Web. *Physica A*, **281**, 69-77.
http://www.nd.edu/~networks/Papers/proceeding.pdf



$v1 - v6 - v4 - v8$ ni niti veriga

$v1 - v6 - v7 - v8$ je veriga, ni pa sprehod

$v8 - v4 - v5 - v2 - v4 - v10$ je enostaven, ni pa osnoven sprehod

$v8 - v4 - v5 - v2 - v3 - v9$ je osnoven sprehod (pot)

$v4 - v10 - v8 - v4$ je sklenjena veriga

$v6 - v5 - v7 - v6$ je cikel

$v1 - v2 - v3 - v9 - v10$
je pot med točkama $v1$ in $v10$, ni pa najkrajša pot (dolžina 4).

Najkrajša pot med $v1$ in $v10$: $v1 - v2 - v4 - v10$ (dolž. 3).

Premer je 5: $v7 - v6 - v5 - v2 - v3 - v9$

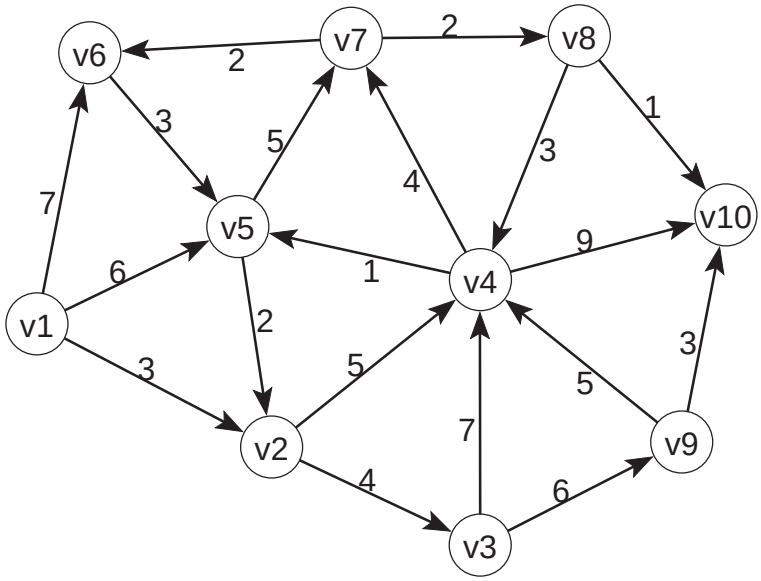
Vrednost poti med dvema točkama je enaka vsoti vrednosti na povezavah, ki sestavljajo to pot. Če so vse vrednosti na povezavah enake 1, je vrednost poti kar enaka dolžini poti.

Najkrajšo pot lahko definiramo glede na ***dolžino*** poti ali ***vrednost*** poti.

Vzemimo npr. iskanje najkrajše letalske poti med dvema krajema pri poznanih letalskih linijah.

1. iskanje poti med krajema, tako da bo *število prestopanj* čim manjše (najkrajša pot glede na dolžino poti v omrežju).
2. iskanje poti med krajema, tako da bo
   - *čas* sedenja v letalu čimkrajši (vrednosti na povezavah so trajanja poletov)
   - *skupna cena* prevoza najnižja (vrednosti na povezavah so cene prevozov)
   - *skupna dolžina poti* čimkrajša (vrednosti na povezavah so oddaljenosti letališč)

Vzemimo isti primer kot prej, le da naj imajo povezave naslednje vrednosti (omrežje flow2.net):



Pot z najmanjšo vrednostjo med točkama $v1$ in $v10$ je:
$\boxed{v1 - v5 - v7 - v8 - v10}$ (dolžina je 4, vrednost je 14).

Najkrajša pot med točkama 1 in 10, po številu povezav na poti, je:
$\boxed{v1 - v2 - v4 - v10}$ (dolžina je 3, vrednost je 17).

Optimalni poti po obeh kriterijih se v tem primeru precej razlikujeta.

Veriga z najmanjšo vrednostjo med točkama $v1$ in $v10$ je:
$\boxed{v1 - v2 - v5 - v4 - v8 - v10}$ (dolžina je 5, vrednost je 10).

