proginda komplexitet

avj

November 2024

1 Bevis med induktion

1.1 Bevisa med induktion: $\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$

Bevis:
Basfall: $n = 1 \implies V.L = \sum_{i=1}^{1} i^2, H.L = \frac{1(1+1)(2\cdot1+1)}{6}$
$V.L = 1^2 = 1$
$H.L = \frac{1\cdot2\cdot3}{6} = \frac{6}{6} = 1 \implies V.L = H.L$
Basfallet OK!
Induktionsantagande(I.A): Vi antar att satsen gäller för P(n) och vill visa att det implicerar att P(n+1) gäller.

$H.L_{n+1} = \frac{(n+1)((n+1)+1)(2(n+1)+1)}{6} = \frac{(n+1)(n+2)(2n+3)}{6} = \frac{(n+1)(2n^2+3n+4n+6)}{6} = \frac{(n+1)(2n^2+7n+6)}{6}$

$V.L_{n+1} = \sum_{i=1}^{n+1} i^2 = \sum_{i=1}^{n} i^2 + (n+1)^2 = [I.A] = \frac{n(n+1)(2n+1)}{6} + (n+1)^2 = \frac{n(n+1)(2n+1)}{6} + \frac{6(n+1)^2}{6} = \frac{n(n+1)(2n+1)+6(n+1)^2}{6} = \frac{(n+1)(n(2n+1)+6(n+1))}{6} = \frac{(n+1)(2n^2+n+6n+6)}{6} = \frac{(n+1)(2n^2+7n+6)}{6} \implies V.L_{n+1} = H.L_{n+1} \implies$ Enligt induktionsaxiomet är satsen nu bevisad!

1.2 Bevisa med induktion: $\sum_{j=1}^{n}(2j-1) = n^2$

Bevis:
Basfall: n = 1 $\implies V.L = \sum_{j=1}^{1}(2j-1), H.L = 1^2 = 1$
$V.L = (2\cdot1-1) = 1 \implies H.L = V.L$ Basfall OK!
Induktionsantagande [I.A]: Vi anatar att satsen gäller för P(n) och vill visa att det implicerar att satsen även gäller för P(n+1)

$V.L_{n+1} = \sum_{j=1}^{n+1}(2j-1), H.L_{n+1} = (n+1)^2$

$V.L_{n+1} = \sum_{j=1}^{n+1}(2j-1) = \sum_{j=1}^{n}(2j-1) + (2(n+1)-1) = [I.A] = n^2 + 2n+2-1 = n^2+2n+1 = (n+1)^2 = H.L_{n+1} \implies$ satsen är nu bevisad enligt induktionsaxiomet.

2 Iterativ korrekthet

Kod:

```
double expIterative(double x, int n) {
    double res = 1.0;

    for (int i = 0; i < n; i++) {
        res *= x;
    }
    return res;
}
```

Tidskomplexiteten för denna funktion är triviell att hitta. Det finns en for loop som itererar från 0 till n, därför är tidskomplexiteten $O(n)$.

Kod med invarianter:

```
double expIterative(double x, int n) {
    double res = 1.0;

    for (int i = 0; i < n; i++) {
        // Invariant: res = 1 * x * x * ... * x = x ^ (i)
        res *= x;
    }
    return res;
}
```

Koden är korrekt eftersom med invarianten ser vi att när i = 0 kommer res att vara lika med 1 vilket den explicit satts till ovanför. Vi antar sen att res kommer vara x^i i början av varje iteration. Vid sista iterationen får vi att $res = x^i = x^{n-1}$ så när vi multiplicerar det med x får vi att $res = x^n$ och vi har nu bevisat korrektheten.

3 Rekursiv korrekthet

Kod:

```
double expRecursive(double x, int n) {
    if (n <= 4) {
        return expIterative(x, n);
    }

    return expRecursive(x, n/2) *
           expRecursive(x, (n + 1)/2);
}
```

T(n) = T(n/2) + O(1) Vi kan använda mästarsatsen för att hitta tidskomplexiteten för den rekursiva funktionen. Vi får a = 2 eftersom vi har två stycken delproblem och får även att b = 2. Vi får även att d = 0 vilket innebär $a > b^d$ vilket gör att tidskomplexiteten blir $O(n^{log_b a}) = O(n^{log_2 2}) = O(n^1) = O(n)$ delvis linjär tid

För basfallen $n =< 4$ vet vi att den kommer ge korrekt resultat eftersom den kallar expIterative som vi visade är korrekt.

Vi antar att funktionen är korrekt för alla P(i), $i < k$ och vill nu visa att P(k) gäller. Eftersom $k > 4$ kommer raden

```
expRecursive(x, n/2) * expRecursive(x, (n + 1)/2);
```

att köras. Eftersom både $n/2$ och $(n+1)/2$ är mindre än k vet vi enligt induktionsantagandet att dessa är korrekta vilket gör att

```
expRecursive(x, n/2) * expRecursive(x, (n + 1)/2);
```

blir $x^{n/2} * x^{(n+1)/2} = x^{(2n+1)/2} = x^{n+(1/2)}$. Eftersom argumentet n, är en int som argument kommer det att avrundas neråt till 0 och det enda vi kommer få kvar är x^n vilket innebär att den rekursiva varianten är korrekt.