DM02 opgaver ugeseddel 2

af Fiona Nielsen

16. september 2003

Øvelsesopgaver 9/9, 10/9 og 11/9

1. Vis, at $1^3 + 3^3 + 5^3 + \ldots + (2n-1)^3 = 2n^4 - n^2$.

   Omskriver til summationsformel:

   $$\sum_{i=1}^{n}(2i-1)^3 = 2n^4 - n^2$$

   Bevis ved induktion over n

   Basis $n = 1$
   Venstresiden

   $$(2\dot{1} - 1)^3 = 1^3 = 1$$

   Højresiden

   $$2\dot{1}^4 - 1^2 = 2 - 1 = 1$$

   Vi ser at hypotesen gælder for vores basistilfælde.

   Induktion $k \geq 1$, Antag at hypotesen holder for alle heltal op til k:

   $$\sum_{i=1}^{k}(2i-1)^3 = 2k^4 - k^2$$

   Vis for k+1:

   $$\sum_{i=1}^{k+1}(2i-1)^3 = 2(k+1)^4 - (k+1)^2$$

1. fortsat fra forrige side...

   Venstresiden

$$\begin{aligned}\sum_{i=1}^{k+1}(2i-1)^3 &= \sum_{i=1}^{k}(2i-1)^3 + (2(k+1)-1)3 \\ &= 2k^4 - k^2 + (2(k+1)-1)^3 \\ &= 2k^4 - k^2 + 8k^3 + 12k^2 + 6k + 1 \\ &= 2k^4 + 8k^3 + 11k^2 + 6k + 1\end{aligned}$$

   Højresiden

$$2(k+1)^4 - (k+1)^2 = 2k^4 + 8k^3 + 11k^2 + 6k + 1$$

   Vi ser at venstresiden er lig højresiden og vi kan konkludere at vores induktionsbevis holder for alle n.

2. Find en lukket formel for $\sum_{i=a}^{n} i$, hvor a er et heltal mellem 1 og n.

   Bevis, at din formel er korrekt.

   Der er flere måder at løse denne opgave på. Een måde er at sjusse sig frem til en formel, og derefter bevise ved induktion at den gælder for alle n.
   En anden måde er at genkende summationen som værende meget lig summationen fra ugeseddel 1, nemlig $\sum_{i=1}^{n} i$. Herfra kan vi udlede en formel for $\sum_{i=a}^{n} i$ ved at opdele summationen:

$$\begin{aligned}\sum_{i=a}^{n} i &= \sum_{i=1}^{n} i - \sum_{i=1}^{a-1} i \\ \text{(fra sidste uge)} &= \frac{n(n+1)}{2} - \frac{(a-1)(a-1+1)}{2} \\ &= \frac{n(n+1) - (a-1)a}{2} \\ &= \frac{n^2 + n - a^2 + a}{2}\end{aligned}$$

   Hermed har vi en lukket formel for summationen. Hvis man bruger denne fremgangsmåde er det ikke nødvendig efterfølgende at bevise formlen, da den er udledt af en allerede bevist formel uden at foretage yderligere antagelser.

1. Et binært træ er et træ, hvor hver knude har 0, 1 eller 2 børn. Et blad er en knude, der har 0 børn. Bevis ved strukturel induktion, at et binært træ med n knuder har højst $\lceil \frac{n}{2} \rceil$ blade.

   Strukturel induktion kan med fordel benyttes til at bevise sætninger for strukturer som har en rekursiv definition, som f.eks. definitionen for et binært træ i denne opgavetekst.

   Vi ønsker at vise påstanden S
   S : "Et binært træ med n knuder har højst $\lceil \frac{n}{2} \rceil$ blade"

   Bevis:
   Basis $n = 1$
   Et træ bestående af én knude har netop ét blad.

$$\lceil \frac{n}{2} \rceil = \lceil \frac{1}{2} \rceil = 1$$

   Induktion $k \geq 1$
   Antag S gælder for et træ med $k \geq 1$ knuder.
   Vis S for et træ med $k + 1$ knuder.

   Lad T være et træ bygget ved den rekursive definition, med to undertræer T_1,T_2 som opfylder S.

   antal knuder i $T_1 = n_1$, antal blade i T_1: $b_1 \leq \lceil \frac{n_1}{2} \rceil$
   antal knuder i $T_2 = n_2$, antal blade i T_2: $b_2 \leq \lceil \frac{n_2}{2} \rceil$
   $\Updownarrow$
   antal knuder i T: $n = n_1 + n_2 + 1 = k + 1$

   Beregner antal blade b i T

$$\begin{aligned} b &= b_1 + b_2 \\ &\leq \lceil \frac{n_1}{2} \rceil + \lceil \frac{n_2}{2} \rceil \\ &\leq \lceil \frac{n_1 + n_2 + 1}{2} \rceil \\ &= \lceil \frac{k+1}{2} \rceil \end{aligned}$$

   $\Updownarrow$

$$b \leq \lceil \frac{k+1}{2} \rceil$$

   Altså kan vi konkludere at S gælder for alle binære træer.

