Damjan Vavpotič

Predstavitev podatkov

Pojem predstavitve podatkov je precej širok in ga srečamo na zelo različnih področjih (npr. predstavitev podatkov z grafikonom, besedna predstavitev podatkov, itd.) S pojmom pa se pogosto srečamo tudi pri računalništvu in informatiki, saj je ena izmed temeljnih nalog informatike prav obdelava in predstavitev podatkov, računalniki pa so danes ključno orodje za ta namen. Načini prestavitve podatkov se razlikujejo predvsem glede na to, kakšen je namen zapisa podatkov. Tako bomo podatke v primeru, da jih želimo arhivirati, predstavili na drugačen način, kot če bomo želeli tudi poiskati določen podatek izmed množice podatkov. Z bolj naprednimi načini predstavitve podatkov lahko modeliramo tudi nekatera dejstva in spoznanja iz realnega sveta. Na primer s pomočjo predstavitve podatkov v obliki drevesa, lahko modeliramo organiziranost šole, s pomočjo predstavitve v obliki grafa pa meje med državami. Seveda pa je v računalništvu potrebno na nek način zapisati tudi programe oz. algoritme, ki tako predstavljene podatke obdelujejo. Če pojem predstavitve podatkov obravnavamo malce širše, lahko rečemo, da moramo uporabiti ustrezno predstavitev podatkov tudi za zapis programov. Gre za povsem poseben način za predstavitev podatkov – govorimo o t.i. formalnih jezikih. Formalnih jezikov pa seveda ne uporabljamo le pri računalništvu, ampak so zelo pogosto v pomoč tudi matematikom. V nadaljevanju si bomo podrobneje pogledali različna področja predstavitve podatkov. Ker pa, kot že rečeno, računalniki brez formalnih jezikov danes ne delujejo, začnimo prav z njimi.

Formalni jezik

V računalništvu (pa tudi logiki in matematiki) lahko formalni jezik opredelimo kot množico nizov simbolov (besed) omejenih s pravili. Abeceda formalnega jezika je množica simbolov, iz katerih lahko sestavimo besede (nize simbolov). Poenostavljeno bi lahko tudi rekli, da je formalni jezik množica besed nad izbrano abecedo. S pojmom formalnega jezika se še posebej pogosto srečamo pri računalništvu, saj v skupino formalnih jezikov spadajo tudi vsi programski jeziki.

S perspektive poučevanja računalniškega razmišlja za otroke so principi formalnih jezikov zanimivi, ker otrokom omogočajo, da sami sestavijo preproste jezike za katere si sproti zamislijo pravila, nato pa ugotavljajo katere besede so del takšnega jezika oz. sestavljajo pravilne besede ipd.

Za lažje razumevanje zgoraj vpeljanih pojmov si poglejmo primer preprostega formalnega jezika – poimenujmo ga jezik J1. Kot abecedo jezika J1 vzemimo naslednje simbole (brez presledkov in vejic):

X , Y

Če ne opredelimo nobenih dodatnih pravil, lahko z jezikom J1 sestavimo neskončno veliko besed. Poskusimo:

»X« , »Y«, »YYYYY«, »YYXYXXXYYYX«, »XXYYXXXY«, itd.

S pomočjo pravil pa lahko nabor veljavnih besed omejimo. Npr. naj velja, da je v J1 le vsak neprazen niz, ki vsebuje največ 2 simbola iz abecede. S tem pravilom smo nabor besed omejili tako močno, da jih je v jeziku J1 ostalo le še 6:

»X« , »Y« , »XX« , »YY« , »YX« , »XY«

Seveda hitro opazimo, da bi se število besed jezika J1 spremenilo tudi v primeru, da spremenimo abecedo.

Hitro si lahko zamislimo tudi bolj zapleten primer – recimo mu jezik J2. Kot abecedo jezika J2 uporabimo naslednje simbole (brez presledkov in vejic):

0 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , + , =

Jezik J2 je opredeljen s pravili:

a) Vsak neprazen niz, ki ne vsebuje simbola »+« ali »=« in se ne prične z »0« je del jezika J2.
b) Niz »0« je v jeziku J2.
c) Niz, ki vsebuje »=« je v jeziku J2 izključno v primeru, da je v nizu en sam »=« in »=« ločuje dva veljavna niza iz jezika J2.
d) Niz, ki vsebuje »+« in hkrati ne vsebuje »=« je v jeziku J2 izključno v primeru, da vsak »+« v nizu ločuje dva veljavna niza iz jezika J2.
e) Noben niz razen tistih, ki jih definirajo zgornja pravila ni v jeziku J2.

Ker so pravila nekoliko bolj zapletena, jih bodo otroci težje pravilno interpretirali. Po drugi strani pa gre za matematične izraze, ki so jim blizu že od prej. Vprašamo jih npr:

- Ali je niz »553+« del jezika J2? Zakaj ne? Odgovor: Ker »+« ne ločuje dveh veljavnih nizov iz J2 (pravilo d).
- In niz »2+2=4«? Seveda! Niz je skladen z vsemi pravili.
- V prejšnjem primeru bodo otroci preverjali tudi semantično pravilnost. Zato jih lahko presenetimo z nizom »1+1=4«. Če preverimo pravila, ugotovimo, da je niz skladen torej ustreza jeziku J2. Opazimo, da je s pravili definirana le sintaksa jezika (torej pravilen način zapisa), ni pa definirana semantika (pomen posameznih simbolov). Zato je del jezika J2 tudi niz »1+1=4«.

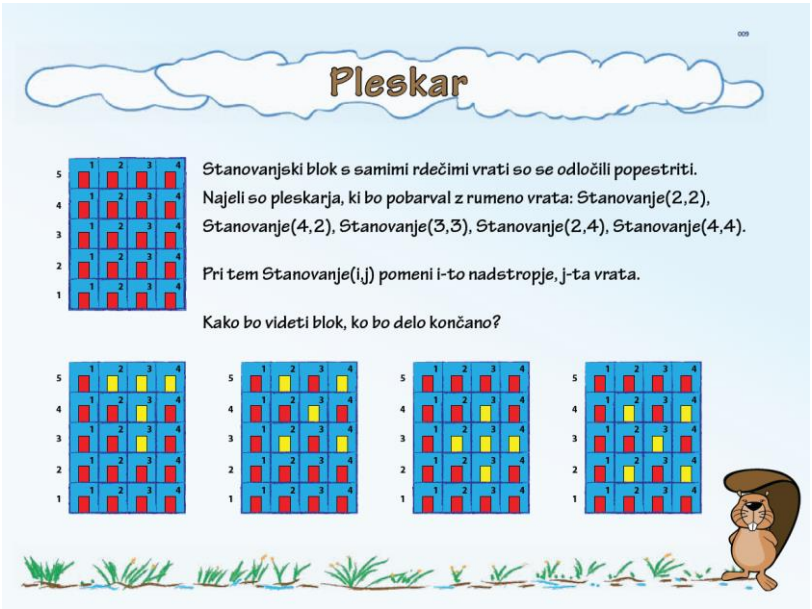
Naloge v okviru bobra se dotikajo področja na nekoliko manj formalen način. Poglejmo si primera dveh nalog.

Bober - Bim, Bam

Komentar:

Pri nalogi morajo otroci zapisati melodijo bim, bam v formalnem jeziku. Čeprav so pravila jezika podana relativno ohlapno, jim je v pomoč primer ding, dong po katerem se zgledujejo in pridejo do ustreznega zapisa.

Bober – Pleskar



Komentar:

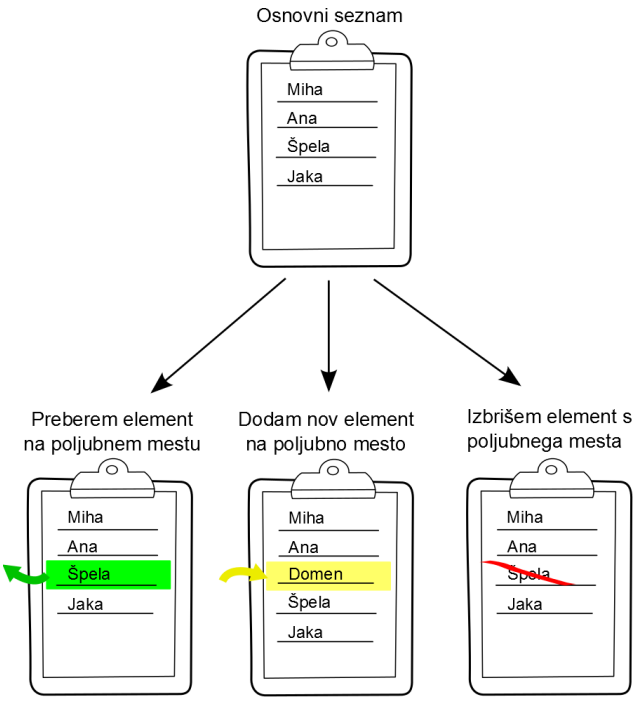
Naloga uporablja formalni jezik za zapis pobarvanih vrat. V tem primeru morajo biti otroci sposobni branja že podanega zapisa v preprostem formalnem jeziku.

