Relazioni di ricorrenza Esercitazione

22 marzo 2022

Altre relazioni di ricorrenza

Consideriamo la sotto-famiglia

- $T(n) = qT(n/2) + cn$ con $T(2)=c$

  per $q=1$ allora $T(n)= \Theta(n)$

  $q=2$ allora $T(n)= \Theta (n \log_2 n)$

  $q>2$ allora $T(n)= O(n^{\log_2 q})$

- $T(n) = 2T(n/2) + cn^2$

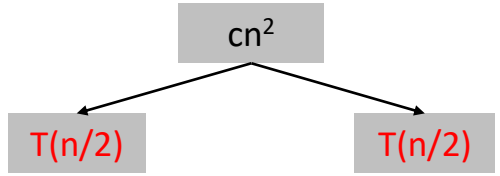
  $T(n) = ?$

Albero di ricorsione

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

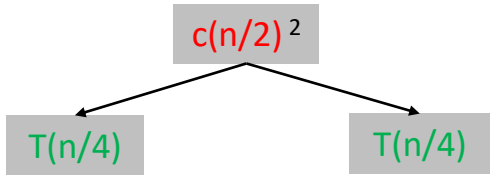
$T(2) = c$

Albero per T(n):

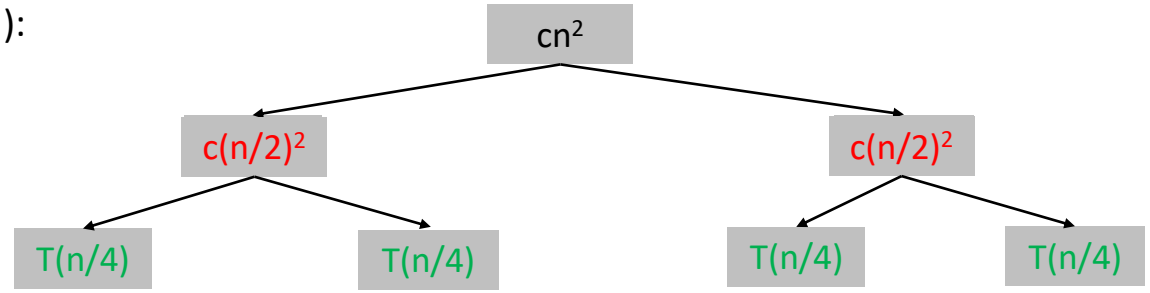


Albero per T(n/2):

$T(n/2) = 2\ T(n/4) + c\ (n/2)^2$



Albero per T(n):



