2-Metodi per la risoluzione di relazioni di ricorrenza lineari

Teoremi, metodi standard e trucchi algebrici

Dott. Simone Staccone
simone.staccone@uniroma2.it

Dipartimento di Ingegneria civile e Ingegneria informatica (DICII)
DAMON Research Group

27 Ottobre 2025

TOR VERGATA
UNIVERSITÀ DEGLI STUDI DI ROMA

Introduzione alle Relazioni di Ricorrenza

Le relazioni di ricorrenza sono uno strumento fondamentale per l'analisi della complessità degli algoritmi ricorsivi.

Esse modellano il tempo di esecuzione $T(n)$ in funzione della dimensione dei sottoproblemi:

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

Introduzione alle Relazioni di Ricorrenza

Esempio: algoritmo ricorsivo per calcolare l'*n-esimo numero di Fibonacci*:

Algorithm Fibonacci Ricorsivo

```
1: function FIBONACCI(integer n) : integer
2:     if n ≤ 1 then
3:         return n
4:     else
5:         return FIBONACCI(n − 1) + FIBONACCI(n − 2)
6:     end if
7: end function
```

Roadmap

Master Theorem

Teorema: Master Theorem

Sia data una relazione di ricorrenza $T(n)$ nella forma

$$T(n) = \begin{cases} a\,T\left(\frac{n}{b}\right) + c\,n^{\beta} & \text{se } n > 1 \\ \Theta(1) & \text{se } n = 1 \end{cases}$$

dove:

- $a \geq 1, b \geq 2$, $a, b \in \mathbb{N}_0$ (costanti intere non negative)
- $c > 0, \beta \geq 0$, $c, \beta \in \mathbb{R}$ (costanti reali)

definito $\alpha = \log_b a$, allora la funzione $T(n)$ è limitata asintoticamente da:

$$T(n) = \begin{cases} \Theta(n^{\alpha}) & \text{se } \beta < \alpha \\ \Theta(n^{\alpha} \log n) & \text{se } \beta = \alpha \\ \Theta(n^{\beta}) & \text{se } \beta > \alpha \end{cases}$$

Consideriamo l'algoritmo `MergeSort`. Studiando la sua ricorrenza possiamo rappresentarlo come una ricorrenza lineare:

Algorithm MergeSort

```
1: function MergeSort(real[] a, integer left, integer right)
2:     if left < right then
3:         center ← ⌊(left + right)/2⌋
4:         MergeSort(a, left, center)
5:         MergeSort(a, center + 1, right)
6:         Merge(a, left, center, right)          ▷ Supponiamo di averla e che costi Θ(n)
7:     end if
8: end function
```

In questo caso:

$$T(n) = 2T\left(\frac{n}{2}\right) + \Theta(n)$$

$$a = 2, \quad b = 2, \quad \beta = 1 \quad \Rightarrow \quad \alpha = \log_2 2 = 1$$

Poiché $\beta = \alpha$, segue che:

$$\boxed{T(n) = \Theta(n \log n)}$$

Roadmap

Teorema sulle ricorrenze lineari

Teorema: Ricorrenze Lineari

Sia data una relazione di ricorrenza $T(n)$ nella forma:

$$T(n) = \begin{cases} \sum_{i=1}^{h} a_i T(n-i) + cn^{\beta} & \text{se } n > h \\ \Theta(1) & \text{se } 1 \leq n \leq h \end{cases}$$

dove:

- $a_1, a_2, \ldots, a_h \in \mathbb{N}_0$ (costanti intere non negative),
- $h \in \mathbb{N}$ (ordine della ricorrenza),
- $c > 0$, $\beta \geq 0$, $c, \beta \in \mathbb{R}$ (costanti reali)

posto $a = \sum_{i=1}^{h} a_i$ allora la funzione $T(n)$ è limitata asintoticamente da:

$$T(n) = \begin{cases} \Theta(n^{\beta+1}) & \text{se } a = 1 \\ \Theta(a^n n^{\beta}) & \text{se } a \geq 2 \end{cases}$$

