Corso di laurea in Informatica
Introduzione agli Algoritmi
Lezioni in modalità mista o a distanza

Equazioni di ricorrenza

Tiziana Calamoneri

SAPIENZA
UNIVERSITÀ DI ROMA

Slides realizzate sulla base di quelle preparate da T. Calamoneri e G. Bongiovanni per il corso di Informatica Generale tenuto a distanza nell'A.A. 2019/20

Equazioni di ricorrenza (1)

- Valutare il costo computazionale di un algoritmo ricorsivo è, in genere, più laborioso che nel caso degli algoritmi iterativi.
- Infatti, la natura ricorsiva della soluzione algoritmica dà luogo a una funzione matematica di costo che, essendo strettamente legata alla struttura dell'algoritmo, è anch'essa ricorsiva.
- Trovare la funzione di costo ricorsiva è piuttosto immediato. Essa però deve essere risolta, altrimenti il costo asintotico non può essere quantificato, e questa è la parte meno semplice.
- La funzione matematica ricorsiva che esprime il costo è anche detta ***equazione di ricorrenza***.

Equazioni di ricorrenza (2)

Esempio: costo computaz. della ricerca sequenziale ricorsiva:

```
def Ricerca_seq_ric(A, v, n=len(A)-1):
      if (A[n]==v):
            return n
            if (n==0)
                  return -1
            else
                  return Ricerca_seq_ric (A, v, n-1)
```

In generale: $T(n)=\Theta(1)+T(n-1)$

Caso base: $T(1)=\Theta(1)$

Equazioni di ricorrenza (3)

- La parte generale dell'equazione di ricorrenza che definisce *T(n)* deve essere sempre costituita dalla somma di almeno due addendi, di cui almeno uno contiene la parte ricorsiva (nell'esempio *T(n-1)*) mentre uno rappresenta il costo computazionale di tutto ciò che viene eseguito al di fuori della chiamata ricorsiva.
- Anche se questa parte dovesse essere un solo confronto, il suo costo non può essere ignorato, altrimenti si otterrebbe che la funzione ha una costo computazionale indipendente dalla dimensione del suo input, e questa è data da quanto illustrato nel caso base. Non possiamo dire che ciò sia impossibile, ma è molto improbabile.

Deve sempre essere presente un **caso base**.

Equazioni di ricorrenza (4)

Esistono alcuni metodi utili per risolvere le equazioni di ricorrenza, che illustreremo:

- metodo iterativo
- metodo di sostituzione
- metodo dell'albero
- metodo principale

Metodo iterativo (1)

Idea:

- sviluppare l'equazione di ricorrenza ed esprimerla come somma di termini dipendenti da *n* e dal caso base

Difficoltà:

- maggiore quantità di calcoli algebrici rispetto agli altri metodi

Metodo iterativo (2)

Esempio: Equazione relativa alla ricerca seq. ricorsiva

$T(n) = T(n-1) + \Theta(1)$

$T(1) = \Theta(1)$

$T(n) = T(n-1) + \Theta(1)$, ma $T(n-1)=T(n-2) + \Theta(1)$

Sostituiamo:

$T(n) = T(n-2) + \Theta(1) + \Theta(1)$, ma $T(n-2)=T(n-3) + \Theta(1)$

$T(n) = T(n-3) + \Theta(1) + \Theta(1) + \Theta(1)$, ma $T(n-3)=T(n-4) + \Theta(1)$

$T(n) = T(n-4) + \Theta(1) + \Theta(1) + \Theta(1) + \Theta(1)$, ecc.

fino a ottenere:

$T(n) = n\,\Theta(1) = \Theta(n)$.

Metodo iterativo (3)

Esempio: Equazione relativa alla ricerca binaria ricorsiva

$T(n) = T(n/2) + \Theta(1)$

$T(1) = \Theta(1)$

$T(n) = T(n/2) + \Theta(1)$ ma $T(n/2) = T(n/2^2) + \Theta(1)$

Sostituiamo:

$T(n) = T(n/2^2) + \Theta(1) + \Theta(1)$ ma $T(n/2^2) = T(n/2^3) + \Theta(1)$

$T(n) = T(n/2^3) + \Theta(1) + \Theta(1) + \Theta(1)$ ma…

…

$T(n) = T(n/2^k) + k\,\Theta(1)$

Quando $k = \log n$ si ha $n/2^k = 1$, quindi per tale valore di k ci fermiamo ed otteniamo:

$T(n) = \Theta(1) + \log n\,\Theta(1) = \Theta(\log n)$.

