RECUPERO DEI DATI
ALGORITMI DI ORDINAMENTO

ALGORITMI E STRUTTURE DATI

Algoritmi Ricorsivi e Complesità

Algoritmi Ricorsivi

Fino a ora abbiamo visto algoritmi iterativi

Possiamo calcolarne la complessità contando il numero di iterazioni.

Possiamo sfruttare la ricorsione scrivendo gli algoritmi?

Come calcolarne la complessità?

Un Algoritmo Ricorsivo . . . Familiare

```
def do_something(A, n=|A|):
  if n>1
    do_something(A, n-1)

    j ← n
    while (j>1 and A[j]<A[j-1]):

      swap(A,j-1,j)
      j←j-1
    endwhile
  endif
enddef
```

Un Algoritmo Ricorsivo . . . Familiare

```
def do_something(A, n=|A|):
  if n>1
    do_something(A, n-1)

    j ← n
    while (j>1 and A[j]<A[j-1]):

      swap(A,j-1,j)
      j←j-1
    endwhile
  endif
enddef
```

Non è familiare questo codice? È insertion sort ricorsivo!

Come Calcolarne la Complessità?

Usando le equazioni ricorsive di complessità!!!

Se $T(|A|)$ il tempo per risolvere il problema su A allora

Come Calcolarne la Complessità?

Usando le equazioni ricorsive di complessità!!!

Se $T(|A|)$ il tempo per risolvere il problema su A allora

$$T(n) = \begin{cases} \Theta(1) & \text{se } n = 1 \\ T(n-1) + \Theta(n) & \text{se } n > 1 \end{cases}$$

Per induzione, $T(|A|) \in \sum_{i=0}^{|A|} \Theta(i) = \Theta(|A|^2)$

Un Esempio Più Spinoso

```
def rec_find(A,v,l=1,r=|A|):
  if r < l
    return 0
  endif

  m ← (l+r)/2
  if A[m]=v
    return m
  endif
  if A[m]>v
    return rec_find(A, v, l, m−1)
  else
    return rec_find(A, v, m+1, r)
  endif
enddef
```

Scriviamo l'Equazione Ricorsiva...

Se $n = 1$ o $A[m] = v$ allora $T(n) \in \Theta(1)$. Altrimenti

$$T(n) \leq T(\lfloor n/2 \rfloor - 1) + \Theta(1)$$

Scriviamo l'Equazione Ricorsiva...

Se $n = 1$ o $A[m] = v$ allora $T(n) \in \Theta(1)$. Altrimenti

$$T(n) \leq T(\lfloor n/2 \rfloor - 1) + \Theta(1) \leq T(n/2) + \Theta(1)$$

Scriviamo l'Equazione Ricorsiva...

Se $n = 1$ o $A[m] = v$ allora $T(n) \in \Theta(1)$. Altrimenti

$$T(n) \leq T(\lfloor n/2 \rfloor - 1) + \Theta(1) \leq T(n/2) + \Theta(1)$$

Poniamo $m \stackrel{\text{def}}{=} \log_2 n$ e $P(m) \stackrel{\text{def}}{=} T(2^m)$

Scriviamo l'Equazione Ricorsiva...

Se $n = 1$ o $A[m] = v$ allora $T(n) \in \Theta(1)$. Altrimenti

$$T(n) \leq T(\lfloor n/2 \rfloor - 1) + \Theta(1) \leq T(n/2) + \Theta(1)$$

Poniamo $m \stackrel{\text{def}}{=} \log_2 n$ e $P(m) \stackrel{\text{def}}{=} T(2^m)$

$$\begin{aligned} P(m) &= T(2^m) \\ &\leq T\left(2^m/2\right) + \Theta(1) \\ &= T\left(2^{m-1}\right) + \Theta(1) = P(m-1) + \Theta(1) \end{aligned}$$

Quindi $P(m) \leq \sum_{i=0}^{m} \Theta(1) = \Theta(m)$, $P(m) \in O(m)$ e ...

$$T(|A|) = P(\log_2 |A|) = O(\log |A|)$$

Quando l'Equazione è Più Complicata?

