KTH, NADA, Henrik Eriksson

2D1320, Tilda, Tentamenslösning 27 oktober 2001

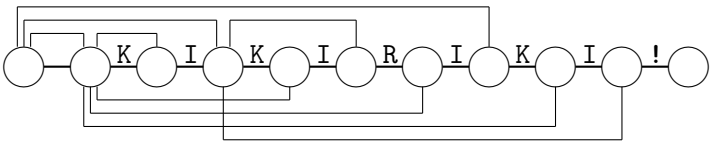
1. *Rekursiv rappkoll*

Rekursiv tanke: För att få reda på vilken stack som är störst poppar man ett element från vardera stacken och sparar i två lokala variabler. Nu är det bara att kolla vilken av dom återstående stackarna som är störst, men innan man returnerar svaret pushar man tillbaka dom båda poppade elementen.
. . .men om båda stackarna är tomma är svaret noll!
. . .men om ena stacken är tom är svaret 1 eller -1!

2. *Rapp trappsortering efter kön*

Man tar ut skiva efter skiva ur kön och stoppar in i trappan med prioriteten `dag+31*(månad+12*år)`. Varje gång det kommer en ny artist tömmer man trappan genom att stoppa in trappans skivor i kön.

3. *Rapp textsökning*



i	next[i]
1	0
2	1
3	0
4	1
5	3
6	1
7	0
8	1
9	3

Om ungefär vart sjunde tecken är `K`, `I`, `R` eller `!` kräver Boyer-Moore kanske 10 jämförelser per 72 tecken. Det är mycket snabbare än automaten.

4. *Svenska Dagbladets ordlek Nian*

Den enda datastruktur som behövs är en array med ett heltal för varje bokstav i alfabetet. Man läser ett ord i ordlistan och använder arrayen till att räkna antalet förekomster av varje bokstav. Sedan går man igenom arrayen och kollar om ordet innehåller just dom nio bokstäverna i rutan. I annat fall går man igenom arrayen en gång till och kollar om ordet innehåller mittenbokstaven och minst tre bokstäver till. Att gå igenonom hela ordlistan på detta sätt är dock inte optimalt. Snabbare är att läsa in ordlistan i en vektor och med binärsökning hitta dom avsnitt som börjar på dom nio bokstäverna.

5. *Lönar sej sortering?*

Osorterad vektor kan kräva en miljon jämförelser för varje sökning, d v s hundra miljoner totalt. Med normal otur tar det hälften så mycket, alltså femti miljoner.

Sortering med quicksort kräver $2N \ln N$ jämförelser, dvs cirka trettio miljoner. Sedan tar varje binärsökning bara tjugo jämförelser, dvs tvåtusen totalt. Det lönar sej att sortera!

6. *Syntax för emotionella komplikationer*

```
<mening>     ::= <subjekt> <somsats> <verb> <objekt> <somsats>
<subjekt>    ::= JAG | DU | HAN | HON
<verb>       ::= ÄLSKAR | HATAR
<objekt>     ::= MEJ | DEJ | HONOM | HENNE
<somsats>    ::=     | SOM <verb> <objekt> <somsats>
```

7. *Abstrakta betyg*

Textsträng är inte bra om man vill räkna betygsmedel. Heltalsvektor är inte bra om man vill veta datum och studentdata.

En abstrakt `Betyg` kan vara en klass med till exempel följande metoder.

```
void registrera(Kurs kurs, Datum datum);
boolean ärRegistrerad(Kurs kurs);
void putBetyg(Kurs kurs, Datum datum, int betyg);
int getBetyg(Kurs kurs);
Datum getDatum(Kurs kurs);
Student getStudent();
```

Betygen kan lagras i en `private int[] registrering`.