ALGORITMER (DATASTRUKTURER) OCH KOMPLEXITET

FÖRELÄSNING 2

ALGORITMER

- ALGORITMANALYS
- MÄSTARSATSEN
- BERÄKNINGSMODELLER
- REPETITION AV SORTERING

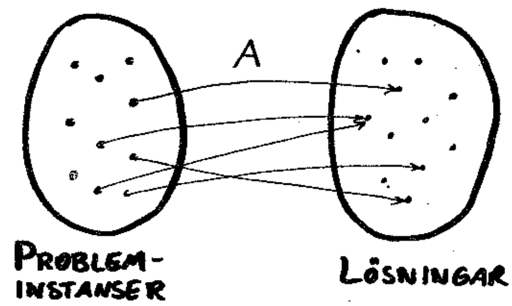
DEFINITION:

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.

EN ALGORITM KAN SES SOM EN FUNKTION

A: PROBLEMINSTANSER $\rightarrow$ LÖSNINGAR



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?

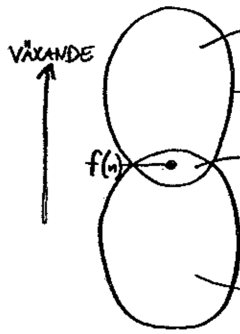
TÄNK PÅ ATT FUNKTIONS- OCH PROCEDURANROP OCKSÅ TAR MINNE.

HUR KOMPLEXITET KAN ANGES

HUR ÄNDRAS KOMPLEXITETEN FÖR VÄXANDE STORLEK n PÅ INDATA?

ASYMPTOTISK KOMPLEXITET — VAD HÄNDER NÄR n VÄXER MOT OÄNDLIGHETEN?

MYCKET ENKLARE OM VI BORTSER FRÅN KONSTANTA FAKTORER.



- $\omega(f(n))$ VÄXER SNABBARE ÄN $f(n)$
- $\Omega(f(n))$ VÄXER MINST LIKA SNABBT SOM $f(n)$
- $\Theta(f(n))$ VÄXER LIKA SNABBT SOM $f(n)$
- $\mathcal{O}(f(n))$ VÄXER HÖGST LIKA SNABBT SOM $f(n)$
- $o(f(n))$ VÄXER LÅNGSAMMARE ÄN $f(n)$

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

(de två första fallen: $\mathcal{O}(f(n))$; de två sista fallen: $\Omega(f(n))$)

LÖSNING TILL VANLIGA REKURSIONSEKVATIONER

SATS: ("MASTER THEOREM")

OM $a \geq 1$, $b > 1$, $d > 0$ SÅ HAR REKURSIONSEKVATIONEN

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

DEN ASYMPTOTISKA LÖSNINGEN

- $T(n) = \Theta(n^{\log_b a})$ OM $f(n) = \mathcal{O}(n^{\log_b a - \varepsilon})$ 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 $af\left(\frac{n}{b}\right) \leq c \cdot f(n)$ FÖR NÅGON KONSTANT $c < 1$ FÖR ALLA TILLRÄCKLIGT STORA n.

EXEMPEL: BINÄRSÖKNING

```
BINSEARCH(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 RETURN BINSEARCH(v[a..m], x)
IF v[a].KEY=x THEN RETURN a
ELSE RETURN 'NOT FOUND'
```

ANALYS:

LÅT $T(n) = $ TIDEN ATT SÖKA BLAND n TAL MED BINSEARCH. (I VÄRSTA FALL)

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

OM $n = 2^m$ FÅR VI $T(n) = \begin{cases} \Theta(1) & \text{om } n=1 \\ T\left(\frac{n}{2}\right) + \Theta(1) & \text{om } n>1 \end{cases}$

MÄSTARSATSEN (MASTER THEOREM) SÄGER DÅ

$n^{\log_2 1} = n^0 \in \Theta(1)$

DÅ ÄR $\underline{T(n) = \Theta(\log n)}$

ANALYS AV PROBLEM

RINGA IN ETT PROBLEMS KOMPLEXITET!

ÖVRE GRÄNS:

GE EN ALGORITM SOM LÖSER PROBLEMET.

ALGORITMENS KOMPLEXITET ÄR EN <u>ÖVRE GRÄNS</u> FÖR PROBLEMETS KOMPLEXITET.

UNDRE GRÄNS:

OFTA SVÅRT ATT ANGE.

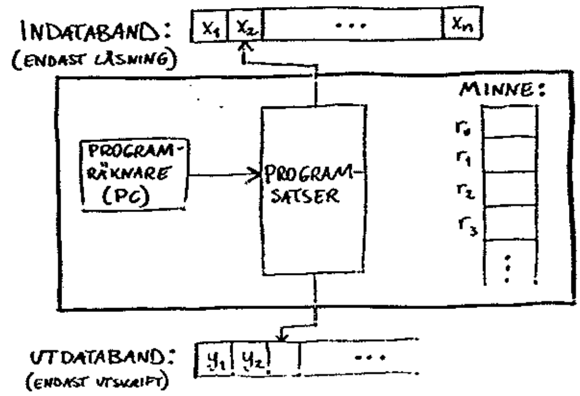
EGENSKAPER HOS PROBLEMET MÅSTE ANVÄNDAS.

