# SI-möte #2, Algoritmer och datastrukturer

Elias Castegren
elca7381@student.uu.se

1 april 2009

## Begrepp

$i$) Hur fungerar Insertion sort? Beskriv algoritmen med ord, inte med kod. Vad är det bästa, respektive sämsta utgångsläget för Insertion sort?

$ii$) Hur fungerar Merge sort? Vilka är de tre stegen i algoritmen? Vad är basfallet i rekursionen?

$iii$) Hur fungerar Quick sort? Beskriv algoritmen med ord, inte med kod. Hur går partitioneringen till i bästa respektive sämsta fall?

$iv$) Vad menas med att en sorteringsalgoritm är stabil?

## Övningar

**1.**
Sortera följande listor i stigande ordning för hand (på papper eller med spelkort) med Insertion sort, Merge sort (dela listorna rakt av på mitten) och Quick sort (välj det första elementet som pivotelement). Räkna antalet heltalsjämförelser (och antalet delningar med Merge sort) för varje algoritm och försök förutsäga vilken algoritm som kommer ge det minsta antalet operationer (du behöver inte räkna med tiden det tar att flytta på ett element).

$i$) `[1,3,2,5,6,4,8,7,9,10]` $ii$) `[6,8,1,3,10,9,7,2,4,5]`

$iii$) `[7,8,3,10,9,2,1,4,6,5]` $iv$) `[10,9,8,6,7,5,4,3,1,2]`

Slumpa en ny lista och sortera på valfritt sätt igen om du vill förstå någon av algoritmerna bättre!

**2.**
Fyll i tabellen med tidskomplexiteten för algoritmerna i bästa, genomsnittliga och värsta fall.

| | Best | Average | Worst |
|---|---|---|---|
| Insertion sort | | | |
| Merge sort | | | |
| Quick sort | | | |

**3.**
När är de olika sorteringsalgoritmerna att föredra? Finns det någon poäng med att använda Insertion sort trots att algoritmen i allmänhet har högre komplexitet? Vad kan det finnas för anledningar till att välja Quicksort framför Merge sort eller tvärtom?

**4.**
Skriv en implementation av Merge sort i ML-kod (helst utan att titta på koden i föreläsningsnoterna) för heltalslistor. Vilka hjälpfunktioner behöver du? Börja med funktionsspecifikationerna! Skriv sedan upp en rekursion och hitta en sluten form.

**5.**
Att i Merge sort dela på listan genom att först hitta mitten med `(length l) div 2` kommer att kräva att listan gås igenom en och en halv gång. Hur skulle man kunna dela listan i två lika stora delar med bara en traversering? Är sortering med den nya delningsfunktionen fortfarande stabil? Påverkas sorteringsalgoritmens tidskomplexitet?

**6.**
Skriv en implementation av Quick sort i ML-kod (fortfarande utan att titta på koden i föreläsningsnoterna) för heltalslistor. Vilka hjälpfunktioner behöver du? Börja med funktionsspecifikationerna! Skriv sedan upp en rekursion och hitta en sluten form för det bästa, genomsnittliga och värsta fallet.

**7.**
Hur kan man göra sorteringsalgoritmerna polymorfa? Du behöver inte skriva om hela koden, beskriv bara hur det skulle gå till i ML.

**8.**
Om man har en datorrepresentation av en vanlig kortlek (eller någon annan mängd av element med bestämd ordning och antal) så är det möjligt att sortera den med tidskomplexiteten $\Theta(n)$. Hur skulle man kunna göra detta? (Fundera på vilken datastruktur man skulle kunna använda för att lagra korten)

---

*Lycka till*!