Podatkovne strukture in abstraktni podatkovni tipi

V prejšnjem poglavju smo videli, da lahko s pomočjo formalnega jezika med drugim predpišemo/opišemo način zapisa oz. predstavitve podatkov. Za potrebe shranjevanja in organizacije podatkov pa načinov zapisa navadno ne razvijamo povsem na novo, ampak uporabljamo različne tipične vrste podatkovnih struktur, ki podpirajo dostop do podatkov in njihovo spreminjanje. Ker nobena vrsta podatkovnih struktur ni idealna za vse namene, je pomembno poznati njihove prednosti in slabosti. Podatkovne strukture (npr. polja, zapise, datoteke, množice) dobimo z združevanjem osnovnih podatkovnih tipov. Takšne strukture podedujejo možne operacije, ki so opredeljene za osnovne podatkovne tipe, ki jih sestavljajo. To sicer programerju omogoča neovirano spreminjanje vrednosti posameznih elementov podatkovne strukture, vendar pa hkrati predstavlja veliko nevarnost pojava napak. Zato se pogosto izkaže, da je smiselno opredeliti oz. omejiti tudi vse dovoljene operacije nad elementi podatkovne strukture. V tem primeru začnemo lahko govoriti o abstraktnih podatkovnih tipih, ki hkrati predstavljajo tudi posplošitev konkretnih podatkovnih struktur. V nadaljevanju se bomo nekaterim abstraktnim podatkovnim tipom posvetili še posebej podrobno, saj se pogosto pojavljajo tudi v okviru Bobra.

Seznam

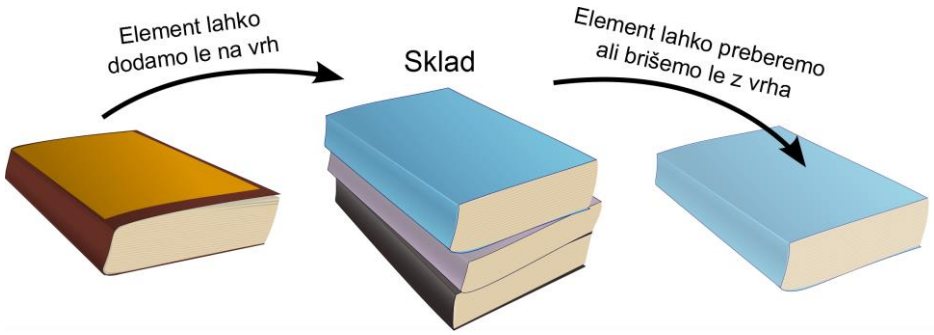
Najprej si poglejmo enega izmed najosnovnejših in hkrati najpreprostejših abstraktnih podatkovnih tipov – seznam. Seznam je zaporedje 0 ali več elementov pri katerem je vrstni red elementov pomemben, možno pa je tudi, da se elementi v seznamu ponavljajo. Operacije, ki so definirane nad seznamom nam omogočajo, da preberemo element iz poljubnega mesta na seznamu, zapišemo nov element na poljubno mesto v seznamu ali zbrišemo element na poljubnem mestu v seznamu (Slika 1).



Slika 1: Seznam

Skladi

Zdaj, ko že poznamo seznam, nam sklad ne bo delal posebnih težav, saj gre le za posebno, nekoliko omejeno vrsto seznama. Pri skladu se elementi lahko berejo, dodajajo ali brišejo vedno le na začetek seznama oz. na vrh sklada. Za lažjo predstavo si zamislimo sklad knjig (Slika 2), kjer je vedno dosegljiva le prva, t.j. tista, ki je najvišje, hkrati pa novo knjigo lahko dodamo le na vrh sklada. Taki podatkovni strukturi v angleščini pravimo tudi LIFO (last-in-first-out).



Slika 2: Sklad

Poleg skladov seveda obstaja še vrsta drugih abstraktnih podatkovnih tipov. Eden zanimivejših je gotovo slovar, ki si ga bomo pogledali v sklopu naslednjega poglavja o drevesih.

Poglejmo si primer naloge iz Bobra, ki se dotika področja skladov.

Bober – Skladi krožnikov

Komentar:

Naloga na preprost način predstavi osnovno zakonitost delovanja sklada t.j., da je vedno dosegljiv le zgornji element sklada. Ker otroci tudi iz izkušenj vedo, da je krožnik iz sredine sklada težko izvleči ne da bi se kaj razbilo, naloga deluje precej intuitivno. Seveda pa lahko nalogo pogledamo tudi iz stališča vrste bobrov, ki jo je mogoče predstaviti z abstraktnim podatkovnim tipom vrsta. V tem primeru lahko ugotavljamo kakšna je razlika med strukturo LIFO (last-in-first-out) – sklad krožnikov in FIFO (first-in-first-out) – vrsta bobrov. Otroke lahko k razmišljanju pripravimo tudi z vprašanjem, ali bi bili zadovoljni, če bi na kosilo čakali skladno s pravili strukture LIFO?

Drevesa

Drevo je abstraktna struktura, ki je sestavljena iz točk oz. vozlišč (angl. *node*) in povezav med vozlišči oz. vej (angl. Branches). Vozlišča, ki nimajo podrejenih vozlišč (oz. otrok) se imenujejo listi (angl. leaf nodes). Vsako končno drevo ima tudi eno vozlišče, ki nima nadrejenih elementov (oz. staršev). Ta element se imenuje koren oz. korensko vozlišče (angl. root node). Posebnost drevesne strukture je, da do od vsake točke do vsake druge točke vodi le ena pot.

Drevesna struktura je tudi grafični način za predstavitev hierarhije. Z drevesnimi strukturami lahko predstavimo različne naravne ali umetne hierarhije. Npr. družinska drevesa, drevo živalskih vrst, drevo razvoja naravnih jezikov, ureditev spletne strani, struktura datotečnega sistema, struktura zaznamkov v brskalniku, itd.

Drevesne strukture pa imajo zelo pomembno vlogo tudi kot posebna oblika podatkovnih struktur. Primer abstraktnega podatkovnega tipa, ki ga pogosto realiziramo s pomočjo

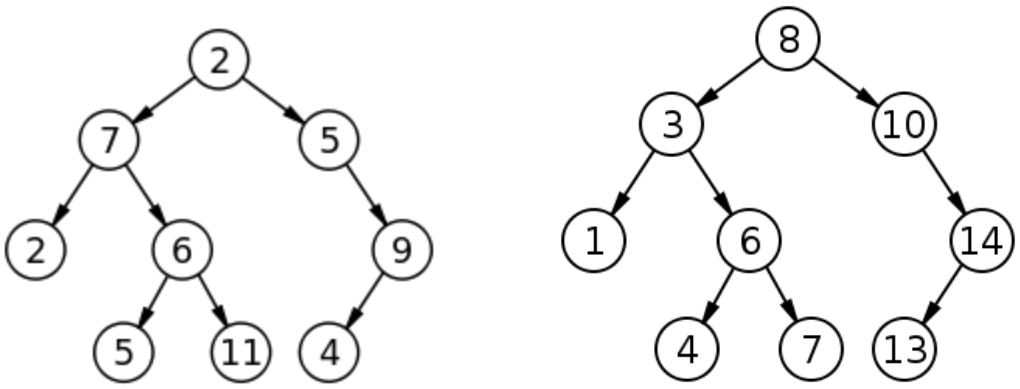
drevesa je slovar. Slovar omogoča samo tri osnovne operacije: vstavljanje, brisanje in iskanje elementa v množici. Slovar je abstraktni podatkovni tip pri katerem je pomembno čim hitrejše iskanje, brisanje in dodajanje elementov. Različne drevesne strukture, omogočajo učinkovito iskanje, brisanje in dodajanje elementov, pa tudi učinkovito iskanje minimalnega in maksimalnega elementa ter iskanje predhodnika in naslednika danega elementa. Kadar s uporabo dreves želimo optimizirati čas iskanja v slovarju, nas začne zanimati poseben podtip dreves t.i. iskalna drevesa.

Preprosta drevesna struktura je dvojiško ali binarno drevo. Za to drevo je značilno, da ima vsako vozlišče največ dva otroka. Z dvojiškim drevesom lahko npr. predstavimo vse prednike neke osebe.

