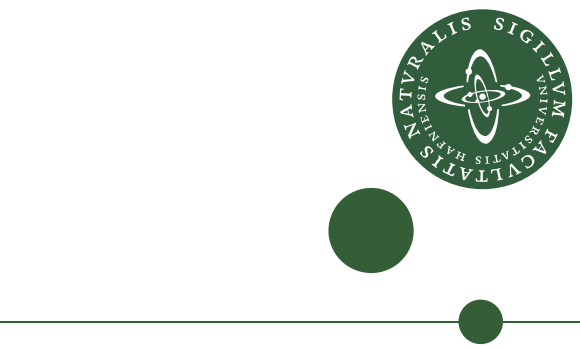


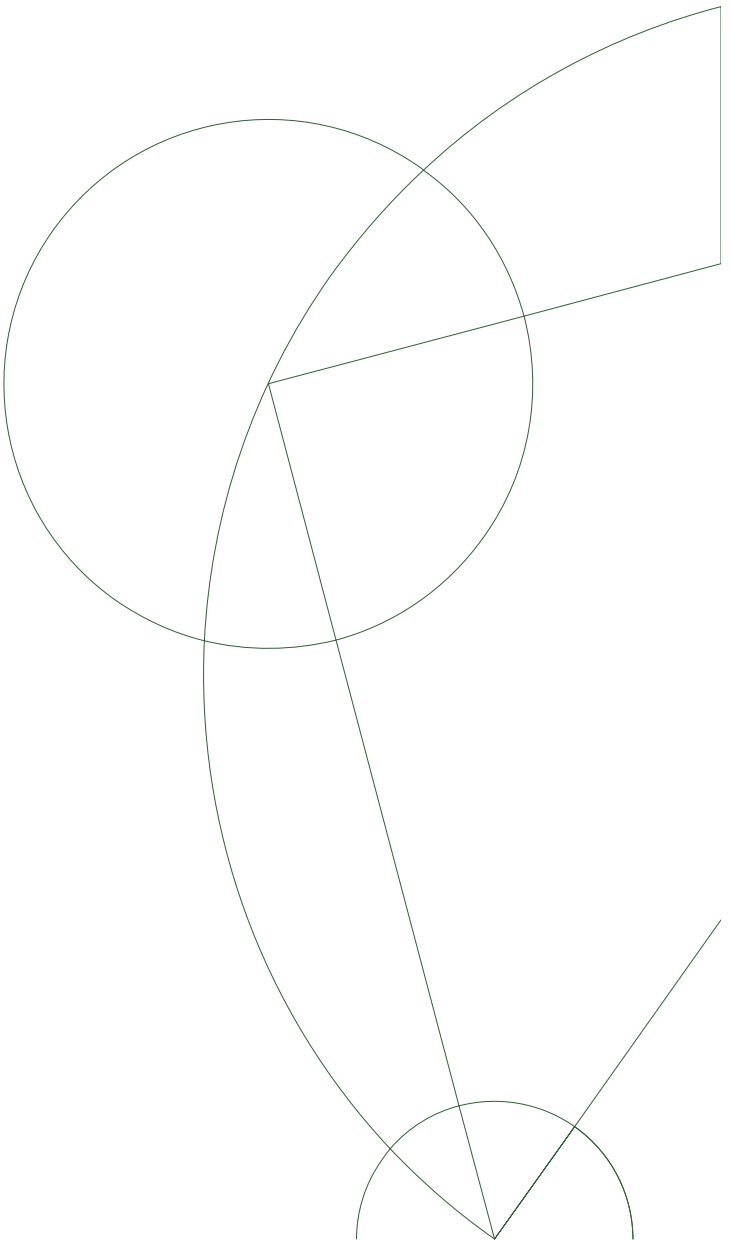
Diskret Matematik og Algoritmer

Aflevering 2i

Adam Frederik Ingwersen Linnemann,
GQR701
Hold 4

Datalogisk Institut
KÃ¿benhavns Universitet

October 3, 2016



Del 1

Antallet af inversioner kan bestemmes ved at tÃęlle det samlede antal 'ikke-sorterede' placeringer af elementerne i A. For arrray bestÃěende af fÃÿlgende $n = 6$ elementer; [2,1,8,4,3,6] er der sammenlagt 5 inversioner.

