Linköpings Tekniska Högskola
Institutionen för Datavetenskap
Tommy Färnqvist

23 oktober 2012

TDDC70/TDDC91 Datastrukturer och algoritmer
Tentamen 2012-10-23, 8–13 (KÅRA, T1, T2)

Examinator: Tommy Färnqvist
Jour: Tommy Färnqvist (telefon 070 4547668).
Max poäng: 26 poäng (betyg 5 = 23p, 4 = 18p, 3 = 13p)
Hjälpmedel: INGA HJÄLPMEDEL TILLÅTNA!!!

VÄNLIGEN IAKTTAG FÖLJANDE

- Lösningar till olika problem skall placeras enkelsidigt på separata blad. Skriv inte två lösningar på samma papper.
- Sortera lösningarna innan de lämnas in.
- 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.
- SE TILL ATT DINA LÖSNINGAR/SVAR ÄR LÄSBARA.
- Lämna plats för kommentarer.

BONUSPOÄNG

- En student som löst de tio föreläsningsrelaterade uppgifterna i 2012 års DALG-mästerskap innan denna tentamens starttid får bonuspoäng motsvarande 3 poäng på denna tentamen.
- Uppgifterna där bonuspoängen kan användas är: uppgift 2, 3, 4, 6 och 8.
- En student behöver inte göra något för att eventuella bonuspoäng ska tas med i poängberäkningen, detta sköts automatiskt vid rättning av tentamen.

Lycka till!

1. Vilka av följande påståenden är sanna och vilka är falska? Svar utan motivering ger inga poäng. (3 p)

   (a) Det finns två funktioner $f(n)$ och $g(n)$ sådana att varken $f(n) \in O(g(n))$ eller $g(n) \in O(f(n))$ gäller. (1)

   (b) $n! \in O(2^n)$. (1)

   (c) I en oriktad enkel graf med n noder och m bågar gäller alltid att $m \in O(n \log n)$. (1)

2. Kolumnen nedan till vänster innehåller strängar som ska sorteras, kolumnen nedan till höger är strängarna i sorterad ordning, övriga kolumner visar innehållet vid något mellanliggande steg i ett anrop till de fyra sorteringsalgoritmerna listade nedan. Para ihop varje algoritm med rätt kolumn. Använd varje algoritm exakt en gång. (2 p)

```
COS ARC ARC ARC REL ARC
PHY CHE CHE ART PHY ART
ELE COS COS CEE PHY CEE
COS COS COS CHE ELE CHE
MAT ECO ECO CHM PHI CHM
MOL ELE EEB COS ORF COS
LIN GEO ELE COS ORF COS
ARC LIN ELE COS COS COS
ECO MAE ENG COS ELE COS
CHE MAT GEO COS EEB COS
MAE MOL LIN COS MUS COS
GEO PHY MAE ECO GEO ECO
ORF ORF MAT ORF ORF EEB
EEB EEB MOL EEB MAT EEB
ENG ENG ORF ENG LIN ELE
ELE ELE PHY ELE COS ELE
COS COS ART MOL COS ELE
ELE ELE CEE ELE ECO ENG
CEE CEE COS ELE CEE GEO
EEB EEB EEB EEB CHE LIN
ART ART ELE PHY ART MAE
MUS MUS MUS MUS MAT MAT
PHI PHI ORF PHI MAE MAT
ORF ORF PHI ORF ELE MOL
COS COS COS GEO COS MUS
PHY PHY PHY PHY MOL ORF
COS COS COS LIN COS ORF
MAT MAT MAT MAT EEB ORF
CHM CHM CHM MAT CHM PHI
ORF ORF ORF ORF ENG PHY
COS COS COS MAE COS PHY
REL REL REL REL ARC REL
--- --- --- --- --- ---
1:a 2:  3:  4:  5:  6:b
```

(a) Indata

(b) Sorterad sekvens

(c) Selection-sort

(d) Insertion-sort

(e) Merge-sort

(f) Heap-sort (in-place)

3. Vi använder en array med indexen 0 till 6 för att implementera en öppet adresserad hashtabell av längd 7 med följande tabell som hashfunktion: (3 p)

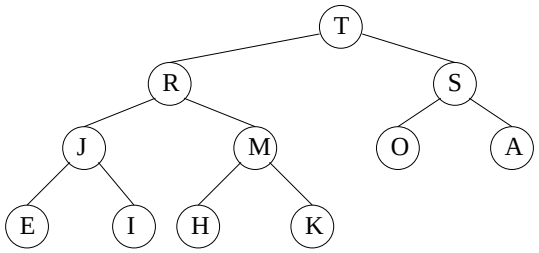
nyckel	hashvärde
A	5
B	2
C	5
D	1
E	4
F	1
G	3

(a) Rita en representation av hashtabellen och dess innehåll efter att vi använt hashfunktionen ovan för att sätta in nycklarna i alfabetisk ordning: A, B, C, D, E, F, G i tabellen. Antag att vi hanterar kollisioner med linjär sondering (*linear probing*). (1.5)

(b) Rita en representation av hashtabellen och dess innehåll efter att vi använt hashfunktionen ovan för att sätta in nycklarna i alfabetisk ordning: A, B, C, D, E, F, G i tabellen. Antag att vi hanterar kollisioner med kvadratisk sondering (*quadratic probing*). (1.5)

4. Nycklarna i den här uppgiften om binära heapar är stora bokstäver och vi använder bokstavsordning för att ordna dessa nycklar. (2 p)

Betrakta följande max-heap representerad som ett binärt träd.