V računalništvu je še zlasti zanimivo binarno iskalno drevo, ki poleg lastnosti binarnega drevesa zadošča tudi naslednjim pogojem:

- levo poddrevo nekega vozlišča vsebuje le elemente z vrednostmi, ki so manjše kot je vrednost tega vozlišča,
- desno poddrevo nekega vozlišča vsebuje le elemente z vrednostmi, ki so večje kot je vrednost tega vozlišča,
- tudi levo in desno poddrevo sta binarni iskalni drevesi,
- vozlišča se ne ponavljajo.

Ključna lastnost binarnega iskalnega drevesa je, da omogoča iskanje, vstavljanje in brisanje elementov s povprečno časovno zahtevnostjo log N, pri čemer je N število vseh vozlišč (ob pogoju, da je drevo poravnano). Na spodnji sliki vidimo preprost primer binarnega iskalnega drevesa.



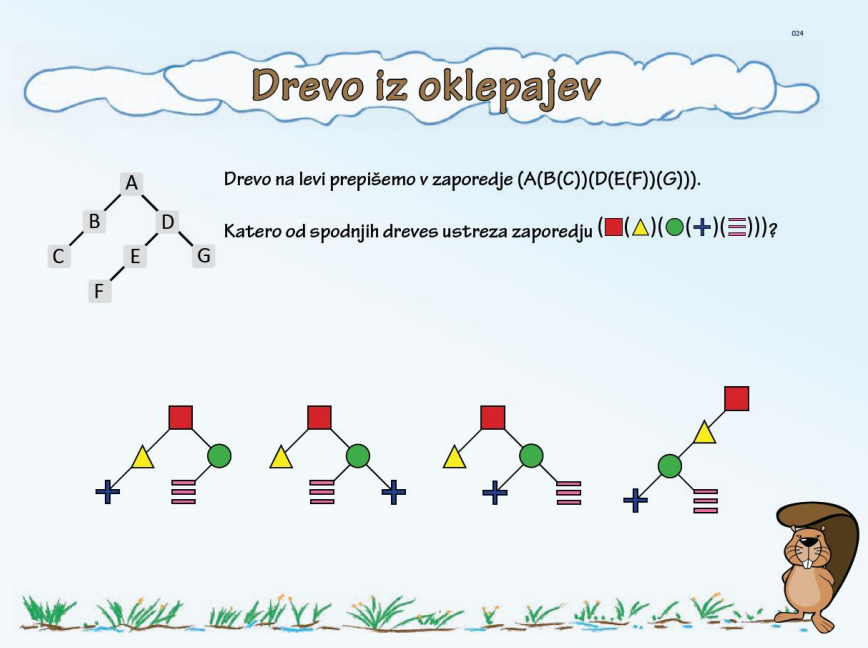
Slika 3: Binarno drevo in binarno iskalno drevo

Kot smo že omenili mora biti iskalno drevo vsaj približno poravnano, da lahko po njem učinkovito iščemo. Seveda pa se lahko v najslabšem možnem primeru binarno iskalno drevo izrodi v seznam, če npr. vanj vstavljamo elemente, ki so že urejeni po vrsti. V tem primeru je časovna zahtevnost reda N, torej enaka kot pri seznamih. Da do takšne situacije ne bi prišlo so bile izdelane še druge oblike iskalnih dreves, ki so delno in popolno poravnana (npr. lomljena drevesa, rdeče-črna drevesa, AVL-drevesa, 2-3 drevesa, B-drevesa, itd.).

Velja omeniti, da lahko drevesa obravnavamo tudi kot posebno vrsto usmerjenega grafa. O tem bomo nekaj več povedali proti koncu poglavja o Grafih.

V okviru Bobra so naloge, ki se dotikajo področja dreves kar pogoste, res pa je, da imamo navadno opravka z nekoliko bolj enostavnimi primerki dreves.

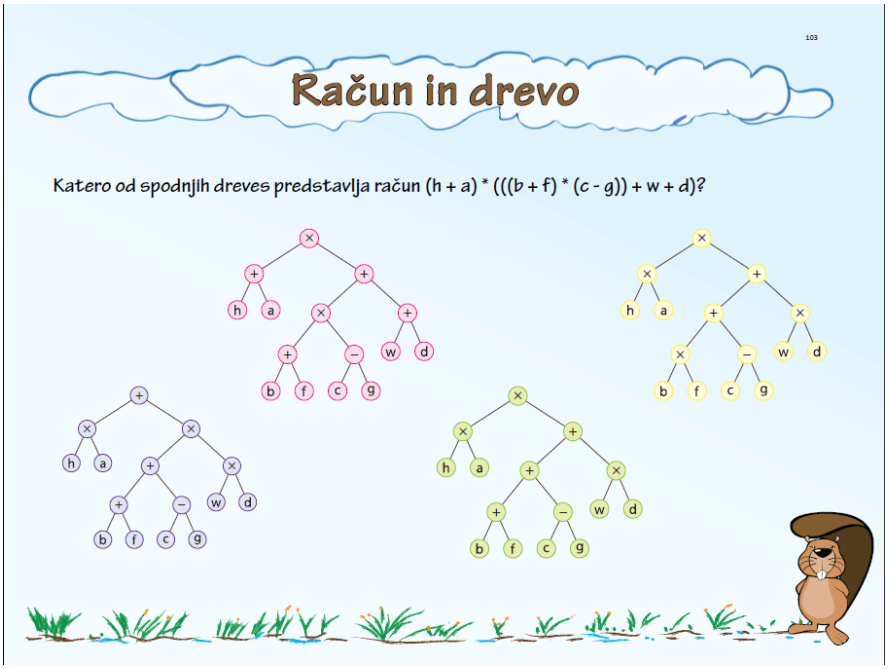
Bober – Drevo iz oklepajev



Komentar:

Drevesa je mogoče predstaviti tudi z drugačnim zapisom. Otroke lahko ob tem spomnimo tudi na formalne jezike.

Bober – Račun in drevo

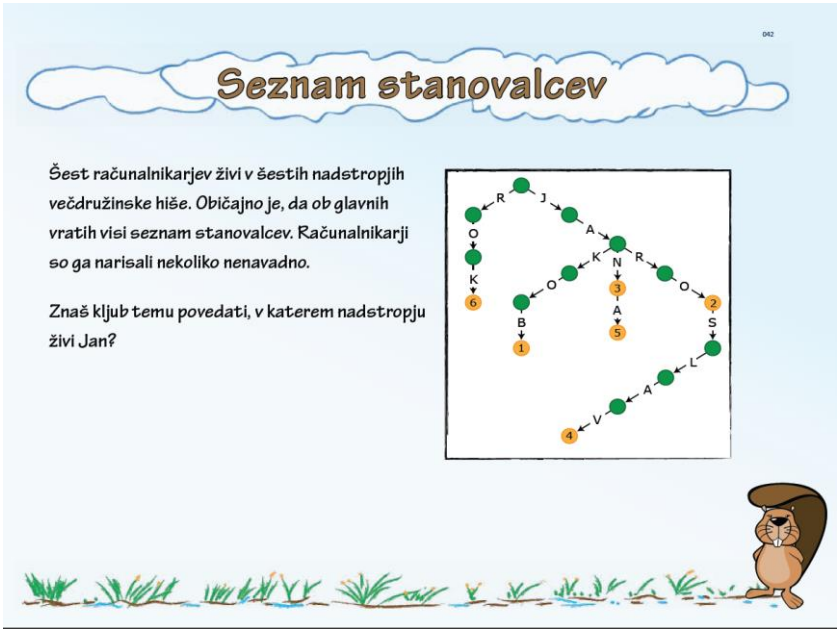


Komentar: Naloga je podobna nalogi Drevo iz oklepajev, le da je na nekoliko višji ravni.

