Esercitazione

30 marzo 2023

Tipologie di esercizi

1. Formalizzazione problema computazionale
2. Notazioni asintotiche:
   a) Via definizione (c, n_0)
   b) Applicando le proprietà
   c) Sequenze di funzioni da ordinare
   d) Confronto tempo di esecuzione di algoritmi
3. Calcolo del tempo di esecuzione di algoritmi (senza chiamate ricorsive)
4. Scrivere la relazione di ricorrenza per il tempo di esecuzione di algoritmi ricorsivi
5. Risolvere relazioni di ricorrenza
6. Tecnica del Divide et Impera

Esercizi svolti in classe

Esercizio 2

Per ciascuna delle seguenti coppie di funzioni $f(n)$ e $g(n)$, dire se $f(n) = O(g(n))$, oppure se $g(n) = O(f(n))$.

- $f(n) = (n^2 - n)/2, \qquad g(n) = 6n$
- $f(n) = n + 2\sqrt{n}, \qquad g(n) = n^2$
- $f(n) = n + \log n, \qquad g(n) = n\sqrt{n}$
- $f(n) = n^2 + 3n, \qquad g(n) = n^3$
- $f(n) = n \log n, \qquad g(n) = n\sqrt{n}/2$
- $f(n) = n + \log n, \qquad g(n) = \sqrt{n}$
- $f(n) = 2(\log n)^2, \qquad g(n) = \log n + 1$
- $f(n) = 4n \log n + n, \qquad g(n) = (n^2 - n)/2$
- $f(n) = (n^2 + 2)/(1 + 2^{-n}), \qquad g(n) = n + 3$
- $f(n) = n + n\sqrt{n}, \qquad g(n) = 4n \log(n^3 + 1)$

(n_0 = 31.869)

PARTITION

Una chiamata alla procedura PARTITION (come studiata) su [6, 1, 3, 9, 8] restituisce

A. 2 B. 3 C. 6 D. Nessuno dei precedenti

```
Partition (A, p, r)
x = A[p]
i = p-1
j = r+1
while True
        do repeat j=j-1 until A[j]≤ x
           repeat i=i+1 until A[i]≥ x
           if i < j
                 then scambia A[i] ↔ A[j]
                 else return j
```

A = [6, 1, 3, 9, 8]

Tempo di esecuzione 4

Il tempo di esecuzione del seguente frammento di pseudocodice è

```
for i=1 to n/2
   PARTITION(A,i,n)
```

A. Θ (log n)
B. Θ (n log n)
C. Θ (n)

D. Nessuna delle risposte precedenti

Esercizi

Scrivere la relazione di ricorrenza soddisfatta dal tempo di esecuzione degli algoritmi nelle prossime slides.

Si noti che **non ci interessa cosa calcolino gli algoritmi, ma soltanto quanto tempo impieghino.**

Supponiamo che il tempo per eseguire **qualcosa** sia c

Esempi

procedure daffy(n)
  if $n = 1$ or $n = 2$ **then do** qualcosa
    else
      daffy($n - 1$);
      for $i = 1$ **to** n **do**
        qualcosa di nuovo
      daffy($n - 1$)

Esempi

procedure elmer(n)
    if $n = 1$ **then do** qualcosa
        else if $n = 2$ **then do** qualcos'altro
        else
            for $i = 1$ **to** n **do**
                elmer($n - 1$)
                fa qualcosa di differente

Esempi

procedure bar(n)
  if $n = 1$ **then do** qualcosa
    else
      for $i = 1$ **to** n **do**
        bar(i)
        fa qualcosa di differente

A. T(n)=T(n-1)+T(n-2) con T(1)=c

B. T(n)=T(n-1)+ Θ(n) con T(1)=c

C. T(n)=T(n-1)+T(n-2)+ Θ(n) con T(1)=c

D. C'è un bug nella procedura.

Relazione di ricorrenza per `ric-fact`

Determinare la relazione di ricorrenza per il tempo di esecuzione dell'algoritmo ricorsivo per il fattoriale

```
int ric-fact (int n)
     if (n==0) return 1
           else return n * ric-fact(n-1)
```

Dalla piattaforma

- (Relazioni di ricorrenza 2)

Nella risoluzione della relazione di ricorrenza

$T(n) = 2\ T(\ n/2) + n$, con $T(1) = c$

col metodo di iterazione, qual è il valore di $T(n)$ alla i-esima iterazione?