Per esempio:

$$T(i) = \begin{cases} \Theta(1) & \text{se } i = 1 \\ 2 * T(i/2) + \Theta(i) & \text{se } i > 1 \end{cases}$$

Tre metodi:

- metodo di sostituzione
- l'albero di ricorsione
- il teorema dell'esperto

Metodo di Sostituzione

Il Metodo di Sostituzione

Due passi:

1. ipotizzo una complessità per $T(n)$
2. dimostro per induzione che esistono delle costanti che soddisfano l'ipotesi

Il Metodo di Sostituzione: Un Esempio

Es. $T(n) = 2 * T(n/2) + O(n)$

1. ipotizzo che $T(n) \in O(n \log n)$
2. assumo che $\exists c \; \forall m < n \; T(m) \leq c * m * \log_2 m$

Il Metodo di Sostituzione: Un Esempio

Es. $T(n) = 2 * T(n/2) + O(n)$

1. ipotizzo che $T(n) \in O(n \log n)$
2. assumo che $\exists c \; \forall m < n \; T(m) \leq c * m * \log_2 m$

$$
\begin{aligned}
T(n) &\leq 2 * T(n/2) + c' * n \\
&\leq 2 * (c * n/2 * log_2(n/2)) + c' * n \\
&= c * n * log_2(n/2) + c' * n \\
&= c * n * log_2(n) - c * n * log_2(2) + c' * n
\end{aligned}
$$

Se scelgo $c' < c$, allora $T(n) \leq c * n * log_2(n)$.

La costante c è la **stessa** per ogni n

Il Metodo di Sostituzione: Come NON fare

Es. $T(n) = 3 * T(n/2) + O(n)$

1. ipotizzo che $T(n) \in O(n * \log n)$
2. assumo che $\exists c \; \forall m < n \; T(m) \leq c * m * \log_2 m$

$$
\begin{aligned}
&\leq 3 * (c * n/2 * log_2(n/2)) + c' * n \\
&= \frac{3}{2} * c * n * log_2(n/2) + c' * n \\
&= \frac{3}{2} * c * n * log_2(n) - c * n * log_2(2) + c' * n \\
&\leq c * \frac{3}{2} * n * log_2(n)
\end{aligned}
$$

Il Metodo di Sostituzione: Come NON fare

Es. $T(n) = 3 * T(n/2) + O(n)$

1. ipotizzo che $T(n) \in O(n * \log n)$
2. assumo che $\exists c \; \forall m < n \; T(m) \leq c * m * \log_2 m$

$$
\begin{aligned}
&\leq 3 * (c * n/2 * log_2(n/2)) + c' * n \\
&= \frac{3}{2} * c * n * log_2(n/2) + c' * n \\
&= \frac{3}{2} * c * n * log_2(n) - c * n * log_2(2) + c' * n \\
&\leq c * \frac{3}{2} * n * log_2(n)
\end{aligned}
$$

La costante per $m < n$ era c, mentre per n è $\frac{3}{2} * c$

Albero di Ricorsione

Albero di Ricorsione

Costruisci un albero (grafo connesso aciclico) in cui ogni nodo:

- rappresenta una chiamata ricorsiva
- corrisponde al costo della chiamata senza i passi ricorsivi
- è collegato alle chiamate ricorsive che genera

Sommando il costo di tutti i nodi, otteniamo il costo totale.

