Tentamen
Datastrukturer för D2
TDA 131

20 december 2003

- Tid: 8.45 - 12.45
- Ansvarig: Peter Dybjer, tel 7721035 eller 405836
- Max poäng: 60. Vart och ett av de sex problemen ger maximalt 10 p.
- Betygsgränser: 3 = 30 p, 4 = 40 p, 5 = 50 p
- Inga hjälpmedel.
- 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. Följande grannmatris representerar en viktad graf.

	A	B	C	D	E	F
A	-	1	4	5	-	-
B	1	-	2	-	2	-
C	4	2	-	5	6	4
D	5	-	5	-	-	2
E	-	2	6	-	-	6
F	-	-	4	2	6	-

Talen i matrisen representerar bågarnas vikter. Ett streck (-) betyder att bågen saknas.

(a) Rita grafen! Sätt ut vikterna på bågarna.

(b) Man kan använda Dijkstras algoritm för att finna kortaste vägen mellan två noder i en graf. Som i lab 4 vill man ofta både veta vilka noder som ligger på den kortaste vägen och hur lång vägen är (dvs summan av de ingående bågarnas vikter). Beskriv i ord hur Dijkstras algoritm gör för att beräkna dessa två saker. (Du får delpoäng om du bara kan beräkna längden men inte nodlistan.) Du behöver inte ge fullständig pseudokod - det räcker om du förklarar vilka datastrukturer du använder och förklarar noggrannt vilken roll de spelar under beräkningen.

.

(c) Dijkstras algoritm arbetar stegvis. Visa hur algoritmen fungerar genom att visa vilka mellanresultat (tillstånd hos datastrukturerna) som Dijkstras algoritm (enligt din beskrivning ovan) lagrar för att kunna returnera den kortaste vägen och denna vägs längd mellan A och F i grafen ovan.

(d) Är det i allmänhet lämpligt att använda en grannmatris för att representera en graf när man implementerar Dijkstra! Varför eller varför inte? Motivera med hjälp av O-komplexiteter.

2. Följande tre algoritmer utför samma uppgift. Givet ett fält med heltal (som kan vara både positiva och negativa) räknar de ut den maximala summan av elementen i en delsekvens (utan gap). Din uppgift är att tala om vilken asymptotisk tidskomplexitet de tre algoritmerna har. Du ska ange bästa O-komplexitetsklass för exekveringstiden uttryckt som funktion av fältets storlek n (`a.length` i koden nedan). Kom ihåg att alla svar måste motiveras ordentligt!

   (a)
```
public static int maxSubsequenceSum(int [] a) {
  int maxsum = 0; int thisSum = 0;
  for(int i = 0, j = 0; j < a.length; j++) {
    thisSum += a[j];
    if(thisSum > maxSum)
      {maxSum = thisSum; seqStart = i; seqEnd = j}
    else if(thisSum < 0) {
      i = j + 1;
      thisSum = 0
      }
    }
  return maxSum;
}
```

   (b)
```
public static int maxSubsequenceSum(int [] a) {
  int maxsum = 0;
  for(int i = 0; i < a.length; i++) {
    int thisSum = 0;
    for(int j = i; j < a.length; j++) {
      thisSum += a[j];
      if(thisSum > maxSum)
        {maxSum = thisSum; seqStart = i; seqEnd = j}
      }
    }
  return maxSum;
}
```

   (c)
```
public static int maxSubsequenceSum(int [] a) {
  int maxsum = 0;
  for(int i = 0; i < a.length; i++) {
    for(int j = i; j < a.length; j++) {
      int thisSum = 0;
      for(int k = i; k <= j; k++) thisSum += a[k];
      if(thisSum > maxSum)
        {maxSum = thisSum; seqStart = i; seqEnd = j}
      }
    }
  return maxSum;
}
```

3. Din uppgift är att implementera en klass för ändliga mängder av heltal med några vanliga mängdoperationer:

```
public class SetInt{

   ... // tillståndsvariabler

   public boolean member(int n);
   public void insert(int n);
   public void delete(int n);
   public void union(SetInt s);
   public void intersect(SetInt s);
}
```

Operationerna ska förstås ha sin vanliga mängdteoretiska betydelse. T ex betyder `intersect(s)` att man ska ersätta den mängd `t` som är lagrad i objektet med snittet av `s` och `t`.

Flera olika datastrukturer kan vara lämpliga för att implementera denna klass - vilken som är bäst beror på vilka operationer som är vanligast! Du ska diskutera följande fall:

(a) `member` är vanligast;

(b) `insert` är vanligast;

(c) `union` och `intersect` är vanligast.

(d) Visa också hur man implementerar `intersect` genom att ge pseudokod eller Javakod.

