adk11
2011-(08)aug-30

- Algoritmanalys
- Mästarsatsen
- Beräkningsmodeller
- Sorteringsalgoritmer

Definition:

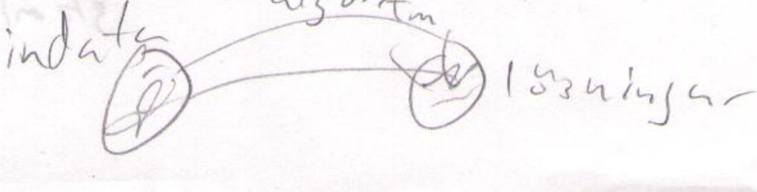
En ALGORITM är en ändlig beskrivning av hur man steg för steg löser ett problem.

En algoritm tar oftast INDATA som beskriver en PROBLEMINSTANS och producerar UTDATA som beskriver probleminstansens LÖSNING(ar)

En algoritm kan ses som en funktion.

A: Probleminstanser → Lösningar

En algoritm kan ses som en funktion.



Analys av algoritmer

Tidskomplexitet
- Hur lång tid tar algoritmen i värsta fallet?

Som funktion av vad?
Vad är ett tidssteg?

Minneskomplexitet

- Hur stort minne behöver algoritmen i värsta fallet?

Som funktion av vad?
Mätt i vad?

Tänk på att funktions- och proceduranrop också tar minne.

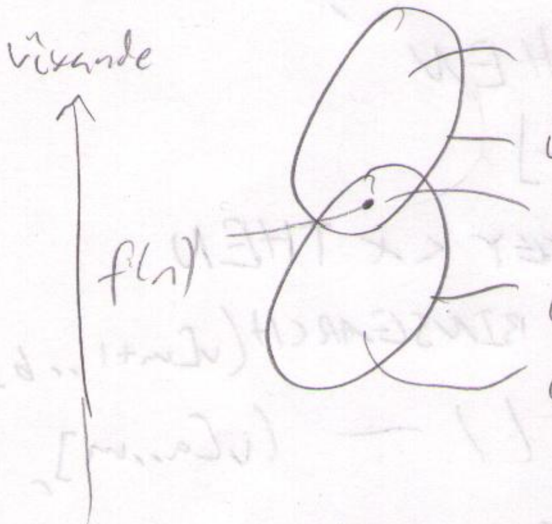
n = antalet tal som skall sorteras

Hur komplexitet kan anges

Hur ändras komplexiteten för växande storlek n på indata?

Mycket enklare om vi bortser från konstanta faktorer

Asymptotisk komplexitet — vad händer när n växer mot oändligheten?



$\omega(f(n))$ växer snabbare än $f(n)$

$\Omega(f(n))$ växer minst lika snabbt som f

$\Theta(f(n))$ växer lika snabbt som $f(n)$

$O(f(n))$ -//- högst lika snabbt som $f(n)$

$o(f(n))$ -//- långsammare än $f(n)$

$$\lim_{n \to \infty} \frac{g(n)}{f(n)} = \begin{cases} 0 & \text{om } g(n) \in o(f(n)) \\ c > 0 & \text{om } g(n) \in \Theta(f(n)) \\ \infty & \text{om } g(n) \in \omega(f(n)) \end{cases}$$

$O(f(n))$ (första två fallen), $\Omega(f(n))$ (sista två fallen)

$$T_{INS}(n) = 2n^2 - \cancel{3n + 8} \in \Theta(n^2) = \left(\Theta(17 n^2)\right)$$

$$3n \in O(n^2)$$

$$3n \notin \Theta(n^2)$$

KONSTANT TID $O(1)$

Exempel: binär sökning

```
BIN SEARCH (v[a..b], x) =
IF a < b THEN
    m ← ⌊(a+b)/2⌋
    IF v[m].KEY < x THEN
        RETURN BINSEARCH(v[m+1..b], x)
    ELSE        ——  // —— (v[a..m], x)
IF v[a].KEY = x THEN RETURN a
ELSE RETURN 'NOT FOUND'
```

$$T(1) = O(1)$$

$$T(n) = O(1) + T\left(\frac{n}{2}\right)$$

Lösning till vanliga rekursionsekvationer

Sats: ("Master theorem")

Om $a \geq 1$, $b > 1$, $d > 0$ så har rekursionsekvationen

$$\begin{cases} T(n) = aT(\frac{n}{2}) + f(n) \\ T(1) = d \end{cases}$$

Den asymptotiska lösningen

- $T(n) = \Theta(n^{\log_b a})$ om $f(n) = O(n^{\log_b a - \varepsilon})$ för något $\varepsilon > 0$
- $T(n) = \Theta(n^{\log_b a} \log n)$ om $f(n) = \Theta(n^{\log_b a})$
- $T(n) = \Theta(f(n))$ om $f(n) = \Omega(n^{\log_b a + \varepsilon})$ för något $\varepsilon > 0$

och $af(\frac{n}{b}) \leq cf(n)$ för någon konstant $c < 1$.