Bober - Družinsko drevo

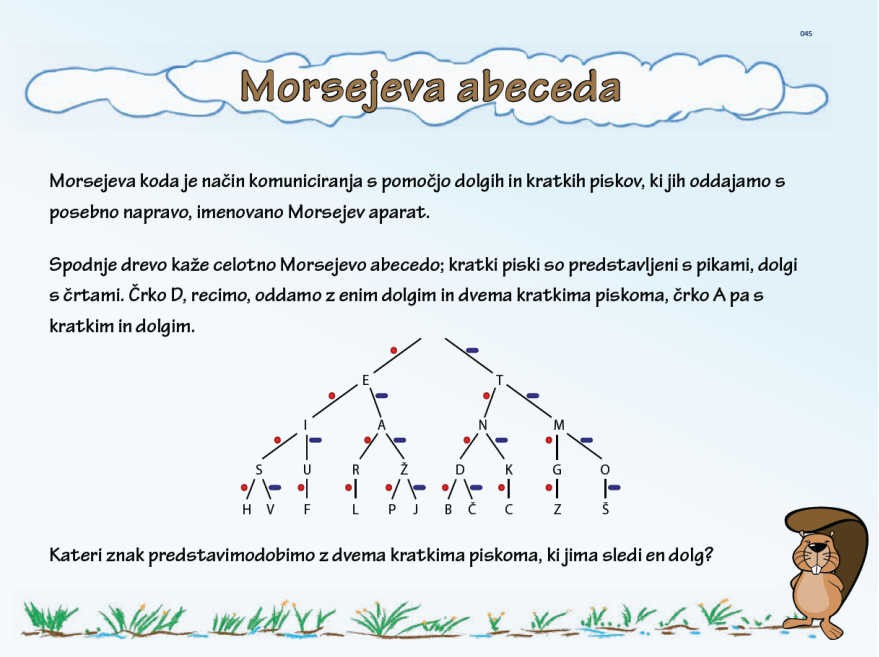
Komentar: Otroke lahko vprašamo, ali za zapis družinskega drevesa zadošča binarno drevo kjer ima lahko vsako vozlišče največ dva otroka. Kaj pa za prikaz drevesa prednikov?

Bober – seznam stanovalcev



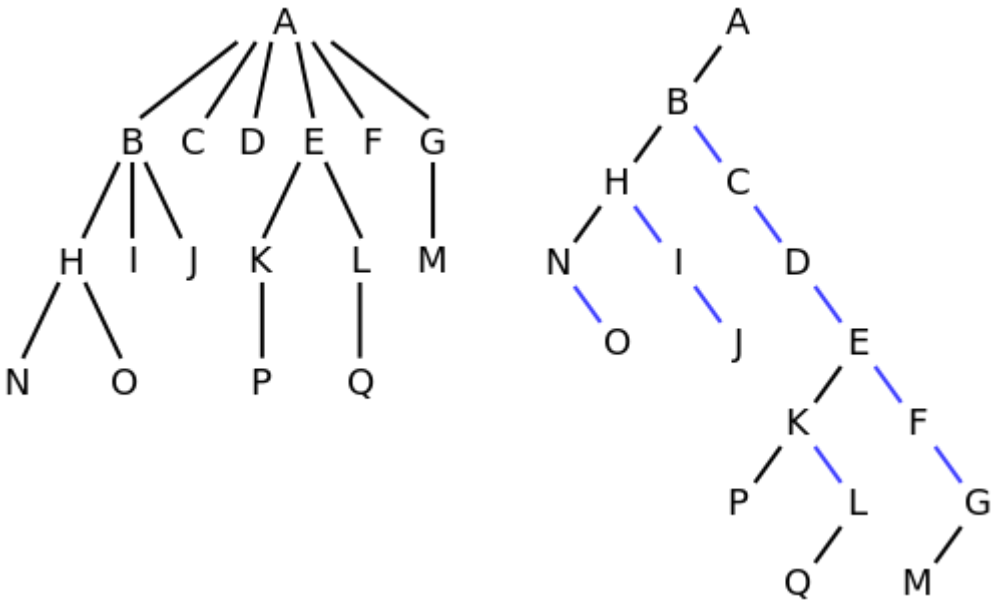
Komentar: Pri nalogi gre za nekoliko posebno drevo, saj na podlagi imen povezav prehajamo med vozlišči vse dokler ne najdemo ustreznega nadstropja. Drevo se s tem nekoliko približa ideji odločitvenih dreves (nadaljujemo po poti na kateri je prava črka), čeprav ne gre za odločitveno drevo, saj nimamo končnih dogodkov. Hkrati pa se naloga dotika tudi ideje iskalnih dreves (poiščemo nadstropje), čeprav pa hitro opazimo, da drevo ne sledi pravilom zasnove iskalnih dreves.

Bober – Morsejeva abeceda



Komentar: Naloga je zelo podobna nalogi Seznam stanovalcev, le da imajo vsa vozlišča svojo vrednost. Res pa je, da je način razmišljanja pri reševanju naloge obrnjen, saj najprej poiščemo črko – vozlišče, nato pa poiščemo pot od korena do vozlišča ter zapisujemo simbole.

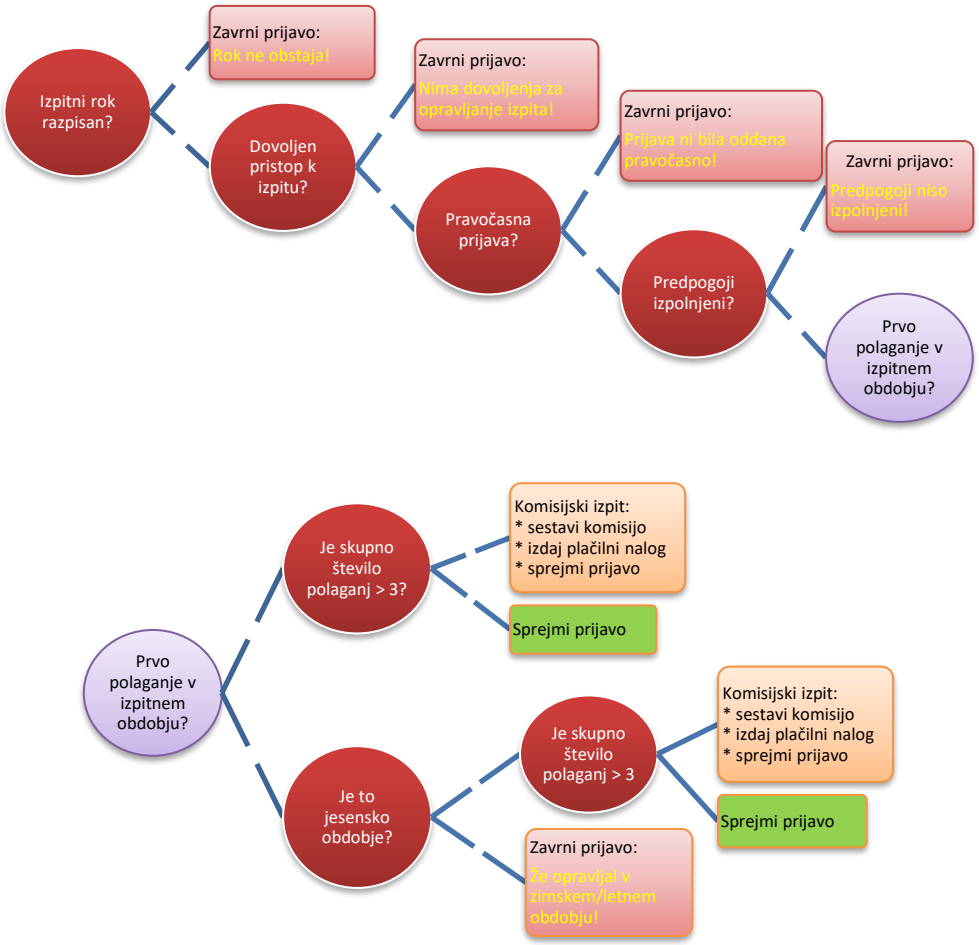
Na koncu kot zanimivost omenimo še, da lahko vsako urejeno drevo zapišemo tudi kot binarno drevo. Postopek pretvorbe na preprost način opišemo takole: *vse druge in nadaljnje potomce nekega vozlišča zaporedno povežemo s prvim potomcem* (glej modre povezave). Če dobro opazujemo na nek način »zvrnemo« vse povezave med predniki in potomci razen povezave med prednikom in prvim potomcem, ki se ohrani.