Albero di Ricorsione: Un Esempio

Consideriamo l'equazione

$$T(n) = \begin{cases} \Theta(1) & \text{se } n = 1 \\ 8 * T(n/2) + \Theta(n^2) & \text{se } m > 1 \end{cases}$$

Albero di Ricorsione: Un Esempio

Consideriamo l'equazione

$$T(n) = \begin{cases} \Theta(1) & \text{se } n = 1 \\ 8 * T(n/2) + \Theta(n^2) & \text{se } m > 1 \end{cases}$$

Scegliamo una funzione in $\Theta(n^2)$ (es. $c * n^2$)

Albero di Ricorsione: Un Esempio

Consideriamo l'equazione

$$T(n) = \begin{cases} \Theta(1) & \text{se } n = 1 \\ 8 * T(n/2) + \Theta(n^2) & \text{se } m > 1 \end{cases}$$

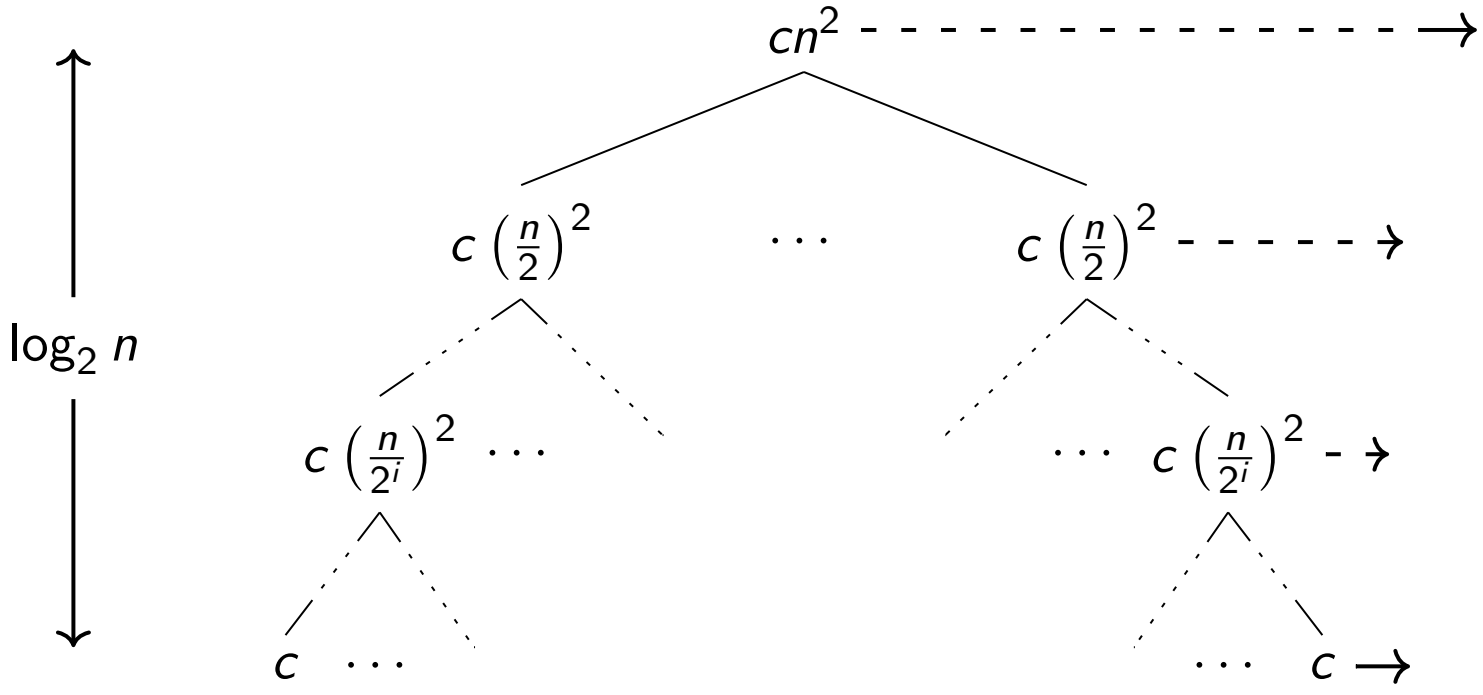
Scegliamo una funzione in $\Theta(n^2)$ (es. $c * n^2$)

Se m è la dimensione dell'input per una chiamata:

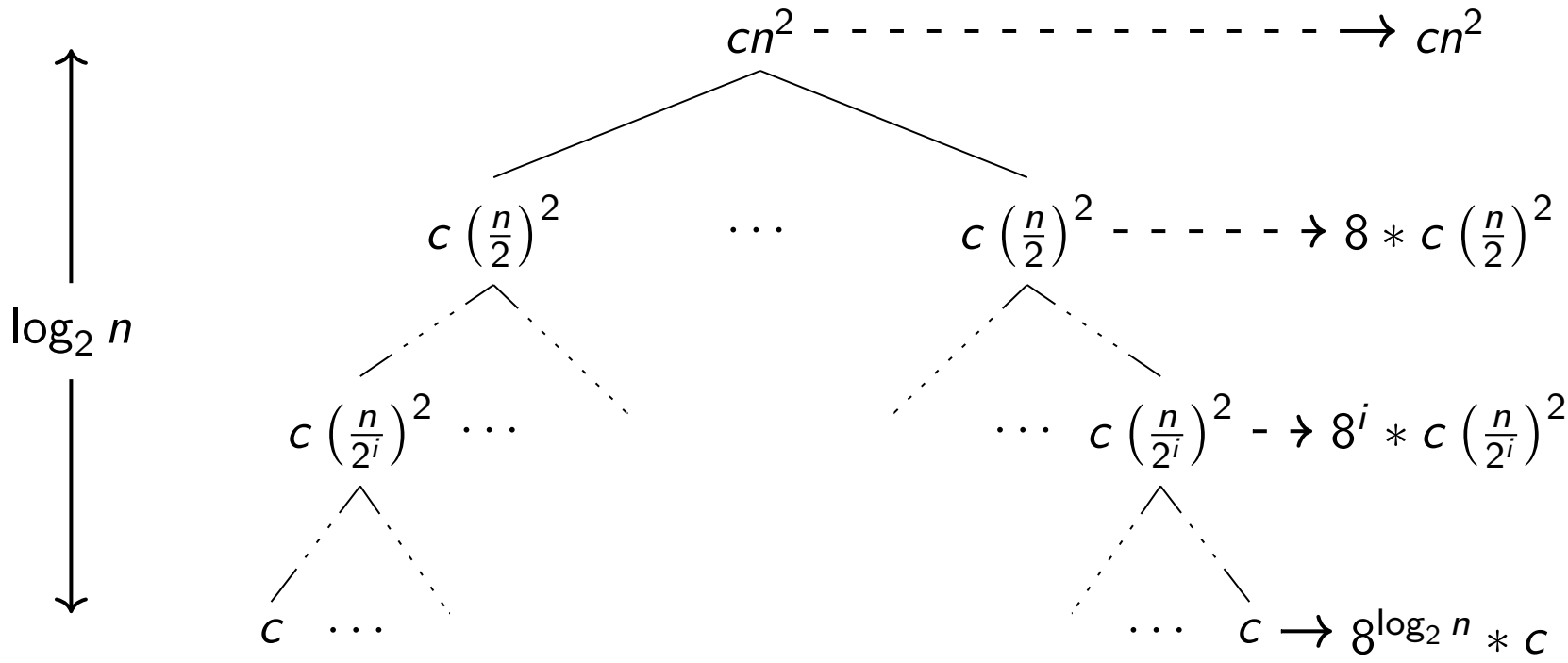
- il costo su una chiamata è cm^2
- ogni nodo dell'albero ha 8 figli (le chiamate ricorsive)
- m si dimezza a ogni chiamata ricorsiva

Albero di Ricorsione: Un Esempio (Cont'd)

Albero di Ricorsione: Un Esempio (Cont'd)