Metodo iterativo (4)

Esempio: Possiamo riscrivere un algoritmo di ricerca binaria ricorsiva diverso dal precedente (e meno furbo!):

```
def Ric_seq_ric2(A, v, i_min=0, i_max=len(A)):
  if (i_max < i_min): return -1
  else return -1;
  m=(i_max+i_min)//2
  if (A[m]==v): return m
  else return (Ricerca_seq_ric2 (A, v,i_min, m-1)
               OR Ricerca_seq_ric2 (A, v, m+1, i_max))
```

costo: $T(n) = 2T(n/2) + \Theta(1)$

$T(1) = \Theta(1)$

Metodo iterativo (5)

Esempio: Equazione relativa alla ricerca sequenziale ricorsiva (seconda versione): $T(n) = 2T(n/2) + \Theta(1)$

$T(1) = \Theta(1)$

$T(n) = 2T(n/2) + \Theta(1)$ ma $T(n/2)= 2T(n/2^2) + \Theta(1)$

Sostituiamo:

$T(n) = 2[2T(n/2^2) + \Theta(1)] + \Theta(1)=2^2T(n/2^2) + 2\,\Theta(1) + \Theta(1)=$

$\ldots=2^k\, T(n/2^k)+\sum_{i=0..k-1} 2^i\, \Theta(1)=\ldots$

… proseguo finché $k=\log n$ …

$\ldots=2^{\log n}\Theta(1)+\sum_{i=0..\log n-1} 2^i\, \Theta(1)=$

$= \Theta(n)$

$(2^{\log n} -1)/(2-1)$

Metodo iterativo (6)

A volte le cose sono un po' più complicate…

Esempio: Equazione relativa al calcolo dell'*n*-esimo numero di Fibomacci: $T(n) = T(n-1) + T(n-2) + \Theta(1)$

$T(0)=T(1) = \Theta(1)$

Provo ad usare il metodo iterativo:

$T(n) = T(n-1)+T(n-2) + \Theta(1) = T(n-2)+2T(n-3) + T(n-4) + 3\Theta(1) =$

$= T(n-3)+3T(n-4)+3T(n-5)+T(n-6) + 6\Theta(1) = \ldots$ 😵

Quando non riusciamo ad ottenere un'espressione che sappiamo gestire possiamo usare le maggiorazioni e minorazioni:

Metodo iterativo (7)

Segue Esempio: Equazione numeri di Fibonacci:

- $T(n) = T(n-1) + T(n-2) + \Theta(1)$
- $T(0)=T(1) = \Theta(1)$

A partire da $T(n)=T(n-1)+T(n-2) + \Theta(1)$ e da $T(n-1) \geq T(n-2)$:

- $T(n) \leq 2T(n-1) + \Theta(1)$: questa disuguaglianza ci permetterà di derivare un limite superiore O;
- $T(n) \geq 2T(n-2) + \Theta(1)$: questa disuguaglianza ci permetterà di derivare un limite inferiore Ω.

Metodo iterativo (8)

Segue Esempio: Equazione numeri di Fibonacci:

- $T(n) = T(n-1) + T(n-2) + \Theta(1)$
- $T(0)=T(1) = \Theta(1)$

$T(n) \leq 2T(n-1) + \Theta(1) \leq 2^2T(n-2) + 2^1\Theta(1) + \Theta(1) \leq$

$\leq 2^3T(n-3) + 2^2\Theta(1) + 2^1\Theta(1) + \Theta(1) \leq \ldots$

$\leq 2^kT(n-k) + \sum_{i=0}^{k-1} 2^i\ \Theta(1)$

Il procedimento si ferma quando $n - k = 1$, ossia $k = n - 1$ per cui otteniamo:

$T(n) \leq 2^{n-1}\ \Theta(1) + \sum_{i=0}^{n-2} 2^i\ \Theta(1) = 2^{n-1}\ \Theta(1) + (2^{n-1} - 1)\Theta(1) = (2^n - 1)\Theta(1)$

e dunque troviamo che $T(n) = O(2^n)$.

