PRESEK
List za mlade matematike, fizike, astronome in računalnikarje

ISSN 0351-6652
Letnik **27** (1999/2000)
Številka 3
Strani 138–141

Martin Juvan:

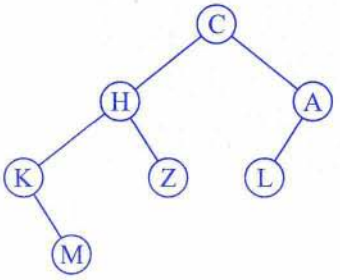
REKONSTRUKCIJA DREVES – 2. del

Ključne besede: računalništvo, programiranje, dvojiška drevesa, pregled po nivojih, vmesni pregled, premi pregled.

Elektronska verzija: http://www.presek.si/27/1395-Juvan.pdf

REKONSTRUKCIJA DREVES, 2. DEL

V prejšnji številki Preseka smo v prvem delu prispevka spoznali tri preglede dvojiških dreves: vmesni in premi pregled ter pregled po nivojih. Za osvežitev spomina si lahko na spodnji sliki ogledate dvojiško drevo in pripadajoče preglede.



vmesni pregled KMHZCLA
premi pregled CHKMZAL
pregled po nivojih CHAKZLM

Spoznali smo tudi, kako lahko iz vmesnega in premega pregleda rekonstruiramo drevo. Tokrat si bomo ogledali, kako je z rekonstrukcijo drevesa, če poznamo pregled po nivojih in enega od preostalih dveh pregledov, vmesnega ali premega.

Programsko kodo bomo pisali v jeziku C, zato si najprej oglejmo, kako v tem jeziku opišemo dvojiško drevo. Deklaracije so podobne kot v pascalu. Vozlišče predstavimo s strukturo (besedica `struct`) – to je C-jevsko poimenovanje sestavljenih tipov, drevo pa zopet s kazalcem na koren:

```
typedef struct vozlisce *Drevo;
typedef struct vozlisce {
  char x;
  Drevo levo, desno;
} Vozlisce;
```

Nekaj osnovnih funkcij za delo z drevesi (in kodo v C-ju) najdete v zbirki nalog *M. Juvan, M. Zaveršnik: C naj bo* (zadnji del 7. poglavja).

Vmesni pregled in pregled po nivojih. Tudi kadar poznamo pregled po nivojih namesto premega pregleda, se lahko rekonstrukcije drevesa lotimo na enak način. Dodatne težave nam povzroča dejstvo, da v pregledu po nivojih vozlišča levega in vozlišča desnega poddrevesa ne tvorijo več strnjenih podnizov (to dejstvo smo s pridom izkoristili pri rekonstrukciji iz premega pregleda).

Zato bo treba prilagoditi parametre rekurzivne funkcije in dopolniti iskanje korena. Funkcija bo imela tri parametre: kazalca `zacV` in `konV` tipa `char *`, ki bosta označevala prvi oziroma zadnji znak vmesnega pregleda, in niz `nivoji`, ki bo vseboval pregled po nivojih, med znake pregleda pa bodo lahko vrinjeni še drugi znaki. Koren poddrevesa, ki ga gradimo na rekurzivnem koraku, bo potem prvi znak iz `nivoji`, ki se pojavi tudi v vmesnem pregledu.

Na kratko poglejmo še korake opisanega postopka na primeru z začetka prispevka. Začnemo z vmesnim pregledom $KMHZCLA$ in pregledom po nivojih $CHAKZLM$. Vzamemo prvi znak iz pregleda po nivojih, to je C, in pogledamo, ali se pojavi v vmesnem pregledu. Tam ga najdemo in tako določimo vmesna pregleda za levo in desno poddrevo. Sledi rekurzivni klic za levo poddrevo: vmesni pregled je $KMHZ$ (del začetnega pregleda do znaka C), pregled po nivojih pa je vsebovan v nizu $HAKZLM$ ("odvečna" znaka sta A in L). Ponovno poiščemo koren. Poskušanje začnemo z znakom H in takoj uspemo. Sledi nov rekurzivni klic: vmesni pregled levega poddrevesa je KM, pregled po nivojih pa je vsebovan v nizu $AKZLM$ (tokrat so "odvečni" znaki A, Z in L). Tokrat prvi kandidat za koren, znak A, ni pravi (ne pojavi se v vmesnem pregledu). Zato poskusimo z naslednjim znakom K. Ta nastopi v vmesnem pregledu in je torej koren poddrevesa na nivoju 2. Zopet sledita rekurzivna klica. Prvi klic za levo poddrevo se takoj konča, saj gre za prazno poddrevo. Sledi drugi rekurzivni klic, ki zgradi drevo z vozliščem M. S tem je poddrevo, katerega koren je K, v celoti zgrajeno. Vrnemo se nivo nazaj. Tam sledi rekurzivni klic za desno poddrevo v drevesu, katerega koren je H. Ta zgradi drevo z vozliščem Z. Sledi vračanje do začetnega nivoja in klic, ki rekurzivno zgradi še desno poddrevo iskanega drevesa.

```
/* Zgradi dvojiško drevo iz vmesnega pregleda in pregleda
   po nivojih. Ne preverja pravilnosti vhodnih podatkov.  */
Drevo zgradi(char *zacV, char *konV, char *nivoji)
{
  char *pomV; /* mesto korena v vmesnem pregledu */
  Drevo kaz; /* pomožni kazalec za novo vozlišče */

  if (zacV > konV) return NULL; /* prazno drevo */
  /* Iščemo koren: prvi znak iz nivoji, ki je v vmesnem pregledu. */
  while (*nivoji != '\0') {
    pomV = zacV;
    while (*pomV != *nivoji && pomV <= konV) pomV++;
    if (*pomV == *nivoji++) break;
  }
```

```
  /* Če so podatki pravilni, smo koren gotovo našli. */
  kaz = malloc(sizeof(Vozlisce)); /* Naredimo novo vozlišče. */
  kaz->x = *pomV;
  /* Rekurzivna klica - zgradimo levo in desno poddrevo. */
  kaz->levo = zgradi(zacV, pomV - 1, nivoji);
  kaz->desno = zgradi(pomV + 1, konV, nivoji);
  return kaz;
} /*zgradi*/
```

