Föreläsning
Datastrukturer (DAT036)

Nils Anders Danielsson

2012-10-29

Motivation

Varför studera datastrukturer?

Motivation

Varför studera datastrukturer?

- Snabbt.
- Lite minne.
- Mindre energiförbrukning.
- Billigare.
- Och korrekt!

Administrativt

Kurshemsidan.

Tidskomplexitet

Hur analyserar man tidskomplexitet?

- *Mäta*.
  Nackdelar: Svårt att generalisera,
  inget bra stöd för design.
- *Räkna instruktioner*.
  Nackdelar: Komplicerat.
- *Förenklad modell*.
  Nackdelar: Inte tillämpligt i alla lägen.

Uniform kostnadsmodell

- Enkel dator.
- Varje instruktion tar en tidsenhet.
  Bara enkla instruktioner,
  men godtyckligt stora tal.
- Oändligt minne.

Uniform kostnadsmodell

- Enkel dator.
- Varje instruktion tar en tidsenhet.
  Bara enkla instruktioner,
  men godtyckligt stora tal.
- Oändligt minne.

- Inte realistisk.
- Fungerar ganska bra om man är försiktig.
- Används ofta i kursen.

Logaritmisk kostnadsmodell

- Enkel dator.
- Tid beräknas i termer av indatas storlek, räknat i antal bitar.
- Oändligt minne.
- Lite mer realistisk, lite krångligare.

Begränsningar

- I/O.
- Cacheminnen.

Analys av enkel algoritm

Insättning i sorterad lista:

```
insert(x, xs) =
  if xs.isEmpty() then
    xs.addFirst(x)
  else
    y = xs.getFirst()
    if x <= y then
      xs.addFirst(x)
    else
      xs.removeFirst()
      insert(x, xs)
      xs.addFirst(y)
```

Analys av enkel algoritm

Insättning i sorterad lista:

```
insert(x, xs) =
  if xs.isEmpty() then
    xs.addFirst(x)
  else
    y = xs.getFirst()
    if x <= y then
      xs.addFirst(x)
    else
      xs.removeFirst()
      insert(x, xs)
      xs.addFirst(y)
```

Bästafallstidskomplexitet uttryckt i termer av listans längd (grov uppskattning):

$$I_b \in \mathbb{N} \to \mathbb{N}$$
$$I_b(0) = 3$$
$$I_b(1+n) = 7$$

Analys av enkel algoritm

Insättning i sorterad lista:

```
insert(x, xs) =
  if xs.isEmpty() then
    xs.addFirst(x)
  else
    y = xs.getFirst()
    if x <= y then
      xs.addFirst(x)
    else
      xs.removeFirst()
      insert(x, xs)
      xs.addFirst(y)
```

Bästafallstidskomplexitet uttryckt i termer av listans längd (grov uppskattning):

$$I_b \in \mathbb{N} \to \mathbb{N}$$
$$I_b(0) = 3$$
$$I_b(1+n) = 7$$

Värstafallstidskomplexitet:

$$I_v \in \mathbb{N} \to \mathbb{N}$$
$$I_v(0) = 3$$
$$I_v(1+n) \leq 8 + I_v(n)$$

Analys av enkel algoritm

Insättning i sorterad lista:

```
insert(x, xs) =
  if xs.isEmpty() then
    xs.addFirst(x)
  else
    y = xs.getFirst()
    if x <= y then
      xs.addFirst(x)
    else
      xs.removeFirst()
      insert(x, xs)
      xs.addFirst(y)
```

Bästafallstidskomplexitet uttryckt i termer av listans längd (grov uppskattning):

$$I_b \in \mathbb{N} \to \mathbb{N}$$
$$I_b(0) = 3$$
$$I_b(1+n) = 7$$

Värstafallstidskomplexitet:

$$I_v \in \mathbb{N} \to \mathbb{N}$$
$$I_v(0) = 3$$
$$I_v(1+n) = 8 + I_v(n)$$

Analys av enkel algoritm

Kan vi lösa rekurrensekvationen?

$$
\begin{aligned}
I_v &\in \mathbb{N} \to \mathbb{N} \\
I_v(0) &= 3 \\
I_v(1+n) &= 8 + I_v(n)
\end{aligned}
$$

$$
\begin{aligned}
I_v(n) &= 8 + I_v(n-1) \\
&= 8 + (8 + I_v(n-2)) \\
&= ... \\
&= 8 + (8 + (... + 3...)) \\
&= 8n + 3
\end{aligned}
$$

Korrekt? Kan försöka bevisa resultatet m h a induktion.