Metodo iterativo (9)

Segue Esempio: Equazione numeri di Fibonacci:

- $T(n) = T(n-1) + T(n-2) + \Theta(1)$
- $T(0)=T(1) = \Theta(1)$

$T(n) \geq 2T(n-2) + \Theta(1) \geq 2^2T(n-4) + 2^1\Theta(1) + \Theta(1) \geq$

$\geq 2^3T(n-6) + 2^2\Theta(1) + 2^1\Theta(1) + \Theta(1) \geq \ldots$

$\geq 2^kT(n-2k) + \sum_{i=0}^{k-1} 2^i\ \Theta(1)$

Il procedimento si ferma quando $n - 2k = 0$ (se n è pari; per n dispari il procedimento termina quando $n - 2k = 1$ e differisce per alcuni dettagli, ma il comportamento asintotico non cambia), ossia $k = n/2$ per cui otteniamo:

$T(n) \geq 2^{n/2}\ \Theta(1) + \sum_{i=0}^{\frac{n}{2}-1} 2^i\ \Theta(1) = 2^{n/2}\ \Theta(1) + (2^{n/2} - 1)\Theta(1) = (2*2^{n/2} - 1)\Theta(1)$ e dunque troviamo che $T(n) = \Omega(2^{n/2})$.

Metodo iterativo (10)

Segue Esempio: Equazione numeri di Fibomacci:

- $T(n) = T(n-1) + T(n-2) + \Theta(1)$
- $T(0)=T(1) = \Theta(1)$

Anche se non siamo in grado di trovare una funzione asintotica precisa (Θ), possiamo comunque concludere che il calcolo dei numeri di Fibonacci con una tecnica ricorsiva richiede un tempo esponenziale in n, visto che

$$k_1 2^{n/2} \leq T(n) \leq k_2 2^n$$

per opportune costanti k_1 e k_2.

Metodo di sostituzione (1)

Idea:

- si ipotizza una soluzione per l'equazione di ricorrenza data;
- si verifica (dimostrando per induzione) se essa "funziona".

Difficoltà:

- si deve trovare la funzione più vicina alla vera soluzione, perché tutte le funzioni più grandi (se stiamo cercando O) o più piccole (se stiamo cercando Ω) funzionano.
- In effetti questo metodo serve soprattutto nelle dimostrazioni mentre si sconsiglia di utilizzarlo nella pratica.

Metodo di sostituzione (2)

Esempio: Equazione relativa alla ricerca seq. ricorsiva

- $T(n) = T(n – 1) + \Theta(1)$
- $T(1) = \Theta(1)$

- Ipotizziamo la soluzione $T(n)=cn$ per una costante opportuna c.
- $T(n)=cn=c(n-1)+ \Theta(1)$, cioè $c= \Theta(1)$;
- $T(1)=c= \Theta(1)$,
- Non è detto che le due costanti nascoste nella notazione asintotica siano le stesse. Per questa ragione, è necessario eliminare la notazione asintotica dall'equazione di ricorrenza, così che le costanti non rimangano "nascoste" nella notazione, conducendo a risultati errati
- …

Metodo di sostituzione (3)

segue Esempio: Equazione relativa alla ricerca seq. ricorsiva

- $T(n) = T(n-1) + c$
- $T(1) = d$. per due costanti c e d fissate.

- Ipotizziamo la soluzione $T(n) = O(n)$, *ossia* $T(n) \leq kn$ dove k è una costante che va ancora determinata.
- N.B. non si può sperare in una soluzione esatta, ma possiamo solo maggiorare o minorare.

Metodo di sostituzione (4)

segue Esempio: Equazione relativa alla ricerca seq. ricorsiva

- $T(n) = T(n-1) + c$
- $T(1) = d$ per due costanti c e d fissate.

- Sostituiamo nel caso base:

  $T(1) \leq k$; poiché sapevamo che $T(1) = d$, la disuguaglianza è soddisfatta se e solo se $k \geq d$.
- Sostituiamo nella formulazione ricorsiva:

  $T(n) \leq k(n-1) + c = kn - k + c \leq kn$; vera se e solo se $k \geq c$.
- La soluzione $T(n) \leq kn$ è corretta per tutti i valori di k tali che $k \geq c$ e $k \geq d$. Poiché un tale k esiste sempre, una volta fissati c e d, $T(n)$ è in $O(n)$.

Metodo di sostituzione (5)

segue Esempio: Equazione relativa alla ricerca seq. ricorsiva

- $T(n) = T(n - 1) + c$
- $T(1) = d$ per due costanti c e d fissate.

