Corso di Perfezionamento

Divide et Impera, Algoritmi Ricorsivi e Ricorrenze

Maria Rita Di Berardini[1]

[1]Dipartimento di Matematica e Informatica
Università di Camerino

17 febbraio 2009

Outline

Tecniche di Programmazione

Tecniche di progettazione di algoritmi:

1. Divide et Impera
2. Programmazione Dinamica (PD for short)
3. Algoritmi golosi

Le ultime due tecniche sono più sofisticate rispetto al Divide et Impera, ma consento di risolve in maniera efficiente molti problemi computazionali

Vedremo come queste tecniche di programmazione vengono usate per risolvere **problemi di ottimizzazione**

Approccio Divide et Impera

L'approccio seguito da questa tecnica di progettazione consiste nel suddividere il problema in input in un certo numero di sottoproblemi di dimensione inferiore

Una volta risolti ricorsivamente i sottoproblemi così generati, si combinano le loro soluzioni per creare la soluzione al problema dato

Distinguiamo tre fasi principali: Divide, Impera e Combina

Determinazione del massimo

Definizione del problema: determinazione del massimo in una sequenza di n elementi

Approccio Divide et Impera:

- **Divide**: gli n elementi della sequenza vengono divisi in due sottosequenze di $\approx n/2$ elementi ciascuna
- **Impera**: si determina il massimo m_1 della prima sequenza ed il massimo m_2 della seconda
- **Combina**: ottiene il massimo della sequenza in input come $max\{m_1, m_2\}$

Algoritmo per la determinazione del massimo

$\textsc{Max}(A, i, j)$
  if $i < j$
    then $m \leftarrow \lfloor (i+j)/2 \rfloor$
      return $\max\{\textsc{Max}(A, i, m), \textsc{Max}(A, m+1, j)\}$

Complessità di $\textsc{Max}(A, 1, n)$ è descritta dalla ricorrenza

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

Algoritmo di ordinamento **MergeSort**

Le tre fasi possono essere così descritte

- **Divide**: gli n elementi della sequenza da ordinare vengono in due sottosequenze di $\approx n/2$ elementi ciascuna
- **Impera**: ordina, usando ricorsivamente il merge sort, le due sottosequenze
- **Combina**: fonde le due sottosequenze per produrre come risposta la sequenza ordinata

Algoritmo di ordinamento **MergeSort**

Il processo di suddivisione si ferma quando la sequenza da ordinare ha lunghezza 1

