KTH, NADA, Henrik Eriksson

2D1320, Tilda, Tentamenslösning 8 mars 1997

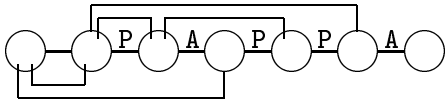
1. *Är stackarna lika?*

Rekursiv tanke: Om båda stackarna är tomma är dom lika; om bara den ena är tom är dom olika; om ingen är tom och topposternas pnr är olika så är dom olika. Annars är dom lika om och endast om stackarna under topposterna är lika.

2. *Lärarträdet*

Binärträdet har två löv (men ritat som allmänt träd får det sex löv). Preorder ger det önskade T M H V I K B L F. Inorder ger H V I M K L F B T och postorder I V H F L B K M T.

3. *Leta efter pappa!*



i	next[i]
1	0
2	1
3	0
4	2
5	1

En snabbare metod är att titta på var femte bokstav i Strindbergs samlade verk. Så länge den inte är P eller A kan man hoppa vidare. När man träffar på ett P eller A får man kolla några bokstäver bakåt. Sedan hoppar man vidare från ett nytt utgångsläge.

4. *Anagram*

Inmatade ord ger en `antal: ARRAY['a'..'ö'] OF CARDINAL` som anger hur många exemplar av varje bokstav man har. Posterna i problemträdet innehåller ett ord och en antalsvektor för återstående bokstäver. För att spara plats kan man i stället för ordet ange ordets index i SAOL-arrayen. Söner skapas genom att man letar vidare i SAOL från fadersordet efter ord som kan bildas av återstående bokstäver. När alla bokstäver har gått åt har man en lösning.

Djupet-först är lämpligaste metoden, eftersom man inte är ute efter kortaste lösningen och eftersom problemträdet är ändligt.

Procedurer och moduler som i fantillgudlabben men med stack i stället för kö.

5. *Hashmissbruk*

Ett balanserat binärträd med `SAOL`s 120259 ordlistord har sjutton nivåer. Lyckad sökning kräver i genomsnitt sexton jämförelser. En hashvektor med 180000 platser kräver i genomsnitt kanske 1.6 jämförelser och är alltså tio gånger snabbare.

Stavaprogrammet finns att studera i föreläsningsanteckningarna nr 8.

6. *Syntax för tro och vetande*

```
<mening>     ::= <sats> . | <sats><konjunktion><mening>
<sats>       ::= <subjekt><predikat>
<subjekt>    ::= JAG | DU
<predikat>   ::= VET | TROR
<konjunktion>::= ATT | OCH
```

7. *Abstrakta skonummer*

`TEXT` är inte bra om man vill jämföra två skonummer. `REAL` är inte bra om man vill ta ett nummer större. `INTEGER` är inte bra om man har engelska halvnummer.

En abstrakt `Skonr.T` kan vara en objekttyp med till exempel följande gränssnitt.

```
INTERFACE Skonr
TYPE T<:AbstraktSkonr;
     AbstraktSkonr = OBJECT
     METHODS
       putsvenskt(n:INTEGER); (* Sätter svenskt skonummer *)
       putengelskt(x:REAL);   (* Sätter engelskt skonummer *)
       getsvenskt():INTEGER;  (* Returnerar svenskt skonummer *)
       getengelskt():REAL;    (* Returnerar engelskt skonummer *)
       equal(skonr:T): BOOLEAN;
       biggerthan(skonr:T): BOOLEAN;
       smallerthan(skonr:T): BOOLEAN;
       nextbigger():T;
       nextsmaller():T;
     END;
END Skonr.
```