- Che cosa sarebbe successo se avessimo ipotizzato $T(n) \leq kn^2$?
- Anche questa soluzione è corretta per tutti i valori di k tali che $k \geq c$ e $k \geq d$.

- A noi interessa stimare $T(n)$ asintoticamente tramite la funzione più piccola, e questo è spesso un obiettivo difficile.

Metodo di sostituzione (6)

segue Esempio: Equazione relativa alla ricerca seq. ricorsiva

- $T(n) = T(n - 1) + c$
- $T(1) = d$ per due costanti c e d fissate.

- Per rendere il nostro risultato stretto, ipotizziamo ora la soluzione $T(n) = \Omega(n)$, *ossia* $T(n) \geq hn$ dove h è una costante che va ancora determinata.
- In modo assolutamente analogo al caso O, sostituiamo nel caso base ottenendo $h \leq d$,
- Sostituiamo nella formulazione ricorsiva dell'equazione di ricorrenza ottenendo $T(n) \geq h(n - 1) + c = hn - h + c \geq hn$ vera se e solo se $h \leq c$.
- Deduciamo dunque che $T(n)$ è un $\Omega(n)$ poiché è possibile trovare un valore h ($h \leq c$, $h \leq d$) *per il quale si ha* $T(n) \geq hn$.

Metodo di sostituzione (7)

segue Esempio: Equazione relativa alla ricerca seq. ricorsiva

- $T(n) = T(n - 1) + c$
- $T(1) = d$ per due costanti c e d fissate.

Dalle due soluzioni: $T(n) = O(n)$ e $T(n) = \Omega(n)$, si ottiene ovviamente che $T(n) = \Theta(n)$.

Metodo di sostituzione (8)

Esempio.

- $T(n)=2T(n/2)+\Theta(1)$
- $T(1)=\Theta(1)$

Dobbiamo eliminare per prima cosa la notazione asintotica:

- $T(n)=2T(n/2)+c$ per due costanti positive **fissate** c e d
- $T(1)=d$

Ipotizziamo la soluzione $T(n) \leq kn$ per qualche k **da determinare** e sostituiamo:

- nel caso base: $d \leq k$
- nel caso generale: $T(n) \leq 2\,k\,n/2 + c = kn + c$ che non è MAI $\leq kn$…

Vuol dire che l'ipotesi è errata? non sempre:

Metodo di sostituzione (9)

segue Esempio.

- T(n)=2T(n/2)+*c*
- T(1)=*d*

Ipotizziamo la nuova soluzione T(n) ≤ *kn-h* per qualche *k* ed *h* **da determinare** e sostituiamo:

- nel caso base: d ≤ *k-h*
- nel caso generale: *T(n)≤2(k n/2-h)+c =kn+-2h+c ≤ kn-h sse h ≥ c.*

Da qui deduciamo che *T(n)=O(n).*

Poi dobbiamo dimostrare anche che T(n)= *Ω(n)*

PER ESERCIZIO

Metodo dell'albero (1)

Idea:

- rappresentare graficamente lo sviluppo del costo computazionale dell'algoritmo, in modo da poterla valutare con maggior facilità rispetto al metodo iterativo

