Tentamen
Datastrukturer D
DAT 036/INN960

18 december 2009

- Tid: 8.30 - 12.30
- Ansvarig: Peter Dybjer, tel 7721035 eller 405836
- Max poäng på tentamen: 60.
- Betygsgränser, CTH: 3 = 24 p, 4 = 36 p, 5 = 48 p, GU: G = 24 p, VG = 48 p.
- Hjälpmedel: *handskrivna* anteckningar på *ett* A4-blad. Man får skriva på båda sidorna och texten måste kunna läsas utan förstoringsglas. Anteckningar som inte uppfyller detta krav kommer att beslagtas!

  Föreläsningsanteckningar om datastrukturer i Haskell, av Bror Bjerner
- Skriv tydligt och disponera papperet på ett lämpligt sätt.
- Börja varje ny uppgift på nytt blad.
- Skriv endast på en sida av papperet.
- **Kom ihåg:** alla svar ska motiveras väl!
- Poängavdrag kan ges för onödigt långa, komplicerade eller ostrukturerade lösningar.
- Lycka till!

1. Vilka av följande påståenden är korrekta och vilka är felaktiga? Motivera!

   (a) Stackar kan implementeras effektivt med länkade listor. (2p)

   (b) En algoritm med komplexiteten $O(n^2)$ är alltid långsammare än en algoritm med komplexiteten $O(n \log n)$. (2p)

   (c) Om ett fält nästan är sorterat är det bättre att använda heapsort än insertionsort. (2p)

   (d) Komplexiteten hos quicksort beror på ordningen mellan elementen i fältet som ska sorteras. (2p)

   (e) Hashtabeller är effektivare än balanserade träd om man ska söka efter data på externminne. (2p)

2. Bestäm O-komplexiteten hos följande kodfragment:

   (a)
   ```
   for (int count = 0; count < n; count++)
   {
      /* instruktionsföljd med tidskomplexitet O(1) */
   }
   ```
   (2p)

   (b)
   ```
   for (int count = 0; count < n; count ++)
   {
      for (int count2 = 1; count2 < n; count2 = count2 * 2)
         {
            /* instruktionsföljd med tidskomplexitet O(n) */
         }
   }
   ```
   (3p)

   (c)
   ```
   for (int i = 0; i < n; ++i)
   {
      for (int j = i; j < n; ++j)
         {
            /* instruktionsföljd med tidskomplexitet O(n) */
         }
   }
   ```
   (3p)

   För att få full poäng räcker det inte att du ger ett korrekt O-uttryck, utan detta uttryck måste också vara så bra som möjligt! Ditt svar måste som vanligt motiveras ordentligt.

3. (a) Mergesort använder sig av operationen `merge` som sammanflätar två sorterade fält till ett sorterat fält. Gör `merge` $O(1)$ jämförelser i bästa fall? Motivera! (2p)

   (b) Har `merge` komplexiteten $O(m + n)$ i värsta fall, om längderna av indatafälten är m och n respekive? Motivera! (2p)

   (c) Diskuterar huruvida `merge` är in-place eller inte! (2p)

   (d) Skriv Haskellprogram

   ```
   T_best_mergesort  : Int -> Int
   T_worst_mergesort : Int -> Int
   ```

   som räknar ut *exakt* hur många jämförelser mergesort gör i bästa fall och värsta fall. Indata är längden av listan.

   Om du vill kan du i stället skriva programmet i Java eller detaljerad pseudokod. (4p)

   (e) Vilka av sorteringsmetoderna insertion sort, selection sort, mergesort och quicksort använder sig av söndra och härska ("divide and conquer") metoden? Svaret måste motiveras utförligt! (2p)

4. (a) Skriv ett program som testar om ett givet binärt träd är ett AVL-träd. Du kan använda Haskell, Java eller detaljerad Java-liknande pseudokod. (6p)

   (b) Vilken O-komplexitet har ditt program? För full poäng krävs att programmet har optimal O-komplexitet för uppgiften. (2p)

   (c) När man sätter in ett element från ett AVL-träd uför man två steg. Först gör man den vanliga operationen för insättning i binära sökträd. Sedan gör man en ombalansering av trädet genom en enkel- eller dubbelrotation. Hur många pekare behöver ändras i värsta fall för denna ombalansering? Motivera! (2p)

5. En oriktad graf kan implementeras som en riktad graf som är *symmetrisk*, dvs om det finns en båge från nod a till nod b så finns det också en båge från nod b till nod a. Vi förutsätter att det inte finns några parallella bågar i den oriktade grafen, dvs det finns högst en oriktad båge mellan varje nodpar a och b.

   (a) Antag först att din riktade graf är implementerad som en grannmatris. Skriv en algoritm i pseudokod som avgör om den riktade grafen är symmetrisk (dvs om den representerar en oriktad graf)! Analysera tidskomplexiteten hos din algoritm! Du ska ange O-komplexiteten beroende på antalet noder n och antalet bågar m. (4p)

   (b) Antag sedan att din riktade graf i stället är implementerad med hjälp av grannlistor och besvara samma frågor som ovan. Dvs skriv en algoritm i pseudokod som avgör om den riktade grafen är symmetrisk (dvs om den representerar en oriktad graf) och analysera O-komplexiteten beroende på antalet noder n och antalet bågar m. (6p)

6. Den ungerske matematikern Paul Erdös (1913-96) skrev totalt 1475 matematiska uppsatser, fler än någon annan i historien. Många av dessa uppsatser skrevs tillsammans med andra: han samarbetade med totalt 511 matematiker. Man säger att dessa 511 har Erdösnummer 1. Vidare säger man att en matematiker har Erdösnummer 2 om han eller hon författat artiklar tillsammans med en matematiker med Erdösnummer 1. Mer allmänt säger man att en matematiker som författat artiklar tillsammans med en matematiker med Erdösnummer n har Erdösnumret $n+1$. (Notera att Erdösnumret är det lägsta tal som tilldelas på detta sätt. Om t ex en matematiker författat artiklar både med Erdös och med en medförfattare till Erdös har alltså matematikern Erdösnumret 1, inte 2.)

   (a) Antag att du har en databas med matematiska artiklar och information om deras författare. Hur kan man skriva ett program som räknar ut en viss matematikers Erdösnummer? Om matematikern inte har något Erdösnummer ska programmet returnera "No Erdös number". Skriv ett program i pseudokod! Om du använder algoritmer och datastrukturer som du lärt dig i kursen behöver du inte ge pseudokod för dem. (7p)

   (b) Vilken O-komplexitet har din algoritm uttryckt som funktion av antalet artiklar och antalet författare i databasen? För full poäng ska komplexiteten vara optimal för detta programmeringsproblem! (3p)