(a) Max-heapen ovan är resultatet av en sekvens av `insert`- och `removeMax`-operationer. Antag att sista operationen i denna sekvens var ett anrop till `insert`. Vilken/vilka nyckel/nycklar skulle kunna vara de(n) som sattes in sist? (1)

(b) Ett anrop till `removeMax` görs på heapen ovan. Rita den resulterande heapen. (1)

5. Randomiserade prioritetsköer. (4 p)

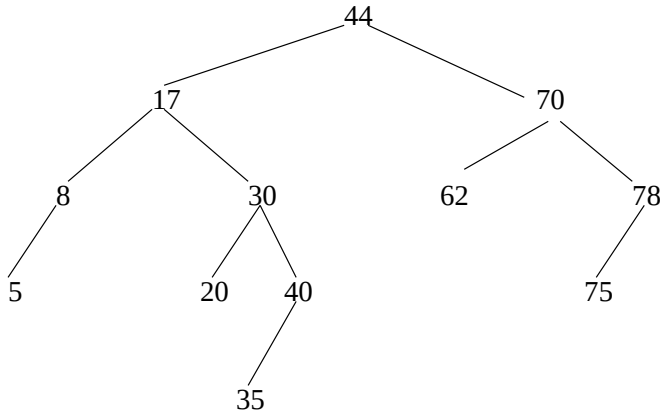
Beskriv, gärna med pseudokod, hur man kan lägga till metoderna `sample()` och `removeRandom()` till en implementation av ADT Priokö. De två metoderna returnerar en nyckel som väljs ut slumpmässigt med likformig sannolikhet bland nycklarna som för tillfället är insatta i prioritetskön, där den senare metoden också tar bort den nyckeln ur prioritetskön.

`PQ()`	skapa en tom prioritetskö
`void insert(Key key)`	sätt in en nyckel i prioritetskön
`Key min()`	returnera den minsta nyckeln
`Key removeMin()`	ta bort och returnera den minsta nyckeln
`Key sample()`	returnera en nyckel vald med likformig sannolikhet
`Key removeRandom()`	ta bort och returnera en nyckel vald med likformig sannolikhet

Antag att du har tillgång till en slumptalsgenereator `Random.uniform(N)` som returnerar slumptal mellan 0 och $N-1$ med likformig sannolikhet. Din implementation av `sample()`

får bara använda konstant tid, medan `removeRandom()` får använda tid proportionell mot $\log N$, där N är antalet nycklar i datastrukturen.

6. Betrakta följande binära träd T: (3 p)



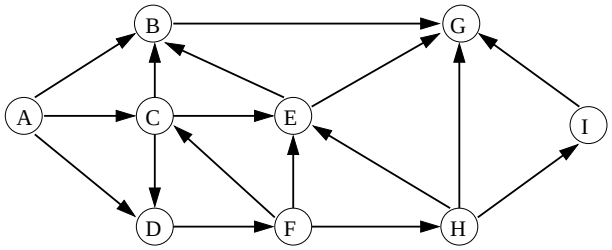
   (a) Trädet T kan ses som ett AVL-träd. Motivera det påståendet genom att hänvisa till definitionen av AVL-träd. (1)

   (b) Se nu T som ett AVL-träd och visa hur följande träd ser ut: $T_1 = Delete(62, T)$. Visa också de eventuella rotationer som behöver utföras efter borttagningen av 62 för att T_1 också ska vara ett AVL-träd. (2)

7. Låt A och B vara två mängder av heltalsnycklar. Vi säger att A *separerar* B om det finns tre nycklar $x < y < z$, sådana att x och z finns i B och y finns i A. Antag att alla nycklar är distinkta, att nycklarna i A lagras i ett balanserat binärt sökträd T_A av lämplig sort och att nycklarna i B lagras i ett balanserat binärt sökträd T_B. Konstruera en algoritm som tar T_A och T_B som indata och avgör ifall A separerar B. Din algoritm ska ha exekveringstid $O(\log n)$, där n är det totala antalet nycklar i A och B, i värsta fallet. Notera att din algoritm får använda sig av ev. interna metoder sökträden har och inte behöver begränsa sig till `insert`, `remove` och `find`. Beskriv din algoritm med pseudokod eller naturligt språk och förklara varför dess värstfallskomplexitet är $O(\log n)$. (4 p)

8. Den här uppgiften handlar om grafer. (5 p)

   (a) Visa i vilken ordning noderna i den riktade grafen nedan besöks av en djupetförstsökning med start i A. Antag att grafen representeras med grannlistor och att grannlistorna är sorterade i bokstavsordning (så att kanten $F \to C$ ligger före kanterna $F \to E$ och $F \to H$ i F:s grannlista). (1)



   (b) Betrakta två noder x och y som båda finns samtidigt på anropsstacken (den som håller reda på alla rekursiva anrop) vid någon tidpunkt i exekveringen av en djupetförstsökning från noden s i en *riktad* graf. Vilka av följande måste vara sanna utsagor? (2)

I. Det finns *både* en riktad stig från s till x *och* en riktad stig från s till y i grafen.

II. Om det *inte* finns någon riktad stig från x till y så finns det en riktad stig från y till x.

III. Det finns *både* en riktad stig från x till y *och* en riktad stig från y till x.

(a) Endast I.

(b) Endast I och II.

(c) Endast I och III.

(d) I, II och III.

(e) Ingen av utsagorna.

(c) Visa hur en exekvering av Dijkstras kortaste väg-algoritm med start i nod A i grafen nedan ser ut. Visa längden av de bästa vägarna funna så långt för varje nod efter varje relaxeringssteg. Rita det motsvarande trädet av kortaste vägar utgående från startnoden. (2)