Albero di ricorsione

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

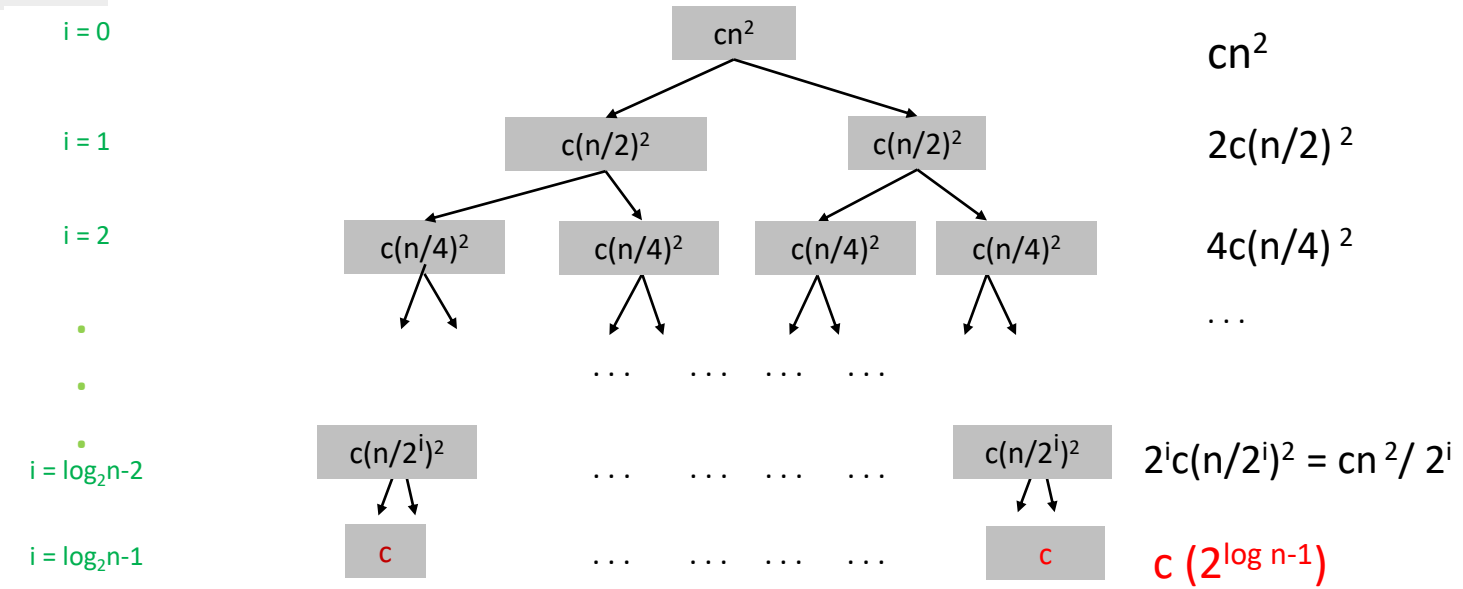
$T(2) = c$

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

sse $i+1 = \log_2 n$

sse $i = \log_2 n - 1$

sse $2^i = n/2$



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

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

(continua)

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

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

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

$$= c\,n/2 + 2\,c\,n^2$$

$$T(n) = O(n^2)$$

Inoltre dalla definizione $T(n) \geq cn^2$. Quindi $T(n) = \Theta(n^2)$.