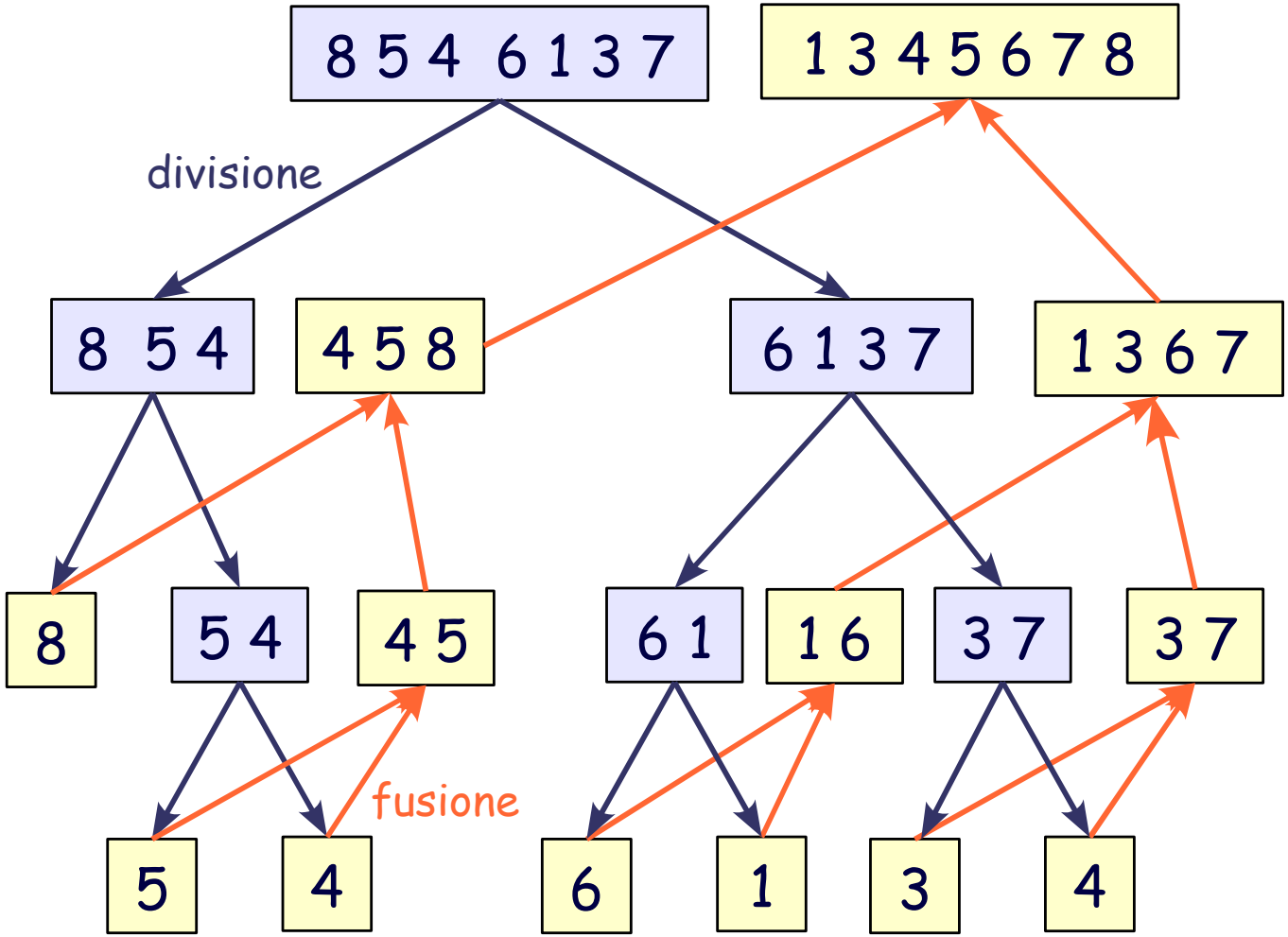
Il passo combina fonde le due sequenze utilizzando una procedura ausiliaria di fusione **merge**(A, p, q, r), dove: A è un array e p, q, r sono indici di elementi dell'array tali che $p < q < r$

merge(A, p, q, r) assume che $A[p, ..., q]$ e $A[q + 1, ..., r]$ siano ordinati e genera $A[p, ..., r]$ ordinato

Algoritmo di ordinamento **MergeSort**

MergeSort$(A, left, right)$
  if $left < right$
    then $mid = \lfloor (left + right)/2 \rfloor$
      MergeSort$(A, left, mid)$
      MergeSort$(A, mid + 1, right)$
      Merge$(A, left, mid, right)$

$A = \{8, 5, 4, 6, 1, 3, 7\}$


8 5 4 6 1 3 7
1 3 4 5 6 7 8
divisione
8 5 4
4 5 8
6 1 3 7
1 3 6 7
8
5 4
4 5
6 1
1 6
3 7
3 7
fusione
5
4
6
1
3
4

