DD1337 läxa

Arvid Kristoffersson

November 2024

1 Induktion

1.1 a)

$$\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$$

Vi visar att påståendet stämmer med hjälp av induktion:
Basfall: $n = 1$
$VL = \sum_{i=1}^{1} i^2 = 1^2 = 1 \quad HL = \frac{1(1+1)(2\cdot 1+1)}{6} = \frac{6}{6} = 1$
$VL = HL$ Basfallet stämmer alltså.
Anta att påståendet stämmer för alla k mindre eller lika med något p där $k, p \in \mathbf{Z}_+$.
Vi visar nu för $p + 1$:

$$VL = \sum_{i=1}^{p+1} i^2 = \sum_{i=1}^{p} i^2 + (p+1)^2$$

Vilket enligt antagande ger:

$$VL = \sum_{i=1}^{p} i^2 + (p+1)^2 = \frac{p(p+1)(2p+1)}{6} + (p+1)^2 = \frac{2p^3 + p^2 + 2p^2 + p}{6} + (p+1)^2 = \frac{2p^3 + 3p^2 + p}{6} + p^2 + 2p + 1$$

$$= \frac{2p^3 + 3p^2 + p}{6} + \frac{6p^2 + 12p + 6}{6} = \frac{2p^3 + 9p^2 + 13p + 6}{6}$$

$$HL = \frac{(p+1)(p+1+1)(2(p+1)+1)}{6} = \frac{(p+1)(p+2)(2p+3)}{6} = \frac{2p^3 + 9p^2 + 13p + 6}{6}$$

$$VL = HL$$

Att påståendet stämmer för alla $n \in \mathbf{N}$ följes av induktionsprincipen.
V.S.B

1.2 b)

Vi ska visa att följande påstående gäller.

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

Vi visar detta med hjälp av induktion:
Basfall: $n = 1$
$VL = \sum_{i=1}^{1}(2i-1) = 2 \cdot 1 - 1 = 1$
$HL = 1^2 = 1$
$VL = HL$ Påståendet stämmer alltså för basfallet
Antа att påståendet stämmer för alla k mindre eller lika med något p, för $p, k \in \mathbf{Z}_+$.
Vi visar för $p + 1$:

$$VL = \sum_{i=1}^{p+1}(2i-1) = \sum_{i=1}^{p}(2i-1) + 2(p+1) - 1$$

Vilket enligt antagande blir:

$$= p^2 + 2(p+1) - 1 = p^2 + 2p + 1$$

$$HL = (p+1)^2 = p^2 + 2p + 1$$

Alltså är:

$$VL = HL$$

Att påståendet stämmer för alla $p \in \mathbf{Z}_+$ följes av induktionsprincipen.
V.S.B

2 Iterativ korrekthet

Vi bevisar dens korrekthet med induktion.
Vi vill visa att expIterative(double x, int n) returnerar x^n för alla $n \in \mathbf{N}$ och $x \in \mathbf{Q}$ eftersom x är en double.
Basfall: $n = 0$
expIterative(x, 1) ger att for-loopen aldrig körs och res initiala värde returneras, alltså 1.0. Vilket stämmer, eftersom $x^0 = 1$.
Anta nu att påståendet stämmer för varje k mindre eller lika med något p, där $k, p \in \mathbf{N}$. Vi visar nu för $p + 1$.
Låt först fr(0, u) $\longrightarrow$ f(x) beteckna en for loop som går från 0 till u och kör någon funktion f(x) där x är vilket steg den är på (0...u).

$$VL = fr(0, p+1) \longrightarrow res = res \cdot x = (fr(0, p) \longrightarrow res = res \cdot x) \cdot x$$

Vilket enligt antagande blir:

$$= (x^p) \cdot x = x^{p+1}$$
$$HL = x^{p+1}$$

$$VL = HL$$

Att påståendet stämmer för alla $n \in \mathbf{N}$ följes av induktionsprincipen.

