EDAA45 Programmering, grundkurs

Läsvecka 5: Sekvensalgoritmer

Björn Regnell

Datavetenskap, LTH

Lp1-2, HT 2016

5 Sekvensalgoritmer

Vad är en sekvensalgoritm?

- En algoritm är en stegvis beskrivning av hur man löser ett problem.
- En sekvensalgoritm är en algoritm där dataelement i sekvens utgör en viktig del av problembeskrivningen och/eller lösningen.
- Exempel: sortera en sekvens av personer efter deras ålder.
- Två olika principer:
  - Skapa **ny sekvens** utan att förändra indatasekvensen
  - Ändra **på plats** (eng. *in place*) i den **föränderliga** indatasekvensen

Några indexerbara samlingar

- Oföränderliga:
  - Kan **ej** ändra elementreferenserna:
    Scala: **Vector**, **List**
- Föränderliga: kan **ändra** elemententreferenserna
  - Kan **ej ändra storlek** efter allokering:
    Scala+Java: **Array**
  - Kan ändra storlek efter allokering:
    Scala: **ArrayBuffer**
    Java: **ArrayList**

Algoritm: SEQ-COPY

Indata : Heltalsarray *xs*
Resultat: En ny heltalsarray som är en kopia av *xs*.

$n \leftarrow$ antalet element i *xs*
ys $\leftarrow$ en ny array med plats för n element
$i \leftarrow 0$
while $i < n$ **do**
  $ys(i) \leftarrow xs(i)$
  $i \leftarrow i + 1$
end
return *ys*

Egenskaper hos några sekvenssamlingar

- `Vector`
  - **Oföränderlig**
  - **Bra till det mesta** och hyffsat **snabb**
- `List`
  - **Oföränderlig**
  - Snabb vid bearbetning **i början**
  - Smidig & snabb vid **rekursiva** algoritmer
  - Långsam vid godtycklig **indexering** och bearbetning **i slutet**
- `Array`
  - **Föränderlig**: **snabb indexering & uppdatering**
  - Kan **ej ändra storlek**; storlek anges vid allokering
  - Har särställning i JVM: ger maskinkod med snabb minnesaccess
- `ArrayBuffer`
  - **Föränderlig**: **snabb indexering & uppdatering**
  - Kan **ändra storlek** efter allokering
- Om prestandakritiskt: undersök om `Array` är snabbare

Vilken sekvenssamling ska jag välja?

- `Vector`
  - Om du vill ha oföränderlighet: `val xs = Vector[MyType](...)`
  - Om du behöver ändra men ej prestandakritiskt:
    `var xs = Vector.empty[MyType]`
  - Om du ännu inte vet vilken som är bäst. (Du kan alltid ändra efter att du mätt prestanda och kollat flaskhalsar.)
- `Array`
  - Om det **verkligen behövs** av prestandaskäl och du kan bestämma en lämplig storlek vid allokering:
    `val xs = Array.fill(initSize)(initValue)`
- `ArrayBuffer`
  - det **verkligen behövs** av prestandaskäl och du behöver kunna ändra storlek efter allokering:
    `val xs = ArrayBuffer.empty[MyType]`