KTH, NADA, Linda Kann

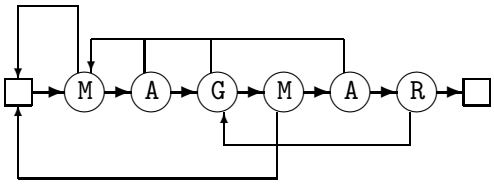
2D1320, Tilda, lösningsskiss och kommentarer för tentan 19 oktober 2005

Maxpoäng = 100. Betygsgränser: 50 poäng ger trea, 70 ger fyra, 90 ger femma.

1. *Hitta en Pokémon*

Pokémon är små monster som man kan bära med sig i fickan (i en pokeboll).

(10p) Konstruera och rita upp en KMP-automat som söker efter eldpokémonen `MAGMAR`. Ange även next-vektorn!



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

2. *Hitta fler pokémon*

I jakten på eldpokémon mäter man upp massor av jordtemperaturer över hela safarizo-

(10p) nen. Föreslå, beskriv och motivera en lämplig komprimeringsmetod.

För full poäng ska man ha valt en lämplig metod, motiverat valet, och beskrivit hur den fungerar i just det här fallet.

3. *Teori*

(20p) Nedan finns fem frågor om algoritmer och datastrukturer. Varje fråga kan ge upp till fyra poäng. Motivering krävs!

a. Duger JUnit om man vill testa hur ett program reagerar på felaktiga indata? *Ja, i JUnit kan man ju kolla om ett särfall kastas när så borde ske.*

b. Utför en algoritm som är O(1) alltid exakt en operation? *Nej, man vet bara att det är ett konstant antal oberoende av indatas storlek. O(1) betyder inte nödvändigtvis hashuppslagning, så eventuella beskrivningar av detta eller slutsatser av hur hashuppslagning beter sig ger mindre än full poäng.*

c. Kan en heap implementeras med en länkad lista? *Ja, men inte med bibehållen effektivitet, eftersom det då tar längre tid att hoppa upp/ner en nivå i det tänkta trädet. För att få full poäng måste man inse att det går med länkad lista (och påpeka att det är dumt).*

d. Går det att ta reda på om ett binärt sökträd är balanserat? *Ja, t ex genom att beräkna maxdjup och jämföra det med tvålogaritmen för antalet noder. En del har svarat man skriver ut trädet och tittar på det", vilket inte ger full poäng (kan bli besvärligt i praktiken för stora träd...)*

e. Är Caesarskrift (att man t ex byter A mot E, B mot F osv) oknäckbart i praktiken? *Nej, det kan knäckas med hjälp av en frekvenstabell (eller genom att prova igenom alla 26 varianterna).*

4. *Pokétyp*

En pokémon har olika egenskaper (t ex namn, styrka och uthållighet), tillhör en huvudtyp (t ex eld, gräs eller vatten), kan utvecklas (t ex kan Pikachu utvecklas till Raichu) och har ett antal attacker (t ex kick eller lullaby) som den kan använda i dueller med andra pokémon. Beskriv en abstrakt datatyp för en pokémon! Hur kan man implementera en abstrakt datatyp i Java? (8p)

I Java kan man beskriva en abstrakt datatyp med ett interface. I vårt Pokemon-interface kan vi ha följande metoder:

```
String namn();
int styrka();
int uthållighet();
String typ();
Pokemon utveckla();
void attack(String attacknamn; Pokemon offer);
```

5. *Pokémonstyrka*

Muk (en giftig pokémon) har styrka som varierar dag för dag, och beror av de två tidigare dagarnas styrka. En Muk som har styrkan 8 när den föds och 16 dagen därpå får styrkan 8+(8+16)/2=20 nästa dag. Dagen därpå får den styrkan 8+(16+20)/2=26. Efter ytterligare en dag har den styrkan 8+(20+26)/2 osv.

Skriv rekursiv tanke och basfall för att beräkna Muks styrka. (5p)

Visa hur din rekursiva tanke fungerar genom att beräkna styrkan för en Muk som föddes för fyra dagar sen med styrkan 4 och hade styrkan 12 dagen därpå. (5p)

Rekursion: Om vår Muk är två dagar gammal eller äldre blir styrkan summan av födelsestyrkan och medelvärdet av de två senaste dagarnas styrkor, alltså $s(n) = s(0) + (s(n-1) + s(n-2))/2$. *I högerledet ska man ha kommit närmare basfallen!*

Basfall: Här krävs två basfall, $s(0)$ *och* $s(1)$*: Om Muk är nyfödd har den födelsestyrka och dagen därpå har den en-dag-gammal-styrka.*

När det gäller beräkningen är det viktigt att det syns i vilken ordning anropen skulle göras, dvs att $s(4)$ *beräknas med hjälp av* $s(3)$ *och* $s(2)$*,* $s(3)$ *beräknas ur* $s(2)$ *och* $s(1)$ *osv.*

6. (12p)

1. Läs in korten till en array, upprepa sedan följande för varje kort: Skriv ut kortet, sök sedan med linjärsökning efter dubbletter i resten av arrayen, och sätt dessa till null. *Inläsning* $O(n)$*, en linjärsökning för varje kort* $nO(n)$*, utskrift* $O(n) => O(n^2)$ *totalt. Fungerar.*
2. Läs in korten till en array. Sortera korten i bokstavsordning med insättningssortering. Skriv sen ut alla kort som inte är identiska med nästa i vektorn. *Inläsning* $O(n)$*, sortering* n^2*, sen går man igenom alla igen och jämför med nästa vilket tar* $O(n) => O(n^2)$ *totalt. Fungerar i princip, men vi måste lägga till utskrift av allra sista elementet.*
3. Hasha in korten i en hashtabell utan krocklistor. Skriv sedan ut alla nullskilda element. *Att hasha in alla korten tar* $O(n)$ *och utskriften* $O(n) => O(n)$ *totalt. Men det fungerar bara om man har en perfekt hashfunktion, annars kan man ju få krockar även för olika kort.*

