

# Inlämningsuppgift 3

Algoritmer och Datastrukturer

**Johan Kämpe**






UIS16

2017-02-XX

## Svar på deluppgifter:

### 1. Ladda ner och packa upp filen metriker.zip från Moodle.

Svar:

 uppgifter.txt	1,25 KB	16 hrs
 main.c	2,93 KB	17 hrs
 sort.c	1,19 KB	17 hrs
 Makefile	126 bytes	17 hrs
 sort.h	199 bytes	21 hrs

### 2. Kompilera och kör programmet med anropet main test.txt.

Finns det felmeddelanden vid körning? Vad skall göras för att få bort dem?

Svar:

Funktionen **check\_if\_sorted** i main.c ger returvärde 0 om en array inte blev korrekt sorterad efter ett anrop av en sorteringsfunktion.

Om detta är fallet skrivs den angivna sorteringsfunktions namn ut. I vårt fall får vi meddelandet: **ERROR: algorithm Insertion sort doesn't sort the array....**

Felet uppstår därför att funktionen **insert\_sort** anropas, men funktionen innehåller ingen kod, och sorterar således inte arrayen.

Det finns flera sätt att få bort felmeddelandena, men det bästa vore antagligen att implementera **insertion\_sort** i filen **sort.c**.

```
void insert_sort(int *array, int size) {  
    //Bygg funktion  
}
```

Andra alternativ är att ta bort anropet till **check\_if\_sorted**, eller att ta bort **insert\_sort** från struktur-arrayen **algorithms** och ändra dess antal **element** till 3.

```
#define NUM_ALGORITHMS 3//4  
  
sort_handle algorithms[NUM_ALGORITHMS] = {  
    {"Bubblesort", ZERO_STAT, bubble_sort},  
    //{"Insertion sort", ZERO_STAT, insert_sort},  
    {"Quicksort", ZERO_STAT, quick_sort},  
    {"Selection sort", ZERO_STAT, select_sort}  
};
```

3. Analysera de tre algoritmerna *bubble sort*, *quick sort* och *selection sort*. Undersök filen *test.txt* och koden.
- Har algoritmen enkel kod jämfört med de andra?
  - Anropar algoritmen sig själv?
  - Hur snabb är algoritmen jämfört med de två andra algoritmerna om arrayen har storleken 10? Ange procent!
  - Ange snabbhet likt uppgift 3c, fast för arraystorlek 100.
  - Ange snabbhet likt uppgift 3c, fast för arraystorlek 1000.

Svar:

#### ANALYS AV BUBBLE SORT

- Bubble sort har den kod som ser mest enkel ut enligt mig, jämfört med de två andra.
- Bubble sort-algoritmen anropar inte sig själv.
- Jag får tiden 0.00 för samtliga algoritmer på arraystorlek 10. Och således kan de anses vara lika snabba för en sådan arraystorlek: **0 %**

num	Bubblesort	Insertion sort	Quicksort	Selection sort	Shell sort
10	0.00	0.00	0.00	0.00	0.00
20	0.00	0.00	0.00	0.00	0.00

- Resultatet i sorteringstid för arraystorlek 100 skiftar mellan körningar, det pendlar mellan 0,00 och 40,00 för samtliga algoritmer. Jag gör antagandet att de är ungefär lika snabba för denna storlek: **0 %**

num	Bubblesort	Insertion sort	Quicksort	Selection sort	Shell sort
100	0.00	0.00	20.00	40.00	20.00
200	120.00	140.00	20.00	60.00	40.00

- Sorteringshastigheter för arraystorlek 1000:

num	Bubblesort	Insertion sort	Quicksort	Selection sort	Shell sort
1000	2.22	2.04	0.10	1.70	0.14
2000	7.50	8.28	0.26	6.72	0.30

Bubble sort sorterar arrayen på **130,6 %** av tiden det tar för selection sort.

Bubble sort sorterar arrayen på **2220 %** av tiden det tar för quick sort.

$$\left(\frac{2,22}{1,70}\right) * 100 \approx 130,6$$

$$\left(\frac{2,22}{0,1}\right) * 100 = 2220$$

**ANALYS AV SELECTION SORT**

- a. Selection sort har, enligt mig, kod som är mer svårförstådd kod än bubble sort, och enklare kod än quick sort.
- b. Selection sort -algoritmen anropar inte sig själv.
- c. Se uppgift c för bubble sort-analysen: **0 %**
- d. Se uppgift d för bubble sort-analysen: **0 %**
- e. Selection sort sorterar arrayen på **76,6 %** av tiden det tar för bubble sort.  
Selection sort sorterar arrayen på **1700 %** av tiden det tar för quick sort.