Fusione di due sottosequenze ordinate

Merge$(A, left, mid, right)$

$m_1 \leftarrow mid - left + 1$ ▷ dim di $A[left, ..., mid]$
$m_2 \leftarrow right - mid$ ▷ dim di $A[mid + 1, ..., right]$
$B[1, \ldots, m_1] \leftarrow A[left, ..., mid]$
$C[1, \ldots, m_2] \leftarrow A[mid + 1, ..., right]$
$i, j \leftarrow 1, \quad k \leftarrow left$
while $i \leq m_1$ and $j \leq m_2$
  do if $B[i] \leq C[j]$
    then $A[k] \leftarrow B[i]$
      $i \leftarrow i + 1$
    else $A[k] \leftarrow C[j]$
      $j \leftarrow j + 1$
  $k \leftarrow k + 1$
if $i \leq m_1$
  then $A[k, \ldots, right] \leftarrow B[i, ..., m_1]$
  else $A[k, \ldots, right] \leftarrow C[j, ..., m_2]$

Analisi della complessità del MergeSort

Il costo dell'algoritmo **mergeSort** è descritto dalla ricorrenza:

$$T(n) = \begin{cases} 2T(n/2) + f(n) & \text{se } n > 1 \\ 1 & \text{se } n = 1 \end{cases}$$

dove con $f(n) = n$ è il costo della procedura di fusione

Cosa è una relazione di ricorrenza

Una **relazione di ricorrenza** – o più semplicemente **ricorrenza** – è una equazione che descrive una funzione in termini del suo valore con input più piccoli

Esistono tre grandi metodi per risolvere le ricorrenze, ossia per ottenere dei limiti asintotici Θ o O

- Il metodo iterativo, o metodo della albero di ricorsione
- Il metodo della sostituzione
- Il metodo dell'esperto che consente di risolvere ricorrenze della forma $T(n) = aT(n/b) + f(n)$ dove $a \geq 1$, $b > 0$ ed $f(n)$ è una funzione data

Metodo iterativo

"Srotoliamo" la ricorsione fino ad ottenere una sommatoria dipendente da n — Esempio:

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

$$\begin{aligned} T(n) &= 1 + T(n/2) \\ &= 1 + 1 + T(n/4) \\ &= 1 + 1 + 1 + T(n/8) \\ &= 1 + 1 + 1 + 1 + T(n/16) \\ &= \dots \\ &= k + T(n/2^k) \end{aligned}$$

Metodo iterativo

$$\begin{aligned} T(n) &= 1 + T(n/2) \\ &= 1 + 1 + T(n/4) \\ &= 1 + 1 + 1 + T(n/8) \\ &= 1 + 1 + 1 + 1 + T(n/16) \\ &= \dots \\ &= k + T(n/2^k) \end{aligned}$$

Continuiamo a srotolare la ricorsione fin quando $n/2^k = 1$, i.e. quando $k = \log_2 n$. Allora:

$$T(n) = \log_2 n + 1 = O(\log_2 n)$$

Metodo iterativo: un altro esempio (meno semplice)

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

$$\begin{aligned} T(n) &= n + T(n/2) \\ &= n + n/2 + T(n/4) \\ &= n + n/2 + n/4 + T(n/8) \\ &= n + n/2 + n/4 + n/8 + T(n/16) \end{aligned}$$

Metodo iterativo: un altro esempio (meno semplice)

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

$$\begin{aligned} T(n) &= n + T(n/2) \\ &= n + n/2 + T(n/4) \\ &= n + n/2 + n/4 + T(n/8) \\ &= n + n/2 + n/4 + n/8 + T(n/16) \\ \\ &= n/2^0 + n/2^1 + n/2^2 + n/2^3 + T(n/2^4) \\ &= \cdots \\ &= \sum_{i=0}^{k-1} n/2^i + T(n/2^k) \end{aligned}$$

Metodo iterativo: un altro esempio (meno semplice)

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

Di nuovo, ci fermiamo quando $k = \log n$ — quindi:

$$\begin{aligned} T(n) &= n \sum_{i=0}^{\log n - 1} (1/2)^i + 1 \\ &= n \, \frac{1-(1/2)^{\log n}}{1-(1/2)} + 1 \\ &= 2n\left(1 - \frac{1}{2^{\log n}}\right) + 1 \\ &= 2n\left(1 - \frac{1}{n}\right) + 1 = 2n - 1 = \Theta(n) \end{aligned}$$