Slika 4: Urejeno drevo lahko zapišemo tudi kot binarno

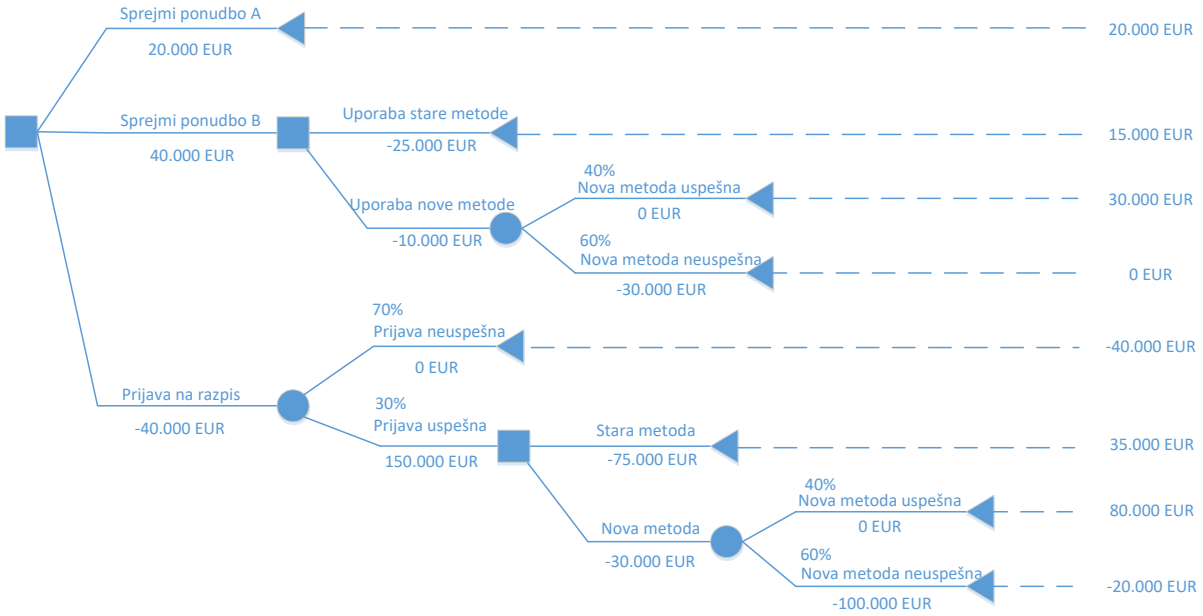
Odločitvena drevesa

Povsem posebna oblika dreves so odločitvena drevesa. Čeprav imajo podobno zgradbo kot običajna drevesa pa je njihov namen drugačen. Uporabimo jih namreč kot pomoč pri odločanju oz. pri iskanju strategije, ki nas bo najverjetneje pripeljala do želenega cilja. Uporabna so zlasti pri primerjavi različnih možnih odločitev in nam omogočajo, da razčlenimo kompleksen odločitveni problem, pri čemer lahko upoštevamo tudi različne verjetnosti dogodkov. Odločitveno drevo sestavljajo 3 različne vrste vozlišč: odločitvena vozlišča, dogodkovna vozlišča in končna vozlišča. Za odločitvenimi vozlišči sledijo povezave, ki modelirajo različne možne alternative, za dogodkovnimi pa povezave, ki modelirajo različne izide in njihove verjetnosti. Končna vozlišča ponazarjajo posledice odločitev. V primeru, da dogodkovnih vozlišč ne potrebujemo jih pri snovanju odločitvenega drevesa lahko izpustimo. Takšna odločitvena drevesa se pojavljajo tudi na Bobru. Slika 5 prikazuje primer odločitvenega drevesa brez dogodkovnih vozlišč.



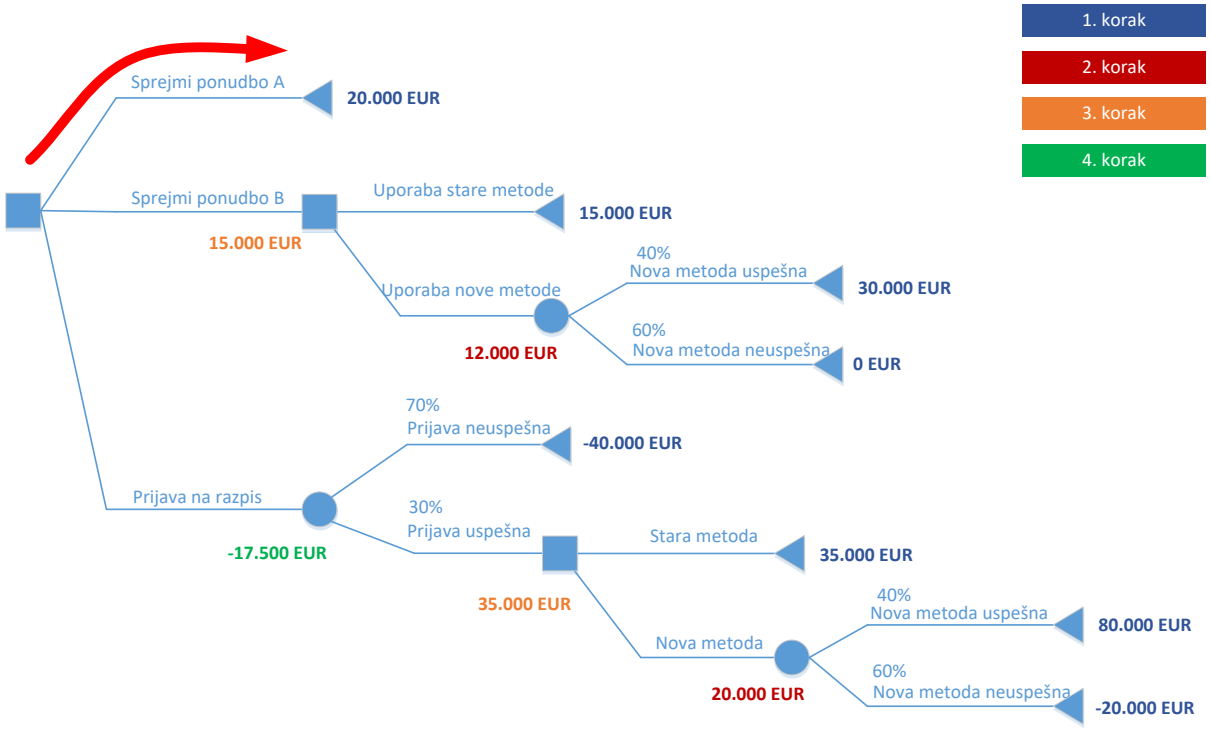
Slika 5: Primer odločitvenega modela brez dogodkovnih vozlišč

Slika 6 prikazuje še primer nekoliko kompleksnejšega odločitvenega drevesa, ki vključuje tudi dogodkovna vozlišča. S tem drevesom si podjetje pomaga pri sprejemanju odločitve ali se bo prijavilo na razpis za nekoliko večji projekt, ali pa bo sprejelo katero od dveh manjših ponudb. Pri tem upošteva tudi različne verjetnosti posameznih dogodkov. S kvadratom so označena odločitvena vozlišča, s krogom dogodkovna, s trikotnikom pa zaključna vozlišča. Pri dogodkovnih vozliščih je z vrednostjo v odstotkih označena verjetnost nastopa posameznega dogodka.



Slika 6: Primer kompleksnejšega odločitvenega drevesa

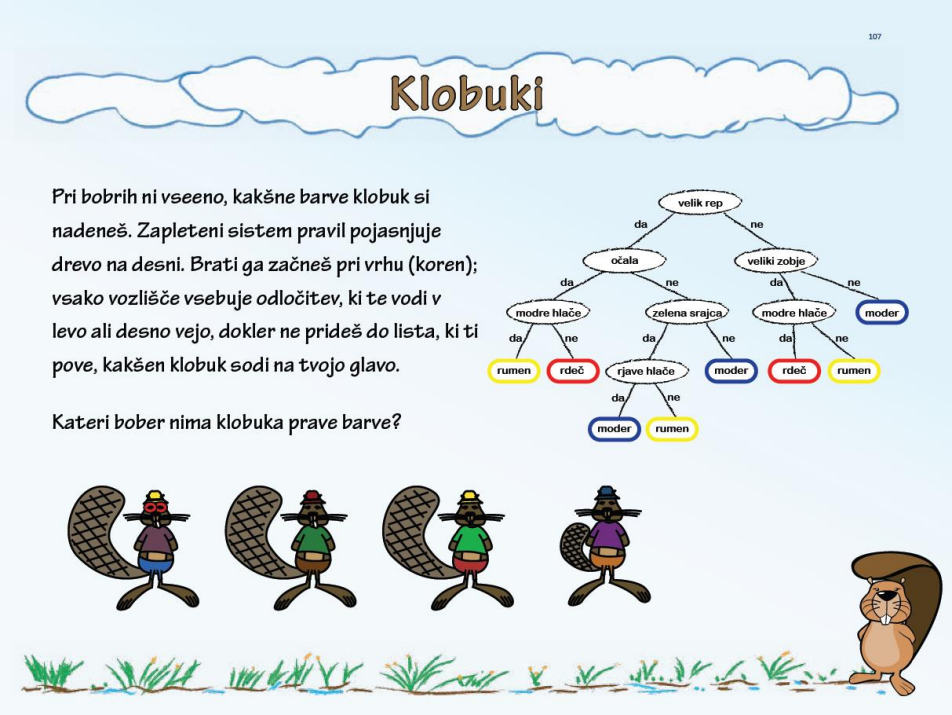
Iskanje optimalne strategije poteka tako, da po drevesu prehajamo v obratni smeri (od terminalnih vozlišč proti začetnemu vozlišču). Pri dogodkovnih vozliščih z upoštevanjem verjetnosti dogodkov preračunamo verjeten izid posameznega vozlišča ter ga vnesemo v drevo. V primeru odločitvenih vozlišč pa izberemo odločitev z najvišjim izidom in ga prav tako vnesemo v drevo. Na koncu optimalno strategijo določimo kot pot z najvišjimi vmesnimi izidi. Postopek prikazuje Slika 7.