Najkrajše poti v programu Pajek

Eno najkrajšo pot med izbranima točkama poiščemo z
Net/Paths between 2 vertices/One Shortest
vnesemo željeni točki – vnesemo lahko oznako točke ali njeno številko.
Na vprašanje
Forget values on lines odgovorimo z Yes, če iščemo najkrajšo pot glede na dolžino poti, oz. z No, če iščemo najkrajšo pot glede na vrednosti na povezavah.

Na vprašanje
Identify vertices in source network odgovorimo z No.

Za rezultat dobimo novo (pod)omrežje, ki vsebuje samo izbrani točki, točke, ki ležijo na najkrajši poti in ustrezne povezave. Dobljeno pot narišemo z uporabo algoritma
Layout/Energy/Kamada Kawai/Fix first and last
(prva in zadnja točka naj bosta v nasprotnih kotih slike).

Pojasnilo: Za razliko od večine programov za analizo (npr. SPSS) kjer po izvedbi analize dobimo rezultat v neki tekstovni obliki, je večina operacij v Pajku takih, da vrne kot rezultat nek nov objekt (npr. novo omrežje, novo razbitje. . .).

Tako je tudi pri iskanju najkrajše poti – rezultat je novo omrežje, ki ga lahko nadalje analiziramo, rišemo…

Pozor: v novem omrežju se točke omrežja spet zaporedoma oštevilčijo s številkami od 1 naprej. Če hočemo 'prepoznati' s katerimi točkami iz osnovnega omrežja imamo opravka, moramo pri risanju omrežja uporabiti označevanje točk z njihovimi oznakami (Options/Mark Vertices Using/Labels ali Ctrl+L).

V usmerjenem omrežju poiščemo **verige** (smeri povezav niso pomembne), tako da usmerjene povezave spremenimo v neusmerjene z ukazom Net/Transform/Arcs to Edges/All.

Primer: flow2.net (omrežje z zadnje slike):
poiščimo najkrajšo pot med $v1$ in $v10$ po obeh kriterijih
poiščimo najkrajšo verigo med $v1$ in $v10$ po obeh kriterijih

Pozor: vedno, ko poženemo neko operacijo v Pajku, se le ta izvede nad izbranim omrežjem, zato moramo pred izvedbo operacije preveriti, če je trenutno res izbrano omrežje, ki ga potrebujemo.

Vse najkrajše poti med izbranima točkama poiščemo z
Net/Paths between 2 vertices/All Shortest
ter odgovorimo na vsa vprašanja enako kot pri iskanju samo ene poti med izbranima točkama.

Seveda se pri iskanju poti med dvema točkama lahko zgodi, da iz ene točke druge ne moremo doseči – pot ne obstaja.

Premer omrežja poiščemo z
Net/Paths between 2 vertices/Diameter
V tem primeru nam Pajek vrne samo točki, ki sta ’najbolj narazen’. Če želimo dobiti tudi ustrezno pot, poženemo običajno iskanje najkrajše poti med dobljenima točkama.

Pozor: Računanje premera omrežja je časovno zahtevna operacija, zato jo izvajamo le na manjših omrežjih!

Primer: Vzemimo omrežje besed (dic28.net) in poiščimo najkrajše poti med izbranimi besedami:

white – yellow
engaged – skeptics
…

Za zelo majhna omrežja lahko poiščemo tudi razdalje (glede na dolžino poti) med vsemi pari točk in število najkrajših poti, ki točke povezujejo.
Uporabimo
Net/Paths between 2 vertices/Geodesics Matrices*

Matrika najkrajših poti (flow2.net):

	1	2	3	4	5	6	7	8	9	10
v1	?	1	2	2	1	1	2	3	3	3
v2	?	3	1	1	2	3	2	3	2	2
v3	?	3	4	1	2	3	2	3	1	2
v4	?	2	3	3	1	2	1	2	4	1
v5	?	1	2	2	3	2	1	2	3	3
v6	?	2	3	3	1	3	2	3	4	4
v7	?	3	4	2	2	1	3	1	5	2
v8	?	3	4	1	2	3	2	3	5	1
v9	?	3	4	1	2	3	2	3	5	1
v10	?	?	?	?	?	?	?	?	?	?

k-sosedi

Točka j se imenuje k-sosed (k-neighbour) točke i, če ima najkrajša pot od točke i do točke j dolžino k.