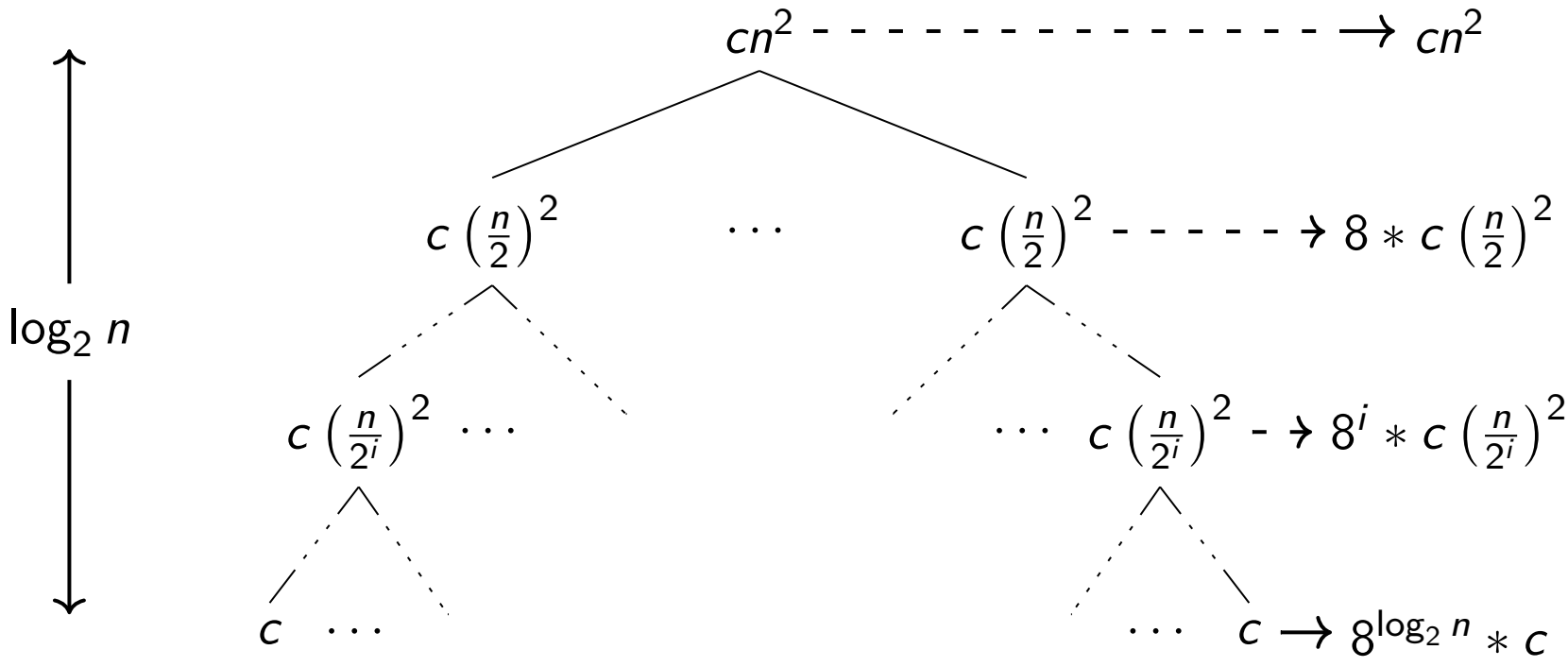


Albero di Ricorsione: Un Esempio (Cont'd)



$$T_M(n) = cn^2\left(1 + 2 + \ldots + 2^i + \ldots + 2^{\log_2 n}\right)$$

Albero di Ricorsione: Un Esempio (Cont'd)



$$T_M(n) = cn^2\left(1 + 2 + \ldots + 2^i + \ldots + 2^{\log_2 n}\right)$$
$$= cn^2\left(2^{1+\log_2 n} - 1\right) = cn^2(2n-1) \in \Theta(n^3)$$

Il Metodo dell'Esperto

Metodo dell'Esperto

Theorem (Teorema dell'Esperto)

Siano $a \geq 1$ *e* $b > 1$ *due costanti,* $f(n)$ *una funzione* ≥ 0 *e*

$$T(n) \stackrel{def}{=} \begin{cases} \Theta(1) & se\ n = 1 \\ a * T(n/b) + f(n) & altrimenti \end{cases}$$

Se, per qualche costante ϵ, $f(n)$ *sta in*

1. $O\left(n^{\log_b a - \epsilon}\right)$, *allora* $T(n) \in \Theta(n^{\log_b a})$;
2. $\Theta\left(n^{\log_b a}\right)$, *allora* $T(n) \in \Theta(n^{\log_b a} * \log n)$;
3. $\Omega\left(n^{\log_b a + \epsilon}\right)$ *e* $a * f(n/b) \leq c * f(n)$ *per qualche* $c < 1$ *e* n *sufficientemente grande, allora* $T(n) \in \Theta(f(n))$.