Slika 7: Primer določitve optimalne strategije

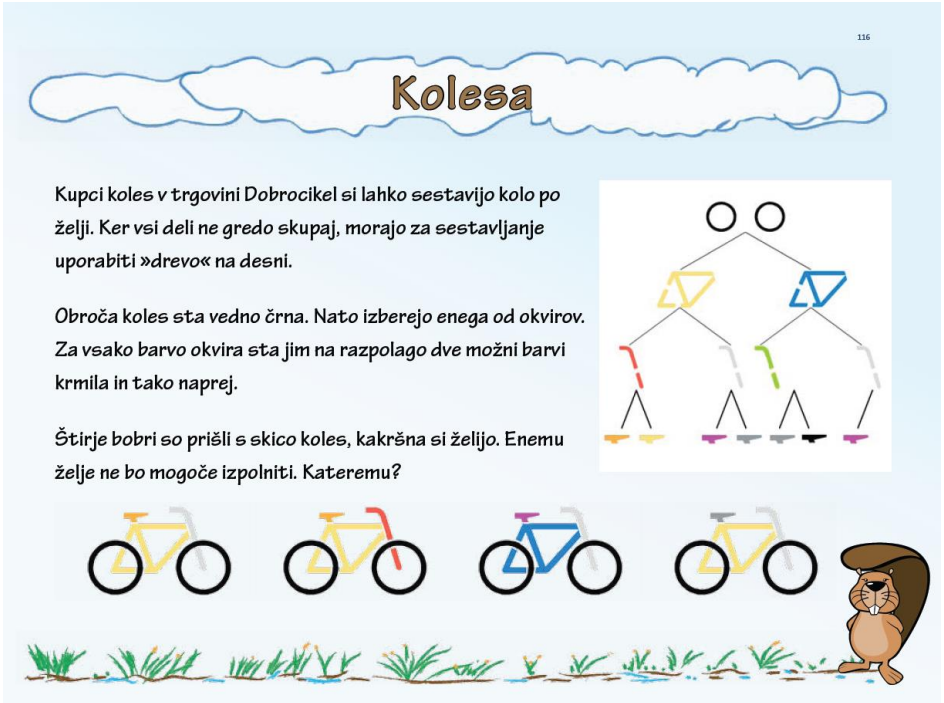
V nadaljevanju si poglejmo nekaj nalog s tekmovanja Bober, ki uporabljajo nekoliko preprostejša odločitvena drevesa.

Bober – Klobuki



Komentar: Naloga prikazuje odločitveno drevo, ki ne uporablja dogodkovnih vozlišč. Na podlagi različnih lastnosti (velikost repa, nošenje očal, velikost zob in barve različnih delov obleke) otroci ugotovijo katera barva klobuka je ustrezna.

Bober - Kolesa



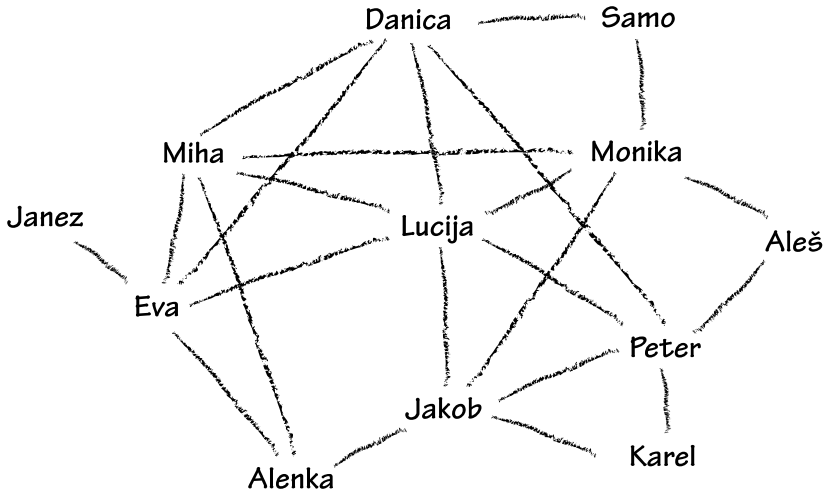
Komentar: gre za nekoliko prirejeno odločitveno drevo, saj odločitvena vozlišča hkrati predstavljajo tudi možne alternative.

Grafi

Graf je abstraktna matematična struktura, sestavljena iz "točk" (včasih jim bomo rekli tudi vozlišča; angl. *vertex* ali *node*) in "povezav" (angl. *edge*) med točkami. Z njo lahko na formalen način predstavimo najrazličnejše probleme. Točke lahko predstavljajo, na primer, osebe, povezave pa relacijo med osebami; dve osebi bosta povezani, če se poznata, sta se že rokovali na neki slavnostni večerji, imata enak vsaj en kos obleke, nimata nobenega enakega kosa obleke, sta preživeli počitnice v isti državi, govorita vsaj en skupni jezik, sta se kdaj peljali z istim vlakom, nista bili nikoli skupaj na gledališki predstavi, imata njuna vrtička skupno mejo... Točke so lahko hiše in dve hiši sta povezani, če se skozi kako okno ene hiše vidi drugo. Točke so lahko križišča in povezave ulice med njimi; ali pa kraji, povezave pa ceste med njimi. Točke so lahko avtobusne postaje in dve postaji sta povezani, če med njima vozi kak avtobus (ki med tema postajama nima vmesnih postaj). Ali pa otoki in dva otoka sta povezana, če med njima vozi neposredna trajektna povezava.

Točka je lahko povezana tudi sama s sabo, kadar je to smiselno.

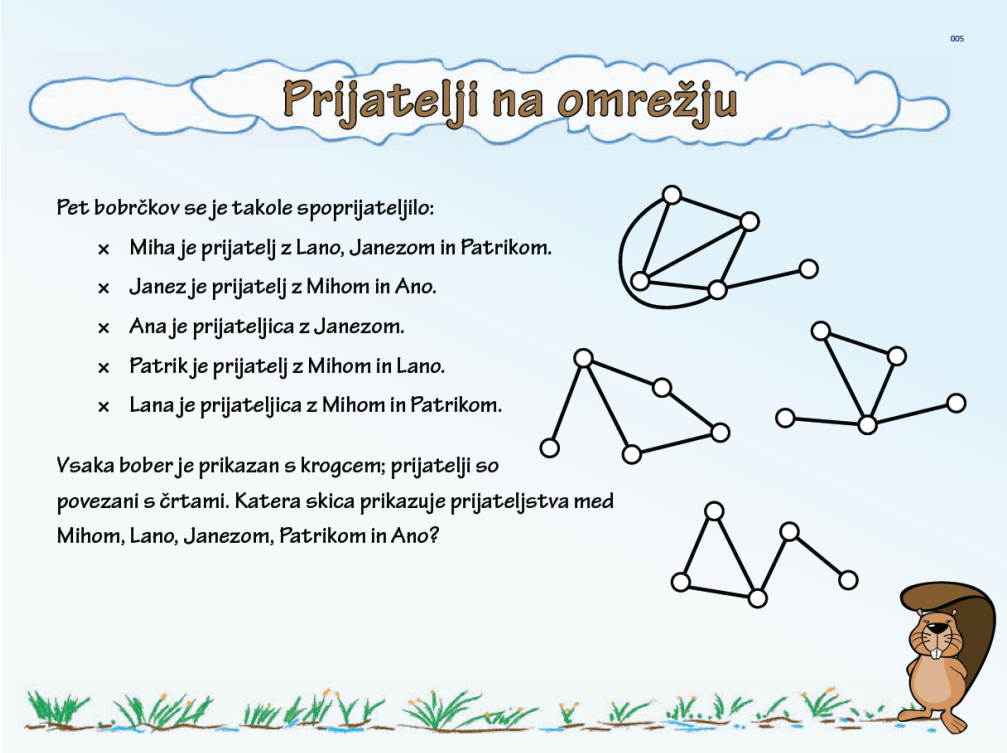
Ker vsaka točka so ustreza nekemu objektu, so točke navadno poimenovane ali kako drugače označene (Slika 8). Točkam so lahko prirejeni tudi kaki drugi podatki. Poleg imena ali oznake imajo lahko, recimo, težo, na primer velikost vrtička ali število prebivalcev kraja.



Slika 8. Primer označenega grafa: sociogram. (Vir: Bober.)

