Linköpings Tekniska Högskola
Institutionen för Datavetenskap
Magnus Nielsen

24 oktober 2017

TDDE22 & 725G96
Datastrukturer och algoritmer
Datortentamen (DAT1)
2017-10-24, 08–12

Examinator:	Erik Nilsson
Jour:	Magnus Nielsen (telefon 073-0822614).
Antal uppgifter:	7, OpenDSA inräknat.
Max poäng:	40 poäng
Preliminära gränser:	**TDDE22:** betyg 5 = 35p, 4 = 27p, 3 = 20p. **725G97:** betyg G = 20p, VG = 30p. (Kan komma att justeras, i båda kurserna)
Hjälpmedel:	Inga hjälpmedel tillåtna!
Bonuspoäng:	Eventuella intjänade bonuspoäng kommer att tillgodoräknas efter att gränsen för 3/G uppnåtts.

VÄNLIGEN IAKTTAG FÖLJANDE

- Du får själv välja om du vill skriva din lösning på papper eller på dator.
- Lösningar till olika problem skall placeras enkelsidigt på separata blad, eller i egen fil. Skriv inte två lösningar på samma papper eller i samma fil. Delproblem får dela papper / fil.
- Om inte annat framgår ska indexering av arrayer / listor börja från 0.
- Papper: Sortera lösningarna innan de lämnas in.
- Filer: Skicka in som lösning till rätt problem i tentaklienten, och döp filen till ett passande namn (exempelvis uppg1.txt).
- MOTIVERA DINA SVAR ORDENTLIGT: avsaknad av, eller otillräckliga, förklaringar resulterar i poängavdrag. Även felaktiga svar kan ge poäng om de är korrekt motiverade.
- Om ett problem medger flera olika lösningar, t.ex. algoritmer med olika tidskomplexitet, ger endast optimala lösningar maximalt antal poäng.
- Papper: Lämna plats för kommentarer.
- SE TILL ATT DINA LÖSNINGAR/SVAR ÄR LÄSLIGA.

Lycka till!

1. **OpenDSA** (12 p)

Efter inloggning i datortentasystemet finns i startmenyn för Linux Mint:

- **"OpenDSA"** öppnar URL för tentamensversion av OpenDSA i Chromium

Logga in på OpenDSA med ditt tilldelade SC-nummer, och använd ditt personnummer som lösenord. SC-numret kan du alltid läsa av från din tenta live-klient som öppnades när du loggade in på datorn. Numret står i fältet för KlientID. Håll din klient öppen, så att du snabbt och smidigt kan ställa frågor till jourhavande lärare, eller skicka in uppgifter.

Genom att klicka på ditt inloggningsnamn kan du hela tiden se i betygsboken hur många av de poänggivande uppgifterna du löst hittills. När du har full poäng i betygsboken är du färdig med den här uppgiften och kan logga ut. På den här uppgiften behöver du inte skicka in något via tentamenssystemet.

2. **Sorteringsalgoritmer** (2 p)

Sorteringsalgoritmer. Svaren behöver ej motiveras.

Givet följande osorterade array:

0	65	75	29	6	68	8	79	36	37	13	63	31	11	94

Efter några iterationer av några olika sorteringsalgoritmer (iteration i bemärkelse fullständig körning av inre loop, alternativt rekursivt anrop) har vi dessa resultat:

A	0	6	8	11	65	75	29	13	68	31	79	36	37	63	94
B	0	6	8	29	65	68	75	79	36	37	13	63	31	11	94
C	0	29	65	75	6	68	8	79	36	37	13	63	31	11	94
D	0	8	6	11	75	68	65	79	36	37	13	63	31	29	94

Matcha delvis sorterad array mot en algoritm. Felaktig matchning ger minuspoäng, dock kan uppgiften ej ge total minuspoäng. För full poäng krävs 4 korrekta svar.

(a) Bubblesort, med nedbubbling av minsta element

(b) Insertionsort

(c) Quicksort, elementet längst till höger i partitionen används som pivot

(d) Selectionsort

3. **Hashtabeller** (4 p)

Vi har en hashtabell med linjär adressering, med några element instoppade. Hashfunktionen är h(x) = x mod size.

9	1	None	3	None	5	15	7	8	18

(a) I vilken ordning har elementen stoppats in? Det finns flera lösningar. Ge fyra korrekta lösningar för maxpoäng. (2)

(b) Om vi tar bort 8 från den färdiga hashtabellen (remove / delete), hur kommer tabellen se ut? Det finns flera olika lösningar. Ge två olika, och förklara dem, för maxpoäng (2)

4. **Sortering** (6 p)

En ny bank, Banks'R'Us'Sorta, har öppnat portarna. De har köpt in ett specialdesignat, toppmodernt och superdyrt system till sina datorer och servrar. Dessvärre glömde de några viktiga detaljer i sin specifikation, och företaget de anlitade, Super Honest Hackers, visade sig vara skattefuskare som numera inte går att få tag på då samtliga inblandade har flytt landet och bosatt sig på en solig ö i ett land utan utlämningsavtal med Sverige. En av de viktiga detaljerna banken glömde specificera var att de måste kunna bokföra summan av alla transaktioner för varje dag (såväl uttag och utgående lån, som insättningar och betalningar).

