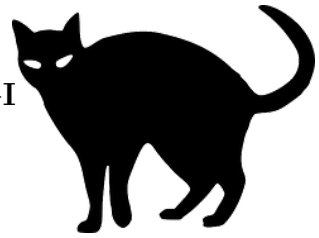
KTH CSC, Linda Kann

DD1320/1325 TENTAMEN I TILLÄMPAD DATALOGI



Fredagen den 23 oktober 2015 kl 8–12

Hjälpmedel: En algoritmbok och ditt eget formelblad. För betyg E krävs att alla E-uppgifter är godkända, för betyg C att alla E- och C-uppgifter är godkända, för betyg A att alla uppgifter är godkända. Lycka till!

1. *Oturligt träd*

Betyg E. Du har startat lotteriet TURIOTUREN där otursförföljda kan vinna på otursnummer! För att snabbt kunna undersöka om ett tal är ett oturstal vill du använda ett binärt sökträd.

Lägg in följande oturstal i ett binärt sökträd, i tur och ordning:
3 4 9 13 17
Rita upp ditt träd (oturligt nog blir det inte balanserat).
Skriv sedan ut trädet i inorder, preorder och postorder.

(10 min)

2. *Otursautomat*

Betyg E. Det vore ju oturligt om ditt företagsnamn redan var upptaget. Konstruera en KMP-automat som söker efter `TURIOTUREN`

Rita upp automaten och ange next-vektorn.

(10 min)

3. *Otur i sökning*

Betyg E. Binärträdssökningen gick inte så snabbt som du hade hoppats, så du satsar på att göra binärsökning i en vektor istället. Visa steg-för-steg vad som händer när vi försöker söka efter talet 17 med binärsökning i vektorn nedan.

12	3	28	9	17

Förklara varför det inte gick bra, och berätta hur man kan lösa problemet.

(10 min)

4. *Vilken datastruktur?*

Betyg E. Att använda rätt datastruktur till algoritmen handlar mer om kunskap än tur. Skriv upp tre olika korrekta meningar givet alternativen nedan:
Vid *xxx* är det lämpligt att använda en *yyy*.

xxx	***yyy***
breddenförstsökning	stack
djupetförstsökning	prioritetskö
bästaförstsökning	kö

(10 min)

VGV

5. *Sortering med otur*

Betyg E. Du använder Quicksort för att sortera men har otur - tiden blir inte blir $O(n \log n)$ utan snarare $O(n^2)$. Hur kan det bli så? Ge ett konkret exempel (med tal som du valt själv) och demonstrera fenomenet. Du kan anta att det är det alltid är det första värdet som väljs som pivotelement.

(10 min)

6. *Oturssyntax*

Betyg E. Givet syntaxen nedan:

```
<skrock> ::= <pred> " " <obj> " "  | <pred> " " <obj> " " <indobj>
<pred> ::= "lägga" | "se" | "spilla" | "krossa" | "gå under"
<obj> ::= "nycklar" | "en svart katt" | "salt" | "en spegel" | "stegen"
<indobj> ::= "på bordet" | "på vägen" | "på golvet"
```

Avgör vilken/vilka av följande meningar som accepteras:

```
se en svart katt
gå under på golvet
lägga salt på vägen
```

(10 min)

7. *Hashning*

Betyg C. Vi har en hashtabell med 10 platser, och hashfunktionen
$h(s) = 2 * len(s)\%10$

Demonstrera inhashning av följade fem fembokstaviga ord
prova
kliva
ramla
gråta
snyta
med linjär respektive kvadratisk probning. Till din hjälp i den kvadratiska probningen kan du använda följande tabell (du behöver inte proba längre än den räcker).

k	k^2	$\sum_{i=1}^{k} i^2$
1	1	1
2	4	5
3	9	14
4	16	30
5	25	55

Diskutera fördelar och nackdelar med respektive *krockhanteringsmetod* i det här fallet. (Hashfunktionen förbättrar vi en annan gång.)

(20 min)

8. *Kryptering*

Betyg C. One-time pad är en krypteringsmetod som fungerar så här:
Givet ett meddelande på binär form

- Slumpa fram en nyckel med lika många binära siffror som meddelandet har
- Gör bitvis *xor* mellan meddelandet och nyckeln

Att ta *xor* med samma nyckel på det krypterade meddelandet avkodar. Exempel:

```
meddelande = 100111
slumpad nyckel = 100101
Kryptering: 100111 xor 100101 = 000010
Dekryptering: 000010 xor 100101 = 100111
```

Gör en jämförelse mellan one-time pad och RSA ur tre relevanta aspekter.

(20 min)

9. *Kontrollera oturstal i en kö*

Betyg A. Oturstalen F(n) definieras som
$F(0) = 0$
$F(1) = 1$
$F(n) = F(n-1) + F(n-2)$ för $n > 1$

Att beräkna oturstalen rekursivt kan vara tidsödande. Som tur är har du en kö som innehåller talen - du ska bara verifiera att det verkligen är en oturs-följd.

Konstruera en algoritm som avgör om en kö innehåller en stigande följd av oturstal, till exempel 0, 1, 1, 2, 3, 5, 8, 13, 21, 34

Du får förutsätta att kön innehåller minst tre tal.
Du får bara använda kön abstrakt, dvs med kö-operationerna `enqueue`, `dequeue` och `isEmpty`. Efteråt ska kön vara oförändrad.

(30 min)

10. *Kodning mot otur*

Betyg A. Ett meddelande (bestående av binärt kodade tecken) som skickas från en dator till en annan kan korrumperas om man har otur; en bit som var ett kan bli noll och vice versa. För att minska risken att ett kodord misstas för ett annat bör inte kodorden vara för lika.

Två kodord har Hammingavstånd d om dom skiljer sig åt i d bitar.
En uppsättning av flera kodord har Hammingavstånd d om det minsta Hammingavståndet mellan två kodord bland dessa är d.

Anta att du har en uppsättning med n kodord, där varje kodord är m bitar långt. Du kan förutsätta att Hammingavståndet för den givna uppsättningen koder är 1.

Konstruera en algoritm för att lägga till *så få bitar som möjligt* till kodorden, så att Hammingavståndet för kodorden blir d.

Till din hjälp har du en funktion *Hamming(koder)* som beräknar Hammingavståndet för en uppsättning kodord.

(30 min)