Consideriamo l'algoritmo ricorsivo per calcolare l'*n-esimo numero di Fibonacci*:

Algorithm Fibonacci Ricorsivo

```
1: function FIBONACCI(integer n) : integer
2:     if n ≤ 1 then
3:         return n
4:     else
5:         return FIBONACCI(n − 1) + FIBONACCI(n − 2)
6:     end if
7: end function
```

La sua relazione di ricorrenza è:

$$T(n) = T(n-1) + T(n-2) + 1$$

La relazione di ricorrenza può essere vista come una ricorrenza lineare di ordine 2:

$$T(n) = T(n-1) + T(n-2) + 1$$

Dove:

$$a_1 = 1, \quad a_2 = 1, \quad \beta = 0$$

$$a = a_1 + a_2 = 2 > 1$$

Poiché $a \geq 2$, la crescita asintotica della funzione è esponenziale:

$$\boxed{T(n) = \Theta(2^n)}$$

Non tutte le relazioni di ricorrenza possono essere risolte facilmente con il **Master Theorem** o le **ricorrenze lineari di ordine costante**.
In questi casi, occorre adottare metodi più *artigianali* e flessibili, quali:

- **Srotolamento**: espandere la ricorrenza per intuire la forma generale.
- **Sostituzione**: ipotizzare la soluzione e dimostrarla per induzione.
- **Trucchi algebrici**: manipolazioni e trasformazioni algebriche per risolvere o semplificare la ricorrenza.

Roadmap

L'idea è di espandere la ricorrenza per un numero finito di passi, per riconoscere un pattern o una somma facilmente risolvibile.

Esempio:

$$T(n) = T(n-1) + n, \quad T(1) = 1$$

Esempio:

$$T(n) = T(n-1) + n, \quad T(1) = 1$$

Srotoliamo:

$$\begin{aligned} T(n) &= T(n-1) + n = \\ &= T(n-2) + (n-1) + n = \\ &= T(1) + 2 + 3 + \cdots + (n-1) + n = \\ &= \sum_{k=0}^{n-1} n - k = \sum_{k=0}^{n-1} n - \sum_{k=0}^{n-1} k = \\ &= n^2 - \frac{n(n-1)}{2} = \frac{n^2}{2} - \frac{n}{2} \end{aligned}$$

$$\boxed{T(n) = \Theta(n^2)}$$

L'idea è di espandere la ricorrenza per un numero finito di passi, per riconoscere un pattern o una somma facilmente risolvibile.

Esempio:

$$T(n) = 3T\left(\frac{n}{3}\right) + n$$

Esempio:

$$T(n) = 3\,T\left(\frac{n}{3}\right) + n$$

Srotoliamo:

$$\begin{aligned} T(n) &= 3\,T\left(\frac{n}{3}\right) + n \\ &= 3\left(3\,T\left(\frac{n}{9}\right) + \frac{n}{3}\right) + n \\ &= 9\,T\left(\frac{n}{9}\right) + 2n \\ &= 3^k\,T\left(\frac{n}{3^k}\right) + kn \end{aligned}$$

Per $k = \log_3 n$, otteniamo:

$$\boxed{T(n) = n\,T(1) + n\log_3 n = \Theta(n \log n)}$$

Roadmap

Metodo della Sostituzione

Consiste nell'indovinare una soluzione $f(n)$ e poi dimostrarne la validità tramite induzione.

Esempio:

$$T(n) = \begin{cases} 2\,T\left(\frac{n}{2}\right) + n & \text{se } n > 1 \\ T(1) & \text{se } n = 1 \end{cases}$$

$$T(n) = 2T\left(\frac{n}{2}\right) + n,$$

Ipotesi induttiva: Si assuma che $T(k) \leq c\,k \log k$ per ogni $k < n$.