Analys av enkel algoritm

Insättningssortering:

```
sort(xs) =
  ys = new LinkedList()
  while not xs.isEmpty() do
    insert(xs.removeLast(), ys)
  return ys
```

Analys av enkel algoritm

Insättningssortering:

```
sort(xs) =
  ys = new LinkedList()
  while not xs.isEmpty() do
    insert(xs.removeLast(), ys)
  return ys
```

Bästafallstidskomplexitet:

$$S_b \in \mathbb{N} \to \mathbb{N}$$
$$S_b(0) = 6$$
$$S_b(1+n) \geq 4 + I_b(n) + S_b(n)$$

Analys av enkel algoritm

Insättningssortering:

```
sort(xs) =
  ys = new LinkedList()
  while not xs.isEmpty() do
    insert(xs.removeLast(), ys)
  return ys
```

Bästafallstidskomplexitet:

$$S_b \in \mathbb{N} \to \mathbb{N}$$
$$S_b(0) = 6$$
$$S_b(1+n) \geq 4 + I_b(n) + S_b(n)$$

Värstafallstidskomplexitet:

$$S_v \in \mathbb{N} \to \mathbb{N}$$
$$S_v(0) = 6$$
$$S_v(1+n) \leq 4 + I_v(n) + S_v(n)$$

Analys av enkel algoritm

Insättningssortering:

```
sort(xs) =
  ys = new LinkedList()
  while not xs.isEmpty() do
    insert(xs.removeLast(), ys)
  return ys
```

Bästafallstidskomplexitet:

$$S_b \in \mathbb{N} \to \mathbb{N}$$
$$S_b(0) = 6$$
$$S_b(1+n) = 4 + I_b(n) + S_b(n)$$

Värstafallstidskomplexitet:

$$S_v \in \mathbb{N} \to \mathbb{N}$$
$$S_v(0) = 6$$
$$S_v(1+n) = 4 + I_v(n) + S_v(n)$$

Analys av enkel algoritm

Kan vi lösa rekurrensekvationerna?

$$
\begin{aligned}
S_b(0) &= 6 \\
S_b(1) &= 4 + I_b(0) + S_b(0) = 13 \\
S_b(2+n) &= 4 + I_b(1+n) + S_b(1+n) = 11 + S_b(1+n)
\end{aligned}
$$

$$
\begin{aligned}
S_b(n) &= 11 + S_b(n-1) \\
&= 11 + (11 + S_b(n-2)) \\
&= \ldots = 11(n-1) + 13 = 11n + 2
\end{aligned}
$$

$$
\begin{aligned}
S_b(0) &= 6 \\
S_b(n) &= 11n + 2 \text{ om } n \geq 1
\end{aligned}
$$

Korrekt? Kan försöka bevisa resultatet m h a induktion.

Analys av enkel algoritm

Kan vi lösa rekurrensekvationerna?

$$
\begin{aligned}
S_v(0) &= 6 \\
S_v(1+n) &= 4 + I_v(n) + S_v(n) \\
&= 4 + (8n+3) + S_v(n) \\
&= 8n + 7 + S_v(n)
\end{aligned}
$$

$$
\begin{aligned}
S_v(n) &= 8(n-1) + 7 + S_v(n-1) \\
&= 8(n-1) + 7 + (8(n-2) + 7 + S_v(n-2)) \\
&= \ldots = \sum_{i=0}^{n-1} (8i+7) + 6 \\
&= 8n\frac{n-1}{2} + 7n + 6 = 4n^2 + 3n + 6
\end{aligned}
$$

Korrekt? Kan försöka bevisa resultatet m h a induktion.

Analys av enkel algoritm

Kan vi lösa rekurrensekvationerna?

$$
\begin{aligned}
S_v(0) &= 6 \\
S_v(1+n) &= 4 + I_v(n) + S_v(n) \\
&= 4 + (8n+3) + S_v(n) \\
&= 8n + 7 + S_v(n)
\end{aligned}
$$

Krångligt!

$$
\begin{aligned}
S_v(n) &= 8(n-1) + 7 + S_v(n-1) \\
&= 8(n-1) + 7 + (8(n-2) + 7 + S_v(n-2)) \\
&= \ldots = \sum_{i=0}^{n-1} (8i+7) + 6 \\
&= 8n\frac{n-1}{2} + 7n + 6 = 4n^2 + 3n + 6
\end{aligned}
$$

Korrekt? Kan försöka bevisa resultatet m h a induktion.

Asymptotisk komplexitet

Detaljerna kanske inte är så viktiga.
Kursen fokuserar på *asymptotisk* komplexitet:
vad händer när storleken går mot oändligheten?

$$\begin{aligned} 8n + 3 &\quad \Rightarrow \quad O(n) \\ 4n^2 + 3n + 6 &\quad \Rightarrow \quad O(n^2) \end{aligned}$$

Asymptotisk komplexitet

1:	2× större	$\Rightarrow$	lika långsam
$\log n$:	2× större	$\Rightarrow$	1 långsammare
n:	2× större	$\Rightarrow$	2× långsammare
$n \log n$:	2× större	$\Rightarrow$	drygt 2× långsammare
n^2:	2× större	$\Rightarrow$	4× långsammare
2^n:	1 större	$\Rightarrow$	2× långsammare

