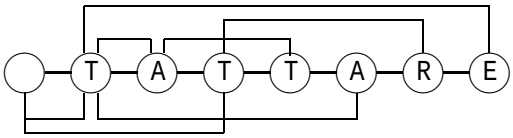
KTH, NADA, Henrik Eriksson

2D1320, Tilda, Tentamenslösning 13 mars 1999

1. *Knuthautomat*



i	next[i]
1	0
2	1
3	0
4	2
5	1
6	3
7	1

2. *Trädkopiering*

```
PROCEDURE Copy(p:Pek):Pek=
BEGIN
  IF p=NIL THEN RETURN NIL END;
  RETURN NEW(Pek,tal:=p.tal,left:=Copy(p.left),right:=Copy(p.right));
END Copy;
```

3. *Armbrytning*

Lägg deltagarnamnen i en kö. Gör upprepade gånger föjande: Ta ut två namn ur kön, matcha dom, putta in segraren sist i kön.
...men finns det bara ett namn i kön är det mästaren.

4. *Hashad historia*

Med size=10000 blir index dom fyra sista siffrorna, dvs bara 366 olika värden av dom tiotusen utnyttjas. Krocklistlängd blir $8000/366 \approx 22$. Med primtalet 1003 blir utspridningen jämn och listlängd $8000/1003 \approx 8$. Viggo skulle ha använt en bitvektor med kanske hundrafemtiotusen bitar varav cirka hälften blivit ettor när man hashat med fjorton hashfunktioner.

5. *Längst väg till gud*

Låt gud bli stamfar och skapa hela trädet breddenförst med hjälp av en kö. I kön läggs poster med ord och faderspekare. Det sista ord som tas ut ur kön har längst väg till gud, i alla fall om vi aldrig skapat dubbletter, dvs dumsöner. Och det kan undvikas med exempelvis ett binärträd för använda ord.

6. *Molekylsyntax*

```
<mol>    ::= <group> | <group><mol>
<group>  ::= <atom> | <atom><num> | ( <mol> ) <num> | [ <mol> ]
```

Proceduren `ReadMol` anropar `ReadGroup` som tjuvtittar och sedan antingen glufsar parentes eller hake, anropar `ReadMol`, kollar, glufsar och eventuellt anropar `ReadNum` eller också anropar `ReadAtom` och eventuellt `ReadNum`. Både `ReadMol` och `ReadGroup` returnerar en heltalsvektor med ett tal för varje grundämne. `ReadMol` adderar ibland två vektorer och `ReadNum` multiplicerar ibland en vektor med ett tal.

7. *Webbtoppen*

Tio genomletningar av vektorn tar 100 000 jämförelser. Inmatning i trappan tar cirka N log N, alltså 130 000 jämförelser och utplockning cirka 20 log N, alltså ytterligare 260 jämförelser. Distributionsräkning innebär att man fyller en vektor med 20 000 nollor och sedan för vart och ett av dom tiotusen talen ökar motsvarande fack med ett. Om talet 17 dyker upp gör man alltså `x[17]:=x[17]+1`. Sedan går man igenom vektorn från index 20 000 och neråt och ser var gränsen går för att man ska tillhöra tio i topp. Slutligen går man igenom dom tiotusen talen och skapar topplistan. Man kan räkna det som cirka 20 000 jämförelser, klart bäst alltså.

8. *Abstrakta böcker*

En abstrakt `Book.T` kan vara en objekttyp med följande gränssnitt.

```
INTERFACE Book
TYPE T<:AbstractBook;
     AbstractBook = OBJECT
     METHODS
       title():TEXT;
       author():TEXT;
       year():INTEGER;
       pages():INTEGER;
       out():BOOLEAN;                (* Anger om boken är utlånad *)
       lender():Lender.T;            (* Låntagartypen är abstrakt *)
       lend(person:Lender.T);        (* Låna ut boken *)
       return();                     (* Återlämna boken *)
     END;
PROCEDURE New(title,author:TEXT; year,pages:INTEGER):T;
END Book.
```

Abstrakta böcker är bättre eftersom man kan ändra representation utan att abstrakta programmen som använder abstrakta böcker. Typiskt vore att man lagrat year med två siffror och inför tusenårsskiftet kommer på att fyra siffror är bättre.