Sekvensielle sorteringsalgoritmer

Sekvensielle algoritmer

- Sorterer fra starten av array
- Bruker typisk to løkker inne i hverandre, $O(n^2)$
- Skal bare se på disse tre algoritmene:
  - Utplukksortering
  - Innstikksortering
  - Bubble sort

Utplukksortering (selection sort)

- Går gjennom arrayen *n* - 1 ganger
- I gjennomløp nr *i (i = 0, 1, 2,… , n - 2)* :
  - Arrayen er sortert fra starten og t.o.m. indeks *i* - 1
  - Finner det minste av de usorterte elementene på indeksene *i*, *i* + 1,… , *n* - 1 (“utplukk”/ “selection”)
  - Setter dette minste elementet inn på indeks *i* med en ombytting (swapping)
- Etter *n* – 1 gjennomløp står det største elementet igjen på index *n* – 1, og hele arrayen er sortert

Utplukksortering, eksempel

64 25 12 22 11

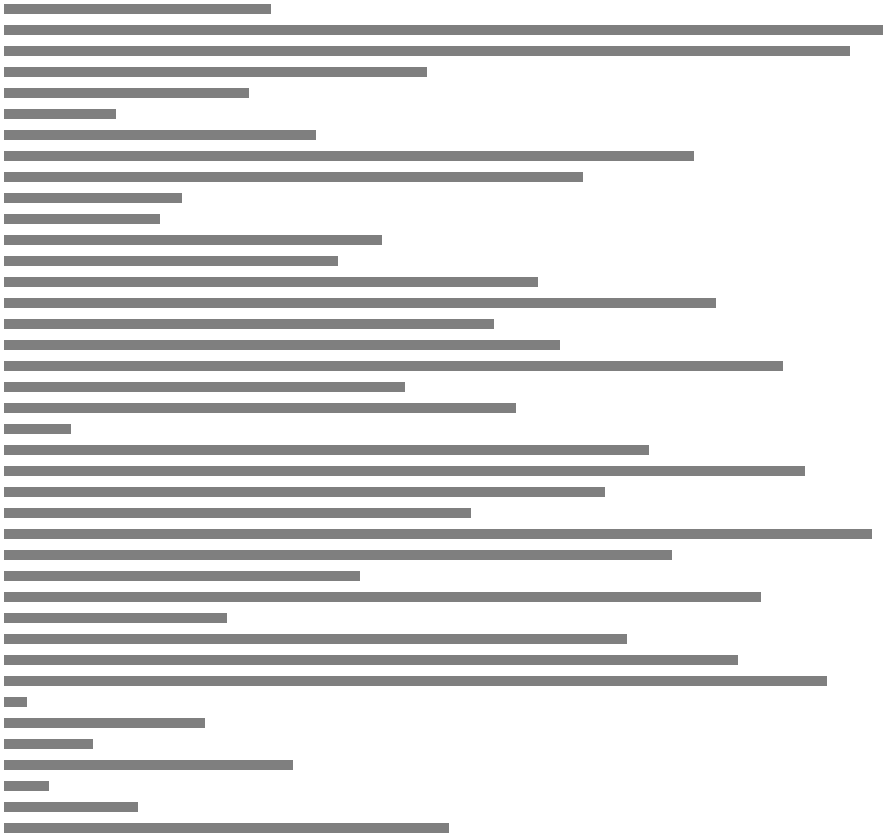
11 25 12 22 64

11 12 25 22 64

11 12 22 25 64

11 12 22 25 64

Utplukksortering, animasjon



Effektivitet og implementasjon

- Utplukksortering programmeres enkelt med to løkker inne i hverandre:
  - Ytre løkke går n - 1 ganger
  - Indre løkke går n - 2, n - 3,... , 2, 1 ganger
  - Totalt: $O(n^2)$
- Effektiviteten av utplukksortering er ikke avhengig av dataene, algoritmen gjør alltid like mange tester og ombyttinger
- Se Java-koden

Innstikksortering (insertion sort)

- Går gjennom arrayen *n* - 1 ganger
- I gjennomløp nr *i (i = 1, 2, 3,… , n - 1)* :
  - Arrayen er sortert fra starten og t.o.m. indeks *i* - 1
  - Setter element nr *i* inn på riktig plass blant de i - 1 første elementene (“insertion”/ ”innstikk”)
  - Innsettingen gjøres ved å flytte alle elementene som er større enn element nr *i* et "hakk" bakover, og deretter sette inn element nr *i* på indeksen som nå er blitt ledig
- Etter *n* – 1 gjennomløp er alle elementer satt inn på riktig plass og hele arrayen er sortert

Innstikksortering, eksempel

3 7 4 9 5 2 6 1 (0 flytt)

3 7 4 9 5 2 6 1 (1 flytt)

3 4 7 9 5 2 6 1 (0 flytt)

3 4 7 9 5 2 6 1 (2 flytt)