Za prejšnje omrežje so oddaljenosti vseh točk od točke $v1$ naslednje:

točka	1	2	3	4	5	6	7	8	9	10
k	0	1	2	2	1	1	2	3	3	3

V programu Pajek izračunamo oddaljenosti vseh točk od izbrane točke z: Net/k-Neighbours/Output
Nato vnesemo željeno točko (številko ali oznako) in pri vprašanju **Maximal distance** vnesemo 0 – želimo pregledati vse oddaljenosti.

Rezultat te operacije je ***razbitje*** **(partition)** – za vsako točko imamo v tabeli izpisano oddaljenost (neko število) od izbrane točke.

Oddaljenosti točk, ki so od izbrane točke nedosegljive, se postavijo ne neko veliko številko (999999998).

Z ukazom Net/k-Neighbours/Output torej izračunamo v koliko korakih lahko pridemo od izbrane točke do vseh drugih točk.

Podobno z ukazom Net/k-Neighbours/Input izračunamo v koliko korakih lahko pridemo od vseh drugih točk do izbrane točke, z ukazom Net/k-Neighbours/All pa iščemo oddaljenosti ne glede na smeri puščic.

Rezultat si v primeru manjših omrežij lahko pogledamo kar tako, da dvakrat kliknemo na dobljeno razbitje ali izberemo ikono Edit/Partition.

Vaji:
Poiščimo oddaljenosti vseh točk od točke $v1$ v omrežju flow2.net.
V omrežju dic28.net poiščimo oddaljenosti vseh besed od izbrane besede.

Pri večjih omrežjih nas ponavadi ne zanimajo oddaljenosti vseh točk od izbrane točke, ampak samo najbolj / najmanj oddaljene točke. Potem, ko izračunamo oddaljenosti od izbrane točke, izpišemo 20 najbližjih točk in frekvenčno porazdelitev oddaljenosti z: Info/Partition in pri vprašanju **Input 1 or 2 numbers: +/highest, -/lowest**
vnesemo -20. Če hočemo izpisati 20 najbolj oddaljenih točk, vnesemo 20. Pri vprašanju **Select minimum frequency** pustimo vrednost 1 (v frekvenčni porazdelitvi prikažemo dan razred, če je v njem vsaj ena enota).

Izločanje k-okolice dane točke

Iz omrežja izrežemo podomrežje s točkami, ki so od dane točke oddaljene za manj kot 2, takole: Izberemo
Operations/Extract from Network/Partition
ter za spodnjo mejo vnesemo 0 za zgornjo pa 2.

Predhodno moramo poskrbeti, da sta v glavnem oknu programa Pajek izbrani omrežje in ustrezno razbitje.

Rezultat preverimo s sliko dobljenega omrežja.

Splošno navodilo za iskanje ukazov v Pajku

Kriterij za razporejanje ukazov po menujih v programu Pajek je naslednji:
ukazi, za izvedbo katerih potrebujejo kot vhod samo eno omrežje, se nahajajo v menuju Net,
ukazi, za izvedbo katerih potrebujejo kot vhod dve omrežji, se nahajajo v menuju Nets,
ukazi, ki za izvedbo potrebujejo dve strukturi (npr. omrežje in razbitje), se nahajajo v menuju Operations.
ukazi, za izvedbo katerih potrebujejo kot vhod samo eno razbitje, se nahajajo v menuju Partition. . . .

Aciklična omrežja

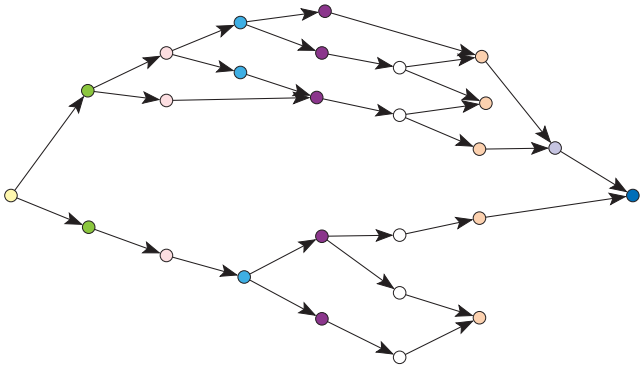
Poseben primer usmerjenega omrežja je ***acikličnо omrežje***:

Usmerjeno omrežje imenujemo *aciklično* omrežje, če v omrežju ni nobenega cikla.

Če začnemo pot v katerikoli točki acikličnega omrežja in sledimo smerem povezav, se v nobenem primeru ne moremo vrniti v začetno točko.