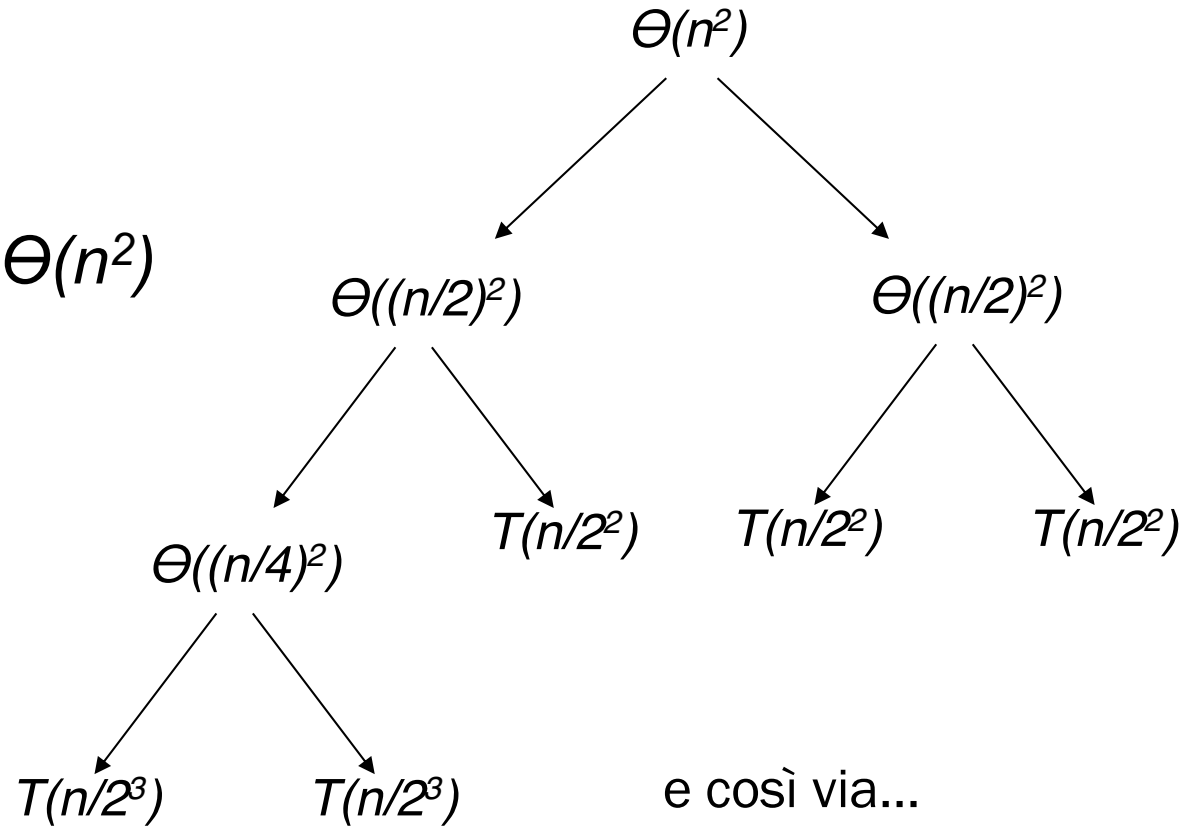
Difficoltà:

- come il metodo iterativo

Metodo dell'albero (2)

Esempio:

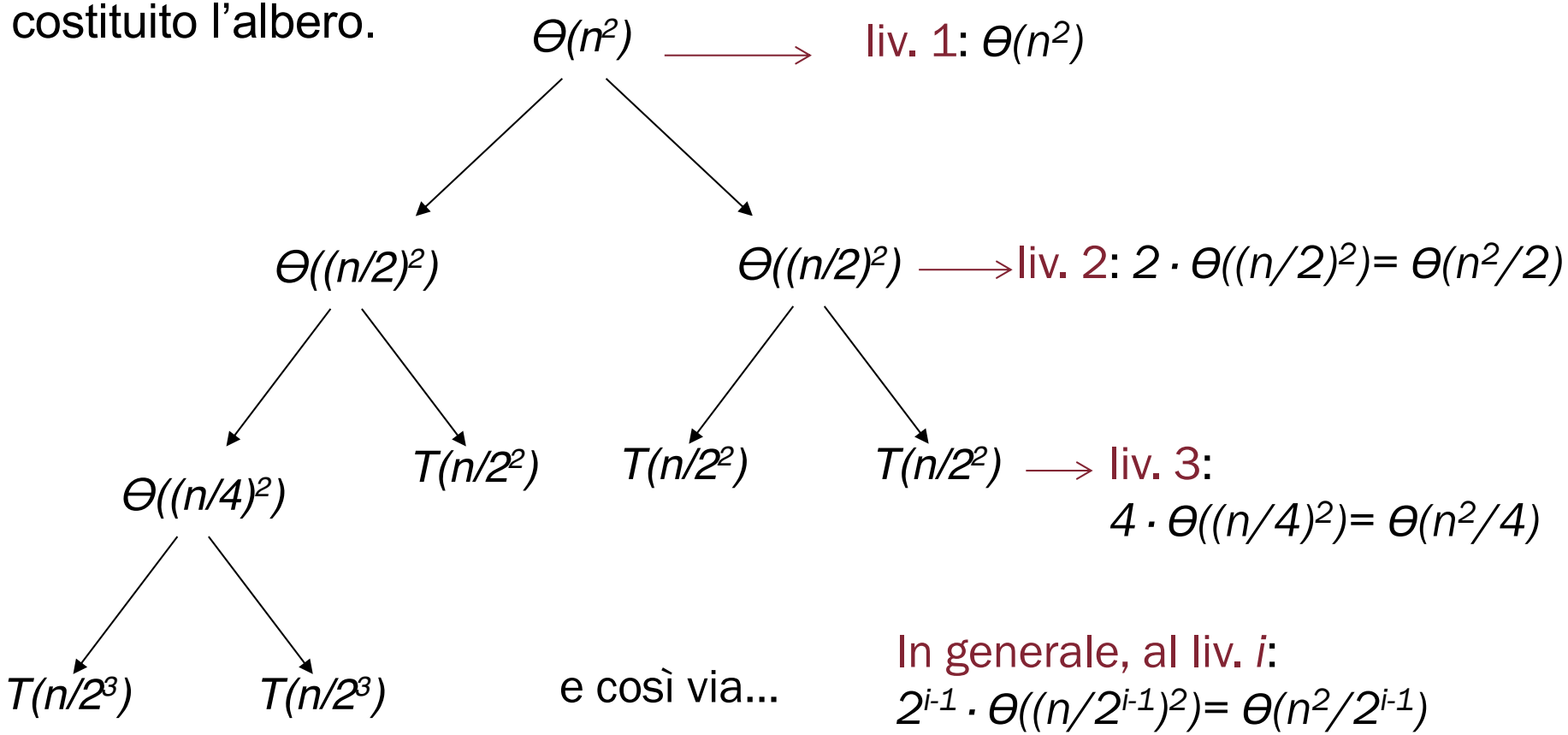
- $T(n) = 2T(n/2) + \Theta(n^2)$
- $T(1) = \Theta(1)$



