ALGORITMIEK: opgaven werkcollege 2

BINAIRE BOMEN

Er moeten enige **recursieve** functies geschreven worden die werken op binaire bomen. We bekijken binaire bomen die behalve een info-veld van type char ook een extra-veld van type integer bevatten.
Een knoop van zo'n binaire boom ziet er dus als volgt uit:

```
struct knoop {
  knoop* links,  // Linker kind.
  knoop* rechts; // Rechter kind.
  char info;     // Info veld.
  int extra;     // Extra veld.
};//knoop
```

Er is al een skeletprogramma beschikbaar, met een klasse binaireboom, waarvan nog niet alle functies zijn ingevuld. Zie: `http://www.liacs.nl/~jlaros/edu/alg.shtml` Van de binaire bomen in de testvoorbeelden zijn de extra-velden nog niet geïnitialiseerd.

Opgave 1. Schrijf een recursieve memberfunctie `int aantalbladeren_p(knoop* k)`, die het aantal bladeren bepaalt van de binaire boom met wortel (ingang) `k`.

Opgave 2. Schrijf een recursieve memberfunctie `int hoogte_p(knoop* k)` die de hoogte berekent van de binaire boom met wortel (ingang) `k`. De hoogte van een binaire boom is het grootste niveau (nivo) dat voorkomt, waarbij de wortel van de boom op niveau 1 zit.

Opgave 3. Schrijf een recursieve memberfunctie `void initextra_p(knoop* k)`, die de extravelden van de binaire boom met wortel (ingang) `k` allemaal op 0 zet.

Opgave 4. Schrijf een recursieve memberfunctie `void vulextra_p(knoop* k, int niveau)`, die het extra-veld in elke knoop van de binaire boom met wortel `k` vult met het niveau van die knoop. (We spreken af: de wortel bevindt zich op niveau 1).

Opgave 5. Schrijf een recursieve memberfunctie `char maxinfowaarde_p(knoop* k)` die de maximale info-waarde (dat is hier de grootste/laatste in lexicografische volgorde) bepaalt die in de binaire boom met wortel (ingang) `k` zit.

Opgave 6. Schrijf een recursieve memberfunctie `void doepostorde_p(knoop* k)`, die inhoud van de binaire boom met wortel (ingang) `k` in *postorde*volgorde afdrukt. Deze functie wordt pas aangeroepen als de extra-velden gevuld zijn, en drukt dan van alle knopen in LRW-volgorde info-veld en extra-veld als volgt af: info,extra info,extra info,extra .....

BINAIRE ZOEKBOMEN

Nu gaan we naar een toepassing van binaire bomen kijken, namelijk de *binaire zoekboom*. Deze bomen hebben de eigenschap dat alle knopen in de linker subboom een kleinere waarde bevatten dan de knoop zelf. Alle knopen in de rechter subboom bevatten een grotere waarde dan de knoop zelf. De functies die hieronder gevraagd worden zijn **niet** recursief, tenzij anders wordt aangegeven.

Opgave 7. Schijf een memberfunctie `knoop *grootstekleinere(knoop *k, knoop *&o)`, die een pointer teruggeeft naar de knoop die de grootste waarde kleiner dan de waarde in `k` bevat. Na afloop van deze functie moet `o` wijzen naar de ouder van de grootste kleinere. De pointer `k` mag bij de aanroep niet `NULL` zijn en de waarde van `o` moet `NULL` zijn bij aanroep.

Opgave 8. Schijf een memberfunctie `knoop *kleinstegrotere(knoop *k, knoop *&o)`, analoog aan de voorgaande opgave.

Opgave 9. Schrijf een recursieve memberfunctie `bool isbzboom_p(knoop *k)`, die `TRUE` teruggeeft als de boom met wortel `k` een binaire zoekboom is en `FALSE` als dit niet zo is. Hint: Gebruik de functies `grootstekleinere()` en `kleinstegrotere()`.

Opgave 10. Schrijf een memberfunctie `knoop *bzoek_p(char waarde, knoop *&o)`, die gegeven een binaire zoekboom een pointer naar de knoop retourneert die `waarde` bevat. Na afloop van deze functie moet `o` wijzen naar de ouder van de gevonden knoop.
Als de waarde niet aanwezig is, moet `NULL` geretourneerd worden (`o` wijst dan dus naar de knoop waar het zoeken ophoudt omdat de waarde niet aanwezig blijkt te zijn). Als de boom leeg is of als `waarde` in de wortel zit, moet `o` op `NULL` blijven staan.
De waarde van `o` moet `NULL` zijn bij aanroep.

Opgave 11. Schrijf een memberfunctie `void bzvoegtoe_p(char waarde)`, die een waarde aan een binaire zoekboom toevoegt indien die nog niet aanwezig is.
Hint: Gebruik de functie `bzoek_p()`.

Opgave 12. Schrijf een memberfunctie `void bzverwijder_p(char waarde)`, die een waarde uit een binaire zoekboom verwijdert, indien aanwezig.
Hint 1: Gebruik de functies `bzoek_p()` en `grootstekleinere()` of `kleinstegrotere()`.
Hint 2: Onderscheid de gevallen: blad, knoop met 1 kind en knoop met 2 kinderen.