Altre relazioni di ricorrenza

Abbiamo considerato una sotto-famiglia

- $T(n) = qT(n/2) + cn$ con $T(2)=c$

  per $q=1$ allora $T(n)= \Theta(n)$

  $q=2$ allora $T(n)= \Theta(n \log_2 n)$

  $q>2$ allora $T(n)= O(n^{\log_2 q})$

- $T(n)=2T(n/2) +cn^2$

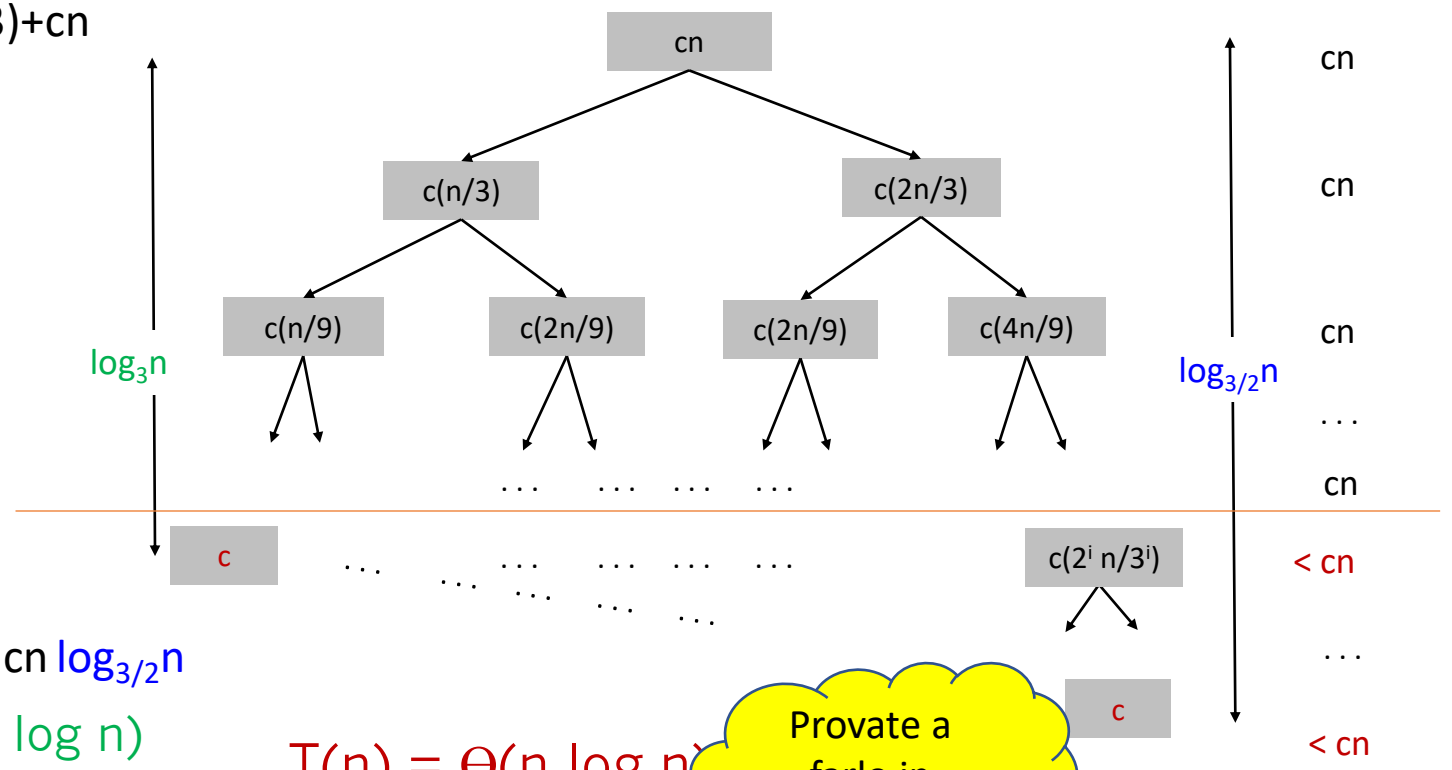
  $T(n)= \Theta(n^2)$

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
$i=\log_{3/2} n$



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

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

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

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

Provate a farlo in linea!!!

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

DEFINIRE PROBLEMA COMPUTAZIONALE (svolto)

Quesito (22 punti) (*Campi di calcetto*)
Dopo il successo del vostro primo campo di calcetto, avete aperto molti altri campi di calcetto, all'interno di un unico complesso.
Ogni giorno raccogliete le richieste per utilizzare i vostri campi, ognuna specificata da un orario di inizio e un orario di fine.
Oramai avete un numero di campi sufficiente ad accontentare sempre tutte le richieste. Volete però organizzare le partite nei campi in modo da accontentare tutti, senza che vi siano sovrapposizioni di orari, ma con il minimo numero possibile di campi (la manutenzione costa!).
Formalizzate il problema reale in un problema computazionale.

Esempi

(svolto)

```
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

(svolto)

```
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

(svolto)

```
procedure bugs(n)
    if n = 1 then do qualcosa
        else
            bugs(n − 1);
            bugs(n − 2);
            for i = 1 to n do qualcosa
```

Alcuni esercizi sulla tecnica del Divide et Impera.

1. a) Descrivere gli aspetti essenziali della tecnica Divide et Impera, utilizzando lo spazio designato.

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

2. Sia $V[1..n]$ un vettore ordinato di 0 e 1.

   Descrivere ed analizzare un algoritmo per determinare il numero di 0 presenti in $V[1..n]$ in tempo $O(\log n)$.

5. a) Fornire lo pseudocodice di un algoritmo ricorsivo che ordini un array $A[1..n]$ nel seguente modo: prima ordina ricorsivamente $A[1..n-1]$ e poi inserisce $A[n]$ nell'array ordinato $A[1..n-1]$.

   b) Analizzare la complessita' di tempo dell'algoritmo proposto al punto b).

6. Sia dato un vettore binario ordinato $A[1..n]$.

   (a) Progettare un algoritmo di complessita' $\Theta(n)$ nel caso peggiore, che conti il numero di occorrenze di 1 nel vettore A.

   (b) Progettare un algoritmo di complessita' $O(\log n)$, che conti il numero di occorrenze di 1 nel vettore A.

7. Sia dato un vettore ordinato $A[1..n]$ di interi distinti. Progettare un algoritmo che determini, in tempo $O(\log n)$, se esiste o meno un intero i tale che $A[i] = i$.

8. Descrivere ed analizzare un algoritmo basato sul paradigma *divide et impera* che dato un vettore *ordinato* $A[1..n]$ di interi strettamente positivi (cioe' per ogni $1 \leq i \leq n$, $A[i] \geq 1$), restituisca il numero di occorrenze di 1 nel vettore A. L'algoritmo deve avere complessita' di tempo $O(\log n)$.

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.