Huvudsats för analys av divide and conquer-algoritmer

Detta papper presenterar en variant av den huvudsats för lösning av de differensekvationer som ofta erhålles vid analys av divide and conquer- algoritmer som boken beskriver i kapitel 4.3 och 4.4. Satsen är mindre generell än den som finns i boken men är lite lättare att förstå och bevisa och den illustrerar hur samspelet mellan antalet delproblem och deras storlek påverkar exekveringstiden.

Divide and conquer-algoritmer

Antag att vi har ett problem av storlek n och att vi löser det genom att dela upp det i a stycken delproblem vardera av storlek n/b. Antag också att arbetet att skapa delproblemen samt att kombinera lösningarna av dessa till en lösning av ursprungsproblemet är $f(n)$. Då kan tiden $T(n)$ för att lösa problemet skrivas med följande differensekvation:

$$T(n) = aT\left(\frac{n}{b}\right) + f(n) \tag{1}$$

där $a > 0$ och $b > 1$ är heltal.

Exempel 1
Binär sökning i en sorterad array. Där är $a = 1$, $b = 2$ och $f(n) = c$

Exempel 2
Samsortering (mergesort). Där är $a = 2$, $b = 2$ och $f(n) = cn$

Vanligt förekommande funktioner $f(n)$ är polynom som i exemplen ovan. Vi väljer därför att formulera och bevisa satsen för $f(n) = cn$.

Påstående
Om

$$\begin{aligned} T(1) &= c \\ T(n) &= aT\left(\frac{n}{b}\right) + cn \end{aligned} \tag{2}$$

där $a > 0$, $b > 1$ och $c \geq 0$ är heltal så gäller att $T(n)$ är

$$O(n) \quad \text{om} \quad a < b \tag{3}$$

$$\Theta(n \log n) \quad \text{om} \quad a = b \tag{4}$$

$$\Theta(n^{\log_b a}) \quad \text{om} \quad a > b \tag{5}$$

Bevis
Vi begränsar oss till fallet att n är en jämn b-potens dvs att $n = b^k, k = \log_b n$.

Genom att använda formel 2 upprepade gånger och utnyttja att argumentet alltid går att dela med b så erhålles:

$$\begin{aligned}
T\left(n\right) &= aT\left(\frac{n}{b}\right) + cn = a\left[aT\left(\frac{n}{b^2}\right) + c\frac{n}{b}\right] + cn \\
&= a^2T\left(\frac{n}{b^2}\right) + cn\frac{a}{b} + cn \\
&= a^3T\left(\frac{n}{b^3}\right) + cn\left(\frac{a}{b}\right)^2 + cn\frac{a}{b} + cn \\
&= \cdots \\
&= a^kT\left(\frac{n}{b^k}\right) + cn\sum_{i=0}^{k-1}\left(\frac{a}{b}\right)^i \qquad (6) \\
&= a^kT\left(1\right) + cn\sum_{i=0}^{k-1}\left(\frac{a}{b}\right)^i \\
&= a^kc + cn\sum_{i=0}^{k-1}\left(\frac{a}{b}\right)^i
\end{aligned}$$

Med hjälp av sambandet $x^{\log y} = y^{\log x}$ blir den första termen

$$a^kc = a^{\log_b n}c = cn^{\log_b a} \tag{7}$$

För att beräkna summationen i den andra termen är det bra att skilja på tre fall:

