**Algoritmer datastrukturer och komplexitet**
**hösten 2011**

# Ommästarprov 1: Algoritmer

Detta mästarprov är avsett för den som ännu inte är godkänd på mästarprov 1. Det är bara en uppgift och den kan bara ge betyg E. Mästarprovet ska lösas **individuellt** och redovisas både skriftligt och muntligt. Inget samarbete är tillåtet, se vidare hederskodexen.

Du ska lämna in den **skriftliga lösningen** på studerandeexpeditionen (Osquars backe 2, plan 2) senast 4 januari 2012 klockan 14.00 eller skicka den senast 6 januari 2012 klockan 14.00 med e-post till cenny@csc.kth.se. Den **muntliga redovisningen** ska göras 9 eller 11 januari 2012. Boka senast 6 januari tid för muntlig redovisning på kurswebbsidan.

Det är viktigt att du förbereder dig inför den muntliga redovisningen. För att en uppgift ska godkännas ska du kunna förklara och motivera algoritmen muntligt och reda ut eventuella oklarheter.

Läs uppgiften mycket noga så att du inte råkar basera dina lösningar på en missuppfattning. Fråga Viggo om något är oklart.

**Olika sorters heapsortering**
En $d$-heap är en generalisering av en vanlig heap. $d$ står för (maximala) antalet barn till varje nod i trädet. Den vanliga heapen är alltså samma sak som en 2-heap eftersom det är ett binärträd. $d$-heapen har samma regler som en vanlig maxheap:

- barn får inte ha större värde än föräldern,
- alla nivåer i trädet är fyllda, utom möjligen den understa (löven); den understa nivån är fylld från vänster fram till en viss position,
- bara två operationer finns: $insert(x)$ som stoppar in ett nytt element $x$ i heapen och $getmax()$ som plockar ut det största elementet i heapen.

a) Beskriv en lämplig datastruktur som kan lagra en 1-heap. Beskriv sedan operationerna $insert(x)$ och $getmax()$ (med pseudokod) på 1-heapen. Analysera tidskomplexiteten för operationerna (uttryckt i antalet element som redan finns i 1-heapen).

b) En $\infty$-heap består bara av två nivåer. Alla element utom rotelementet ligger på den andra nivån. Beskriv en lämplig datastruktur som kan lagra en $\infty$-heap. Beskriv sedan operationerna $insert(x)$ och $getmax()$ (med pseudokod) på $\infty$-heapen. Analysera tidskomplexiteten för operationerna.

*Vänd!*

Betrakta nu följande algoritm:

DHeapSort($H, v[1..n]$) = // $H$ är en tom D-heap
    **for** $i \leftarrow 1$ **to** $n$ **do** $H.insert(v[i])$
    **for** $i \leftarrow n$ **downto** 1 **do** $v[i] \leftarrow H.getmax()$
    **return** $v$

c) Analysera tidskomplexiteten för DHeapSort då heapen är en 1-heap, en 2-heap respektive en $\infty$-heap. Du får använda kända tidskomplexitetsresultat från kursen i din analys.

d) Visa i vart och ett av de tre fallen (D=1, 2 och $\infty$) steg för steg med figurer hur heapen byggs upp och monteras ned då DheapSort körs med indata $v = [17, 4, 19, 2]$.

e) Analysera hur sorteringsalgoritmen jämför elementen i $v[1..n]$ med varandra i dom tre fallen (1, 2 och $\infty$) och tala för varje fall om vilken sorteringsalgoritm från kursen det motsvarar.

Slutsats: Tre vanliga sorteringsalgoritmer kan ses som specialfall av den mycket enkla algoritmen DHeapSort!