Insertion sort og selection sort

Tölvunarfræði 2, vor 2012

Hallgrímur H. Gunnarsson

Háskóli Íslands

2012-02-17

Röðun

Röðun er klassískt vandamál í tölvunarfræði.

Til margar lausnir:

- Insertion sort: $O(n^2)$ verst, $O(n)$ best, $O(n^2)$ meðal
- Selection sort: $O(n^2)$ verst, $O(n^2)$ best, $O(n^2)$ meðal
- Mergesort: $O(n \log n)$ verst, $O(n \log n)$ best, $O(n \log n)$ meðal
- Quicksort: $O(n^2)$ verst, $O(n \log n)$ best, $O(n \log n)$ meðal
- Heapsort: $O(n \log n)$ verst, $O(n \log n)$ best, $O(n \log n)$ meðal

Líka til reiknirit sem byggja ekki á samanburðum. Við munum skoða eitt slíkt, radix sort, seinna á misserinu.

Abstract röðun

Markmið: raða hvaða tagi sem er

Spurning: hvernig getur sort() aðferð borið saman tvö gildi án þess að vita tag þeirra?

Callback er tilvísun á kóða sem er hægt að framkvæma:

- Notandi lætur sort() fá fylki af hlutum
- sort() kallar á compareTo() til að bera saman tvö gildi

Útfærsla á callbacks:

- Java: viðmót (interfaces)
- C: fallbendar (function pointers)
- C#: delegates

Comparable interface

```
public interface Comparable
{
    // Notkun: k = a.compareTo(b);
    // Fyrir:  a og b eru af tagi sem má bera saman
    // Eftir:  k < 0   ef a < b
    //         k == 0  ef a = b
    //         k > 0   ef a > b
    int compareTo(Object o);
}
```

Innbyggð tög sem eru samanburðarhæf: Integer, Double, String, Date, File...

Getum gert okkar eigin klasa samanburðarhæfa með því að útfæra Comparable

Dæmi um klasa sem útfærir Comparable

```
public class Date implements Comparable<Date>
{
    private final int month, day, year;

    public Date(int m, int d, int y)
    {
        month = m; day   = d; year  = y;
    }
    public int compareTo(Date that)
    {
        if (this.year  < that.year ) return -1;
        if (this.year  > that.year ) return +1;
        if (this.month < that.month) return -1;
        if (this.month > that.month) return +1;
        if (this.day   < that.day  ) return -1;
        if (this.day   > that.day  ) return +1;
        return 0;
    }
}
```

Almenn röðun

```
static void sort(Comparable[] f, int i, int j)
{
  // ...

      if (f[k].compareTo(f[k-1] < 0)
        // ...

  // ...
}
```

Hjálparföll

Tvö þægileg hjálparföll.

`less`. Er a minna en b?

```
private static boolean less(Comparable a, Comparable b)
{
  return a.compareTo(b) < 0;
}
```

`swap`. Víxla gildunum í sætum a og b í `f[]`

```
private static void swap(Comparable[] f, int a, int b)
{
  Comparable tmp = f[a];
  f[a] = f[b];
  f[b] = tmp;
}
```

Prófun – athuga hvort fylki sé í vaxandi röð

Allt fylkið:

```
private static boolean isSorted(Comparable[] a)
{
    for (int i = 1; i < a.length; i++)
        if (less(a[i], a[i-1]))
            return false;
    return true;
}
```

Tiltekið svæði í fylkinu:

```
private static boolean isSorted(Comparable[] a, int i, int j)
{
    for (int k = i+1; k < j; k++)
        if (less(a[k], a[k-1]))
            return false;
    return true;
}
```

Selection sort – demo

http://www.sorting-algorithms.com/selection-sort

http://algs4.cs.princeton.edu/lectures/21DemoSelectionSort.mov

Selection sort

```
// Notkun:  ssort(f,i,j);
// Fyrir:   f[i..j-1] er svæði í f
// Eftir:   Búið er að raða svæðinu f[i..j-1] í vaxandi röð.
//          Röðunaraðferðin er selection sort.
static void ssort(Comparable[] f, int i, int j)
{
  for (int p = i; p < j; p++) {
    // | minnst í vaxandi röð | ??? |
    //  i                       p     j
    int min = p;
    for (int k = p+1; k < j; k++) {
      // f[min] <= f[p..k-1]
      if (less(f[k], f[min]))
        min = k;
    }
    // f[min] <= f[p..j-1]
    swap(f, p, min);
  }
}
```

Insertion sort – demo

http://www.sorting-algorithms.com/insertion-sort

http://algs4.cs.princeton.edu/lectures/21DemoInsertionSort.mov

Insertion sort

```
// Notkun:  isort(f,i,j);
// Fyrir:   f[i..j-1] er svæði í f
// Eftir:   Búið er að raða svæðinu f[i..j-1] í vaxandi röð.
//          Röðunaraðferðin er insertion sort.
public static void isort(Comparable[] f, int i, int j)
{
  for (int p = i+1; p < j; p++)
    // f[i..p-1] er í vaxandi röð
    for (int k = p; k > i && less(f[k], f[k-1]); k--)
      // f[i..p] er í vaxandi röð, nema hvað e.t.v.
      // er f[k] of aftarlega
      swap(f,k-1,k);
}
```