Tidskomplexiteten är $O(N)$ (alltså linjär) för att den gör en $O(1)$ operation N gånger.

3 Rekursiv korrekthet

3.1 korrekthet

Vi visar även denna med induktion. Vi vill visa att expRecursive(double x, int n) returnerar x^n för alla $n \in \mathbf{N}$ och $x \in \mathbf{Q}$.
Basfall: $n = 0, n = 1, n = 2, n = 3, n = 4$ När n antar något av dessa värden returnerar funktionen expIterative(x, n) vilket vi redan har bevisat stämmer för alla $n \in \mathbf{N}$. Alltså stämmer det även för dessa basfall.
Antag att påståendet att expRecursive(x, k) returnerar x^k för alla k mindre eller lika med något $4 \leq p \in N$. Vi visar nu för $p + 1$:

$$VL = expRecursive(x, p+1) = expRecursive(x, \lfloor (p+2)/2 \rfloor) \cdot expRecursive(x, \lfloor (p+1)/2 \rfloor)$$

Om p är jämnt får vi:

$$VL = expRecursive(x, p/2 + 1) \cdot expRecursive(x, p/2)$$

Detta blir enligt antagande (notera att $p/2 + 1$ och $p/2$ är mindre än $p + 1$ för alla $p \in \mathbf{N}$:

$$expRecursive(x, p/2+1) \cdot expRecursive(x, p/2) = x^{p/2+1} \cdot x^{p/2} = x^{p/2+1+p/2} = x^{p+1}$$

och om p är udda får vi:

$$VL = expRecursive(x, (p+1)/2) \cdot expRecursive(x, (p+1)/2)$$

Vilket enligt antagande blir:

$$expRecursive(x, (p+1)/2) \cdot expRecursive(x, (p+1)/2) = x^{(p+1)/2} \cdot x^{(p+1)/2} = x^{(p+1)/2+(p+1)/2} = x^{p+1}$$

$$HL = x^{p+1}$$

$$VL = HL$$

stämmer för både jämna och udda p.
Att påståendet stämmer för alla $n \in \mathbf{N}$ följes av induktionsprincipen. **V.S.B**

3.2 tidskomplexitet

Mästarsatstermerna a, b och f(n) är i detta fall $a = 2, b = 2, f(n) = O(1)$. Då får vi fallet i mästarsatsen där $T(n) = O(n^{log_b(a)}) = O(n^{log_2(2)}) = O(n)$
Tidskomplexiteten är alltså linjär ($O(n)$).

Man kan också bevisa det mer intuitivt. Fallet när $n <= 4$ antar vi vara konstant eftersom det maximalt blir 4 operationer. Tänk dig att funktionen bildar ett träd där varje nod har två barn vars värden är hälften av nodens, ända tills noden uppfyller $n <= 4$. Notera att ett värde n kan halveras maximalt logaritmiskt antal gånger bas 2. Detta innebär att trädet får en höjd som är $log_2(n)$. Vi kan se det som att trädet har logaritmiskt antal lager. Notera att antalet noder i varje lager dubbleras varje gång också. Eftersom varje tidigare nod bildar har två barn. Alltså blir antalet noder i lager p (räknat från toppen till botten från $0 \longrightarrow log_2(n)$) 2^p. Eftersom varje nod genomför konstant antal operationer ($O(1)$) blir komplexiteten samma som antalet noder. Tänk dig nu att vi summerar alla 2^p: alltså $\sum_{p=0}^{log_2(n)} 2^p$. Man kan tänka på detta som summan av alla tal $\leq p$ som har endast en bit i sin binära representation. Detta binära tal kommer att vara på formen 11...1 med $log_2(n)$ antal bitar. Värdet av detta binära tal blir därför $2 \cdot n - 1$ vilket är hur många noder det finns ungefär. Tidskomplexiteten blir alltså $O(n)$.