$$\left(\frac{1,70}{2,22}\right) * 100 \approx 76,6$$

$$\left(\frac{1,70}{0,1}\right) * 100 = 1700$$

**ANALYS AV QUICK SORT**

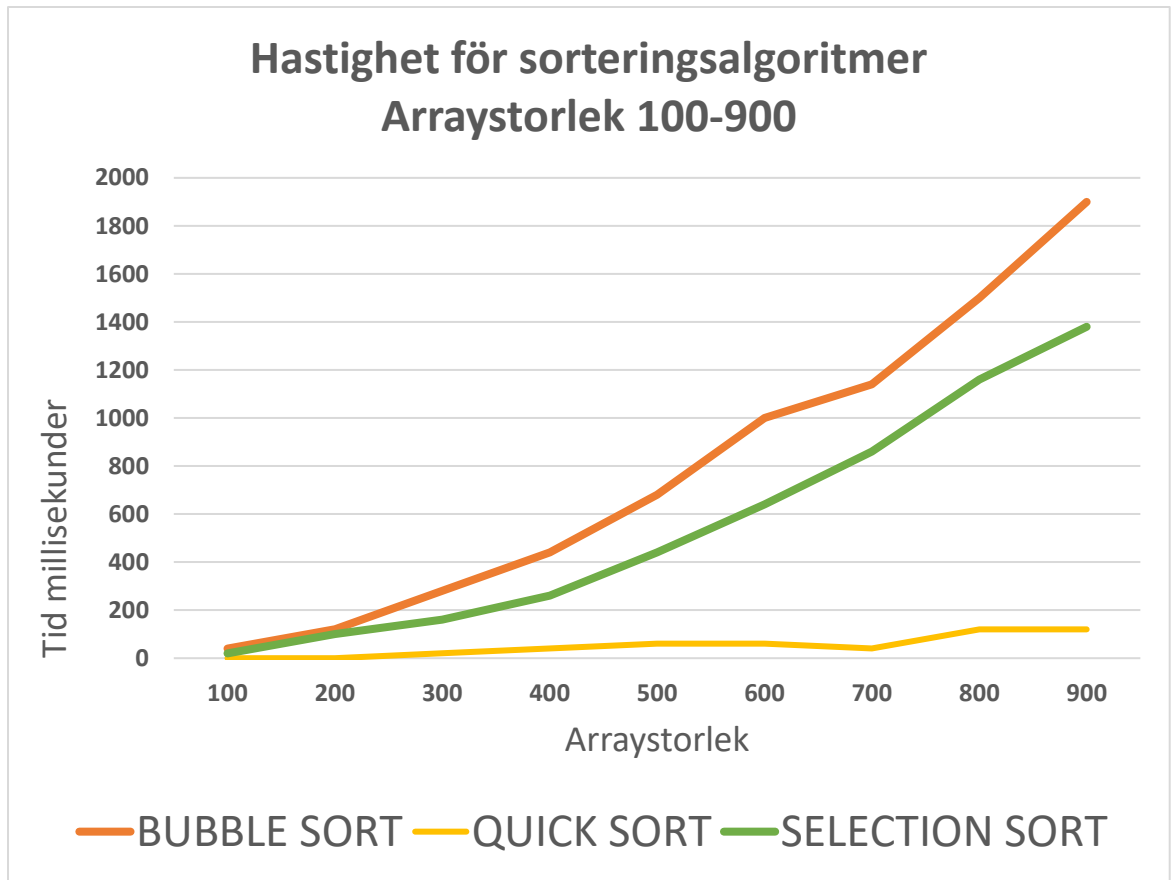
- a. Quick sort har, enligt mig, kod som är mycket mer svårförstådd än de två andra algoritmernas kod. Den använder sig dessutom av två extra stödfunktioner, utöver funktionen `_swap`.
- b. Quick sort-algoritmen anropar sig själv.
- c. Se uppgift c för bubble sort-analysen: **0 %**
- d. Se uppgift d för bubble sort-analysen: **0 %**
- e. Quick sort sorterar arrayen på **4,5 %** av tiden det tar för bubble sort.  
Quick sort sorterar arrayen på **5,9 %** av tiden det tar för selection sort.

$$\left(\frac{0,1}{2,22}\right) * 100 \approx 4,5$$

$$\left(\frac{0,1}{1,70}\right) * 100 \approx 5,9$$

**4. Rita upp de tre algoritmerna i ett diagram för arraystorlekarna 100-900.**

Svar:

**5. Implementera insertion sort i `sort.c`**

Svar: Se kod.

**6. Utvärdera insertion sort enligt uppgift 3**

- a. *Insertion sort* har, enligt mig, kod som är likställt avancerad med *Selection sort*, dvs. svårare än bubble sort, fast enklare än *quick sort*.
- b. Insertion sort-algoritmen med min implementation anropar inte sig själv.
- c. Se uppgift c för bubble sort-analysen: **0 %**
- d. Se uppgift d för bubble sort-analysen: **0 %**
- e. Selection sort sorterar arrayen på **91,9 %** av tiden det tar för bubble sort.  
Selection sort sorterar arrayen på **120 %** av tiden det tar för selection sort.  
Selection sort sorterar arrayen på **2040 %** av tiden det tar för quick sort.

$$\left(\frac{2,04}{2,22}\right) * 100 \approx 91,9$$

$$\left(\frac{2,04}{1,70}\right) * 100 = 120$$

$$\left(\frac{2,04}{0,1}\right) * 100 = 2040$$

**7. Implementera shell sort i *sort.c***

Svar: Se kod.

**8. Utvärdera shell sort enligt uppgift 3**

- a. *Shell sort* har, enligt mig, mycket mer avancerad kod än *Selection sort*, *bubble sort* och *Insertion sort*. Jag har svårt att avgöra om den känns enklare än koden för quick sort.
- b. Shell sort-algoritmen med min implementation anropar inte sig själv.
- c. Se uppgift c för bubble sort-analysen: **0 %**
- d. Se uppgift d för bubble sort-analysen: **0 %**
- e. Shell sort sorterar arrayen på **6,3 %** av tiden det tar för bubble sort.  
Shell sort sorterar arrayen på **8,2 %** av tiden det tar för selection sort.  
Shell sort sorterar arrayen på **6,7 %** av tiden det tar för insertion sort.  
Shell sort sorterar arrayen på **140 %** av tiden det tar för quick sort.

$$\left(\frac{0,14}{2,22}\right) * 100 \approx 6,3$$

$$\left(\frac{0,14}{1,70}\right) * 100 \approx 8,2$$

$$\left(\frac{0,14}{2,04}\right) * 100 \approx 6,7$$

$$\left(\frac{0,14}{0,1}\right) * 100 = 140$$