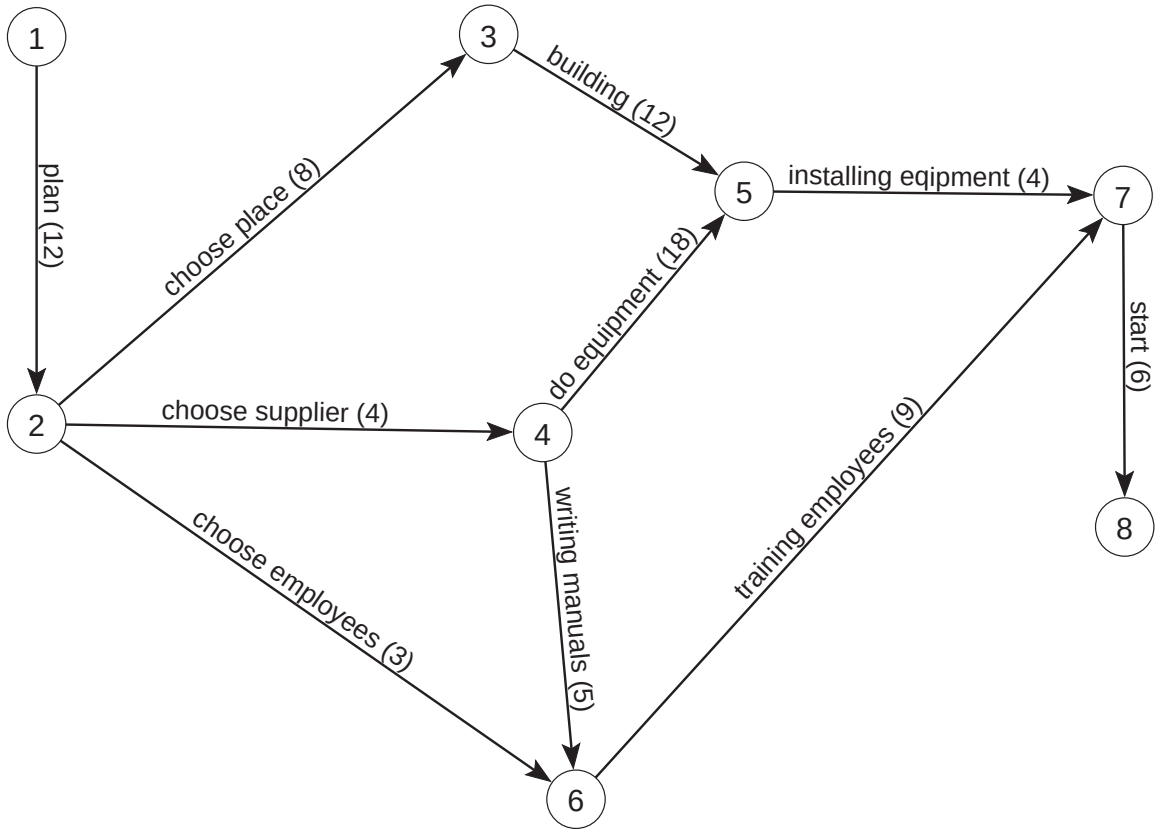
Primeri acikličnih omrežij:

Omrežje citiranj



V omrežju citiranj je zanimivo vprašanje kateri avtor je najbolj vpliven in katero citiranje je najpomembnejše. Obe oceni temeljita na štetju vseh poti, ki vodijo skozi dano točko oz. uporabljajo dano povezavo (število poti v acikličnem omrežju je končno). V Pajku se ustrezna procedure skriva pod Citation Weights.