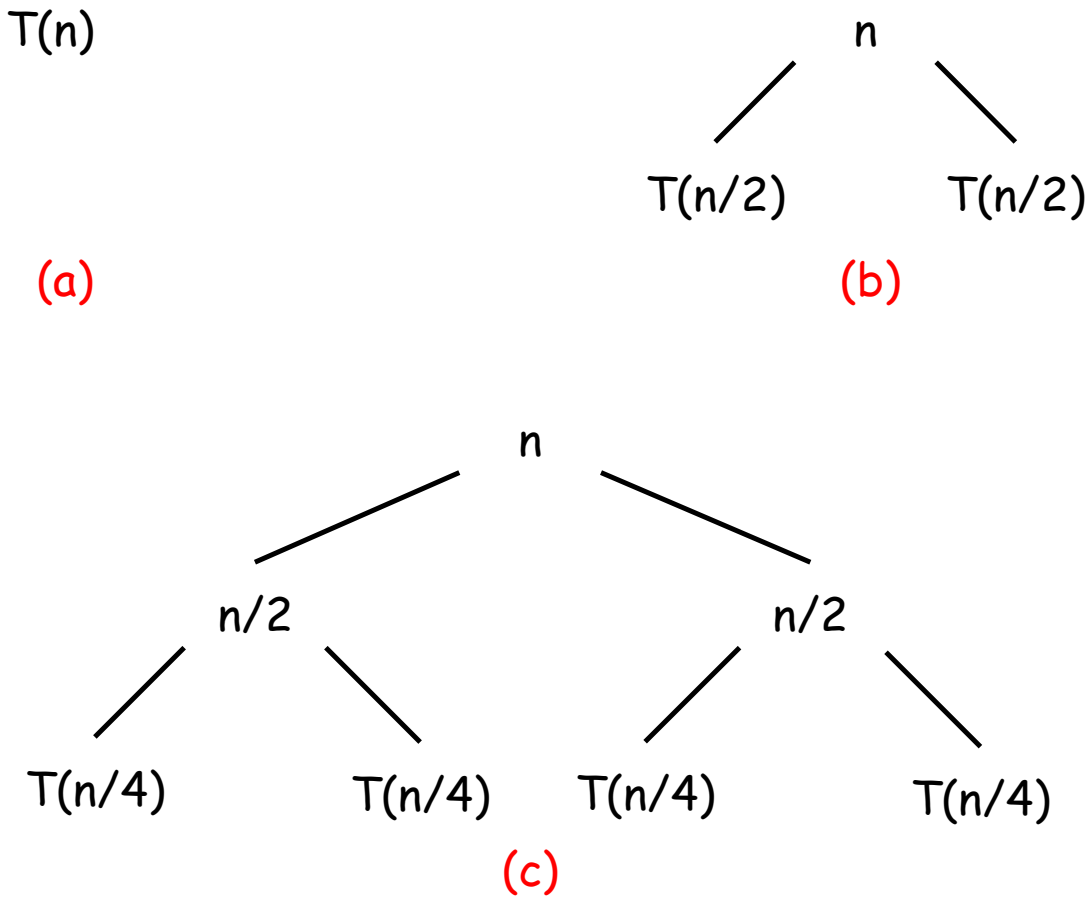
Metodo degli alberi di ricorsione

È una variante del metodo iterativo in cui usiamo degli alberi per rappresentare i costi delle varie chiamate ricorsive