EXEMPEL:

- MÅSTE TITTA PÅ ALLA INDATA $\Rightarrow \Omega(n)$
- MÅSTE PRODUCERA HELA UTDATA
- BESLUTSTRÄD — ETT VISST ANTAL OLIKA FALL MÅSTE SÄRSKILJAS

BERÄKNINGSMODELL 1: RAM

(RANDOM ACCESS MACHINE)



PROGRAMMET BESTÅR AV VANLIGA SATSER SOM UTFÖRS SEKVENSIELLT (INTE PARALLELLT).

VARJE SATS KAN BARA LÄSA OCH PÅVERKA ETT KONSTANT ANTAL MINNESPLATSER. BARA EN SYMBOL KAN LÄSAS/SKRIVAS I TAGET. VARJE SATS TAR KONSTANT TID.

PÅ GRUND AV $O()$-NOTATIONENS ROBUSTHET KAN VI STRUNTA I VILKA VÄRDEN KONSTANTERNA HAR

KOSTNADSMÅTT

ENHETSKOSTNAD

— VARJE OPERATION TAR EN TIDSENHET

— VARJE VARIABEL TAR EN MINNESENHET

BERÄKNINGSMODELL: RAM

BITKOSTNAD

— VARJE BITOPERATION TAR EN TIDSENHET

— VARJE BIT TAR UPP EN MINNESENHET

BERÄKNINGSMODELLER $\begin{cases}\text{RAM MED BEGRÄNSAD ORDLÄNGD}\\ \text{TURINGMASKIN}\end{cases}$

ANVÄNDS NÄR ALGORITMEN RÄKNAR MED TAL AV GODTYCKLIG STORLEK.

EXEMPEL: ADDITION AV TVÅ n-BITSHELTAL

TID $O(1)$ MED ENHETSKOSTNAD

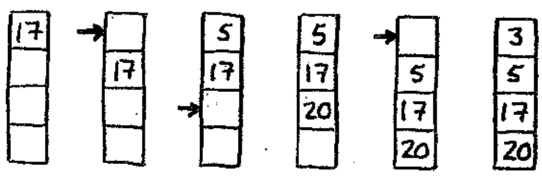
TID $O(n)$ MED BITKOSTNAD

INSÄTTNINGSSORTERING

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

Ex) SORTERA IN TALEN 17 5 20 3



ANTAL OP:

$\Theta(n^2)$

	I MEDELTAL	I VÄRSTA FALLET
JÄMFÖRELSER	$\frac{n^2}{4}$	$\frac{n^2}{2}$
FLYTTAR	$\frac{n^2}{4}$	$\frac{n^2}{2}$

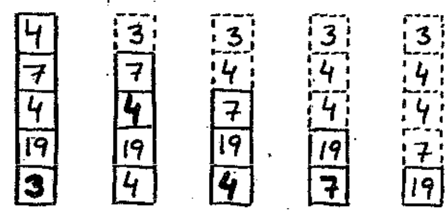
MINNESÅTGÅNG: 0 EXTRA ELEMENT

URVALSSORTERING

ALGORITM:

- VÄLJ UT DET MINSTA ELEMENTET
- BYT PLATS PÅ MINSTA OCH FÖRSTA ELEMENTET
- FORTSÄTT PÅ SAMMA SÄTT MED RESTEN AV ELEMENTEN

Ex)



ANTAL OPERATIONER:

$\frac{n(n-1)}{2}$ DVS $\Theta(n^2)$ JÄMFÖRELSER

$n-1$ PLATSBYTEN

DETTA GÄLLER OBEROENDE AV HUR INDATA ÄR ORDNAT

MINNESÅTGÅNG:

1 EXTRA ELEMENT

EXEMPEL: MERGESORT

MERGESORT(v[i..j]) =

```
IF i<j THEN
   m ← ⌊(i+j)/2⌋
   MERGESORT(v[i..m])
   MERGESORT(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 $n=2^m$ FÅR VI $T(n)=\begin{cases}\Theta(1) & \text{om } n=1\\ 2T(\frac{n}{2})+\Theta(n) & \text{om } n>1\end{cases}$

"MASTER THEOREM": EFTERSOM $n^{\log_2 2}=n^1\in\Theta(n)$

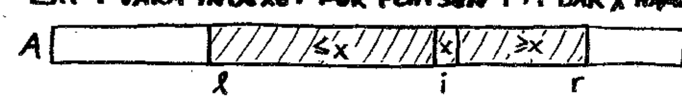
SÅ ÄR $T(n)=\Theta(n\log n)$

MINNESÅTGÅNG: n EXTRA ELEMENT (FÖR MERGE)

QUICKSORT — $O(n \log n)$ I GENOMSNITT

ALGORITM SOM SORTERAR ARRAYEN $A[l..r]$:

1. PARTITIONERA A EFTER ETT GODTYCKLIGT ELEMENT X, DVS 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 $l\geq r$, DVS DÅ HÖGST ETT ELEMENT FINNS I ARRAYEN.

ALTERNATIV 2: DÅ $r-l<10$ DVS DÅ HÖGST TIO ELEMENT FINNS I ARRAYEN. SORTERA DE ÅTERSTÅENDE ELEMENTEN MED INSÄTTNINGSSORTERING.