Linköpings Tekniska Högskola
Institutionen för Datavetenskap
Tommy Färnqvist

29 oktober 2013

TDDI16 Datastrukturer och algoritmer
Tentamen (TEN1) 2013-10-29, 14–18 (G32, G33, TER4)

Examinator: Tommy Färnqvist
Jour: Tommy Färnqvist (telefon 070 4547668).
Max poäng: 22 poäng (betyg 5 = 19p, 4 = 15p, 3 = 11p)
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!

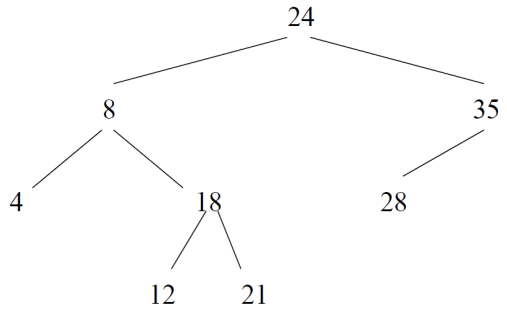
1. Vilka av följande påståenden är sanna och vilka är falska? Svar utan motivering ger inga poäng. (4 p)

   (a) Låt A vara en array med heltal ordnade i icke-ökande ordning. Då representerar A en giltig heap. (1)

   (b) Räknesortering (Counting Sort) kan sortera varje sekvens av indata av längd n i $O(n)$ tid. (1)

   (c) Definiera $f_n = f_{n-1} + f_{n-2}$, med $f_0 = 0$ och $f_1 = 1$. (f_n är det n:te Fibonaccitalet.) Då gäller $f_n \in \Omega\left(\left(\frac{3}{2}\right)^n\right)$. (2)

2. Betrakta följande binära sökträd. (4 p)



   (a) Antag att trädet ovan är ett AVL-träd och visa (i) hur trädet ser ut efter att 10 satts in i trädet, och (ii) hur trädet ser ut efter påföljande borttagning av 28. (2)

   (b) ~~Utför samma operationer under antagandet att trädet ovan är ett splay-träd.~~ (2)

3. En dequeue (Double Ended QUEue) är en mer flexibel datatyp än både stack och kö — den tillåter insättning och borttagning av data från bägge ändar. Om du behövde implementera en dequeue med hjälp av enbart två instanser av de mindre flexibla datatyperna, vilket skulle bli snabbast: 2 stackar, 2 köer eller 1 stack och 1 kö? Beskriv din implementation och ge tidskomplexiteten för `InsertFront`, `RemoveFront`, `Insert Tail` och `RemoveTail`. (3 p)

4. Ge en effektiv algoritm för att avgöra om det finns ett heltal i sådant att $i = A_i$ i en array av heltal $A_0 < A_1 < \cdots < A_{n-1}$. Vad blir exekveringstiden för din algoritm? (3 p)

5. Betänk en prioritetskö implementerad med hjälp av en heap. Heapen implementeras med en array som lagrar värden på platserna med index 1 till och med n. Antag att det är en max-heap, så att det största värdet lagras på platsen med index 1. (3 p)

   (a) På vilka platser kan det näst största värdet i heapen lagras (under antagandet att alla värden är distinkta)? (1)

   (b) På vilket intervall av platser kan det minsta värdet i heapen lagras (återigen under antagandet att alla värden är distinkta)? (1)

   (c) Vad är tidskomplexiteten i värsta fallet för att hitta det minsta värdet i heapen? (1)

6. Vi använder en array med indexen 0 till 6 för att implementera en hashtabell där kollisioner hanteras med hjälp av separat länkning (separate chaining). Hashfunktionen som används är $h(k) = k \bmod 7$. Visa en representation av tabellen efter att nycklarna 5, 28, 19, 15, 20, 33, 12, 17, 10 satts in i den initialt tomma tabellen. (2 p)

7. Nedan följer en implementation av algoritmen Quick Select. När ett anrop till algoritmen terminerar (inklusive alla rekursiva anrop) finns det efterfrågade värdet på position k i arrayen. (3 p)

```
Q_Select_One ( int A[ ], const int k, const int Left, const int Right )
{
    if( Left < Right ) {
        int Pivot = A [Left];
        unsigned int i = Left, j = Right + 1;

        for( ; ; ) {
            while( A[ ++i ] < Pivot && i<Right );
            while( A[ --j ] > Pivot );
            if( i < j )
                Swap( A[ i ], A[ j ] );
            else
                break;
        }
        Swap( A[ j ], A[ Left ] ); // Move pivot to sorted position.

        if( k < j )       Q_Select_One( A, k, Left, j-1 );
        else if( k > j )  Q_Select_One( A, k, j+1, Right );
    }
}
```

(a) Antag att ett anrop till funktionen ovan görs som `Q_Select_One( A, 3, 0, 8)` med följande innehåll i `A`. (1)

0	1	2	3	4	5	6	7	8
14	24	8	16	32	71	26	25	12

Visa innehållet i `A` när exekveringen av anropet precis ska gå in i villkorssatserna som avgör om ett rekursivt anrop ska göras. Vilket rekursivt anrop, om något, utförs?

(b) Hur många jämförelser gör `Q_Select_One` i värsta fallet som en funktion av både n och k, där n är värdet av `Right-Left+1` i ursprungsanropet? När inträffar det värsta fallet? (2)