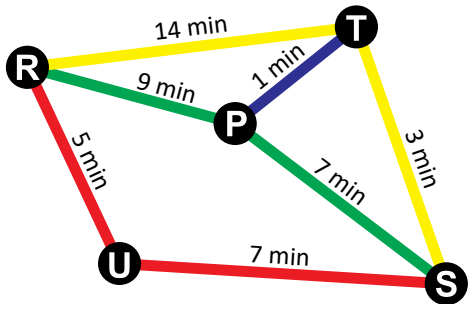
Označen graf – sociogram srečamo tudi med nalogami na Bobru.

Bober – Prijatelji na omrežju



Komentar: Naloga prikazuje več različnih grafov (sociogramov) iz katerih so bila odstranjena imena vozlišč t.j. imena prijateljev. Otroci morajo na podlagi zgradbe grafa sami ugotoviti, kateri graf prikazuje podano situacijo.

Tudi povezave imajo lahko oznake in druge podatke. Povezavam je pogosto prirejena številka, kot recimo dolžina poti med krajema, pogostost avtobusnih povezav med postajama, dolžina meje med vrtičkoma ali število voženj z vlakom, ko sta se dve osebi peljali skupaj. Primer kaže Slika 9.



Slika 9. Graf z označenimi povezavami; barva povezave pomeni avtobusno progo, številke pa povedo čas vožnje. (Vir: Bober.)

Graf je lahko usmerjen (*directed*) ali neusmerjen (*undirected*). V grafu, ki pove, ali se neka hiša vidi skozi okno druge, smer povezave pove, katero hišo vidimo iz katere. Usmerjene povezave rišemo kot puščice. Povezave, ki povedo, da dva osebi govorita skupni jezik, bodo neusmerjene.

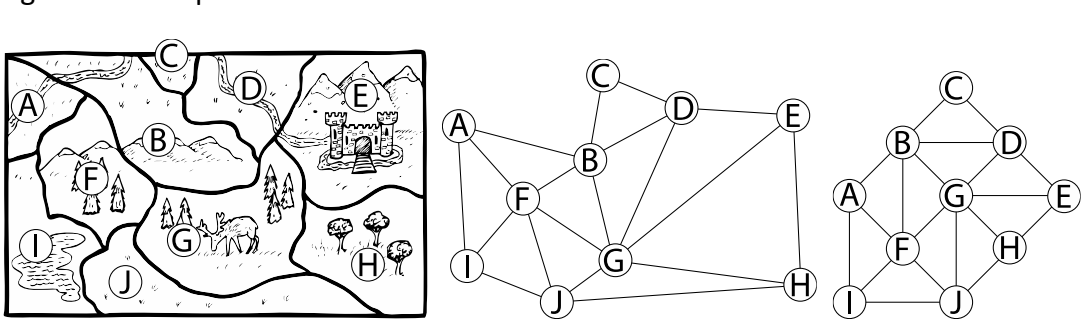
Na Bobru srečamo tudi naloge, ki se dotikajo usmerjenih grafov.

Bober - Pogrinjki

Komentar: Na podlagi sledenja vozlišč po usmerjenih povezavah grafa bodo otroci ugotovili katera miza je pripravljena skladno s pravili grafa.

Med parom točk imamo lahko eno ali več povezav. Če želimo povedati, da med dvema krajema vodi več cest, to pokažemo z več povezavami med njima. Večkratne povezave v neusmerjenih grafih sicer niso običajne, pač pa jih pogosto srečamo v usmerjenih grafih, kadar relacija med parom točk velja v obe smeri.

Ko rišemo grafe, se ne oziramo na postavitev točk: dva grafa, ki imata iste točke in povezave med njimi, sta enaka ne glede na to, kako razpostavimo točke in kje vlečemo povezave (Slika 10). Tudi, kadar graf predstavlja objekte, ki so fizično razporejeni, recimo, na zemljevidu, jih lahko rišemo v skladu s to razporeditvijo ali pa tudi ne. Pri risanju grafov gledamo predvsem na preglednosti in uporabnost.



Slika 10. Zemljevid (levo), njegova predstavitev v obliki grafa, kjer so povezane pokrajine s skupno mejo (na sredi) in pregledneje narisan isti graf (desno). (Vir: CS Unplugged in Vidra.)

Grafi so praktični, ker z njimi prevedemo najrazličnejše probleme na isti splošni, abstraktni problem. Kot primer vzemimo graf otokov in trajektov med njimi in graf oseb, v katerem povežemo tiste, ki govorijo kak skupni jezik. Tipičen problem, ki nas zanima v prvem primeru, je, ali je mogoče z določenega otoka priti na nek drug otok in kako to storiti s čim manj vožnjami ali v čim krajšem času. V drugem primeru bi se rada določena oseba pogovarjala z neko drugo, čeprav morda ne govorita nobenega skupnega jezika; zanima nas, ali obstaja "veriga" prevajalcev med njima in kako jih izbrati, da bo čim krajša. Opazimo, da gre, ko nalogo prevedemo v jezik grafov, pravzaprav za en in isti problem.

Matematiki opazujejo predvsem lastnosti grafov. V gornjem primeru vprašanje "ali obstaja?" sprašuje o lastnosti, ki ji pravimo povezanost: V računalništvu nas zanimajo tudi algoritmi na grafih. Gornji vprašanji "kako?" sprašujeta, ko ju povemo v jeziku grafov, po najkrajši poti med dvema točkama v grafu, zato ju rešujemo z enakim postopkom.

Lastnosti grafov

Matematično graf definiramo z dvema množicama: množico točk V in množico povezav E; množica povezav je podmnožica kartezičnega produkta V×V. V strogo formalno, matematične zapise v tem besedilu sicer ne bomo silili, kjer bo slovenščina dovolj jasno in nedvoumno opravila svoje delo.

Številu povezav, ki jih ima točka, rečemo stopnja točke (*vertex degree*).

Podgraf je poljubna podmnožica točk in vse povezave med njimi. Če si v grafu, ki kaže avtobusne povezave po vsej Sloveniji, izberemo le eno pokrajino ali mesto, dobimo podgraf.

Povezave, poti in obhodi

Pot (angl. *path*) med dvema točkama je zaporedje povezav, ki nas pripelje od ene točke do druge. Če so povezave usmerjene, moramo pri razmišljanju o poteh paziti tudi na smeri povezav: ko gremo od ene točke do druge, morajo biti vse povezave obrnjene v smer potovanja. Povezave in točke se lahko tudi ponavljajo; v isto točko ali po isti povezavi smemo iti večkrat.

Pri poteh nas pogosto zanima njihova dolžina. To lahko definiramo kot število povezav, iz katerih je sestavljena. Če gremo iz Ljubljane na Jesenice in od ondod v Kranjsko goro, smo napravili pot dolžine 2.

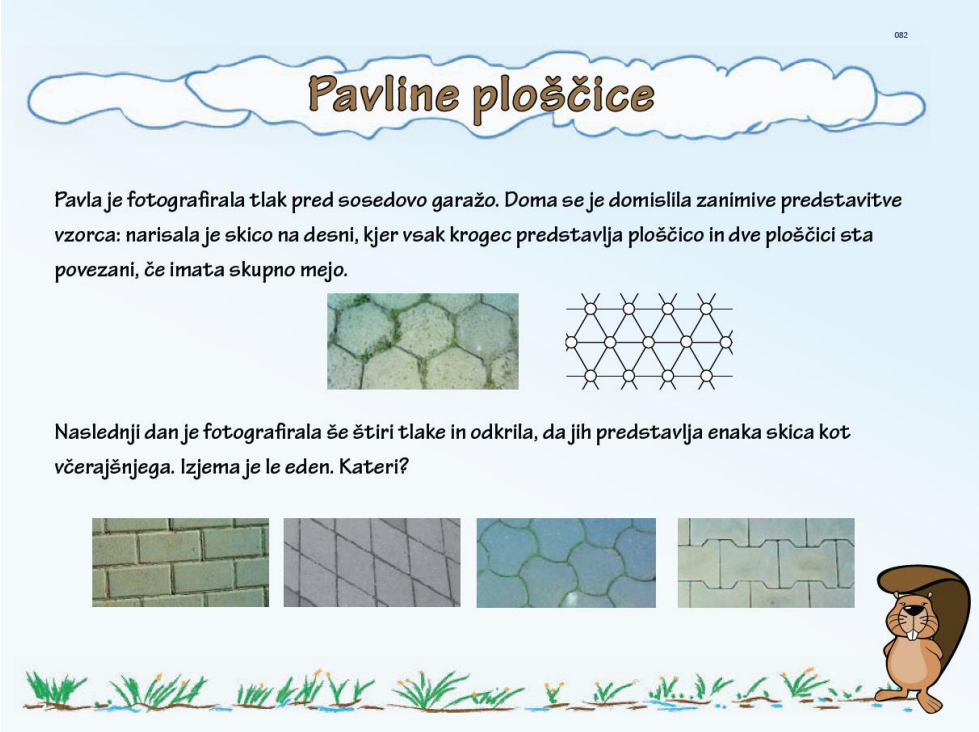
Graf je povezan (*connected*), če obstaja pot med vsakim parom točk, torej, če je iz vsake točke možno priti do vsake druge. V usmerjenih grafih govorimo tudi o šibki povezanosti (*weakly connected*): graf je le šibko povezan, če se pri kakem paru točk zgodi, da obstaja pot iz ene točke v drugo, nazaj pa ne.