Metodo dell'albero (3)

segue Esempio:

Una volta completato l'albero, il costo computazionale è dato dalla somma dei contributi di tutti i livelli (cioè le "righe" in cui sono disposti i nodi) di cui è costituito l'albero.



Metodo dell'albero (4)

segue Esempio:

In questo caso il numero di livelli dell'albero ha un valore tale che $n/2^{i-1}=1$, ossia $i-1=\log n$ da cui $i=\log n+1$.

Sommiamo i contributi di tutti i livelli:

$$\sum_{i=1}^{\log n+1} \Theta(\frac{n^2}{2^{i-1}}) = n^2 \sum_{j=o}^{\log n} \Theta(\frac{1}{2^j}) = \Theta(n^2)$$

Metodo principale (1)

Idea:

- avere una "ricetta" meccanica per risolvere un'equazione di ricorrenza

Difficoltà:

- funziona solo quando l'equazione è della forma $T(n)=aT(n/b)+f(n)$ con $T(1)=\Theta(1)$

Metodo principale (2)

Enunciato del teorema principale:

Dati $a \geq 1$, $b > 1$, una funzione asintoticamente positiva $f(n)$ ed un'equazione di ricorrenza di forma $T(n) = aT(n/b) + f(n)$, $T(1) = \Theta(1)$ vale che:

- Se $f(n) = O(n^{\log_b a - \varepsilon})$ per qualche costante $\boldsymbol{\varepsilon} > 0$ allora $T(n) = \Theta(n^{\log_b a})$
- Se $f(n) = \Theta(n^{\log_b a})$ allora $T(n) = \Theta(n^{\log_b a} \log n)$
- Se $f(n) = \Omega(n^{\log_b a + \varepsilon})$ per qualche costante $\boldsymbol{\varepsilon} > 0$ e se $a\, f(n/b) \leq c\, f(n)$ per qualche costante $c < 1$ e per n sufficientemente grande, allora $T(n) = \Theta(f(n))$.

Metodo principale (3)

Il teorema principale ci dice che in ciascuno dei tre casi vengono confrontati fra loro *f(n)* e $n^{log_b a}$ ed il costo computazionale è governato dal maggiore dei due:

- se (caso 1) il più grande dei due è $n^{log_b a}$, allora il costo è $\Theta(n^{log_b a})$;
- se (caso 3) il più grande dei due è *f(n)*, allora il costo è $\Theta(f(n))$;
- se (caso 2) sono uguali, allora si moltiplica *f(n)* per un fattore logaritmico.

