Tentamen i Algortitmer och datastrukturer DV1 2000-03-17

Skrivtid: 1500 – 2000
Hjälpmedel: Räknedosa, *Några användbara matematiska samband*
Varje delfråga kan, om inget annat anges, ge högst 1 poäng.
Svar skall motiveras om inte annat anges.

Lycka till!

Uppgifter

1. I samband med algoritmanalys använder man ofta begreppen *Ordo, Omega* och *Theta*. Definiera dessa begrepp.

2. Vid testkörning av ett program som implementerar trädsortering på slumptal uppmättes tiden 1 sekund för att sortera 1000 tal. Uppskatta hur lång tid som krävs för att sortera 10^6 tal.

3. Följande funktioner är avsedda att skriva ut ett binärt träd i *nivåorder*:

```
void visitLevel(node * r, int level) {
  if ( r!=NULL && level>=1 )
     if ( level==1 )
       printf( "%d ", r->data );
     else {
       visitLevel( r->left , level-1 );
       visitLevel( r->right, level-1 );
     }
}

void visitTree(node * r) {
  int i;
  for ( i = 1; i <= height(r); i++ )
      visitLevel( r, i ); printf("\n");
}
```

   (a) Deklarera typen `node` som skall representera en nod i trädet.

   (b) Implementera funktionen `height` som beräknar höjden.

   (c) Hur beror körtiden för programmet av antalet noder i trädet? Du behöver bara göra analysen för ett komplett träd med höjd h (dvs h helt fyllda nivåer) (2p)

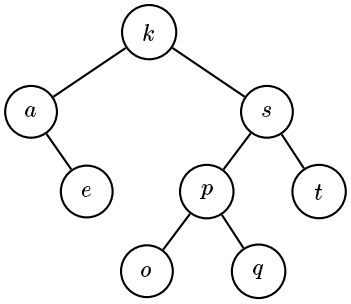
4. Beräkna den

   (a) maximala och

   (b) den minimala (2p)

   interna väglängden för ett träd med n noder? Det räcker att beakta $n = 2^i - 1$ noder där i är ett godtyckligt positivt heltal.

5. Givet följande binära sökträd med bokstäver som nycklar:



   (a) Vad menas med ett *rödsvart* träd? Visa att ovanstående träd är ett sådant.

   (b) Lägg in nycklarna f och r så att trädet förblir ett rödsvart träd. Om du använder annat än standardalgoriterna måste du redovisa dessa. (Alternativt kan du behandla detta som ett AVL-träd och göra AVL-trädsinlägg.)

   (c) Rita ett rödsvart träd som inte är ett AVL-träd.

6. Använd nycklarna 11, 21, 3, 42, 17, 2, 19, 12, 26, 15, 1 (i denna ordning) för att bygga upp hashtabeller med m platser enligt nedan. Som hashfunktion skall

$$h(k) = k \bmod m$$

   och, i förekommande fall, stegfunktionen

$$g(k) = k \bmod (m-2) + 1$$

   användas.

   (a) En *länkad* hashtabell. Tabellens storlek m skall vara 7.

   (b) En öppen hashtabell (alla nycklar i tabellen) med *linjär* kollisionshantering och med tabellstorleken $m = 13$. Hur många försök kräver en *misslyckad* sökning i genomsnitt i just denna tabell?

   (c) En öppen hashtabell med *dubbel hashning* med tabellstorleken $m = 13$. Hur många försök kräver en *lyckad* sökning i genomsnitt i just denna tabell?

   (d) Vad menas med *ordnad hashning*? Vad är fördelarna med denna metod?

   (e) Antag att n nycklar lagras i en länkad hashtabell med m platser. Härled formler för hur *lyckad* respektive *misslyckad* sökning beror på m och n. (2p)

7. (a) Definiera begreppet *heap* (i datastruktur- och algoritmsammanhang).

   (b) Placera talen från uppgift 6 i en heap. Du behöver inte redovisa hur du gjort bara resultatet är en heap. Redovisa därefter hur operationerna *insert(7)* och *deleteMin()* tillgår.