För alla tillräckligt stora n.

Analys av problem

Ringa in ett problems komplexitet!

Övre gräns:

Ge en algoritm som löser problemet. Algoritmens komplexitet är en övre gräns för problemets komplexitet.

Undre gräns:

Ofta svårt att ge. Egenskaper hos problemet måste användas.

Exempel:

- Måste titta på alla indata $\Rightarrow \Omega(n)$
- Måste producera hela utdatat
- Beslutsträd – Ett visst antal olika fall måste särskiljas

Lösning till vanliga rekursionsekvationer

Sats: ("Master theorem")

Om $a \geq 1$, $b > 1$, $d > 0$ så har rekursionsekvationen

$$\begin{cases} T(n) = aT(\frac{n}{2}) + f(n) \\ T(1) = d \end{cases}$$

Den asymptotiska lösningen

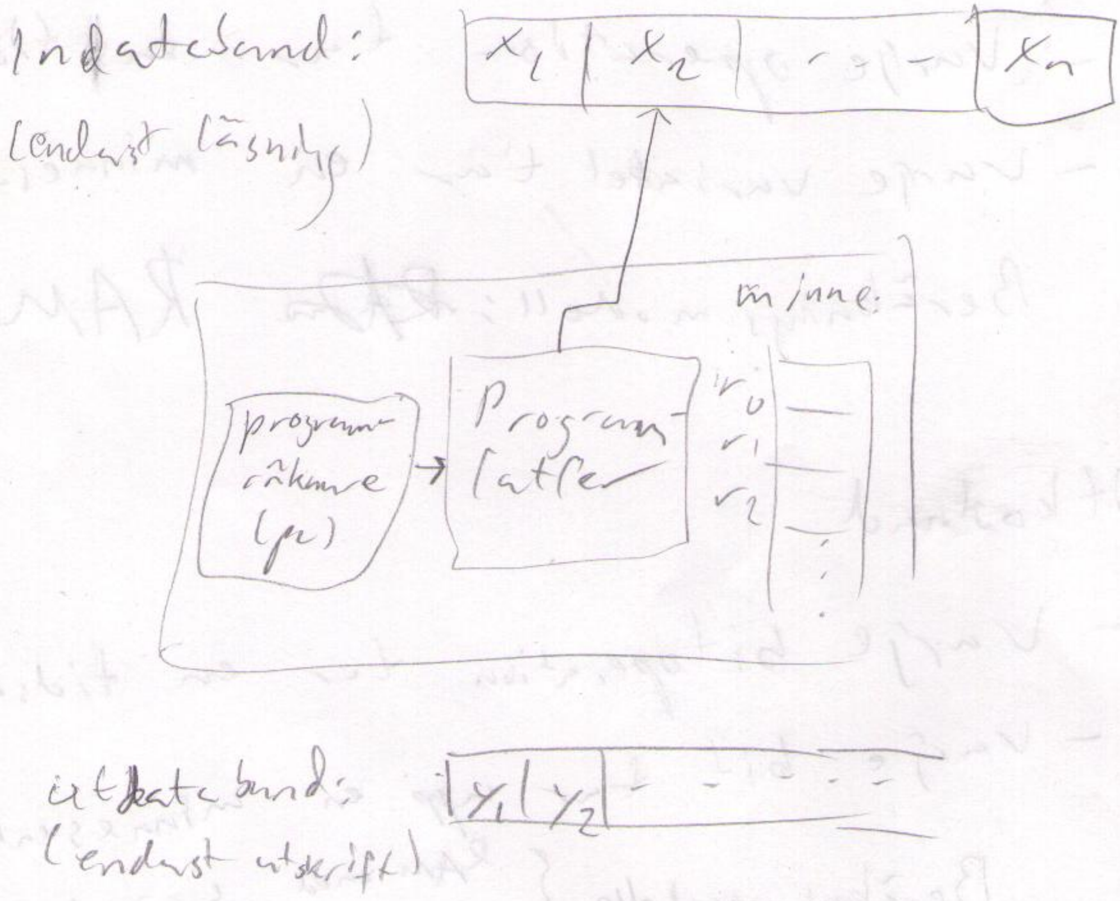
- $T(n) = \Theta(n^{\log_b a})$ om $f(n) = O(n^{\log_b a - \varepsilon})$ för något $\varepsilon > 0$
- $T(n) = \Theta(n^{\log_b a} \log n)$ om $f(n) = \Theta(n^{\log_b a})$
- $T(n) = \Theta(f(n))$ om $f(n) = \Omega(n^{\log_b a + \varepsilon})$ för något $\varepsilon > 0$

och $af(\frac{n}{b}) \leq cf(n)$ för någon konstant $c < 1$.