1. Eksamen januar 96 opg. 2.

   Opgaven sammenligner Udvalgssortering (Selection sort) med Indsættelsessortering (Insertionsort)
   (begge kan slås op i næsten hvilkensomhelst algoritmebog, hvis du gerne vil se flere varianter eller analyser af disse to algoritmer)

   (a) Angiv kompleksiteten ved sorteret input. $U : O(n^2)$ og $I : O(n)$

   (b) Kompleksiteten for lister i c-uorden, $c > 0$ $U : O(n^2)$ og $I : O(cn)$

   (c) Ændring så algoritmerne er $O(cn)$ når listen er i c-uorden
   For U: Man kan nøjes med at kigge c pladser frem hvis arrayet er i c-uorden
   Dvs. den indre løkke kan ændres til :

```
if (i+c) < n
    limit = i + c
else
    limit = n
for j = i+1 to limit do
```

   For I: ingen ændring nødvendig

2. Cormen et al. 2.2-4. Hvordan kan man ændre næsten en hvilkensomhelst algoritme til at have en god best-case køretid?

   For alle algoritmer der har en veldefineret best-case eller et inputtilfælde hvor output er meget nemt at beregne, kan man tilføje - allerførst i algoritmen - et tjek om input er denne best-case og i givet tilfælde løse problemet og returnere.

   For udvalgssortering kan man f.eks. inden den første løkke, lave en enkelt kørsel gennem arrayet for at tjekke om det allerede er sorteret. I givet fald skal algoritmen bare returnere, uden at foretage sig andet.
   På denne måde kan vi give udvalgssortering en best-case på bare $\Theta(n)$!

   Men *husk* at det ikke hjælper en døjt på algoritmens generelle kompleksitet og køretid at give den en god bestcase. I tilfældet med en sorteringsalgoritme, vil det kun være i én ud af $n!$ (n fakultet) tilfælde at køretiden forbedres, hvis vi antager at alle inputkombinationer er lige sandsynlige og dermed vil det være bogstaveligt talt spild af tid at prøve at forbedre dette ene tilfælde.
   Til gengæld: hvis vi har en algoritme som bruges et sted hvor en særlig speciel input-kombination hyppigt opstår, kan det måske betale sig at tjekke om algoritmen kan forbedres netop på det hyppige input, og opnå en lille forbedring i performance.

1. Cormen et al. 2-2. Korrekthed af Bubblesort

   (a) Hvad skal vises førend vi har bevist algoritmens korrekthed?
   1. Terminering
   2. Sorteret output
   3. Outputarray A' er en mulig permutering af elementerne fra input-arrayet A.

   (b) Løkkeinvariant I, Linie 2-4
   Invariant: $S_j : A[j] \leq A[j+1 \ldots n]$
   Initialization: $j = n$, $S_n : A[n] \leq A[n]$ Ok.
   Maintenance: for $j \leq n$ (Husk at løkken tæller *nedad*)
   Induktionsantagelse: $S_j : A[j] \leq A[j+1 \ldots]$
   undersøger to tilfælde:

   1. if-betingelsen holder, dvs. $A[j] \leq A[j-1]$
   $\Rightarrow A[j-1] < A[j]$ (fordi elementerne ombyttes)
   $\Rightarrow A[j-1] \leq A[j \ldots] = S_{j-1}$ Ok.

   2. if-betingelsen fejler, dvs. $A[j] > A[j-1]$
   $\Rightarrow A[j-1] \leq A[j \ldots] = S_{j-1}$ Ok.

   Termination: $j = i$

   $\Rightarrow A[i] \leq A[i+1 \ldots n]$

   (c) Løkkeinvariant II, Linie 1-4
   Invariant: $I_i : A[1] \leq A[2] \leq \ldots \leq A[i-1]$, dvs. $A[1..i-1]$ indeholder de i-1 mindste elementer fra A i sorteret (stigende) rækkefølge.
   Initialization: $i = 1$, $I_1 : A[1..0] = \emptyset$, et tomt array er sorteret. Ok.
   Maintenance: for $i \geq 1$
   Induktionsantagelse: $I_i : A[1] \leq A[2] \leq \ldots \leq A[i-1]$
   I hver eneste iteration går vi direkte til den indre løkke, I:

   $\Rightarrow A[i] \leq A[i+1..n] \Rightarrow A[1] \leq A[2] \leq \ldots \leq A[i]$
   $\Leftrightarrow I_{i+1}$ Ok.

   Termination: $i = n+1$

   $\Rightarrow I_{n+1} \Leftrightarrow A[1] \leq A[2] \leq \ldots \leq A[n+1-1]$
   $\Leftrightarrow A[1] \leq A[2] \leq \ldots \leq A[n]$

1. fortsat fra forrige side...

   (a) Hvad er worst-case køretid for Bubblesort? Sammenlign med Insertion-sort.

   Bubblesort gennemgår begge løkker uanset input. Når den karakteristiske operation vælges til at være sammenligningen i 3. linie får vi følgende.

   Den ydre løkke gentages n gange.
   Den indre løkke (og dermed sammenligningen):

$$\begin{aligned}(n-1)+(n-2)+\ldots+2+1 &= \sum_{i=1}^{n-1} i \\ &= \frac{n(n+1)}{2} - n \\ &= \frac{n^2}{2} - \frac{n}{2} \in \Theta(n^2)\end{aligned}$$

   Altså, $W(n) = A(n) = B(n) = \Theta(n^2)$.

   De tilsvarende køretider for insertion sort kan du slå op i bogen, eller i dine noter fra forelæsningen i uge 37.

Bemærkninger

Strukturen for bevis af invarianter gennemgås s. 30-31 i Cormen.

Et lille tip til den skriftlige eksamen:
Når du svarer på opgaver der indeholder et matematisk bevis, som f.eks. et induktionsbevis, så skriv en sætning eller to ved siden af, hvor du forklarer med ord, hvorfor beviset gælder. På den måde viser du at du har forståelse for algoritmen, plus at du sikrer dig mod en evt skrive- eller tastefejl i det matematiske bevis ødelægger helhedsindtrykket af besvarelsen.