4. Hasha in korten i Viggos bloomfilter med fjorton hashfunktioner. Gå sedan igenom filen igen och skriv ut alla kort som ger träff i bloomfiltret. *Komplexitet $O(n)$ som hashningen ovan, men fungerar inte. Eftersom vi går igenom filen när vi ska skriva ut korten och inte sållar bort några kort så kommer alla kort att skrivas ut.*

5. Sortera in korten i ett binärt sökträd, som i labb 3, och skriv sedan ut trädet i inorder. *Stoppa in n kort i trädet tar $nO(logn)$, skriva ut tar $O(n) => O(nlogn)$ totalt. Fungerar bra om man i insättningmetoden låter bli att lägga in ett kort om man upptäcker att det redan fanns med.*

6. Läs in korten till en stack. Upprepa följande till stacken är tom: Poppa första kortet och skriv ut det, poppa sedan resten av korten, utom dom som är identiska med första kortet, till en andra stack och poppa tillbaka rubbet till första stacken. *Första kortet jämförs med resten (som vid urvalssortering) =>$O(n^2)$ totalt. Fungerar om man när man poppar till den andra stacken slänger bort alla kort som är identiska med det första.*

Om två moment utförs efter varann, t ex inläsning $O(n)$ och sortering $O(n^2)$ så blir den totala komplexiteten summan, dvs $O(n) + O(n^2)$. Produkten $O(n^3)$ skulle vi bara få om vi sorterade n gånger.

7. *Arbeta med Pokémon*

Du är så uppskattad i ditt arbete som pokémontränare att du kan välja och vraka bland erbjudandena. Givet ett antal förslag (anbudsgivare, startdag, slutdag, arvode) vill du planera nästa månad så att den blir så lönsam som möjligt.

(4p) Rita först upp problemträdet (roten och några noder i de första två nivåerna räcker).

(12p) Föreslå en algoritm som finner det schema som ger störst inkomster. Motivera ditt val av algoritm och beskriv hur du skulle implementera algoritmen.

I varje nod finns schema och totalinkomst. Tomt schema i roten, barnen har varsitt anbud i schemat, barnbarnen har två.

Man kan använda djupetförstsökning för att gå igenom alla möjligheter och använda en maxinkomst-variabel som uppdateras varje gång vi hittar ett lönsammare schema. Breddenförstsökning fungerar också, men man får inte avbryta direkt när den hittar ett fullt schema - vi söker ju inte efter ett schema med så få anbud som möjligt. Bästaförst-sökning fungerar också, om man går igenom hela trädet, men det är mindre effektivt eftersom prioritetskö inte är lika snabb som stack/kö.

I beskrivningen av implementationen måste just det här problemets klurigheter tas med, t ex hur sönerna representeras, hur man går igenom anbuden, hur man vet att schemat är fullt osv.

Många har en girig lösning (ibland maskerad som en bästaförst-sökning som bryter så fort den kommer till slutet). Ibland är lösning girig rakt igenom (söker bara en stig i trädet), och ibland söker den hela trädet fram till näst sista nivån men bryter så fort den fyllt schemat. Det ger inte nödvändigtvis den bästa lösningen eftersom det kan finnas krockande anbud som skulle givit större inkomster. Man brukar få mellan två och sex poäng för en sådan lösning. (Man kan dessutom få fyra poäng för problemträdet om det är riktigt.)

8. *Pokémonturnering*

I en Pokémonturnering får varje deltagare spela mot en annan i första omgången. Förlorarna slås ut och vinnarna fortsätter till nästa omgång osv ända tills en ensam segrare
(4p) blir kvar. Rita upp ett litet exempel på en turnering. Hur många matcher kommer att

spelas om turneringen har n deltagare?

(10p) Det är inte rättvist att den som förlorade i finalen ska få silverpokalen, eftersom den kan ha haft turen att möta halvdana motståndare på vägen till semifinal. Hur ska man gå tillväga för att hitta en värdig andrapristagare och hur många matcher krävs?

Så här kan turneringen se ut med sju deltagare:

```
Mudkip V Sceptile   Slugma V Aron   Latias V Hitmonlee   Manetric

     Sceptile     V      Aron            Latias    V     Manetric

               Sceptile           V            Manetric

                                  Sceptile
```

Med n deltagare i turneringen blir det $n - 1$ *matcher.*

För att hitta en värdig andrapristagare kan man se efter vilka vinnaren mött genom hela turneringen och låta dom möta varandra. Denna silverturnering får $\log n$ *deltagare och* $\log n - 1$ *matcher.*

Labbtider

För dig som har labbar kvar finns det sex labbtillfällen under de två första veckorna i november då du kan få hjälp och redovisa:

Datum	**Tid**	**Sal**
Tor 3 nov	15-17	Gul, Brun
Fre 4 nov	08-10	Gul, Brun
Ons 9 nov	08-10	Gul, Brun
Tor 10 nov	13-15	Gul, Brun
Fre 11 nov	10-12	Gul, Brun
Mån 14 nov	08-10	Gul, Brun

OM DU INTE REDAN GJORT DET – FYLL I KURSENKÄTEN PÅ WEBBEN!