Fall 1: $a < b$

$$\sum_{i=0}^{k-1}\left(\frac{a}{b}\right)^i < \sum_{i=0}^{\infty}\left(\frac{a}{b}\right)^i = \frac{1}{1-a/b} \tag{8}$$

vilket ger

$$T\left(n\right) < cn^{\log_b a} + \frac{c}{1-a/b}n < cn + \frac{c}{1-a/b}n \tag{9}$$

som är $O(n)$

Fall 2: $a = b$

$$\sum_{i=0}^{k-1}\left(\frac{a}{b}\right)^i = \sum_{i=0}^{k-1}1^i = k = \log_b n \tag{10}$$

vilket ger

$$T\left(n\right) = cn^{\log_b b} + cn\log_b n = cn + cn\log_b n \tag{11}$$

som är $\Theta\left(n\log n\right)$

Fall 3: $a > b$

$$\sum_{i=0}^{k-1} \left(\frac{a}{b}\right)^i = \frac{(a/b)^k - 1}{a/b - 1} \tag{12}$$

$$(a/b)^k = (a/b)^{\log_b n} = n^{\log_b (a/b)} = n^{\log_b a - \log_b b} = n^{\log_b a - 1} \tag{13}$$

Således

$$\begin{aligned} T(n) &= cn^{\log_b a} + cn\frac{(a/b)^k - 1}{a/b - 1} = cn^{\log_b a} + cn\frac{n^{\log_b a - 1} - 1}{a/b - 1} \\ &= cn^{\log_b a} + \frac{cn^{\log_b a} - cn}{a/b - 1} \end{aligned} \tag{14}$$

som är $\Theta(n^{\log_b a})$ eftersom $\log_b a > 1$

Därmed är alla tre fallen bevisade.

Övning 1
Bevisa att om

$$\begin{aligned} T(1) &= c \\ T(n) &= aT\left(\frac{n}{b}\right) + c \end{aligned} \tag{15}$$

där $a > 0$, $b > 1$ och $c \geq 0$ är heltal så gäller att $T(n)$ är

$$\Theta(\log n) \quad \text{om} \quad a = 1 \tag{16}$$

$$\Theta(n^{\log_b a}) \quad \text{om} \quad a > 1 \tag{17}$$

Övning 2
Produkten av två stycken 2×2-matriser kan definieras som

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \times \begin{pmatrix} A_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} = \begin{pmatrix} A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{12} + A_{12}B_{22} \\ A_{21}B_{11} + A_{22}B_{21} & A_{21}B_{12} + A_{22}B_{22} \end{pmatrix} \tag{18}$$

Denna formel kan användas för att, på divide and conquer-sätt definiera multiplikation av godtyckliga matriser. Om vi för enkelhets skull antar att n är "tillräckligt" delbart med 2 så kan multiplikation av $n \times n$-matrisern A och B definieras enligt ovan där A_{ij} och B_{ij} är kvadratiska delmatriser av dimension $(n/2) \times (n/2)$. En multiplikation av två stycken $n \times n$-matriser kan alltså utföras med hjälp av åtta multiplikationer och fyra additioner av $(n/2) \times (n/2)$-matriser.

Hur många skalära aritmetiska operationer (additioner och multiplikationer) av tal använder ovanstående algoritm?

Övning 3
Enligt V. Strassen kan matrisprodukten ovan också definieras genom att man beräknar sju nya $(n/2) \times (n/2)$-matriser:

$$\begin{aligned}
M_1 &= (A_{12} - A_{22})(B_{21} + B_{22}) \\
M_2 &= (A_{11} + A_{22})(B_{11} + B_{22}) \\
M_3 &= (A_{11} - A_{21})(B_{11} + B_{12}) \\
M_4 &= (A_{11} + A_{12})B_{22} \\
M_5 &= A_{11}(B_{12} - B_{22}) \\
M_6 &= A_{22}(B_{21} - B_{11}) \\
M_7 &= (A_{21} + A_{22})B_{11}
\end{aligned} \tag{19}$$

varefter de fyra delarna av produktmatrisen erhålles genom

$$\begin{aligned}
C_{11} &= M_1 + M_2 - M_4 + M_6 \\
C_{12} &= M_4 + M_5 \\
C_{21} &= M_6 + M_7 \\
C_{22} &= M_2 - M_3 + M_5 - M_7
\end{aligned} \tag{20}$$

(Kan ganska enkelt verifieras för 2×2-matriser) Är denna metod snabbare eller långsammare än metoden i övning 2?