Tölvunarfræði 2
Svör við skyndiprófi

Snorri Agnarsson

25. febrúar 2010

Allar spurningar gilda 16.7%. Bestu 6 svör gilda til einkunnar. Engin hjálpargögn eru leyfileg. Merkið öll svarblöð með **nafni** og **númeri dæmahóps** (D1, D2 eða D3).

1. **Spurning:** Fyllið inn þar sem spurningarmerkin eru, þ.e. skrifið hvað á að koma í stað `?1?`, o.s.frv. Þetta eru samtals átta svör.

```
// Notkun: k = leita(f,i,j,x);
// Fyrir:  f[i..j-1] er í minnkandi röð
// Eftir:  f[i..j-1] er óbreytt, i <= k <= j, og
//         f[i..k-1] > x >= f[k..j-1]
int leita( double[] f, int i, int j, double x ) {
  if( i==j ) return i;
  int m = (i+j)/2;
  if( f[m] ?1? x )
    return leita(f,i,?2?,x);
  else
    return leita(f,?3?,j,x);
}

int leita( double[] f, int i, int j, double x ) {
  int p=i, q=j;
  while( ?4? ) {
    // | >x | óþekkt | <=x |
    //  i    p        q     j
    int m = (?5?)/2;
    if( f[m] ?6? x )
      p = ?7?;
    else
      q = ?8?;
  }
  return p;
}
```

Svar:

```
// Notkun: k = leita(f,i,j,x);
// Fyrir:  f[i..j-1] er í minnkandi röð
// Eftir:  f[i..j-1] er óbreytt, i <= k <= j, og
//         f[i..k-1] > x >= f[k..j-1]
int leita( double[] f, int i, int j, double x ) {
  if( i==j ) return i;
  int m = (i+j)/2;
  if( f[m] <= x )
    return leita(f,i,m,x);
  else
    return leita(f,m+1,j,x);
}

int leita( double[] f, int i, int j, double x ) {
  int p=i, q=j;
  while( p!=q ) {
    // | >x | óþekkt | <=x |
    //  i    p        q     j
    int m = (p+q)/2;
    if( f[m] > x )
      p = m+1;
    else
      q = m;
  }
  return p;
}
```

2. **Spurning:** Fyllið inn þar sem spurningarmerkin eru. Þetta er selection sort.

```
// Notkun: swap(f,i,j);
// Fyrir:  f[i] og f[j] eru sæti í f
// Eftir:  Búið er að víxla gildunum í f[i] og
//         f[j]
void swap( double[] f, int i, int j );

// Notkun: sort(f,i,j);
// Fyrir:  f[i..j-1] er ekki-tómt svæði í f
// Eftir:  Búið er að raða f[i..j-1]
void sort( double[] f, int i, int j ) {
  int p = ?1?;
  while( ?2? ) {
    // | minnstu gildi í vaxandi röð | óþekkt |
    //  i                            p        j
    int q = ?3?;
    while( ?4? ) {
      // a) | minnstu gildi í vax. röð | óþekkt |
```

```
      //     i                              p        j
      // b) p < q <= j
      // c) f[p] er minnst af f[p..q-1]
      if( f[q] ?5? f[p] )
        swap(f,p,q);
      q++;
    }
    p++;
  }
}
```

Svar:

```
// Notkun: swap(f,i,j);
// Fyrir:  f[i] og f[j] eru sæti í f
// Eftir:  Búið er að víxla gildunum í f[i] og
//         f[j]
void swap( double[] f, int i, int j );

// Notkun: sort(f,i,j);
// Fyrir:  f[i..j-1] er ekki-tómt svæði í f
// Eftir:  Búið er að raða f[i..j-1]
void sort( double[] f, int i, int j ) {
  int p = i;
  while( p!=j ) {
    // | minnstu gildi í vaxandi röð | óþekkt |
    //  i                             p        j
    int q = p+1;
    while( q!=j ) {
      // a) | minnstu gildi í vax. röð | óþekkt |
      //     i                            p        j
      // b) p < q <= j
      // c) f[p] er minnst af f[p..q-1]
      if( f[q] < f[p] )
        swap(f,p,q);
      q++;
    }
    p++;
  }
}
```

3. **Spurning:** Hver er fastayrðing lykkju í ytri lykkju insertion sort?

