Linköpings Tekniska Högskola
Institutionen för Datavetenskap
Tommy Färnqvist

12 januari 2013

TDDC70/TDDC91 Datastrukturer och algoritmer
Tentamen (TEN1) 2013-01-12, 14–19 (TER2)

Examinator: Tommy Färnqvist
Jour: Tommy Färnqvist (telefon 070 4547668).
Max poäng: 27 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.

Lycka till!

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

   (a) Om $t_1(n) \in O(f(n))$ och $t_2(n) \in O(f(n))$, så gäller $t_1(n) \in O(t_2(n))$. (1)

   (b) $4^{n-1} \in O(3^n)$ (1)

   (c) $2^{3\log_2(n)} \in \Omega(n^2)$ (1)

2. Analysera den asymptotiska exekveringstiden för nedanstående kodsnuttar. (3 p)

   (a) (1.5)

```
sum = 0;
for (i = 0; i < n; i++)
    for (j = 0; j < i * i; j++)
        for (k = 0; k < j; k++)
            sum++;
```

   (b) (1.5)

```
sum = 0;
for (i = 1; i < n; i++)
    for (j = 1; j < i * i; j++)
        if (j % i == 0)
           for (k = 0; k < j; k++)
               sum++;
```

3. Köer och stackar (5 p)

   (a) Antag att en kö Q är implementerad med hjälp av en ringbuffer. Vad är den minsta storleken ringbuffern kan ha om följande operationer ska ge korrekt beteende: (1)

   $$enqueue(5, Q), enqueue(3, Q), dequeue(Q), enqueue(2, Q), enqueue(8, Q), dequeue(Q), dequeue(Q), enqueue(9, Q), enqueue(1, Q), dequeue(Q), dequeue(Q)$$

   Motivera ditt svar.

   (b) Visa vilka värden som returneras genom att utföra sekvensen av operationer ovan. (0.5)

   (c) Visa hur ringbuffern (av minimal storlek) ser ut efter att sekvensen av operationer ovan har utförts. (0.5)

   (d) Föreslå en datastruktur som stödjer stackoperationerna `push` och `pop` samt en tredje operation `findMin` som returnerar det minsta elementet i datastrukturen, samtliga i $O(1)$ värstafallstid. (1.5)

   (e) Visa att om vi lägger till en fjärde operation `deleteMin` som hittar och tar bort det minsta elementet, så måste minst en av de fyra operationerna använda $\Omega(\log n)$ tid. (1.5)

4. Binära sökträd (3 p)

   (a) Rita de binära sökträd vars postordertraverseringar ger följande sekvenser av nycklar: (1.5)

   - 6, 17, 13, 23, 20, 30, 31, 27
   - 5, 16, 10, 21, 18
   - 6, 16, 13, 20, 22

   Vilka av dem, om några, är AVL-träd?

(b) Visa resultatet av att sätta in 2, 1, 4, 5, 9, 3, 6, 7 i ett initialt tomt AVL-träd. (1.5)

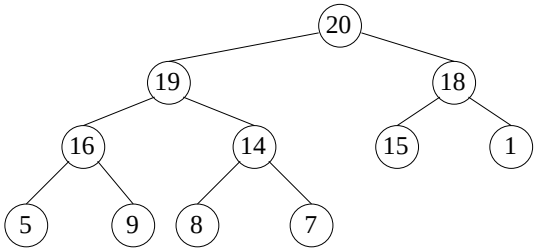
5. Kolumnen nedan till vänster innehåller strängar som ska sorteras, kolumnen nedan till höger är strängarna i sorterad ordning, övriga kolumner visar innehållet vid något mellanliggande steg i ett anrop till de fem sorteringsalgoritmerna listade nedan. Para ihop varje algoritm med rätt kolumn. Använd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

```
john aviv aviv yort alan alan
tzha azhu azhu tzha andy andy
fyau ctai ctai vyas anna anna
ctai ddix ddix ravi aviv aviv
nbal fyau fyau sida azhu azhu
ddix john john oleg azhu azhu
sguo kuan kuan sguo ctai ctai
azhu nbal nbal peck ddix ddix
aviv sguo oleg ctai fyau fyau
sida sida sguo nbal john john
kuan tzha sida kuan kuan kuan
vyas vyas tzha lily kwak kwak
oleg kwak vyas fyau oleg levy
levy levy levy levy levy lily
kwak muir kwak kwak vyas muir
muir oleg muir muir muir nbal
peck peck peck azhu peck oleg
ravi ravi ravi aviv ravi peck
alan alan alan alan sida ravi
yort andy yort john yort sguo
azhu anna azhu azhu nbal sida
andy azhu andy andy tzha tzha
anna lily anna anna sguo vyas
lily yort lily ddix lily yort
---- ---- ---- ---- ---- ----
1:a  2:   4:   5:   6:   7:b
```

(a) Indata
(b) Sorterad sekvens
(c) Selection-sort
(d) Insertion-sort
(e) Merge-sort
(f) Heap-sort (in-place)

6. Heapar (3 p)

(a) Betrakta följande max-heap representerad som ett binärt träd. (1)



Sätt in nyckeln 26 i den binära heapen ovan. Illustrera vilka operationer som utförs och hur slutresultatet ser ut.

(b) Ett anrop till `removeMax` genomförs på den binära heapen som blir svaret i (a). Illustrera vilka operationer som utförs och hur slutresultatet ser ut. (1)

(c) Sant eller falskt? Givet en max-heap med n distinkta nycklar blir resultatet av att sätta in en nyckel som är större än de n nycklarna i heapen och sedan anropa `removeMax` exakt samma heap som den vi ursprungligen hade. (1)

7. Vi använder en array med indexen 0 till 10 för att implementera en öppet adresserad hashtabell av längd 11 med $h(k) = k \bmod 11$ som hashfunktion. Rita en representation av hashtabellen och dess innehåll efter var och en av följande operationer: (4 p)

$$Insert(15, c),\ Insert(4, a),\ Insert(26, b),\ Delete(15),\ Insert(5, d),\ Insert(4, e)$$

(a) när linjär sondering (*linear probing*) används för att hantera kollisioner (2)

(b) när dubbel hashning, med $h'(k) = 5 - (k \bmod 5)$ som sekundär hashfunktion, används för att hantera kollisioner. (2)

I båda fallen hanteras borttagning med *deleted bit*-tekniken.

8. En sammanhängaande oriktad graf $G = (V, E)$ kallas *2-färgbar* om och endast om det finns en funktion $f : V \to \{\mathsf{red}, \mathsf{blue}\}$ sådan $f(v) \neq f(w)$ när det finns en båge mellan v och w; med andra ord, inga grannar tilldelas samma färg. Konstruera en algoritm med polynomisk värstafallstid som avgör ifall en sammanhängande oriktad graf är 2-färgbar eller ej. Förklara noggrannt varför din algoritm fungerar och varför den bara använder polynomisk tid. Tips: använd en girig approach. (4 p)