DMA 2021
Ugeopgave 2

Aditya Fadhillah

2. oktober 2021

OPGAVEN

Del 1
Lad $n = 7$ og $A = [5, 4, 6, 7, 2, 1, 5]$. Hvor mange inversioner er der i A?

$A[0] = 5$ har 3 inversioner, altså (5,4),(5,2), (5,1)
$A[1] = 4$ har 2 inversioner, altså (4,2),(4,1)
$A[2] = 6$ har 3 inversioner, altså (6,2),(6,1), (6,5)
$A[3] = 7$ har 3 inversioner, altså (7,2),(7,1), (7,5)
$A[4] = 2$ har 1 inversioner, altså (2,1)
$A[5] = 1$ har 0 inversioner
$A[6] = 5$ har 0 inversioner
Det vil sige at der er i alt 12 inversioner i arrayet A

Del 2
For hvert n, hvor mange inversioner kan et array af længde n maksimalt have?
Hint: Du vil måske finde det nyttigt først at kigge på konkrete små værdier for n, og dernæst forsøge at finde sammenhæng. Du kan måske også få brug for at kigge i CLRS afsnit A.1.

Man kan bruge sumformlen for at finde antallet af inversioner, i det med at man går igennem hele arrayet, og for hver index prøver man at finde antallet af mindre værdier på højre siden af arrayet. Så for at få det maksimalt antal af inversioner, skal arrayet være omvendt sorteret, så det næste index er mindre end den forrige. Og når man så lægger inversionerne sammen vil det ligne sumformlen, da det største værdi adderes med mindre og mindre værdier. Men man skal først modificerer sumformlen, før man kan anvende den til udregningen af inversioner. Sumformlen finder summen af hele værdier i arrayet, altså

$$1 + 2 + ... + n - 1 = \frac{n * (n + 1)}{2}$$

Men da man antager at arrayet er omvendt sorteret, dvs. at den største værdi som ligger længst til venstre, kan ikke være en inversion for et andet tal. Derfor ændres sumformlen til:

$$\frac{(n - 1) * ((n - 1) + 1)}{2}$$

eller

$$\frac{n * (n - 1))}{2}$$

Del 3
Lav pseudokode for en algoritme $CountInversions(A, n)$, der tæller antallet af inversioner i et array A af størrelse n. Din pseudokode skal have nummererede linjenumre.

Count Inversion

```
1 CountInversion(A, n)
2     c = 0
3     for i = 0 to n - 2
4         for j = i + 1 to n - 1
5             if A[i] > A[j]
6                 c = c + 1
7     return c
8
```

Del 4
Analyser din pseudokode fra del 3: find køretiden og angiv den med Θ-notation. (Hvis du er omhyggelig kan du finde en algoritme med køretid $\Theta(nlogn)$, men dette er ikke et krav)

Den første del kører kun en gang og tager c_1 skridt

$$c = 0 \ , \ c_1$$

i-loopet kører $n-1$

$$\text{for } i = 0 \text{ to } n-2 \text{ , } (n-1) * c_2$$

og j.loopet kører $\frac{n*(n-1))}{2}$ gange

$$\text{for } j = i+1 \text{ to } n-1 \text{ , } \frac{n*(n-1))}{2} * c_3$$

Køretiden på pseudokoden er $T(n) = c_1 + (n-1) * c_2 + \frac{n*(n-1))}{2} * c_3$, og det vil så sige at pseudokoden asymptotisk køretid er på $\Theta(n^2)$