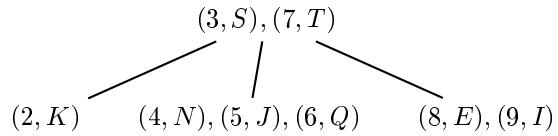
I deluppgift (a), (b) och (c) ska du föreslå minst två datastrukturer som du anser vara huvudalternativ och jämföra dem med varandra. Du ska motivera varför du anser dem vara bäst utifrån den asymptotiska tidskomplexiteten hos de olika implementeringarna av operationerna! Tänk på att även om du i första hand ska ta hänsyn till komplexiteten hos de vanligaste operationerna, måste du även i andra hand ta hänsyn till komplexiteten hos de mindre vanliga.

4. (2,4)-träd är ett slags balanserade sökträd som kan användas för att implementera lexika. Vi antar här att dessa lexika är "funktionella", dvs en nyckel k kan bara förekomma i högst ett par (k, v).

   Våra (2,4)-träd har följande egenskaper. (Definitionen avviker något från den i Goodrich och Tamassia eftersom vi förutsätter funktionella lexika.)

   - Varje nod lagrar ett, två eller tre par (k, v) av nycklar k och element v. Vi antar här att nycklarna är heltal.
   - En intern nod som lagrar n par av nycklar och element har $n + 1$ barn.
   - Följande sökträdsegenskap gäller:
     - om en nod lagrar ett par (k_1, v_1) och paret (k, v) ligger i det första (vänstra) delträdet till noden så gäller $k < k_1$. Om (k, v) ligger i det andra (högra) delträdet så gäller $k_1 < k$.
     - om en nod lagrar två par $(k_1, v_1), (k_2, v_2)$ och (k, v) ligger i det första delträdet så gäller $k < k_1$. Om (k, v) ligger i det andra delträdet så gäller $k_1 < k < k_2$. Om (k, v) ligger i det tredje delträdet så gäller $k_2 < k$.
     - om en nod lagrar tre par $(k_1, v_1), (k_2, v_2), (k_3, v_3)$ och (k, v) ligger i det första delträdet så gäller $k < k_1$. Om (k, v) ligger i det andra delträdet så gäller $k_1 < k < k_2$. Om (k, v) ligger i det tredje delträdet så gäller $k_2 < k < k_3$. Om (k, v) ligger i det fjärde delträdet så gäller $k_3 < k$.
   - Alla lövnoder har samma djup.

   Här är en bild på ett (2,4)-träd:



   (a) Definiera en klass i Java som implementerar (2,4)-träd. Klassen ska bara ha *en* metod. Denna metod är en sökmetod som tar en nyckel (ett heltal) k som indata och returnerar ett element v om (k, v) finns lagrat i någon nod i (2,4)-trädet. Annars ska metoden returnera `NO_SUCH_KEY`.

   (b) Rita två bilder som visar hur

   i. ovanstående (2,4)-träd,

   ii. det tomma (2,4)-trädet

   representeras i minnet!

   Tänk på att rita alla relevanta pekare och/eller fält som representationerna använder.

   (c) Definiera också en konstruerarmetod som skapar ett tomt (2,4)-träd!

Eftersom du inte har någon Java-bok tillgänglig kommer vi att ha överseende med smärre syntaxfel. Logiska fel ger dock poängavdrag.

5. (a) Skriv en algoritm i pseudokod som givet en oriktad graf beräknar antalet sammanhängande komponenter i grafen! En sammanhängande komponent är en maximal sammanhängande delgraf. En delgraf är sammanhängande om varje par av noder i delgrafen kan förbindas med en väg.

   (b) Vilken O-tidskomplexitet har din algoritm uttryckt som en funktion av antalet noder n och antalet bågar m som finns i grafen? Motivera utifrån pseudokoden.

   (c) Skriv en algoritm i pseudokod som givet en riktad graf avgör om grafen saknar cykler, dvs om grafen är en DAG ("Directed Acyclic Graph").

   (d) Ange även här O-tidskomplexiteten! Motivera!

   För att få maximal poäng på uppgiften måste dina algoritmer ha så god asymptotisk tidskomplexitet som möjligt.

6. Du har fått i uppgift att skriva en artikel för ett uppslagsverk om hashning. Artikeln riktar sig till en person som har elementära kunskaper om programmering, men inte vet något ytterligare om datalogi. Den ska innehålla 150 - 300 ord. Du ska alltså kort och klart sammanfatta det viktigaste du vet om hashning på ett populärvetenskapligt sätt. Försök ge läsaren svar på frågor som "Vad är hashning?", "Vad används det till?", "Vad har metoden för för- och nackdelar?".

   Poängsättningen grundas både på hur pass välskriven artikeln är, om den är "på rätt nivå", och förstås, om den gör en bra avvägning av vad som är viktigaste att veta!