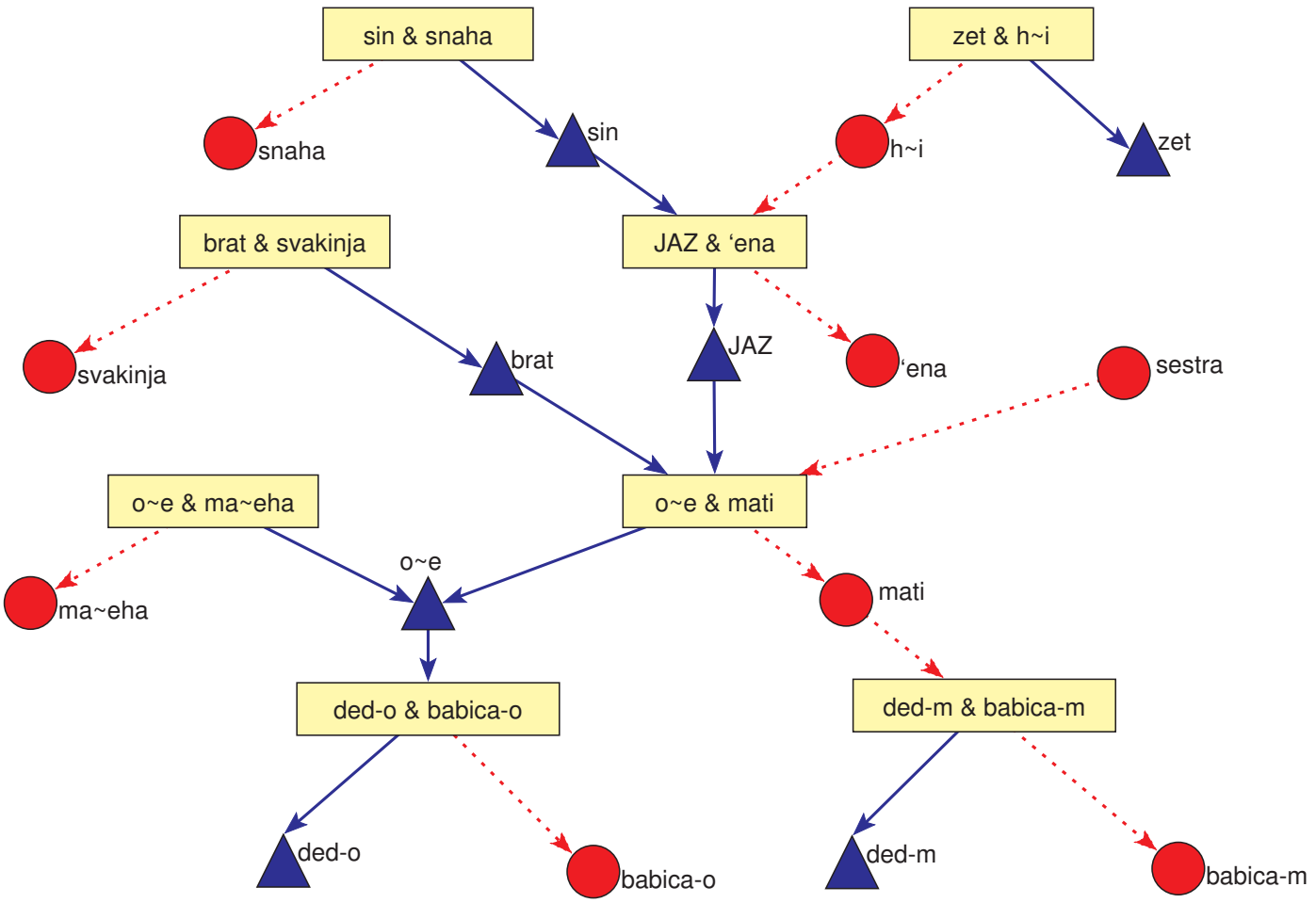
Omrežje CPM (*Critical Path Method*) iz področja mrežnega planiranja.



V omrežju CPM nas zanima *krtitična pot*, ki določa trajanje celotnega projekta.

V našem primeru je kritična pot 1 – 2 – 4 – 5 – 7 – 8, trajanje pa 44. V Pajku se ustrezna procedure skriva pod Critical Path Method.