A. $T(n) = 2^i T(n/2^i) + n$
B. $T(n) = 2^i T(n/2^i) + 2^i\ n$
C. $T(n) = 2iT(n/2i) + c\ n$
D. Nessuno dei precedenti

Dalla piattaforma

- (Soluzione relazione di ricorrenza 1)

La soluzione della relazione di ricorrenza $\mathbf{T(n)=2T(n/2)+n^3}$ è

- (Soluzione relazione di ricorrenza 2) ***(vista ieri)***

La soluzione della relazione di ricorrenza $\mathbf{T(n)=T(n-1)+n}$ è

- (Soluzione relazione di ricorrenza 3)

La soluzione della relazione di ricorrenza $\mathbf{T(n) = 2T(n/2) + c}$ è:

- (Soluzione relazione di ricorrenza 4)

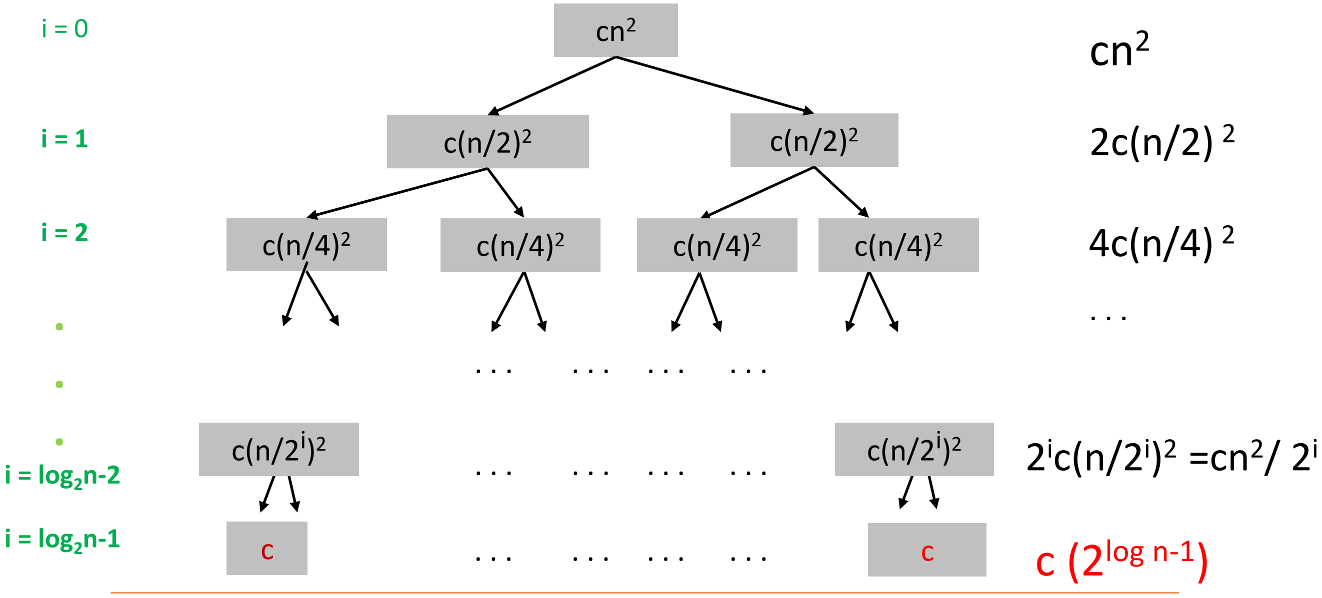
La soluzione della relazione di ricorrenza $\mathbf{T(n) = 4T(n/2) + n}$ è:

Albero di ricorsione

$T(n) = 2\,T(n/2) + cn^2$

$T(2) = c$

$n/2^i = 2$ sse $n = 2^{i+1}$ sse $i+1 = \log_2 n$ sse $i = \log_2 n - 1$ sse $2^i = n/2$