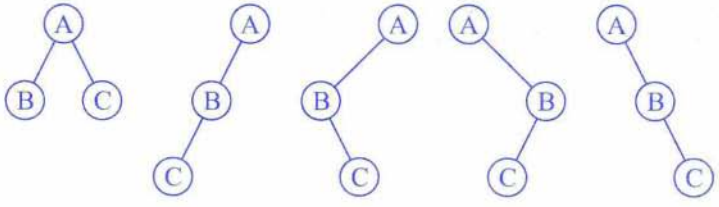
Če je sta pregleda drevesa shranjena v nizih `opis1` in `opis2` ter je `d` spremenljivka tipa `Drevo`, potem funkcijo pokličemo z

```
d = zgradi(opis1, opis1 + strlen(opis1) - 1, opis2);
```

Tu je `strlen` funkcija iz knjižnice `<string.h>`, ki vrne dolžino niza.

Poglejmo še, kako je s časovno in prostorsko zahtevnostjo funkcije. O prostorski zahtevnosti ne moremo povedati nič novega. Glede uporabe pomnilnika se funkcija obnaša praktično enako kot funkcija za rekonstrukcijo dreves iz vmesnih in premih pregledov iz prvega dela prispevka. Pri časovni zahtevnosti pa se funkciji na prvi pogled razlikujeta. Tokrat smo namreč uporabili še dodatno zanko za iskanje korena (poleg zanke, v kateri poiščemo mesto korena v vmesnem pregledu, ki jo imamo v obeh funkcijah). Vendar pa natančnejša analiza pokaže, da je število operacij še vedno omejeno s kvadratno funkcijo števila vozlišč drevesa (je pa vodilni koeficient te funkcije nekoliko večji kot pri prejšnjem primeru). Ključna je ugotovitev, da nobenega znaka iz vmesnega pregleda ne primerjamo dvakrat z istim znakom iz pregleda po nivojih (natančna utemeljitev te ugotovitve zahteva nekoliko daljši razmislek, ki presega okvir tega prispevka).

Premi pregled in pregled po nivojih. Premi pregled in pregled po nivojih sta si precej podobna, zato ni presenetljivo, da drevesa z njima niso enolično določena. Premi pregled ABC in pregled po nivojih ABC ima npr. naslednjih pet dreves:



Enolična rekonstrukcija drevesa iz premega pregleda in pregleda po nivojih tako v splošnem ni mogoča. To ugotovitev lahko še nekoliko poostrimo. Za konec vas bom poskušal prepričati, da velja naslednja trditev: *Za vsako dvojiško drevo $\mathcal{D}$ z vsaj dvema vozliščema obstaja vsaj še eno dvojiško drevo $\mathcal{D}' \neq \mathcal{D}$, ki ima enak premi pregled in enak pregled po nivojih kot drevo $\mathcal{D}$.* Takole razmišljajmo. Naj bo x najbolj desno vozlišče na zadnjem nivoju (pri drevesu iz zgleda je to vozlišče M). Mimogrede opazimo, da je x zadnje obiskano vozlišče pri premem pregledu in pri pregledu po nivojih. Ker ima drevo $\mathcal{D}$ po predpostavki vsaj dve vozlišči, x ni koren drevesa. Naj bo y njegov oče (v zgledu je to vozlišče K). Ločimo dve možnosti. Če je x edini sin vozlišča y (tako je v zgledu), potem je x lahko levi ali pa desni sin. Če je levi, ga lahko prestavimo na desno, če je desni, pa na levo, v obeh primerih pa ima novo drevo enak premi pregled in enak pregled po nivojih kot drevo $\mathcal{D}$. Kadar pa ima y dva sinova, lahko poddrevo s tremi vozlišči, katerega koren je, zamenjamo s katerimkoli od petih dreves z zadnje slike, pa še vedno dobimo drevo z enakima pregledoma.

Martin Juvan