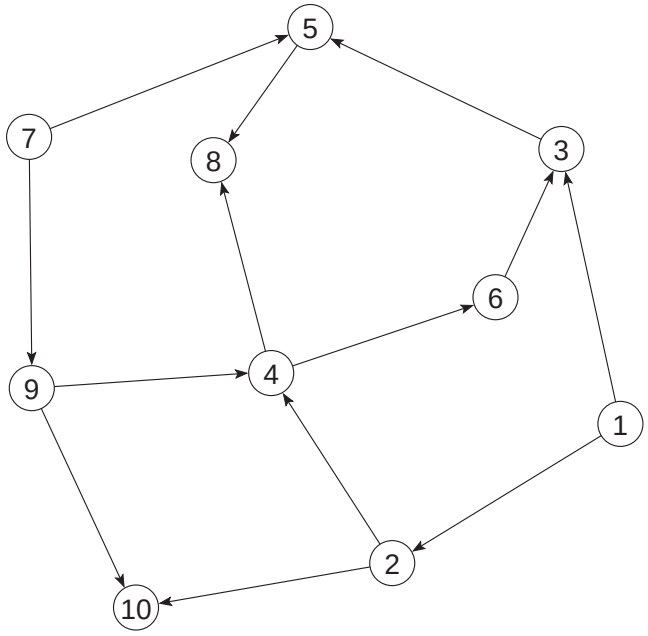
Rodovniki – genealogije



V acikličnem omrežju obstajajo ***prve točke*** – točke v katere ne vodi nobena povezava in ***zadnje točke*** – točke iz katerih ne vodi nobena povezava.

Za vsako točko acikličnega omrežja lahko izračunamo ***globino***: Začetnim točkam pripišemo globino 1, jih izločimo iz omrežja ter tako dobimo novo aciklično omrežje. Iz novodobljenega omrežja spet izločimo vse začetne točke, jim pripišemo globino 2 in postopek nadaljujemo.

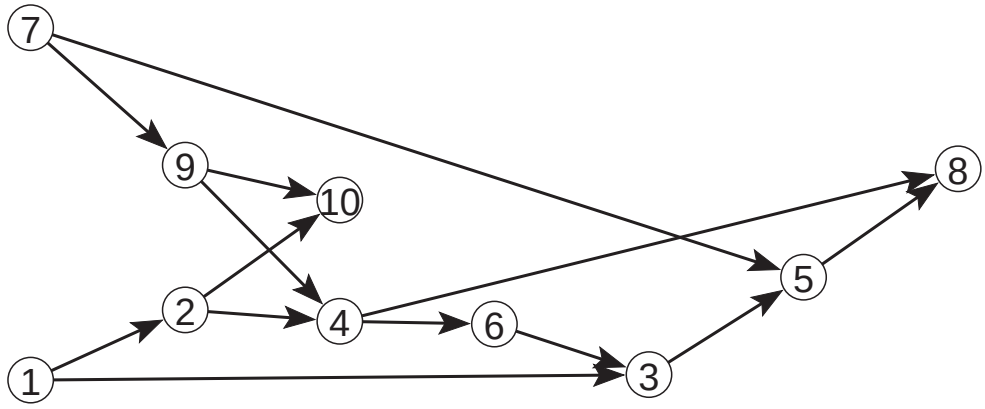
Primer: Za točke v acikličnem omrežju



so **globine**

toč.	1	2	3	4	5	6	7	8	9	10
gl.	1	2	5	3	6	4	1	7	2	3

Dobljene **globine** lahko uporabimo za **prikaz omrežja po nivojih** (v smeri x):



V Pajku izračunamo globine točk acikličnega omrežja z Net/Partitions/Depth/Acyclic

Če nato izberemo ukaz Draw/Draw-Partition, se na sliki omrežja točke pobarvajo z barvo, ki ustreza globini. Uporabljene barve lahko pogledamo z Options/Colors/Partition Colors v oknu Draw, ali na http://vlado.fmf.uni-lj.si/pub/networks/pajek/part_col.pdf

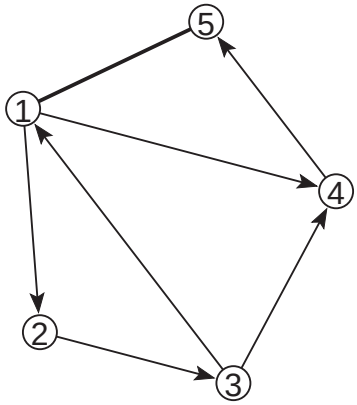
Sliko narišemo po nivojih z ukazom Layers/in y direction. Če zelimo sliko ročno popravljati, a bi pri tem radi ohranili nivojsko sliko, lahko proglasimo koordinato y za nepremično z ukazom Move/Fix y – s tem onemogočimo spremembo koordinate y katerekoli točke.

Stopnja točke

Vhodna stopnja točke **(input degree)**: število povezav, ki vstopa v točko.

Izhodna stopnja **(output degree)**: število povezav, ki izstopa iz točke.

Skupna stopnja točke **(all degree)**: skupno število povezav, ki ima eno krajišče v točki.



