DMA

Ugeopgave 2

Matti Andreas Nielsen

4. oktober 2016

Indhold

1 Opgaven

A er i denne opgave et array det indeholder n heltal A[0], . . . , A[n 1]. En inversion er et par (i, j) sådan at A[i] > A[j] og i < j. Antallet af inversioner i A[0..n 1] kan man se som et mål for hvor langt A er fra at være sorteret

2 Del 1

Lad n = 6 og A være et array der indeholder tallene (2,1,8,4,3,6). Hvor mange inversioner er der i A?

$$A = \begin{bmatrix} 2 & 1 & 8 & 4 & 3 & 6 \end{bmatrix}$$

Der er 5 inversioner i A, man tæller simpelthen bare hvor mange tal der er mindre end det givne tal fra venstre mod højre. Hvorefter man går videre til det næste tal, indtil der ikke er nogle tal tilbage.

3 Del 2

For hvert n, hvor mange inversioner kan A 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.

Hvis A er listen, i er indexet og n er mængden af elementer i vores liste, så for ethvert A[i] kan man højst have n - i inversioner. Hvilket er det samme som de resterende elementer i listen fra det nuværende index.

For at sige dette på en anden måde, kan man sige at det værste tilfælde vi kan have er hvis A er totalt usorteret. Så får man:

$$(n-1) + (n-2) + (n-3) + (n-(n-1))$$

Som er det samme som

$$0.5 * (n-1) * n$$

4 Del 3

Lav pseudokode, der tæller antallet af inversioner i A.

pseudokode:
1 ... countInversions(A)
2 ... inversionCount = 0
3 ... for i to A.Length-2
4 ... for j+i to A.length-1

```
5 ... if A[i] > A[j] then
6 ... inversionCount++
```

F:

```
open System

let countInversions (A : int array) =
let mutable inversionCount = 0
for i in 0 .. A.Length-2 do
for j in i+1 .. A.Length-1 do
if A.[i] > A.[j] then
inversionCount <- inversionCount + 1
(inversionCount)

[<EntryPoint>]
let main argv =
let A = [| 2; 1; 8; 4; 3; 6|]
let inversionCount = countInversions A
printf "Inversion count:
0 // return an integer exit code
```

5 Del 4

Analyser din pseudokode fra del 3 og find køretiden på din algoritme.

Definition af vores udtryk:

$$n = A.Length$$

t_i= mængden af gange vores ydre check bliver eksekveret

t_j = mængden af gange vores indre loop bliver eksekveret

$$c = \text{konstant}$$

CountInversions(A)	cost	times
inversionCount = 0	c1	1
for i to A.Length-2	c2	n - 1
for j+i to A.Length-1	c3	$\sum_{i=0}^{n-1} t_i$
if A[i] > A[j] then	c4	$\sum_{j=i+j}^{n-1}(t_j - 1)$
inversionCount++	c5	$\sum_{j=i+j}^{n-1}(t_j - 1)$

En manuel opsummering ville se således ud:

$$T(n) = c1 * 1 + c2 * (n-1) + c3\sum_{i=0}^{n-1} t_i + c4\sum_{i=0}^{n-1}(t_i - 1) + c5\sum_{i=0}^{n-1}(t_i - 1)$$

Men vi er mere interesseret i en generalisering af sumfølgen vi kan udtrække fra algoritmen.

Udfra vores analyse kan vi se at vi har 2 forløkker, og at vi får en talfølge som ser således ud

$$(n-1)+(n-2)+(n-3)+(n-(n-1))$$

det kan vi se fordi det indre loop plusser j med i.

Vores talfølge

$$(n-1)+(n-2)+(n-3)+(n-(n-1))$$

er det samme som følgen

$$\sum_{k=1}^{n} k$$

fordi vores talrække stiger med 1 hver gang.

Dette er det samme som

$$(n^2+n)/2$$

og kan man omskrive til

$$0.5*n^2+0.5*n$$

og i bigO fjerner vi konstanter så det giver os

$$n^2$$