Då det hört att f.d. DALG-studenter är något av det bästa man kan få tag på om man vill ha effektiva och snabba databehandlingslösningar anställer de dig för att lösa problemet. Företaget som designade systemet har dessvärre gjort minimalt arbete och databasen låter dig bara specificera år i sökningarna, inte månad och datum. Därtill har databasen ingen inre sortering utan den skickar tillbaka en lista med samtliga transaktioner som gjorts för det specificerade året i en till synes slumpmässig ordning. I varje transaktion sparas ett ID (oändligt stort heltal: första transaktionen för året får ID 1, andra får ID 2, osv), ett datum (datumklass), belopp (positivt eller negativt), kontoinformation, etc.

Du anser att listan måste sorteras för att underlätta summeringen, och du har lyckats lista ut att det första ID't på en ny dag avrundas uppåt till närmaste högre miljon för att kunna hålla isär transaktionerna för olika dagar. Vi kan förutsätta att minst en transaktion utförs varje dag. Efter noggrannt avvägande väljer du att sortera baserat på datum.

(a) Vilken eller vilka sorteringsalgoritmer väljer du att använda? Motivera. (3)

(b) Vilken tidskomplexitet har din sorteringslösning? (1)

(c) Antag att vi inte har tillgång till datum. Hur förändras sorteringssituationen? Vilka nya val och avvägningar gör du? (2)

5. **Algoritmer och Tidskomplexitet** (6 p)

(a) Beskriv hur algoritmen merge sort fungerar. (3)

(b) Beräkna tidskomplexiteten med avseende på n för följande funktion: (1)

```
public int calc1(n) {
     res = 0;
     for (int i = 0; i < n+n; i++) {
          res += i;
     }
     return res;
}
```

(c) Beräkna tidskomplexiteten med avseende på n för följande funktion: (1)

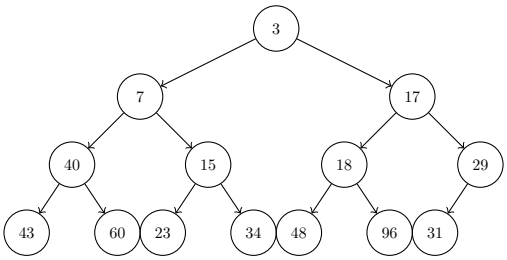
```
public int calc2(n) {
     res = 0;
     for (int i = 0; i < n; i++) {
          res++;
     }
     return res;
}
}
```

(d) Beräkna tidskomplexiteten med avseende på n för följande funktion: (1)

```
public int calc3(n) {
     res = 0;
     for (int i = n; i > 0; i=i/2) {
          res += calc2(i);
     }
     return res;
}
```

6. **Träd** (6 p)

Studera följande träd:



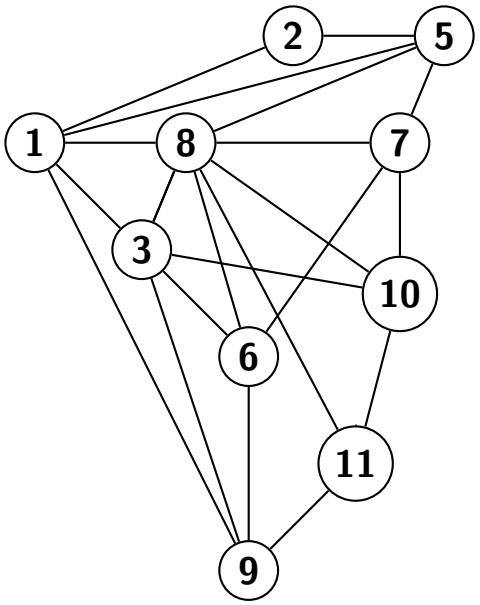
(a) Är trädet ett binärt sökträd? Motivera! (2)

(b) Är trädet balanserat? Motivera! (2)

(c) Är trädet en min-heap, max-heap eller ingetdera? Motivera! (2)

7. **Graftraversering** (4 p)

Studera följande graf:



Det räcker med en korrekt lösning på a) och b). Du behöver alltså inte räkna upp flera tänkbara lösningar.

(a) I vilken ordning kommer noderna att traverseras med bredden först sökning, om vi utgår från 9? (2)

(b) I vilken ordning kommer noderna att traverseras med djupet först sökning, om vi utgår från 9? (2)