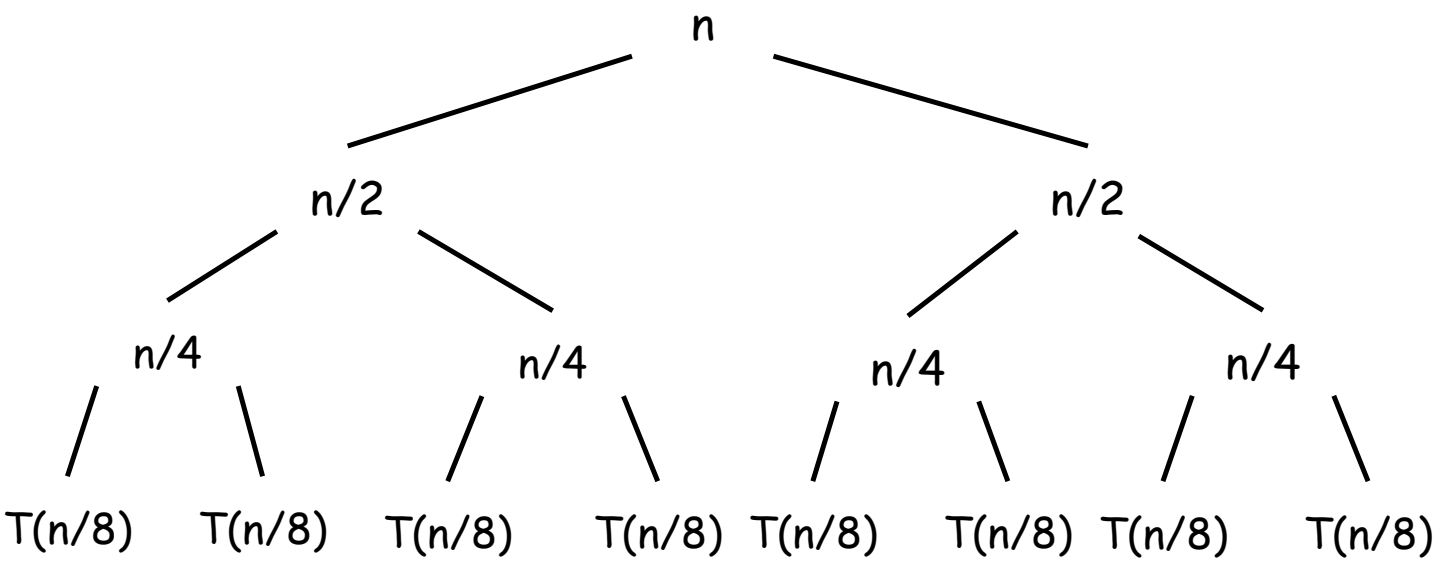
Esempio

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

Alberi di ricorsione: un esempio

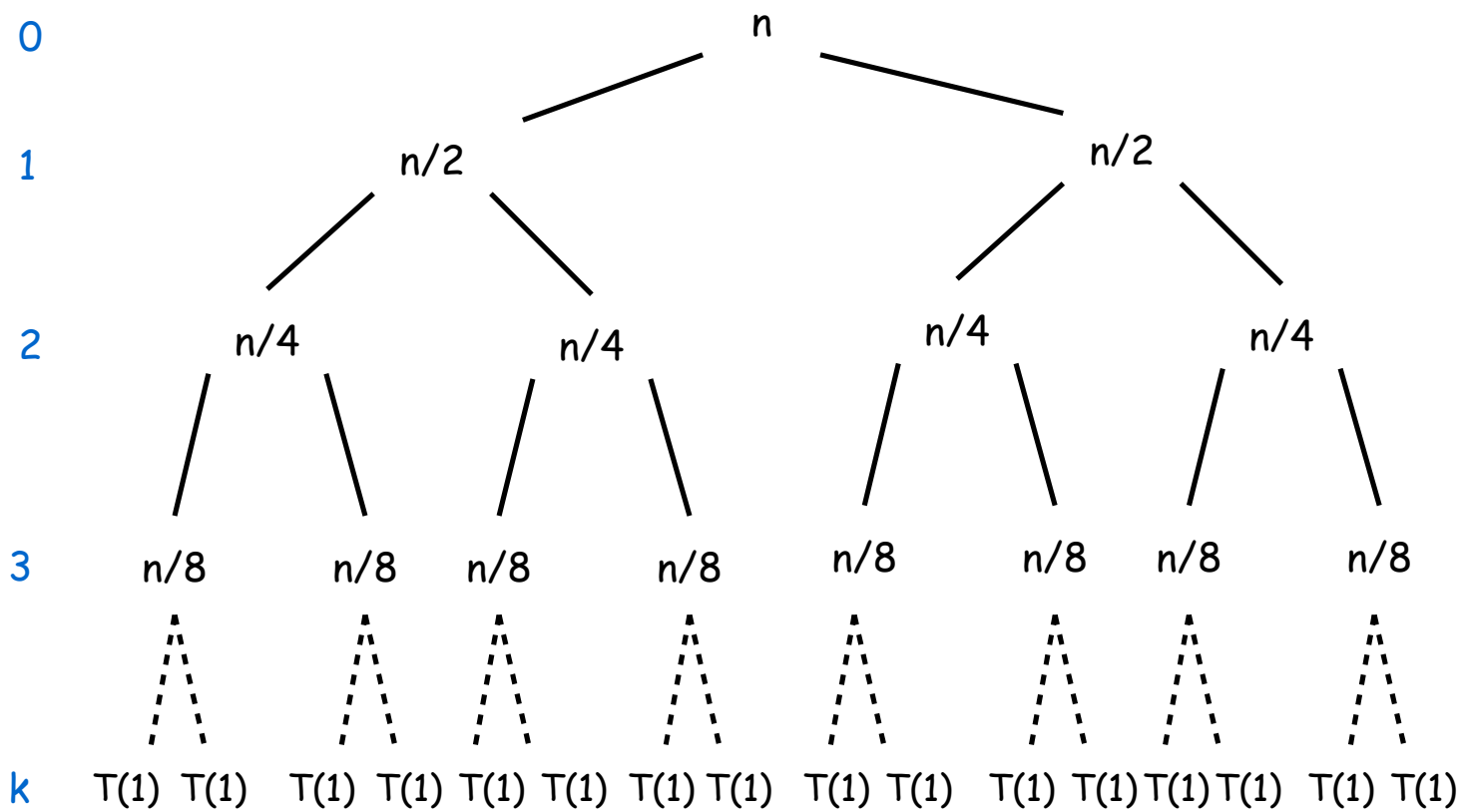


Alberi di ricorsione



(d)

Alberi di ricorsione



Alberi di ricorsione

Sia i un generico livello dell'albero di ricorsione — $i = 0, \ldots, k$

- Il livello i-esimo ha 2^i nodi
  (al livello 0 abbiamo $1 = 2^0$ nodi, ed ogni livello il doppio dei nodi del precedente)
- Ciascun nodo al livello i ha un costo pari a $n/2^i$
  (il nodo al livello 0 ha un costo pari a $n = n/1 = n/2^0$, e ogni volta che scendiamo di livello il costo dei nodi si dimezza)
- il costo complessivo dei nodi al livello i è

$$2^i \times n/2^i = n$$

Alberi di ricorsione

Quale è il costo complessivo della chiamata $T(n)$?

$$T(n) = \sum_{i=0}^{k} costo\_livello(i) = \sum_{i=0}^{k} n = n \cdot (k+1)$$

Ci rimane da determinare il valore di k

L'ultimo livello corrisponde ad una serie di chiamate di $T(1)$ (ci fermiamo quando la dimensione del problema è 1)

k è tale che $n/2^k = 1$, e quindi $k = \log n$

Ricapitolando

$$T(n) = n \cdot (k+1) = n(\log n + 1) = \Theta(n \log n)$$

Metodo della sostituzione

Metodo della sostituzione: "indovinare" una possibile soluzione ed usare l'induzione matematica per dimostrare che la soluzione è corretta