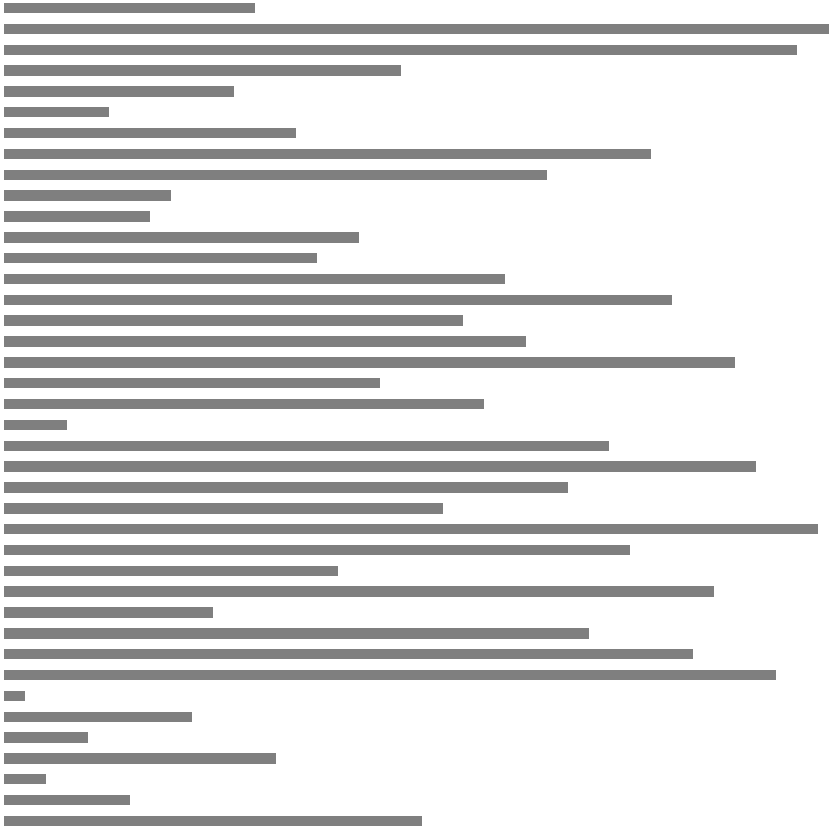
3 4 5 7 9 2 6 1 (5 flytt)

2 3 4 5 7 9 6 1 (2 flytt)

2 3 4 5 6 7 9 1 (7 flytt)

1 2 3 4 5 6 7 9

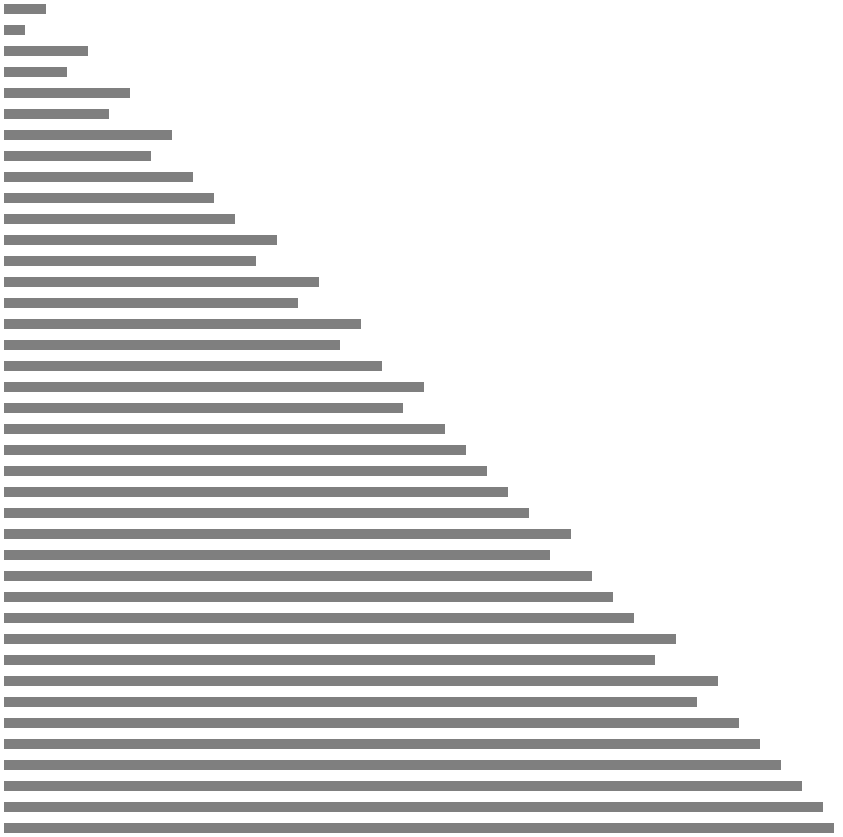
Innstikksortering, animasjon



Effektivitet og implementasjon

- Programmeres med to løkker inne i hverandre:
  - Ytre løkke går n - 1 ganger
  - Indre løkke går inntil element i står riktig, maksimalt: 1, 2, 3,..., n - 1 ganger
  - Worst-case: $O(n^2)$
- Effektivitet av innstikksort. avhenger av dataene:
  - Er $O(n)$ for nesten sorterte arrayer
  - Gjennomsnittlig (random data) $O(n^2)$
- Se Java-koden