Applicazioni

Consideriamo:

$T(n) = 9 * T(n/3) + O(n)$: $a = 9$, $b = 3$ e $f(n) \in O(n^{\log_3 9 - \epsilon})$
con $\epsilon = 1$ quindi Caso 1 $\Rightarrow$ $T(n) \in \Theta(n^2)$

Applicazioni

Consideriamo:

$T(n) = 9 * T(n/3) + O(n)$: $a = 9$, $b = 3$ e $f(n) \in O(n^{\log_3 9 - \epsilon})$
con $\epsilon = 1$ quindi Caso 1 $\Rightarrow$ $T(n) \in \Theta(n^2)$

$T(n) = T(2 * n/3) + 1$: $a = 1$, $b = 3/2$ e
$f(n) \in \Theta(n^{\log_{3/2} 1}) = \Theta(1)$ quindi Caso 2 $\Rightarrow$ $T(n) \in \Theta(\log n)$

Applicazioni

Consideriamo:

$T(n) = 9 * T(n/3) + O(n)$: $a = 9$, $b = 3$ e $f(n) \in O(n^{\log_3 9 - \epsilon})$
con $\epsilon = 1$ quindi Caso 1 $\Rightarrow$ $T(n) \in \Theta(n^2)$

$T(n) = T(2 * n/3) + 1$: $a = 1$, $b = 3/2$ e
$f(n) \in \Theta(n^{\log_{3/2} 1}) = \Theta(1)$ quindi Caso 2 $\Rightarrow$ $T(n) \in \Theta(\log n)$

$T(n) = 3 * T(n/4) + n * \log n$: $a = 3$, $b = 4$ e $f(n) \in \Omega(n^{\log_4 3 + \epsilon})$
con $\epsilon \approx 0.2$ quindi Caso 3 $\Rightarrow$ $T(n) \in \Theta(n * \log n)$

Applicazioni

Consideriamo:

$T(n) = 2 * T(n/2) + n * \log n$: $a = 2$, $b = 2$ e $f(n) \in \Omega(n^{\log_2 2})$.

Sfortunatamente,

$$\lim_{n \to \infty} \frac{n * \log n}{n^{1+\epsilon}} = 0$$

per ogni $\epsilon > 0$ e **NON** si può applicare il teorema dell'esperto.

Dimostrazione

Lemma

Siano $a \geq 1$ e $b > 1$ due costanti, $f(n)$ una funzione ≥ 0 e

$$T(n) \stackrel{def}{=} \begin{cases} \Theta(1) & se\ n = 1 \\ a * T(n/b) + f(n) & se\ n = b^i \end{cases}$$

dove i è un naturale positivo. Allora

$$T(n) \in \Theta(n^{\log_b a}) + \sum_{j=0}^{\log_b n - 1} a^j f(n/b^j)$$

Dim. tramite l'albero di ricorsione.

Dimostrazione (Cont'd)

Lemma

Siano $a \geq 1$ e $b > 1$ due costanti, $f(n)$ una funzione ≥ 0 e:

$$g(n) \stackrel{def}{=} \sum_{j=0}^{\log_b n - 1} a^j f(n/b^j)$$

1. *se $f(n) \in O\left(n^{\log_b a - \epsilon}\right)$, allora $g(n) \in O(n^{\log_b a})$;*
2. *se $f(n) \in \Theta\left(n^{\log_b a}\right)$, allora $g(n) \in \Theta(n^{\log_b a} \log n)$;*
3. *se $a * f(n/b) \leq cf(n)$ per qualche $c < 1$ e n sufficientemente grande, allora $g(n) \in \Theta(f(n))$.*

Dim. tramite sostituzione delle condizioni