INF102 Algoritmar, datastrukturar og programmering - Innlevering 2 hausten 2013

Dette er ei obligatorisk innlevering **med frist fredag 18. oktober kl. 16:00** som du må få godkjend for å få gå opp til eksamen. Resultatet tel 10% av sluttkarakteren. Innleveringa skal vera individuell. Vi godtar berre innlevering via MiSide si innleveringsmappe, og alle filene skal pakkast til ei zip-fil. Alle filer som ikkje er Java-kode skal vera på pdf-format. For å få full utteljing må algoritmar og program vera lettleselege.

I alle implementasjonsoppgåvene kan du fritt gjera bruk av Java-kode som kan lastast ned frå læreboka si heimeside.

Oppgåve 1

a) Tenk deg at du har fått ein sekk med n ulike hengelåsar og like mange nøklar, og at du skal finna kva nøkkel som svarer til kva lås. Kvar nøkkel har same storleik som nøyaktig ein lås, men alle storleikane er ulike. Ved å prøva ein nøkkel i ein lås vil du få vita om nøkkelen er for liten for låsen, for stor for låsen, eller om han passar akkurat. Du kan ikkje samanlikna to nøklar eller to låsar.

Finn ein effektiv metode for å løysa dette problemet.

b) I ein usortert tabell med 5 distinkte verdiar vil vi finna *medianen*, altså elementet som er mindre enn to andre element og større enn to andre element. Ved å bruka prinsippet til ein kjent sorteringsalgoritme kan du finna medianen v.h.a. berre sju samanlikningar. Forklar korleis dette kan gjerast og kva sorteringsalgoritme som ligg til grunn. Merk at du kan avbryta sorteringa så snart du har funne medianen, slik at du ikkje treng sortera heile tabellen.

c) Kall tabellelementa x, y, z, u og v, og forklar korleis du kan finna medianen med berre seks samanlikningar. Om du vil kan du gjerne basera forklaringa på figurar (hassediagram). *Hint*: Start med å samanlikna x med y og z med u, for deretter å samanlikna den største av x og y med den største av z og u.

d) I standardvarianten av `Quicksort` blir første tabellelement, `a[0]`, brukt som *partisjonselement*, dvs. algoritmen flyttar om på elementa slik at `a[0]` får ein deltabell med mindre element til venstre og ein deltabell med større element til høgre for seg. I alternative variantar prøver ein å velja partisjonselementet slik at dei to deltabellane blir omlag like store.

I denne oppgåva skal vi sjå på ein variant av Quicksort der vi brukar medianen til 5 tabellelement som partisjonselementet. Når vi har funne indeksen `med` til medianen, byter vi om `a[lo]` og `a[med]`, og partisjonerer som før. Medianen skal finnast blant 5 jamnt fordelte tabellelement, dvs blant elementa `a[0]`, `a[4]`, `a[8]`, `a[12]` og `a[16]` i tilfellet `lo=0` og `hi=16`.

Skriv og implementer ein slik variant av `Quicksort`. Bruk gjerne javakode frå læreboka som utgangspunkt for implementasjonen. For å finna medianen, implementer ein metode `int median5(Comparable[] a, int i, int j, int k, int l, int m)` som returnerer indeksen til medianen av `a[i]`, `a[j]`, `a[k]`, `a[l]` og `a[m]`.

Du skal kunna kjøra programmet frå kommandolina slik

```
java Quick5 < tiny.txt
```

og programmet skal då skriva ut (fila tiny.txt er henta frå læreboka):

```
A
E
E
L
M
O
P
R
S
T
X
```

e) Samanlikn kjøretida til den nye varianten av `Quicksort` med kjøretida til standardvarianten henta frå læreboka ved hjelp av eksperiment på tilfeldig genererte tabellar. Start med tabellengde $n = 8000$, og i kvart eksperiment doblar du n, like til `Quicksort` treng meir enn eit minutt på å sortera tabellen. I kvart eksperiment skal 10 tilfeldige tabellar genererast, og begge algoritmane skal kjørast på desse tabellane.

Plott kjøretidene og kommenter. Løner det seg å bruka tid på å finna medianen?

Oppgåve 2

I denne oppgåva skal vi samanlikna kjøretidene for innsetting i symboltabellar for to ulike implementasjonar: Lenka lister og binære søketre.

a) Eit program les n element med distinkte nøkkelverdiar frå ei fil, og set elementa inn i ein symboltabell. Når symboltabellen er implementert som ei kjeda liste, viser det seg at innsetting av dei n elementa tar m gonger så lang tid som når symboltabellen er implementert som eit binært søketre. Kor stor er n? Svar på spørsmålet for $m = 10$, $m = 100$ og $m = 1000$, og gå ut frå at kjøretida er proporsjonal med talet på samanlikningar som blir gjort.

Dersom du treng løysa likningar på forma $x = c \lg x$, kan du bruka vedlagte program (sjå fillageret på MiSide) slik (konstanten c blir gitt inn via kommandolina):

```
java Lg c
```

b) Last ned implementasjonane med lenka liste og binært søketre frå læreboka. Modifiser `main`-funksjonane slik at dei genererer n flyttal og set dei inn i ein symboltabell. For å gjera det enkelt kan du la datafelta (`val`) vera like nøkkelfelta (`key`). Samanlikn kjøretidene for dei to implementasjonane ved å bruka verdiar for n rundt det du fann i oppgåve a, og vurder om observasjonane samsvarer med den teoretiske analysen du gjorde i oppgåve a.