Innstikksortering av nesten sorterte data, animasjon



Bubble sort *

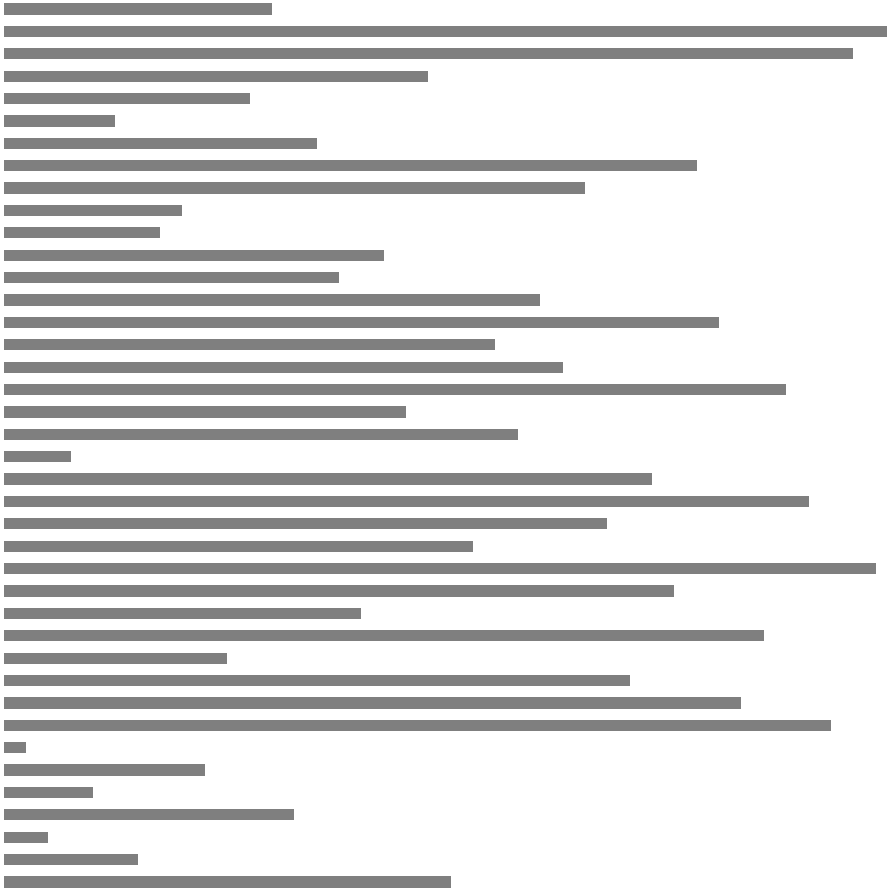
- I gjennomløp nummer *i*, *i = 0, 1, 2,, … , n - 2* :
  - Arrayen er sortert fra starten og opp t.o.m. indeks *i* - 1
  - Setter minste verdi i usortert del av array på indeks *i*
  - Finner minste verdi ved å starte i indeks *n* - 1 og swappe med foregående element hvis dette er større
  - Fortsetter å swappe minste nabolement fremover, til det minste av de usorterte elementene står i indeks *i*
  - Små verdier vil "boble" oppover i arrayen, store verdier "synker" nedover ("sink sort"/ "percolation sort")
- Etter *n* – 1 gjennomløp er hele arrayen er sortert

*: Litt anderledes enn i læreboka

Bubble sort, eksempel, $n = 5$

$i = 0$	$i = 1$	$i = 2$	$i = 3$
64 10 12 22 11	10 64 11 12 22	10 11 64 12 22	10 11 12 64 22
64 10 12 11 22	10 64 11 12 22	10 11 64 12 22	10 11 12 22 64
64 10 11 12 22	10 64 11 12 22	10 11 12 64 22	
64 10 11 12 22	10 11 64 12 22		
10 64 11 12 22			

Bubble sort, animasjon



Effektivitet og implementasjon

- Bubble sort programmeres med to løkker i hverandre:
  - Ytre løkke går n - 1 ganger
  - Indre løkke går n - 1, n - 2, n - 3,... , 2, 1 ganger
  - Alltid $O(n^2)$
- Effektivitet av bubble sort avhenger av dataene:
  - Går langsomt hvis det er mye ombytting av verdier
  - Kan enkelt programmes til å avbryte med en gang arrayen er sortert (0 swaps i et gjennomløp)
- Se Java-koden

Sammenligning av $O(n^2)$ metoder

- Effektivitet avhenger av hvor mye flytting av data som gjøres – en swap tar *mye* lenger tid enn en if-test

Utplukksortering	• Forutsigbar, alltid samme effektivitet
Instikksortering	• Ofte mest effektiv, lite swapping • Rask for nesten-sorterte arrayer • Bedre enn “smarte” algoritmer for små *n*
Bubble sort	• Langsom pga. mye swapping, lite brukt

- Se testprogram for sorteringsalgoritmer