Passo base:

$$T(2) = 2T(1) + 2 = 2 \cdot 1 + 2 = 4 \leq c \cdot 2 \cdot \log 2 = 2c$$

Poniamo $c = 2$ e abbiamo soddisfatto il passo base

$$T(n) = 2T\left(\frac{n}{2}\right) + n,$$

Passo induttivo:

$$\begin{aligned} T(n) &= 2T\left(\frac{n}{2}\right) + n \leq \\ &\leq 2c\frac{n}{2}\log\left(\frac{n}{2}\right) + n = cn\log\left(\frac{n}{2}\right) + n = \\ &= cn\log n - cn\log 2 + n \end{aligned}$$

ora dobbiamo dimostrare che è vera la seguente disequazione:

$$cn\log n - cn + n \leq cn\log n$$

sviluppando la disequazione otteniamo un valore di c per cui la disqeuazione è vera $-cn + n \leq 0 \rightarrow c \geq 1$ e quindi:

$$\boxed{T(n) = O(n\log n)}$$

$$T(n) = 2T\left(\frac{n}{2}\right) + n$$

Ipotesi induttiva: Si assuma che $T(k) \geq c\,k \log k$ per ogni $k < n$.

Passo base:

$$T(2) = 2T(1) + 2 = 2 \cdot 1 + 2 = 4 \geq c \cdot 2 \cdot \log 2 = 2c$$

Ponendo $c = 1$ la disuguaglianza risulta vera, quindi il passo base è soddisfatto.

$$T(n) = 2T\left(\frac{n}{2}\right) + n$$

Passo induttivo:

$$\begin{aligned} T(n) &= 2T\left(\frac{n}{2}\right) + n \geq \\ &\geq 2c\frac{n}{2}\log\left(\frac{n}{2}\right) + n = \\ &= cn\log\left(\frac{n}{2}\right) + n = cn(\log n - \log 2) + n = \\ &= cn\log n - cn\log 2 + n \end{aligned}$$

ora dobbiamo dimostrare che è vera la seguente disequazione:

$$cn\log n - cn + n \geq cn\log n$$

sviluppando la disequazione otteniamo un valore di c per cui la disqeuazione è vera $-cn + n \geq 0 \rightarrow c \leq 1$ e quindi:

Roadmap

Trucchi algebrici

Dobbiamo stimare asintoticamente la seguente ricorrenza:

$$T(n) = \begin{cases} T(\sqrt{n}) + 1 & n > 1 \\ 1 & n = 1 \end{cases}$$

Trucchi algebrici

$$T(n) = T(\sqrt{n}) + 1$$

Possiamo effettuare un cambio di variabile per stimare la funzione $T(n)$ attraverso l'utilizzo di una seconda funzione $S(m)$ che avrà lo stesso andamento asintotico. Per questo poniamo $n = 2^m$ (quindi $m = \log_2 n$) e definiamo

$$S(m) := T(2^m)$$

Allora

$$S(m) = T(2^m) = T\left(\sqrt{2^m}\right) + 1 = T\left(2^{m/2}\right) + 1 = S\left(\frac{m}{2}\right) + 1.$$

Trucchi algebrici

La ricorrenza

$$S(m) = S\left(\frac{m}{2}\right) + 1$$

che applicando il master theorem:

$$S(m) = \Theta(\log(m))$$

sostituendo:

$$T(n) = \Theta(\log(m)) = \Theta(\log(\log(n)))$$

$$\boxed{T(n) = \Theta(\log(\log(n)))}$$

Bibliografia

- C. Demetrescu, I. Finocchi, G. F. Italiano.
  Algoritmi e strutture dati.
  Collana di istruzione scientifica, McGraw-Hill, 2004.
- T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein.
  Introduzione agli algoritmi e strutture dati, Terza edizione. MIT Press, 2009.
- Wikipedia, *Master theorem*,
  `https://en.wikipedia.org/wiki/Master_theorem_(analysis_of_algorithms)`