Consideriamo di nuovo la ricorrenza

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

e dimostriamo, applicando il metodo della sostituzione, che

$$T(n) = O(n)$$

Metodo della sostituzione

Dobbiamo dimostrare che che esistono delle costanti positive c ed n_0 tali che $0 \leq T(n) \leq cn$ per ogni $n \geq n_0$

Caso base $n = 1$: $T(1) = 1 \leq c1 = c$ per ogni costante $c \geq 1$ (positiva)

Passo induttivo: assumiamo $T(n') \leq cn'$ per ogni $n' < n$. Allora

$$\begin{aligned} T(n) &= n + T(n/2) \\ &\leq n + cn/2 \\ &= n(1 + c/2) \quad \text{se scegliamo } c \geq 2^1 \\ &\leq cn \end{aligned}$$

[1] Infatti se $c \geq 2$, allora $1 \leq c/2$ e $1 + c/2 \leq c/2 + c/2 = c$

Teorema dell'esperto – Teorema Master

Permette di analizzare algoritmi basati sulla tecnica del Divide et Impera:

- dividi il problema (di dimensione n) in a sotto-problemi di dimensione n/b
- risolvi i sotto-problemi ricorsivamente
- ricombina le soluzioni

Sia $f(n)$ il tempo per dividere e ricombinare istanze di dimensione n. La relazione di ricorrenza è data da:

$$T(n) = \begin{cases} aT(n/b) + f(n) & \text{se } n > 1 \\ 1 & \text{se } n = 1 \end{cases}$$

Teorema dell'esperto – Teorema Master

Si consideri la ricorrenza

$$T(n) = \begin{cases} aT(n/b) + f(n) & \text{se } n > 1 \\ 1 & \text{se } n = 1 \end{cases}$$

Distinguiamo tre casi:

1. $f(n) = O(n^{\log_b a - \varepsilon})$ per qualche costante $\varepsilon > 0$
   allora: $T(n) = \Theta(n^{\log_b a})$
2. $f(n) = \Theta(n^{\log_b a})$,
   allora: $T(n) = \Theta(n^{\log_b a} \log_2 n)$
3. $f(n) = \Omega(n^{\log_b a + \varepsilon})$ per qualche costante $\varepsilon > 0$ e $af(n/b) \leq cf(n)$, per qualche costante $c < 1$
   allora: $T(n) = \Theta(f(n))$

Applicazioni del teorema dell'esperto

$$T(n) = n + 2T(n/2)$$

$$a = b = 2,\ \log_b a = 1$$
$$f(n) = \Theta(n) = \Theta(n^{\log_b a})$$
(caso 2 del teorema master)

$$\Downarrow$$

$$T(n) = \Theta(n^{\log_b a} \log_2 n) = \Theta(n \log_2 n)$$

Applicazioni del teorema dell'esperto

$$T(n) = c + 9T(n/3)$$

$$a = 9,\ b = 3,\ \log_b a = \log_3 9 = 2$$
$$f(n) = c = O(n) = O(n^{\log_b a - \varepsilon}) = O(n^{2-\varepsilon}) \text{ con } \varepsilon = 1$$
(caso 1 del teorema master)

$$\Downarrow$$

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

Applicazioni del teorema dell'esperto

$$T(n) = n + 3T(n/9)$$

$$a = 3,\ b = 9,\ \log_b a = \log_9 3 = 1/2\ (3 = \sqrt{9} = 9^{1/2})$$
$$f(n) = n = \Omega(n) = \Omega(n^{\log_9 3+\varepsilon}) \text{ con } \varepsilon = 1/2$$
inoltre,
$$af(n/b) = 3f(n/9) = 3(n/9) = 1/3n \leq cf(n), \text{ per } c = 1/3$$
(caso 3 del teorema master)

$$\Downarrow$$

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