$$T(n) = c\,(2^{\log n - 1}) + \sum_{i=0}^{\log_2 n - 2} cn^2/\,2^i =$$

$$= c\,n/2 + c\,n^2 \sum_{i=0}^{\log_2 n - 2} (1/2)^i \leq c\,n/2 + 2\,c\,n^2$$

$$T(n) \geq c\,n^2$$

Quindi $T(n) = \Theta(n^2)$

Albero di ricorsione per $T(n)=2T(n/2)+cn^3$

$T(n) = 2\,T(n/2)+cn^3$
$T(2) = c$

$n/2^i = 2$ sse $n = 2^{i+1}$

sse $i+1 = \log_2 n$

sse $i = \log_2 n - 1$

sse $2^i = n/2$

$i = 0$ — cn^3

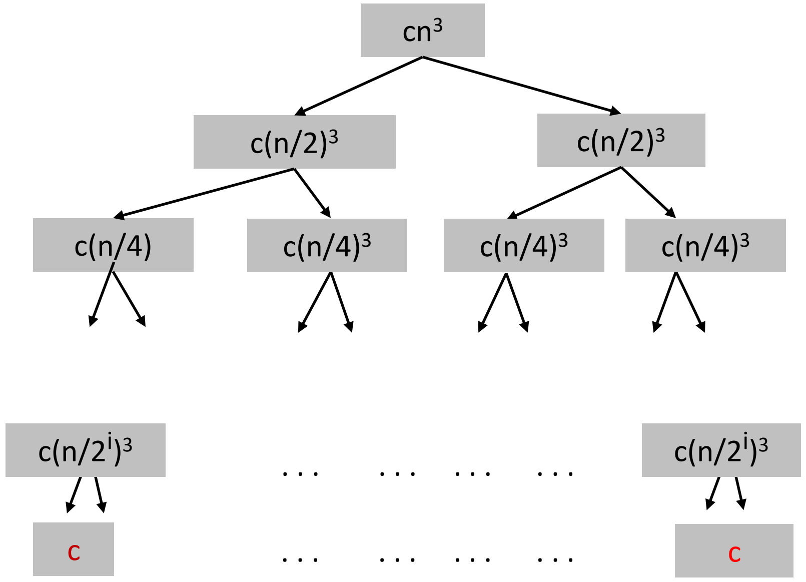
$i = 1$ — $2c(n/2)^3$

$i = 2$ — $4c(n/4)^3$

...

$i = \log_2 n-2$ — $2^i c(n/2^i)^3 = cn^3/(2^i)^2$

$i = \log_2 n-1$ — $c\,(2^{\log n-1})$



$$T(n) = c\,(2^{\log n - 1}) + \sum_{i=0}^{\log_2 n - 2} cn^3/4^i =$$

$$= c\,n/2 + c\,n^3 \sum_{i=0}^{\log_2 n - 2} (1/4)^i \le c\,n/2 + 4/3\,c\,n^3$$

$$T(n) \ge c\,n^3$$

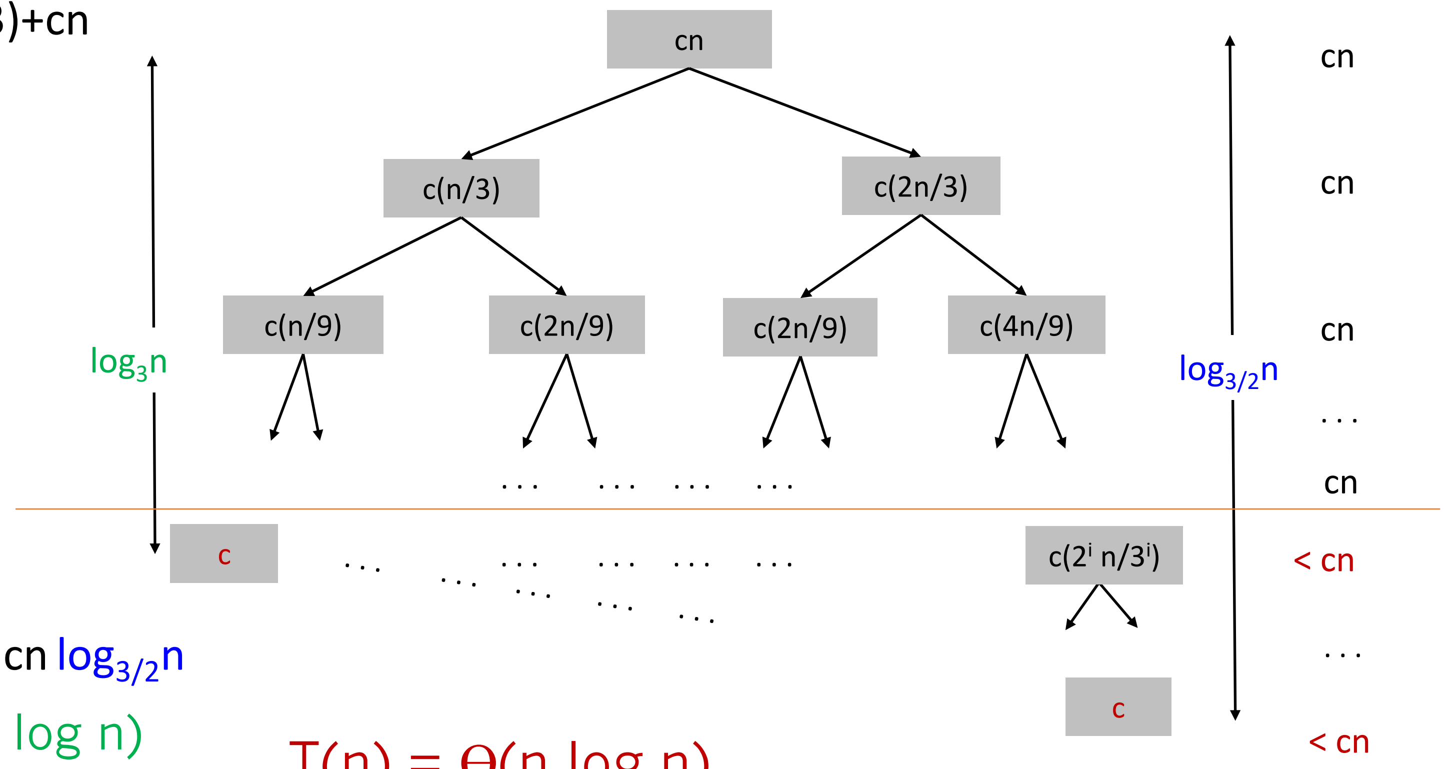
Quindi $T(n) = \Theta(n^3)$

Albero di ricorsione «sbilanciato»