Ordo-notation (en variant)

Antag att $T \in \mathbb{N} \to \mathbb{N}$, $f \in \mathbb{N}^+ \to \mathbb{R}$.

$$
\begin{aligned}
& T(n) = O(f(n)) \\
& \quad \Leftrightarrow \\
& \exists n_0 \in \mathbb{N}^+, c \in \mathbb{R}^+. \\
& \forall n \geq n_0. \quad T(n) \leq c f(n)
\end{aligned}
$$

Ordo-notation (en variant)

Antag att $T \in \mathbb{N} \to \mathbb{N}$, $f \in \mathbb{N}^+ \to \mathbb{R}$.

$$
\begin{aligned}
& T \in O(f) \\
& \quad \Leftrightarrow \\
& \exists n_0 \in \mathbb{N}^+, c \in \mathbb{R}^+. \\
& \forall n \geq n_0. \quad T(n) \leq c f(n)
\end{aligned}
$$

Ω

Antag att $T \in \mathbb{N} \to \mathbb{N}$, $f \in \mathbb{N}^+ \to \mathbb{R}$.

$$
\begin{aligned}
& T \in \Omega(f) \\
& \quad \Leftrightarrow \\
& \exists n_0 \in \mathbb{N}^+, c \in \mathbb{R}^+. \\
& \forall n \geq n_0. \quad T(n) \geq c f(n)
\end{aligned}
$$

Antag att $f \in \mathbb{N}^+ \to \mathbb{R}$.

$$\Theta(f) = O(f) \cap \Omega(f)$$

Antag att $T \in \mathbb{N} \to \mathbb{N}$, $f \in \mathbb{N}^+ \to \mathbb{R}$.

$$T \in \Theta(f)$$

$$\Leftrightarrow$$

$$\exists n_0 \in \mathbb{N}^+, c, d \in \mathbb{R}^+.$$

$$\forall n \geq n_0. \quad cf(n) \leq T(n) \leq df(n)$$

Analys av enkel algoritm

```
insert(x, xs) =
  if xs.isEmpty() then  // O(1)
    xs.addFirst(x)      // O(1)
  else
    y = xs.getFirst()   // O(1)
    if x <= y then      // O(1)
      xs.addFirst(x)    // O(1)
    else
      xs.removeFirst()  // O(1)
      insert(x, xs)     // Rekursivt anrop
      xs.addFirst(y)    // O(1)
```

Notera att $O(1) + O(1) = O(1)$.
Bästa fallet $O(1)$, värsta fallet $O(n)$, där $n = |\texttt{xs}|$.

Analys av enkel algoritm

```
insert(x, xs) =
  if xs.isEmpty() then   // Θ(1)
    xs.addFirst(x)       // Θ(1)
  else
    y = xs.getFirst()    // Θ(1)
    if x <= y then       // Θ(1)
      xs.addFirst(x)     // Θ(1)
    else
      xs.removeFirst()   // Θ(1)
      insert(x, xs)      // Rekursivt anrop
      xs.addFirst(y)     // Θ(1)
```

Notera att $\Theta(1) + \Theta(1) = \Theta(1)$.
Bästa fallet $\Theta(1)$, värsta fallet $\Theta(|\texttt{xs}|)$.

Analys av enkel algoritm

```
sort(xs) =                                  // Värsta fallet:
  ys = new LinkedList()                     // O(1)
                                            // |xs| gånger:
  while not xs.isEmpty() do                 //   O(1)
    insert(xs.removeLast(), ys)             //   O(1) + O(|ys|)
  return ys                                 // O(1)
```

Notera att $O(1) + O(|\texttt{ys}|) = O(|\texttt{ys}|)$.
Totalt:

$$O\left(\sum_{i=0}^{|\texttt{xs}|-1} i\right) = O\left(|\texttt{xs}|\frac{|\texttt{xs}|-1}{2}\right) = O(|\texttt{xs}|^2)$$

Asymptotisk notation

Notationen är okänslig för vissa implementationsdetaljer/modellvariationer:

$$n = \Theta(n)$$
$$2n + 1 = \Theta(n)$$
$$100n + 10^8 = \Theta(n)$$
$$\lfloor 10^{-8} n \rfloor = \Theta(n)$$

Asymptotisk notation

Notera:

$$1 = O(n)$$
$$1 = O(n^2)$$
$$n = O(n \log n)$$
$$n = O(n^2)$$

Asymptotisk notation

Asymptotisk notation + rekursion/induktion kan vara förrädiskt. Var försiktiga.

(Se övning P1.)

Asymptotisk notation

Vad är bäst: $\Theta(n)$ eller $\Theta(n^2)$?

Sammanfattning

- Tidskomplexitet.
- Kostnadsmodeller.
- Asymptotisk komplexitet/notation.
- Analys av algoritmers tidskomplexitet.