Teorema Master

Alessandro Straziota

Master Theorem

Quando si utilizza la tecnica algoritmica *Divide et Impera* si divide il problema in sottoproblemi distinti, si trova *ricorsivamente* la soluzione di tutti i sottoproblemi e si "fondono" le diverse sottosoluzioni per trovare quella globale.
Ricorsivamente a loro volta i sottoproblemi sono risolti alla stessa maniera (fase di *Divide*), finché non si arriva a un *passo base*, il quale viene risolto in tempo costante, dando il via alla "risalita" durante la quale si fondono tutte le soluzioni locali (fase di *Impera*).

Possiamo quindi osservare che il consto di un algoritmo che utilizza questa tecnica può essere espresso con la seguente *equazione di ricorrenza*

$$T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n)$$

dove a è il numero di sottoproblemi in cui l'istanza viene suddivisa, b è la grandezza del sottoproblema e $f(n)$ è il costo che ci vuole per l'*Impera*.

È facile notare che il costo dell'impera deve essere *polinomiale* in n, in quanto sarebbe davvero troppo costoso un algoritmo che fonde ogni sottoistanza in tempo esponenziale.
Perciò poniamo

$$f(n) \in O(n^d)$$

Così facendo possiamo esprimere l'equazione di ricorrenza dell'algoritmo divide et impera in questione come

$$T(n) \leq a \cdot T\left(\frac{n}{b}\right) + c \cdot n^d$$

per qualche costante $c > 0$.

Proviamo quindi ora a "srotolare" l'equazione finché non riconosciamo un pattern che ci permetta di riscriverla sotto forma di una equazione ben definita, priva di ricorsioni.

$$
\begin{aligned}
T(n) &\leq a \cdot T\left(\frac{n}{b}\right) + c \cdot n^d \\
&= a\left[aT\left(\frac{n}{b^2}\right) + c\left(\frac{n}{b}\right)^d\right] + cn^d \\
&= a^2T\left(\frac{n}{b^2}\right) + cn^d\left(\frac{a}{b^d}\right) + cn^d \\
&= a^2\left[aT\left(\frac{n}{b^3}\right) + c\left(\frac{n}{b^2}\right)^d\right] + cn^d\left(\frac{a}{b^d}\right) + cn^d \\
&= a^3T\left(\frac{n}{b^3}\right) + cn^d\left(\frac{a}{b^d}\right)^2 + cn^d\left(\frac{a}{b^d}\right) + cn^d \\
&= a^3T\left(\frac{n}{b^3}\right) + cn^d\left[\left(\frac{a}{b^d}\right)^2 + \left(\frac{a}{b^d}\right) + 1\right] \qquad (1) \\
&\vdots \\
&= a^k \cdot T\left(\frac{n}{b^k}\right) + cn^d \cdot \sum_{i=0}^{k-1}\left(\frac{a}{b^d}\right)^i \\
&\vdots \\
&= a^{\log_b n} \cdot \cancelto{\Theta(1)}{T\left(\frac{n}{b^{\log_b n}}\right)} + cn^d \cdot \sum_{i=0}^{\log_b n-1}\left(\frac{a}{b^d}\right)^i
\end{aligned}
$$

A questo punto è facile poter calcolare con precisione il costo dell'algoritmo divide et impera perché nell'equazione abbiamo rimosso le ricorsioni.

Come prima cosa possiamo riscrivere $a^{\log_b n}$ come

$$a^{\log_b n} = \left(b^{\log_b a}\right)^{\log_b n} = \left(b^{\log_b n}\right)^{\log_b a} = n^{\log_b a}$$

Ora non resta che svolgere la serie, che osservandola si può notare che è una *serie geometrica*. Ponendo $p = \dfrac{a}{b^d}$ possiamo risolverla in tre modi differenti a seconda se $p < 1$, $p = 1$ o $p > 1$.

Caso 1: se $p < 1$ allora è vero che $a < b^d$ e anche che $d > \log_b a$, quindi possiamo andare a fare le seguenti sostituzioni

$$\begin{aligned} T(n) &\leq n^{\log_b a} + cn^d \cdot \sum_{i=0}^{\log_b n - 1} p^i \\ &= n^{\log_b a} + cn^d \cdot \frac{1 - p^{\log_b n}}{1 - p} \\ &= n^{\log_b a} + cn^d \cdot \Theta(1) \\ &= O(n^d) \end{aligned} \tag{2}$$

Caso 2: se $p = 1$ allora è vero che $a = b^d$ e anche che $d = \log_b a$, quindi possiamo andare a fare le seguenti sostituzioni

$$\begin{aligned} T(n) &\leq n^{\log_b a} + cn^d \cdot \sum_{i=0}^{\log_b n - 1} \cancelto{1}{\left(\frac{a}{b^d}\right)^i} \\ &= n^{\log_b a} + cn^{\cancelto{\log_b a}{d}} \cdot \log_b n \\ &= O(n^{\log_b a} \log n) \end{aligned} \tag{3}$$

Caso 3: se $p > 1$ allora è vero che $a > b^d$ e anche che $d < \log_b a$, quindi possiamo andare a fare le seguenti sostituzioni

$$\begin{aligned}
T(n) &\leq n^{\log_b a} + cn^d \cdot \sum_{i=0}^{\log_b n - 1} p^i \\
&= n^{\log_b a} + cn^d \cdot \frac{p^{\log_b n} - 1}{p-1} \\
&= n^{\log_b a} + cn^d \cdot \frac{\frac{a^{\log_b n}}{(b^d)^{\log_b n}} - 1}{p-1} \\
&= n^{\log_b a} + cn^d \cdot \frac{\frac{n^{\log_b a}}{n^d} - 1}{p-1} \\
&\vdots \\
n \to \infty &= n^{\log_b a} + c\cancel{n^d} \cdot \frac{n^{\log_b a}}{\cancel{n^d} p} \\
&= O(n^{\log_b a})
\end{aligned} \tag{4}$$

Ricapitolando, quando bisogna calcolare una equazione di ricorrenza del tipo

$$T(n) = a \cdot T\left(\frac{n}{b}\right) + O(n^d)$$

possiamo ricorrere al *Teorema Master* e calcolare $T(n)$ come segue

$$T(n) = \begin{cases} O(n^d) & \text{se } a < b^d \\ O(n^d \log_b n) & \text{se } a = b^d \\ O(n^{\log_b a}) & \text{se } a > b^d \end{cases}$$