   Svar:

```
| í vaxandi röð | óþekkt |
 0              i        n
```

4. **Spurning:** Hver eftirfarandi getur verið fastayrðing seinni lykkju heapsort?

   (a)
   ```
   | minnst í vaxandi röð | hrúga (heap) |
   0                      i              n
   ```

   (b)
   ```
   | minnst í vax. röð | uppfyllir hrúguskilyrði |
   0                   i                         n
   ```

   (c)
   ```
   | hrúga (heap) | stærst í vaxandi röð |
   0              i                      n
   ```

   Svar:

   ```
   | hrúga (heap) | stærst í vaxandi röð |
   0              i                      n
   ```

5. **Spurning:**

   (a) Hvaða fastayrðingu má nota í fyrri lykkju heapsort?

   (b) Hver er tímaflækja heapsort?

   Svar:

   ```
   | hrúga (heap) | óþekkt |
   0              i        n
   ```

   eða (betra)

   ```
   | óþekkt | uppfyllir hrúguskilyrði |
   0        i                         n
   ```

   og tímaflækjan er $O(n \log n)$.

6. **Spurning:**

   (a) Lýsið hugmyndinni í merge-sort

   (b) Hver er tímaflækja merge-sort?

   Svar: Til að raða runu gilda með mergesort skiptum við runununni til helminga (eða því sem næst) í tvær runur, röðum þeim með mergesort og samröðum síðan útkomunum. Ef runa inniheldur færri en tvö gildi þarf auðvitað ekkert að gera.

   Tímaflækjan er O($n \log n$).

7. **Spurning:** Gefið er eftirfarandi C++ stef:

```
// Notkun:  skipta(f,i,j,k,n);
// Fyrir:   f[i..j-1] er ekki-tómt svæði í fylkinu f
// Eftir:   Búið er að víxla gildum í svæðinu og gefa
//          k og n gildi þ.a.
//           1) i <= k < n <= j og
//           2) f[i..k-1] < p og
//           3) f[k..n-1] = p og
//           4) f[n..j-1] > p
//          fyrir eitthvert p sem fyrir var í svæðinu.
void skipta( double f[], int i, int j, int& k, int& n );
```

Skrifið quicksort stef (með lýsingu - notkun, fyrir og eftir) með hjálp þessa stefs. Ekki þarf að forrita skipta stefið.

Svar:

```
// Notkun: sort(f,i,j);
// Fyrir:  f[i..j-1] er svæði í f.
// Eftir:  Búið er að raða svæðinu í vax. röð.
void sort( int[] f, int i, int j )
{
  if( j-i < 2 ) return;
  int k,n;
  skipta(f,i,j,k,n);
  sort(f,i,k);
  sort(f,n,j);
}
```

8. **Spurning:** Hver er tímaflækja eftirfarandi röðunaraðferða? Hvers konar tímaflækju er um að ræða (versti tími, meðaltími, o.s.frv.)?

   (a) Insertion-sort

   (b) Selection-sort

   (c) Quicksort

   (d) Merge-sort

   (e) Radix-sort (gerið ráð fyrir þriggja stafa tölum)

   (f) Heapsort

   Svar:

   (a) Insertion-sort — $O(n^2)$ í versta falli.

   (b) Selection-sort — $O(n^2)$ í versta falli.

   (c) Quicksort — $O(n^2)$ að meðaltali ($O(n^2)$ í versta falli nema það sé sérstaklega vel forritað til að losna við það).

   (d) Merge-sort — $O(n \log n)$ í versta falli (og alltaf).

(e) Radix-sort (gerið ráð fyrir þriggja stafa tölum) — $O(n)$ í versta falli (og alltaf).

(f) Heapsort — $O(n \log n)$ í versta falli (og alltaf).

9. **Spurning:** Gerið grein fyrir meginreglunni um upplýsingahuld: Hver er reglan og hver er tilgangur hennar?

   Svar: Fela skal sérhverja veigamikla útfærsluákvörðun í einingu. (Eða: Gefa skal notanda einingar nægilega miklar upplýsingar um eininguna til að nota hana, en ekki meiri, og gefa skal smið einingar nægilega miklar upplýsingar um eininguna til að smíða hana, en ekki meiri).