V omrežju so:
vhodne stopnje točk: (2, 1, 1, 2, 2)
izhodne stopnje točk: (3, 1, 2, 1, 1)
skupne stopnje točk: (4, 2, 3, 3, 2)

V programu Pajek izračunamo stopnje točk z
Net/Partitions/Degree

ter nato izberemo Input, Output ali All. Za manjša omrežja lahko pregledamo dobljene stopnje za vse točke, tako da dvakrat kliknemo na dobljeni rezultat, izberemo File/Partition/Edit ali pritisnemo na ustrezno ikono.

Če potem, ko izračunamo stopnje točk, z ukazom Draw/Draw-Partition narišemo sliko omrežja, bo barva vsake točke določena z izračunano stopnjo. Če prenesemo dobljeno razbitje po stopnjah v vektor z uporabo Partition/Make Vector, lahko s klicem Draw/Draw-Vector narišemo sliko omrežja, kjer bodo velikosti točk določene z vrednostjo v vektorju. Obe lastnosti (barvo in velikost) prikažemo na sliki z uporabo Draw/Draw-Partition-Vector.

Pri večjih omrežjih pa nas ponavadi zanima samo nekaj točk z največjimi stopnjami in frekvenčna porazdelitev stopenj, kar zahtevamo z Info/Partition. Podrobnosti o tem ukazu so opisane pri določanju k-sosedov. Razlika je le v tem, da so nas tam zanimale najbližje točke (vnesli smo negativno število), tu pa nas zanimajo točke z najvišjimi stopnjami (vnesemo pozitivno število).

Primer: V omrežju angleških besed (dic28.net) poiščimo besede, ki imajo največ sosedov.

Še nekaj ukazov

Premikanje v določenih smereh onemogočimo z ukazi Move/Fix/x, Move/Fix/y in Move/Fix/Radius.

Razporeditev točk omrežja na mreži ali koncentričnih krožnicah dosežemo z uporabo ukazov
Move/Grid in Move/Circles
Vnesemo še željeno velikost mreže ali gostoto koncentričnih krožnic. Pri premikanju točk se izbrana točka vedno postavi na najbližjo izbrano pozicijo.

Primer: mreza.net.

Pri omrežjih z vrednostmi na povezavah z izbiro Options/Values of Lines določimo ali so vrednosti na povezavah podobnosti, različnosti ali pa jih ne upoštevamo. To informacijo uporabimo pri risanju: v primeru podobnosti naj bodo točke povezane s povezavami z velikimi vrednostmi čimbliže skupaj, v primeru različnosti pa čimdlje narazen.

Primer: mreza.net.

Z ukazom Options/Interrupt določimo koliko časa naj se optimizira slika pri energijskih risanjih.

Z ukazom Options/ScrollBar On/Off postavimo drsnik v okno Draw. Drsnik lahko deluje na dva načina:

- v primeru, da je izbrana cela slika, s pomočjo drsnika vrtimo sliko
- v primeru, da je izbran del slike (Zoom), pa s pomočjo drsnika premikamo vidno okno.

Pri shranjevanju slik v oblike EPS, SVG ali VRML imamo navoljo več dodatnih nastavitev (Export/Options), s pomočjo katerih lahko natančneje opišemo izgled slike.

Opis parametrov je na voljo na:
http://vlado.fmf.uni-lj.si/pub/networks/pajek/epsdef.htm

Slike shranjene v obliki EPS prikazujemo s programom GsView.

Vaje

1. V omrežju flow2.net poiščite najkrajši poti med $v1$ in $v10$ po kriteriju dolžine poti in vrednosti poti.
   Poiščite še najkrajše verige po obeh kriterijih.
2. V omrežju besed dic28.net poiščite najkrajše poti med izbranimi besedami, npr.
   white – yellow
   engaged – skeptics
3. V omrežju flow2.net poiščite oddaljenosti vseh točk od točke $v5$.
4. V omrežju besed dic28.net poiščite oddaljenosti vseh besed od besede yellow. Izločite in narišite podomrežje z vsemi besedami (vključno z besedo yellow), ki so od besede yellow oddaljene za največ 3.
5. Narišite aciklično omrežje wirth.net (omrežje vnesite sami) po nivojih s čimmanj križanji povezav.
6. V omrežju angleških besed dic28.net poiščite besede, ki imajo največ sosedov.