Kadar graf ni povezan, razpade na nepovezane podgrafe. Rečemo jim tudi komponente. Vsaka komponenta je povezana – iz vsake točke lahko pridemo do vseh drugih točk iz komponente. Med točkami iz različnih komponent pa ni povezav. Kot primer vzemimo graf avtobusnih povezav na Hrvaškem: graf razpade na komponente, pri čemer ena, največja, predstavlja celinski del, manjše komponente pa ustrezajo posameznih otokom.

Zgodi se lahko tudi, da kakšna komponenta vsebuje eno samo točko. Takšna točka je lahko hrvaški kraj Unije, ki nima nobenih avtobusnih povezav (saj gre za edini kraj na istoimenskem otoku).

Povezavam so pogosto prirejene številke. Te imajo lahko različne pomene. V konkretnih primerih povedo, recimo, pogostost avtobusov, število skupnih jezikov ali kako dobro sogovornika govorita določen skupni jezik, kako "dobra prijatelja" sta dve osebi ali kako verjetna je določena relacija. V takšnih kontekstih temu številu rečemo *teža povezave* in v algoritmih, ki jo upoštevajo, želimo uporabiti povezave s čim večjo težo.

Bober – Pavline ploščice



Komentar: Skica, ki jo je narisala Pavla je pravzaprav graf. Da bi ugotovili, katere postavitve so ustrezne moramo biti pozorni na število skupnih mej, ki jih je Pavla predstavila s povezavami med točkami grafa.

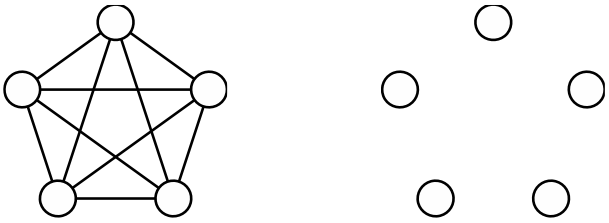
Drug pomen povezave je *cena*: v konkretnih nalogah je to lahko cena avtobusa ali trajekta ali pa dolžina poti med krajema. V algoritmih, ki upoštevajo ceno, poskušamo izbrati povezave s čim nižjo ceno. Ko iščemo, na primer, najkrajšo pot med točkama, je to pot z najmanjšo vsoto cen povezav. (V tem odstavku z "dolžino poti" očitno mislimo nekaj drugega kot malo višje; tam je bila dolžina *število* povezav, tu pa je *vsota* njihovih cen oz. dolžin.)

Če se pot konča v isti točki, kot se je začela, ji rečemo obhod (*cycle*). Med obhodi obstajata dva, ki imata posebni imeni. Hamiltonov obhod se začne in konča, kot vsi obhodi, v isti točki, prek vseh ostalih točk grafa pa gre natančno enkrat. Hamiltonova pot je podobna reč, le da se ne konča v isti točki, v kateri se je začela.

Eulerjev obhod in Eulerjeva pot sta podobna, le da ne gresta enkrat v vsako točko temveč enkrat po vsaki povezavi.

Posebni grafi

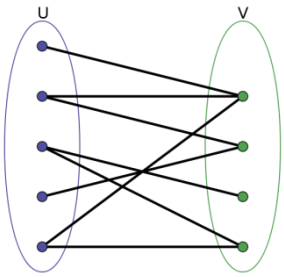
Graf je poln (*complete*) (Slika 11, levo) če so vsi pari točk neposredno povezani. Polni grafi so relativno dolgočasna zadeva. Pač pa so včasih zanimivi polni podgrafi: v (nepolnih) grafih včasih iščemo poln podgraf na petih točkah ali, v manj matematičnem jeziku, takšno peterico točk, da je vsaka neposredno povezana z vsako.



Slika 11. Poln in prazen podgraf na petih točkah.

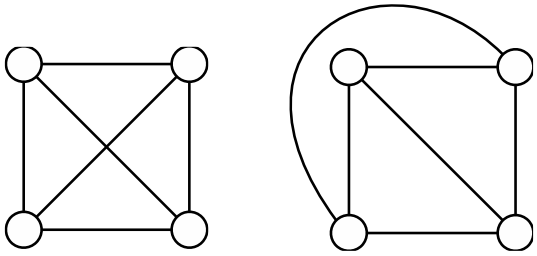
Graf je prazen, če v njem ni povezav, le točke (Slika 11, desno). Prazni grafi so absolutno dolgočasna zadeva.

Graf je dvodelen (*bipartite*), če lahko njegove točke razdelimo v dve podmnožici tako, da obstajajo povezave le med točkami iz različnih podmnožic, ne pa tudi znotraj podmnožice. Primer takšnega grafa so plesni pari: objekti so plesalci na nekem plesu in dve točki sta povezani, če sta plesalca odplesala kak ples. Dva dela tega grafa so plesalke in plesalci; povezave obstajajo le med točkami iz različnih delov (pri čemer ni potrebno, da je vsaka ženska plesala z vsakim moškim), ne pa tudi znotraj delov (ob predpostavki, da nobena ženska ni plesala z žensko in moški ne z moškim).



Slika 12: Dvodelen graf - med točkami, ki so znotraj množic U in V ni povezav

Graf je ravninski (*planar*), če ga lahko narišemo tako, da se povezave v njem ne sekajo. Vsak graf, ki ima vsaj dve povezavi, lahko seveda narišemo tudi tako, da se povezave sekajo; ravninskost pravi, da bi graf *lahko* narisali brez sekanja povezav, če bi se potrudili. Ravninskost je lastnost grafa, ne njegove grafične predstavitve.



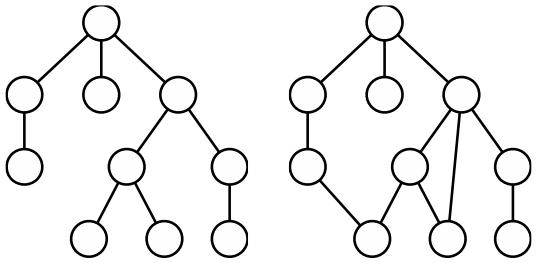
Slika 13. Graf na levi je planaren, čeprav se povezave na sliki sekajo; narišemo ga lahko namreč tudi tako, kot kaže desna slika.

Čisto posebna vrsta grafov so tudi drevesa. Tako posebna, da smo jih obravnavali ločeno: drevo je usmerjen graf, v katerem v vsako točko vodi natančno ena povezava. Izjema je le ena točka, ki nima povezav in ji pravimo koren. Drevo z *n* točkami ima, očitno, *n*-1 povezav.

Čeprav so povezave v drevesih usmerjene, jih pogosto rišemo brez puščic; pač pa drevo navadno rišemo od zgoraj navzdol (ali od spodaj navzgor ali z leve na desno), tako da se razume, da vse povezave vodijo le navzdol (ali navzgor ali na desno).

Podobna reč so usmerjeni aciklični grafi (*directed acyclic graph*; ker je ime dolgo, vrsta grafa pa pomembna, zanje uporabljamo kratico DAG, včasih pa jo uporabljamo celo kot besedo, *dag*). Zanje velja, da, kot ime pove, nimajo ciklov: iz nobene točke ne moremo priti nazaj vanjo. Spet jih lahko rišemo brez puščic, če jih, tako kot drevesa, rišemo od zgoraj navzdol.

V drevesu lahko do vsake točke pridemo le na en način, v usmerjenem acikličnem grafu pa na več načinov. Za razliko od iskanja poti v splošnih grafih, pa so poti v usmerjenih acikličnih grafih bolj "obvladljive": lažje jih je našteti ali prešteti.



Slika 14. Drevo (levo) in usmerjen aciklični graf (desno).

Bober – drevo živali

Komentar: Različne vrste oz. podvrste živali predstavimo s točkami, povezave med točkami pa predstavljajo pripadnost neke podvrste določeni vrsti. Če graf pogledamo malo bolje, lahko ugotovimo, da so povezave usmerjene (pripadnost podvrste vrsti) ter da v vsako točko vodi točno ena pot. Gre torej za drevo.