   Tilgangurinn er að gera viðhald (breytingar á einingunni og þar með á útfærsluákvörðuninni) mögulegar og auðveldar án þess að það valdi verulegri hættu á að heildarkerfið hrynji vegna brostinna forsendna.

10. **Spurning:** Bubblesort er vel þekkt röðunaraðferð sem virkar þannig að til að raða fylki f[0..n-1] endurtökun við aftur og aftur sömu lykkju þar sem rennt er eftir fylkinu frá vinstri til hægri (líka hægt að fara frá hægri til vinstri) og víxlum gildumn í samliggjandi sætum ef gildið í vinstra sæti er stærra en gildið í hægra sæti. Að lokum kemur að því í einhverri umferð ytri lykkju að engar víxlanir þarf í innri lykkju, og þá er röðun lokið.

    (a) Hver er tímaflækja bubblesort í versta tilfelli?

    (b) Hver er tímaflækja bubblesort í besta tilfelli?

    (c) (Aukaspurning, þarf ekki að svara) Hver er tímaflækja bubblesort að meðaltali? Hvaða nýjar forsendur þarf að gefa sér til að svara þeirri spurningu?

    Svar:

    (a) $O(n^2)$.

    (b) $O(n)$, til dæmis ef fylkið var þegar raðað.

    (c) $O(n^2)$ miðað við þá forsendu að gildin sem verið er að raða séu öll mismunandi og að sérhver umröðun gildanna sé jafn líkleg.

11. **Spurning:** Gerið ráð fyrir að til sé skilgreining í C++ eða Java á grunnklasa eða skilum (interface), IntBag, fyrir poka heiltalna, með boðum get, put og isEmpty með venjulegum lýsingum fyrir poka.

    (a) Hverja af eftirfarandi klösum er þá eðlilegt að útfæra sem undirklasa af IntBag:

    - Klasi, IntStack, fyrir hlaða heiltalna.
    - Klasi, IntQueue, fyrir biðröð heiltalna.
    - Klasi, IntPriQueue, fyrir forgangsbiðröð heiltalna.
    - Klasi, IntSet, fyrir mengi heiltalna.

    (b) Fyrir þá klasa úr upptalningunni sem ekki er eðlilegt að útfæra sem undirklasa IntBag, rökstyðjið að það sé ekki eðlilegt með því að sýna að eðlilegar lýsingar á einhverju boði geti ekki uppfyllt þær söksemdafærslureglur sem til þarf.

Svar: IntStack, IntQueue og IntPriQueue geta verið undirklasar IntBag. IntSet getur það ekki vegna þess að í IntSet þyrfti annað hvort að þrengja forskilyrði fyrir put aðgerð þannig að hún sé ekki leyfileg ef mengið inniheldur þegar töluna sem setja skal í mengið, eða að víkka eftirskilyrði með því að gefa kost á því að reynt sé að setja tölu í mengið, en það valdi ekki breytingu á menginu.

12. **Spurning:** Gerið ráð fyrir að til sé klasi IntPriQueue í C++ eða Java fyrir forgangsbiðröð heiltalna af ótakmarkaðri stærð, sem hefur sjálfgefinn smið og boðin get, put, og isEmpty með eðlilegum lýsingum þ.a. get boðið fjarlægir og skilar minnstu tölunni í forgangsbiðröðinni.

    Notið þennan klasa til að skrifa röðunarstef fyrir fylki heiltalna. Munið að hafa skýra notkunarlýsingu (Notkun, Fyrir, Eftir).

    Ef get og put boðin hafa tímaflækju $O(f(n))$, þar sem n er fjöldi gilda í forgangsbiðröðinni, og isEmpty hefur tímaflækjuna $O(1)$, hver er þá tímaflækja röðunarstefins?

    Svar:

```
// Notkun: sort(f);
// Eftir:  Búið er að raða f í vax. röð.
void sort( int[] f )
{
  IntPriQueue q = new IntPriQueue();
  int i;
  for( i=0 ; i!=f.length ; i++ )
  {
    // q inniheldur tölurnar úr f[0..i-1]
q.put(f[i]);
  }
  for( i=0 ; i!=f.length ; i++ )
  {
    // f[0..i-1] inniheldur minnstu tölur í röð
// af þeim sem voru í f áður.  Hinar eru í q.
f[i] = q.get();
  }
}
```

    Tímaflækjan er $O(nf(n))$.