KTH, Nada, Alexander Baltatzis

2D1320, Tildatenta lösningsförslag
Lördagen den 12 mars 2004 kl 8–13

1. *Hitta lekisleken*

(6p) (a) Next-vektorn

```
L E K I S L E K A R
0 1 1 1 1 0 1 1 4 1
```

(6p) (b) Tabell med tecken c och tillstånd i.

```
c = L E K I S L E K S S S A K E R _ I _ L E K I S L E K A R
i = 1 2 3 4 5 6 7 8 9 4 1 1 1 1 1 1 1 1 2 3 4 5 6 7 8 9 10
```

2. *Lekande skratt*

(4p) (a) Definiera testfall

Godkännes	*Godkänns inte*
TIHIHIHI	HAHATIHI
HAHAHA	HOHOHO
HAHAHIHIHAHA	HA
TIHIHAHAHIHIHI	HIHIHAHIHI

(10p) (b) *Skriv en syntax för lillebrors skratt.*

```
<skratt>     ::= <TIHI> | <TIHI><DELSKRATT> | <DELSKRATT>
<TIHI>       ::= TIHI | TI<HI>
<DELSKRATT>  ::= <HIHA> | <HIHA><DELSKRATT>
<HIHA>       ::= <HI> | <HA>
<HI>         ::= HIHI | HIHIHI
<HA>         ::= HAHA | HAHAHA
```

Kommentar: Det går att göra kortare och elegantare syntax genom att använda en symbol ϵ som står för ingenting.

3. *Dåligt leksakssortiment*

(a) *Vilka sorteringsalgoritmer behåller den tidigare inbördes sorteringen?*

(12p) För att lösa uppgiften måste man själv hitta på några data och sortera för hand. Först m.a.p på datum och sedan m.a.p på namn. Bubbel- och insättningssortering bibehåller sorteringsordningen. Visa med exempel.

Urvalssortering förstör den tidigare sorteringen. Nedan är ett exempel på urvalssortering på tredje varvet d.v.s två fält är rättsorterade (fetmarkeras nedan). Notera hur *Isak 010101* flyttas långt ner i listan vid platsbytet och *Isak 020202* sorteras in på "fel" plats.

Alex , 010101	**Alex** , 010101	**Alex** , 010101
Alex , 020202	**Alex** , 020202	**Alex** , 020202
*Isak , 010101	**Alex** , 030303	**Alex** , 030303
Linda, 010101	*Linda, 010101	**Isak** , 020202
Isak , 020202	*Isak , 020202	*Linda, 010101
Johan, 010101	Johan, 010101	Johan, 010101
Linda, 020202	Linda, 020202	Linda, 020202
Johan, 030303	Johan, 030303	Johan, 030303
Isak , 030303	Isak , 030303	*Isak , 030303
*Alex , 030303	Isak , 010101	Isak , 010101(!)
Linda, 030303	Linda, 030303	Linda, 030303

(6p) (b) Quicksort har liknande platsbyten stora platsbyten som urvalssortering är i allmänhet inte en stabil sortering. Mergesort kan man likna lite som insättningssortering. I samsorteringen av dellistorna görs insättning i slutliga listan. Mergesort är stabil.

Det går att skriva partitioneringen så att man får stabil sortering. Man kan klanta till implementeringen av samsorteringen av listor i mergesort så att den inte blir stabil.

Radixsort gick vi inte igenom på föreläsningen men den står beskriven i Baileys bok och är stabil. Det går att få full poäng på deluppgiften utan att känna till radixsort.

Kommentar: Det står olika mycket om stabil sortering i böckerna. Många böcker skriver vilka som är stabila. Oavsett vilken hjälp boken ger så är det svåra i den här uppgiften att själv konstruera en datamängd och visa skillnaderna.

4. *Djupsinta leksaker*

(5p) (a) *Beskriv en datastruktur som representerar kuben.*

Det finns många sätt att representera kuben. Man kan se kuben som 8 hörn som kan vara placerade på 8 platser och roteras på tre sätt. I så fall en 8 stor vektor med klassen hörn som har två fält: id och rotation. Där id är 1 till 8 och rotation 1 till 3.

(5p) (b) *Antag att du har tillgång till en metod som kan göra en sidvridning med din datastruktur nämligen:*

DinDatastruktur till = **vridAxel**(**int** axel, **DinDatastruktur** från);

Från ett givet läge kan vi komma till tre nya lägen genom att vrida ett steg kring någon av kubens tre axlar. En rekursiv djupet-förstalgoritmen blir i princip:

```
void sok(DinDatastruktur läge) {
  if (kollaOmKlar(läge)); // KLAR!! avbryt programmet
  for (int i = 1; i <= 3; i++) {
    sok(vridAxel(i, läge));
  }
}
```

För en iterativ djupet-förstsökning krävs en stack.

(3p) (c) *Om man vrider en axel fyra gånger så får man samma situation igen. Om man inte tar hänsyn till det blir djupet-förstsökningen ineffektiv.*

I algoritmen ovan kommer precis det att hända, kuben kommer att rotera kring axel nummer 1 i evighet och det kan man säga är inte bara oeffektivt, det är fel.

(7p) (d) *Beskriv utförligt hur man hanterar jämförelser med din datastruktur och hur man effektiviserar djupet-förstsökningen.*

Någon slags jämförelse måste man ha i vilket fall som helst för att kunna avgöra om kuben är löst.