Metodo principale (4)

Si noti che "più grande" e "più piccolo" in questo contesto significa ***polinomialmente*** più grande (o più piccolo), data la presenza all'esponente di ε. In altre parole, *f(n)* deve essere asintoticamente più grande (o più piccola) rispetto a $n^{log_b a}$ di un fattore n^{ε} per qualche $\varepsilon > 0$.

In effetti fra i casi 1 e 2 vi è un intervallo in cui *f(n)* è più piccola di $n^{log_b a}$, ma non polinomialmente.

Analogamente, fra i casi 2 e 3 vi è un intervallo in cui *f(n)* è più grande di $n^{log_b a}$, ma non polinomialmente.

Metodo principale (5) – caso 1

Esempio.

$T(n) = 9T(n/3) + \Theta(n)$ e $T(1) = \Theta(1)$

- $a = 9$, $b = 3$
- $f(n) = \Theta(n)$
- $n^{\log_b a} = n^{\log_3 9} = n^2$

Poiché $f(n) = \Theta(n^{\log_3 9 - \varepsilon})$ con $\boldsymbol{\varepsilon} = 1$, siamo nel **caso 1**, per cui
$T(n) = \Theta(n^{\log_b a}) = \Theta(n^2)$.

Metodo principale (6) – caso 2

Esempio.

$T(n) = T(\frac{2}{3}n) + \Theta(1)$ e $T(1) = \Theta(1)$

- $a = 1$, $b = 3/2$
- $f(n) = \Theta(1)$
- $n^{\log_b a} = n^{\log_{3/2} 1} = n^0 = 1$

Poiché $f(n) = \Theta(n^{\log_b a})$ siamo nel **caso 2**, per cui $T(n) = \Theta(n^{\log_b a} \log n) = \Theta(\log n)$.

Metodo principale (7) – caso 3

Esempio.

$T(n) = 3T(n/4) + \Theta(n \log n)$ e $T(1) = \Theta(1)$

- $a = 3$, $b = 4$
- $f(n) = \Theta(n \log n)$
- $n^{\log_b a} = n^{\log_4 3} \approx n^{0,7}$

Poiché $f(n) = \Omega(n^{\log_4 3+\varepsilon})$ con, ad esempio, $\varepsilon = 0,2$, siamo nel **caso 3** se possiamo dimostrare che $3\frac{n}{4}\log\frac{n}{4} \leq c\, n \log n$, per qualche $c < 1$ ed n abbastanza grande. Ponendo $c = ¾$ otteniamo:

$$3\frac{n}{4}\log\frac{n}{4} \leq \frac{3}{4} n \log n$$

che è vera, quindi $T(n) = \Theta(n \log n)$.

Metodo principale (8)

Esempio.

$T(n) = 2T(n/2) + \Theta(n \log n)$ e $T(1) = \Theta(1)$

- $a = 2$, $b = 2$
- $f(n) = \Theta(n \log n)$
- $n^{\log_b a} = n^{\log_2 2} = n$

Ora, $f(n) = \Theta(n \log n)$ è asintoticamente più grande di $n^{\log_b a} = n$, ma non polinomialmente più grande. Infatti, $\log n$ è asintoticamente minore di n^{ε} per qualunque valore di $\boldsymbol{\varepsilon} > 0$. Di conseguenza non possiamo applicare il metodo del teorema principale.

Esercizi per casa

Esercizi

Calcolare la soluzione delle seguenti equazioni di ricorrenza con tutti e quattro i metodi ove possibile:

- $T(n)=2T(n/2)+\Theta(n)$ $T(1)=\Theta(1)$
- $T(n)=3T(n/2)+\Theta(n)$ $T(1)=\Theta(1)$
- $T(n)=3T(n/4)+\Theta(n)$ $T(1)=\Theta(1)$
- $T(n)=2T(n/2)+\Theta(n)$ $T(1)=\Theta(1)$
- $T(n)=4T(n/2)+\Theta(n)$ $T(1)=\Theta(1)$
- $T(n)=2T(n/2)+\Theta(n^3)$ $T(1)=\Theta(1)$
- $T(n)=16T(n/4)+\Theta(n^2)$ $T(1)=\Theta(1)$
- $T(n)=T(n-1)+\Theta(n)$ $T(1)=\Theta(1)$
- $T(n)=3T(n/2)+\Theta(n \log n)$ $T(1)=\Theta(1)$