# Tölvunarfræði 2/2a
# Svör við skyndiprófi

Snorri Agnarsson

25. febrúar 2009

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 vaxandi 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 vaxandi 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.

```
// 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 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? ) {
      // | minnstu gildi í vax. röð | óþekkt |
      //  i                         p        j
```

```
      //     p < q <= j
      //     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 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 ) {
      // | minnstu gildi í vax. röð | óþekkt |
      //  i                          p        j
      //     p < q <= j
      //     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 |
 i               k        j
```

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:**

   (a)
   ```
   | óþekkt | uppfyllir hrúguskilyrði |
    0        k                         n
   ```
   Eða
   ```
   | hrúga | óþekkt |
    0       k        n
   ```
   Eða
   ```
   | óþekkt | uppfyllir hrúguskilyrði |
    0        k                        n
   ```
   Eða
   ```
   | hrúga | óþekkt |
    0       k      n
   ```

   (b) $O(n \log n)$ þar sem $n$ er fjöldi sæta í fylkinu.

6. **Spurning:**

   (a) Lýsið hugmyndinni í merge-sort

   (b) Hver er tímaflækja merge-sort?

   **Svar:**

   (a) Til að raða runu gilda skiptum við rununni í tvær runur (óraðaðar) til helminga (u.þ.b.), röðum síðan smærri rununum og samröðum síðan þessum smærri runum til að fá raðaða heildarrunu (mergjum, *merge*). Ef upphaflega runan var styttri en tvö gildi þarf auðvitað ekkert að gera til að raða.

(b) $O(n \log n)$ þar sem $n$ er fjöldi gilda í rununni.

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

8. **Spurning:**

Hverjar af eftirfarandi röðunaraðferðum má nota til að raða fylki með víxlunum, þ.e. án þess að geyma gildin sem verið er að raða annars staðar en í fylkinu?

(a) Insertion-sort

(b) Selection-sort

(c) Quicksort

(d) Merge-sort

(e) Radix-sort

(f) Heapsort

**Svar:** Allar nema merge-sort og radix-sort.

9. **Spurning:**

   Gerið grein fyrir meginreglunni um upplýsingahuld. Hver er reglan og hver er tilgangur hennar?

   **Svar:** Meginreglan um upplýsingahuld, samkvæmt upphaflegri skilgreiningu frá D.L. Parnas, hljóðar svona: Sérhverja veigamikla hönnunarákvörðun[1] skal fela í einingu. Einnig mætti orða þetta svona, í tveimur reglum:

   - Gefa skal notanda einingar nægilega miklar upplýsingar um smíð hennar til að nota eininguna, en ekki meiri.
   - Gefa skal smið einingar nægilega miklar upplýsingar um notkun hennar til að smíða eininguna, en ekki meiri.

10. **Spurning:** Gerið ráð fyrir að til sé klasi PQI 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.

    **Svar:**

    Miðum við C++ í þessu svari.

```
// Notkun: sort(f,n);
// Fyrir:  f[0..n-1] er svæði í f.
// Eftir:  f[0..n-1] inniheldur sömu gildi og
//         áður, en í vaxandi röð.
double sort( double[] f, int n )
{
  IPQ q;
  int i=0;
  while( i!=n )
  {
    // IPQ inniheldur tölurnar úr
    // f[0..i-1].  0<=i<=n.
    q.put(f[i++]);
  }
  i=0;
  while( !q.isEmpty() )
  {
    // f[0..i-1] inniheldur minnstu tölurnar
    // af þeim sem upphaflega voru í f[0..n-1].
    // Hinar eru í q.
    f[i++] = q.get();
  }
}
```

[1] Með „hönnunarákvörðun“ á Parnas við það sem ég myndi frekar kalla smíðaákvörðun eða útfærsluákvörðun.

11. **Spurning:** Forritið eftirfarandi fall:

```
// Notkun: prim(n);
// Fyrir:  n er jákvæð heiltala
// Eftir:  Búið er að skrifa prímþætti n á
//         aðalúttak.
void prim( int n ) {
  ???
}
```

**Svar:**

```
void prim( int n ) {
  int p=2, m=n;
  while( m != 1 ) {
    // Allar skrifaðar tölur eru prímtölur
    // Margfeldi m og skrifaðra talna er n
    // Engin prímtala minni en p gengur upp í m
    // p >= 2
    if( m%p == 0 ) {
      cout << p << endl;
      m = m/p;
    }
    else
      p++;
  }
}
```