För alla tillräckligt stora n.

Beräkningsmodell 7: RAM

(Random access machine)



Programmet består av ...

Kostnadsmått

Enhetskostnad

- Varje operation tar en tidsenhet
- Varje variabel tar en minnesenhet

Beräkningsmodell: ~~RAm~~ RAM

Bitkostnad

- Varje bitoperation tar en tidsenhet
- Varje bit tar upp en minnesenhet

Beräkningsmodeller { RAM med begränsad ordlängd / turingmaskin

Används när algoritmen räknar med tal av godtycklig storlek

Exempel: Addition av två n-bitheltal

Tid $O(1)$ med enhetskostnad

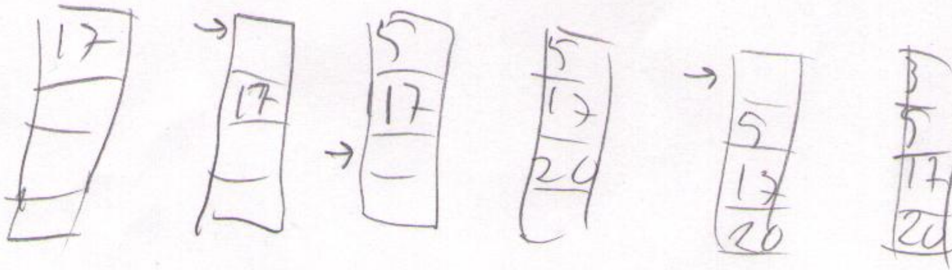
Tid $O(n)$ med bitkostnad

Insättningssortering (enkel)

Algoritm:

- Placera in första elementet på första platsen
- För varje nytt element x som ska sorteras in
  - Leta reda på var x ska in
    - Förskjut alla element till höger om den platsen ett steg
    - Sätt in x

Exempel) Sortera in talen 17 5 20 3



Antal operationer: $O(n^2)$

i medeltal	i värsta fall

Urvalssortering (enkel)

Algoritm:

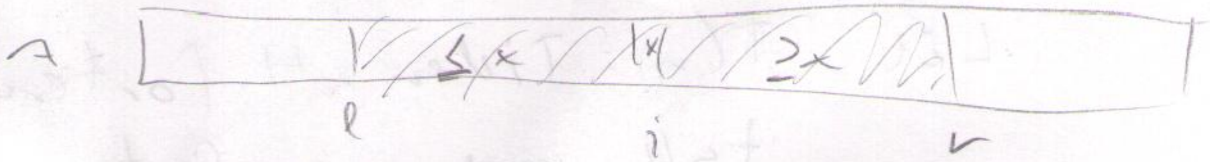
- Välj ut det minsta elementet
- Byt plats på minsta och första elementet
- Fortsätt på samma sätt med resten av elementen

Exempel:

Quicksort — $O(n \log n)$ i genomsnitt

1. Partitionera A efter ett godtyckligt element x, det vill säga ordna om A så att alla element som är $<x$ kommer till vänster, alla element som är $>x$ kommer till höger.

   Låt i vara indexet för platsen i A där x hamnar.



2. Anropa Quicksort rekursivt på den vänstra delen av A (från index l till $i-1$).
3. Anropa Quicksort rekursivt på den högra delen av A (från index $i+1$ till r)

När ska rekursionen avbrytas?

Alternativ 1: Då $l \ge r$ det vill säga då högst ett element finns i

Exempel: Mergesort

MERGESORT $(v[i..j])$ =
IF $i<j$ THEN
$m \leftarrow \left\lfloor \frac{i+j}{2} \right\rfloor$
MERGESORT $(v[i...m])$
— || — $(v[m+1..j])$
MERGE $(v[i..j], v[i...m], v[m+1..j])$

Analys:

Låt $T(n)$ = Tiden att sortera n tal med mergesort

$$T(n) = \begin{cases} \Theta(1) & \text{om } n=1 \\ T(\lfloor \frac{n}{2} \rfloor) + T(\lceil \frac{n}{2} \rceil) + \Theta(n) & \text{om } n>1 \end{cases}$$

Om $n = 2^n$ får vi T . . . . .

"Master theorem" eftersom $n^{\log_2 2} = n^1 \in \Theta(n)$

2,