$T(n)=T(n/3)+T(2n/3)+cn$
$T(1)=c$

$n/3^i=1$ sse
$n=3^i$ sse
$i=\log_3 n$

$n\,(2/3)^i=1$ sse
$n=(3/2)^i$ sse
$i=\log_{3/2} n$



$cn \log_3 n \leq T(n) \leq cn \log_{3/2} n$

$T(n) = \Omega(n \log n)$

$T(n) = O(n \log n)$

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

Provate a farlo con unrolling!!!

Massimo con D&I

Descrivere ed analizzare un algoritmo **basato sulla tecnica Divide et Impera** che dato un array A[1, ..., n] di interi ne restituisca il massimo.

Esercizi da svolgere

(possibilmente sulla piattaforma)

(Soluzione relazione di ricorrenza 3)

La soluzione della relazione di ricorrenza **$T(n) = 2T(n/2) + c$** è:

(Soluzione relazione di ricorrenza 4)

La soluzione della relazione di ricorrenza **$T(n) = 4T(n/2) + n$** è:

T(n)=T($\sqrt{n}$)+1 con T(2)=1

Relazione di ricorrenza 1 (soluzione)

- Risolvere la seguente relazione di ricorrenza con 2 diversi metodi di risoluzione, nell'ipotesi che n sia una potenza di 2.

$$T(1)=a$$
$$T(n)=T(n/2)+c$$

- Cosa potete dire della soluzione nel caso generale che n non sia necessariamente una potenza di 2?

Relazione di ricorrenza 2 (soluzione)

- Si consideri la seguente relazione di ricorrenza.

$$T(0) = 1$$
$$T(1) = 3$$
$$T(n) = T(n - 2) + n$$

Quanto valgono T(6) e T(9)?

- Risolvere la relazione di ricorrenza con tutti i metodi possibili.

Ricerca ternaria (D&I)

- Progettare un algoritmo per la ricerca di un elemento **key** in un array ordinato **A[1..n]**, basato sulla tecnica **Divide-et-impera** che nella prima fase divide l'array in **3** parti «uguali» (le 3 parti differiranno di al più 1 elemento).
- **Scrivere** la relazione di ricorrenza per il tempo di esecuzione dell'algoritmo proposto. Potete supporre che n sia una potenza di 3.

- **Risolvere** la relazione di ricorrenza. (*fra qualche lezione*)

- **Confrontare** il tempo di esecuzione ottenuto con quello della ricerca binaria. (*fra qualche lezione*)

Ricerca del k-esimo minimo

a) Indicare le varie fasi in cui è suddiviso un algoritmo basato sulla tecnica Divide et Impera.

b) Descrivere un algoritmo basato sulla tecnica Divide et Impera che, dati:

un array A[1..n] di caratteri distinti del nostro alfabeto {a, b, c, … , z}, nell'ordine usuale a < b <c < … < z, e un intero k, con $1 \leq k \leq n$, calcoli il k-esimo minimo di A.

Tale algoritmo non dovrà fare alcun ricorso ad algoritmi di ordinamento, ma dovrà utilizzare la procedura Partition nella sua prima fase. Si potrà ottenere il massimo del punteggio solo se l'algoritmo è descritto tramite pseudo-codice. In ogni caso, è necessario spiegare verbalmente il funzionamento dell'algoritmo proposto e giustificarne la correttezza.

Esempio: se A[1..7] = [e, h, b, a, d, f, g] e k = 3 l'algoritmo dovrà restituire d.

c) Analizzare la complessità di tempo nel caso peggiore dell'algoritmo proposto al punto b).

Occorrenze consecutive di 2 (D&I) (dalla piattaforma)

Si scriva lo pseudo-codice di un algoritmo ricorsivo basato sulla tecnica Divide et Impera che prende in input un array di interi positivi e restituisce il massimo numero di occorrenze **consecutive** del numero ‘2’.

Ad esempio, se l'array contiene la sequenza <2 2 3 6 2 2 2 2 3 3> allora l'algoritmo restituisce 4. Occorre specificare l'input e l'output dell’algoritmo.