DMA 2021
Ugeopgave 2

Aditya Fadhillah

22. september 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 til at finde antallet af inversioner på et array, Men man skal først modificerer den. Sumformlen finder summen af hele værdier i arrayet, altså

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

Men i det med at den først index ikke kan være inversion for noget skal sumformlen ændres til:

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

eller

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

for eksempel med array $A = [7, 6, 5, 4, 3, 2, 1]$ kan jeg ved hjælp af den modificeret sumformlen finde frem til at der er 21 inversioner.

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

```
CountInversion(A, n)
    c = 0
    for i = 0 to n - 2
        for j = i + 1 to n - 1
            if A[i] > A[j]
                c = c + 1
    return c
```

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)

Algoritmen har en køretid på $\Theta(n^2)$ da der er en loop for j som gøre at algoritmen først skal kører j n-gange før i adderes med 1, hvor den derefter så igen skal kører j n-gange. Det fortsætter så indtil $i = n - 1$. Men da man så kun kører i fra 0 til $n - 2$, og at j kun kører fra $i + 1$ til $n - 1$ vil køretiden være $\Theta(n^2 - \frac{n*(n-1))}{2}) - 1$. Men da jeg kun er intereseret i det størst voksende led af køretiden, kan jeg fjerne konstanten -1 og den langsomt voksende led $-\frac{n*(n-1))}{2}$. Køretiden vil derfor være $\Theta(n^2)$