Del 2

En generel formel for at finde det maksimale antal inversioner, k, i et array bestÃěende af n vilkÃěrlige elementer kan bestemmes ved:

$$k = \frac{n(n-1)}{2}$$

Dette kan bl.a. skrives pÃě fÃÿlgende mÃěde ved iterativ pseudokode:

Algorithm 1 Maksimale antal inversioner givet n

```
1: function MaxInv(n)
2:     v = 0
3:     for i = 1 to n
4:         v = (i-1) + v
5:     return v
```

Del 3

Denne delopgave kan lÃÿses pÃě adskillige mÃěder - bÃěde rekursivt og iterativt. Her er valgt en iterativ tilgang, der bygger pÃě lineÃęr sÃÿgning. Algoritmen opererer pÃě fÃÿlgende mÃěde:

1. PositionÃľr starten af den fÃÿrste lÃÿkke (i=0) ved A's 0'te element
2. PositionÃľr herefter starten af den anden lÃÿkke foran i (j=i+1)
3. Lad j iterere fra j=1 til n over nedenstÃěende betingelse:
   (a) Hvis vÃęrdien af det i'te element er strengt stÃÿrre end det j'te, da:
   (b) DenotÃľr variablen 'inv' med sin hidtige vÃęrdi plus 1
4. Afslut ved at returnere den akkumulerede vÃęrdi for 'inv'

Algoritmen er beskrevet i et mere koncist format i pseudokoden nedenfor.

Algorithm 2 TÃęl antal inversioner i A

```
   function CountInv(A, n)
2:     inv = 0
       for i = 0 to n - 2
4:         for j = i+1 to n - 1
               if A[i] > A[j]
6:             inv = inv + 1
       return inv
```

En alternativ lÃÿsning til den 'fladpandede' lineÃęre sÃÿgning ville vÃęre, at anvende merge-sort med et if-statement og en tÃęlle-variabel i den afsluttende 'merge'-del.

Del 4

For at analysere algoritmens kÃÿretid, anvendes tilgangen beskrevet i CLRS 2.2. Her er vi umiddelbart kun interesserede i de led, der pÃĕvirker kÃÿretiden, nÃĕr $n \leftarrow \infty$, sÃĕ at sige. Af denne Ãĕrsag, betragtes udelukkende de iterative led i $CountInv(A, n)$. Leddene har en multiplikativ relation. Det sidste for-loop aftager i antallet af gentagelser som funktion af i's placering, mens det fÃÿrste for-loop gentages $n - 1$ gange.

$$rt \approx \Theta((n-1) \cdot \frac{(n-1)((n-1)+1)}{2}) =$$
$$\Theta((n-1) \cdot \frac{n^2 - n}{2}) =$$
$$\Theta(\frac{n \cdot (n-1)^2}{2})$$

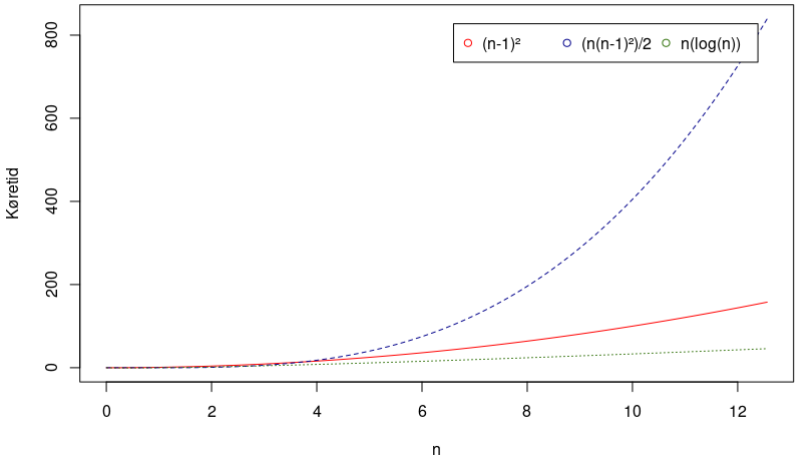


Figure 1: Algoritmens kÃÿretid sammenlignet med n^2

Ved at plotte kÃÿretiden mod f.eks. n^2, ses det, at algoritmen ved store n vil opnÃĕ eksponentielt hÃÿjere kÃÿretider i takt med at $n \leftarrow \infty$. Dette mÃĕ anses for at vÃęre en direkte effekt af det voksende antal mulige sammensÃętninger. For at konstruere plottet er der anvendt R-kode, som er vedlagt i bilag.

Bilag

```
fun1 <- function(x) x^2
fun2 <- function(x) (x*(x-1)^2)/2
fun3 <- function(x) x*log2(x)
x<-seq(0,4*pi,0.01)

data <- cbind(fun1(x), fun2(x), fun3(x))

matplot(x, data, type = 'l',
        col = c("#ff0000", "#00008b", "#31720a"),
        ylab = "KÃÿretid", xlab = "n")
legend("topright", inset = .05,
        legend = c("(n-1)Âš", "(n(n-1)Âš)/2", "n(log(n))"),
    pch = 1,
    col = c("#ff0000", "#00008b", "#31720a"), horiz = TRUE)
title("Sammenligning af kÃÿretider")
```