



# Sortering i lineær tid

---

29.09.2005



# Generelt

---

- Linear tid vil si  $(n)$  som forventet kjøretid
- Baserer seg ikke på sammenligninger
- Beste sortering basert på sammenligninger har  $(n \cdot \log(n))$  som forventet kjøretid
- Input-settet må ha spesielle egenskaper
- 3 metoder beskrives og vises i dag
- Counting sort
- Radix sort
- Bucket sort



# Counting sort (s168 i boka)

---

- Antar at alle elementer som sorteres er et heltall i intervallet fra 0 til  $k$ , alternativt i intervallet  $a$  til  $b$  der  $b - a = k$
- Dersom  $k = O(n)$ , vil sortering ta  $O(n)$
- Stabil, dvs elementer med lik verdi beholder sin opprinnelige rekkefølge.
- Animasjon:  
<http://www.cse.iitk.ac.in/users/dsrkg/cs210/applets/sortingII/countingSort/countingSort.html>



# Counting sort kode

---

- NB! Bruker 1 som første indeks i en liste
- for  $i \leftarrow 1$  to  $k$  do
- $c[i] \leftarrow 0$
- for  $j \leftarrow 1$  to  $n$  do
- $c[A[j]] \leftarrow c[A[j]] + 1$
- //  $c[i]$  now contains the number of elements equal to  $i$
- for  $i \leftarrow 2$  to  $k$  do
- $c[i] \leftarrow c[i] + c[i-1]$
- //  $c[i]$  now contains the number of elements  $\leq i$
- for  $j \leftarrow n$  downto  $1$  do
- $B[c[A[i]]] \leftarrow A[j]$
- $c[A[i]] \leftarrow c[A[j]] - 1$



# Radix sort (s170 i boka)

---

- Sorterer en og en del av det som skal sorteres
- Eksempel: kortstokk, tall, ord
- Kjøretid:  $n$  elementer som skal sorteres som deles inn i  $d$  deler som kan ha  $k$  mulige verdier  $\rightarrow (d * (n + k))$
- Må ha støtte av en stabil sorteringsalgoritme
- Animasjon (flere animasjon på samme side):  
<http://oopweb.com/Algorithms/Documents/AnimatedAlgorithms/VolumeFrames>



# Bucket sort (s174 i boka)

---

- Optimal i en uniform fordeling av input-elementer
- Setter elementer inn i båser(buckets) for så og sorterer hver bøtte. Til slutt setter en sammen dette
- Ideen er at ved å sette ting i bås slipper en å sammenligne det med det i de andre båsene når det skal sorteres.
- Kjøretid: Innsetting i båser tar  $O(n)$  worst case. Sorteringen i hver bås er avhengig av hvor mange elementer som finnes i den enkelte bås.

# Bucket sort eksempel

A		B			
1	0,78	0	/		
2	0,17	1	-->	0,12	--> 0,17
3	0,39	2	-->	0,21	--> 0,23
4	0,26	3	-->	0,39	--> 0,26
5	0,72	4	/		
6	0,94	5	/		
7	0,21	6	-->	0,68	
8	0,12	7	-->	0,72	--> 0,78
9	0,23	8	/		
10	0,68	9	-->	0,94	

- Sorter tall mellom 0 og 1
- Båser fra (0)0 - 0.1, (1)0.1 - 0.2 etc.
- A, usortet liste
- B, båsene med elementene lagret i lenket liste. De er sortert etter å ha blitt plassert