Med den föreslagna datastrukturen blir det en loop som går igenom vektorn och kollar att första hörnet ligger på plats ett och är rätt roterat, andra hörnet ligger på plats två o.s.v.

För att effektivisera djupet-förstsökningen måste vi jämföra med tidigare undansparade lösningar. För att göra uppslagningen snabbt kan vi spara undan tidigare lösningar i en hashtabell. Då behöver vi en hashcode. Den skulle vi t.ex. kunna tillverka genom att konkatenera alla siffror i våran datastruktur till en stor (8x2x2=32-siffrigt) sträng eller BigNum och ta hashCode för den.

Ett annat sätt är att stoppa in i ett binärträd. Då måste vi utöka jämförelsefunktionen så att man kan avgöra vilken av två instanser DinDatastruktur som är minst. Det är inte uppenbart. I fallet ovan krävs en loop som jämför fält för fält och avbryter när något fält anses som mindre.

(3p) (e) Om man vet något om utgångsläget (vad gjorde lillebror) så kan man välja en annan metod. När skulle man t.ex. hellre välja bredden-först för att lösa kuben?

Om lillebror bara har gjort några få drag så vet vi att det finns en kort väg till lösningen. Då är det smartare med bredden-förstsökning som söker kortaste lösningen.

5. *Abstrakta leksaker*

(3p) (a) *Antag att man i ett större grupparbete skulle vilja bygga en robot som löste kuben. Det finns flera arbetsuppgifter t.ex. att styra robotens armar, läsa av sensorer eller skriva algoritmerna som löser kuben.*

Vad är det i allmänhet för fördelar med att använda abstrakta datatyper?

Fördelar är att kunna parallellisera utvecklingsarbetet. De som implementerar gränssnittet kan förbättra implementeringen utan att påverka de övriga grupperna.

(5p) (b) *Hitta på och beskriv en eller ett par abstrakta datastrukturer och några metoder som skulle kunna vara vettiga i robotprojektet.*

(3p) (c) *Deklarera några av metoderna du beskrivit i b och beskriv vettiga parametrar och vettigt returvärde till dessa metoder.*

Ett vettigt gränssnitt är t.ex. DinDatastruktur. I föregående uppgift gick vi igenom några jämförelsemetoder som måste finnas kvar. De som skriver algoritmen kanske kommer något otroligt smart sätt att representera kuben som förenklar algoritmen. De som pysslar med robotarmen kanske representerar kuben internt på något annat sätt, sidan man håller i + sidan man ska rotera.

Kommentar: Alla rimliga godkänns, för att få full poäng måste man dock hänvisa till motiveringen i a

6. *Önkelistan*

(1p) (a) *Pappa har sorterat om önskelistan i plånboksordning och lagt dem i en kö (billigast först). Hur ser kön ut?*

(8p) (b) *Visa hur prioritetskön ser ut i vektorform efter varje insättning.*

För att lösa den här uppgiften kan det vara bra att numrera lillebrors önskningar. Lillebrors och pappas värderingar ser ut så här:

```
Lillebrors lista     Pappas lista
 1_Bri   299:-       9_Pok    49:-
 2_Jär   399:-       8_Fär    50:-
 3_Har   499:-       7_Dra    69:-
 4_Rad   295:-       6_Leg    99:-
 5_Cyk  1500:-       4_Rad   295:-
 6_Leg    99:-       1_Bri   299:-
 7_Dra    69:-       2_Jär   399:-
 8_Fär    50:-       3_Har   499:-
 9_Pok    49:-       5_Cyk  1500:-
```

Insättning i heap:

```
9_Pok
8_Fär 9_Pok
7_Dra 9_Pok 8_Fär
6_Leg 7_Dra 8_Fär 9_Pok
4_Rad 6_Leg 8_Fär 9_Pok 7_Dra
1_Bri 6_Leg 4_Rad 9_Pok 7_Dra 8_Fär
1_Bri 6_Leg 2_Jär 9_Pok 7_Dra 8_Fär 4_Rad
1_Bri 3_Har 2_Jär 6_Leg 7_Dra 8_Fär 4_Rad 9_Pok
1_Bri 3_Har 2_Jär 5_Cyk 7_Dra 8_Fär 4_Rad 9_Pok 6_Leg
```

(9p) (c) *Rita storasysters heap i trädform.*

```
                Bri
           Har          Järl
      Cyk   Dra   Fär   Rad
    Pok Leg
```

Hur skulle trädet skrivas ut om man skrev ut det i pre- resp. postorder?

```
Bri Har Cyk Pok Leg Dra Järl Fär Rad
Pok Leg Cyk Dra Har Fär Rad Järl Bri
```

Beskriv en algoritm och de datastrukturer som behövs för det. Om man gör det iterativt med en slinga behövs en stack. Den rekursiva lösningen behöver bara nodklassen.

```
void print(Nod p) {
  if (p != null) {
    if (preorder) System.out.println(p);
    print(p.right);
    print(p.left);
    if (postorder) System.out.println(p);
  }
}
```

Kommentar: Det går att svara med en rekursiv tanke precis som i hemtal 3.

(4p) (d) *Antag att man skulle vilja skriva ut hela trädet nivå för nivå. Beskriv en algoritm och de datastrukturer som behövs för det.*

En bredden-först algoritm (se andra X-tentor) som inte avbryter utan håller på tills kön är tom.

Kommentar: Frågan avsåg egentligen ett allmänt träd men eftersom det specifika trädet är härlett från en heap så är det OK att svara att man skriver ut heapvektorn.