

Algoritmi di ordinamento ottimali

Problema dell' ordinamento per confronto:

Lower bound - $\Omega(n \log(n))$

considerazioni teoriche

Upper bound - $O(n^2)$

IS,SS,BS

Proviamo a costruire un algoritmo ottimale.

Notiamo che:

IS, BS e SS utilizzano un **approccio incrementale**

alla k -esima iterazione essi producono una sequenza ordinata di k elementi

Notare - E' facile capire che algoritmi basati su questo approccio hanno complessita' (nel caso medio e nel caso peggiore) $\Theta(n^2)$. Infatti:

- per ordinare tutta la sequenza ho bisogno di n iterazioni;
- ad ogni iterazione devo posizionare un nuovo elemento. Per fare cio' ho bisogno in media di $k/2$ confronti:

$$\sum_{k=1}^n \frac{k}{2} = \Theta(n^2)$$

L' approccio incrementale non e' l'unico approccio possibile:

Approccio divide-et-impera (divide-and-conquer)

- Il problema e' diviso in un certo numero di sotto-problemi (**divide**)
- I sottoproblemi vengono risolti separatamente (**impera**);
- Le soluzioni dei sottoproblemi vengono combinate per ottenere la soluzione del problema iniziale (**combina**).

Proviamo ...

Consideriamo il problema dell'ordinamento. Confrontiamo l'algoritmo:

Insertion-sort(A)

Con l'algoritmo:

Divide1(A)

$n = \text{length}[A]$

$q = \lfloor (1+n)/2 \rfloor$

Insertion-sort(A,1,q)

Insertion-sort(A,q+1,n)

Merge(A,1,q,n)

$\lfloor x \rfloor = \text{max intero} \leq x$

→ ordina sotto-sequenza $A[1.....q]$

→ ordina sotto-sequenza $A[q+1.....n]$

→ Fusione di due sequenze ordinate

Consideriamo che:

Insertion-Sort $\rightarrow T_{is}(n) = \Theta(n^2) \rightarrow T_{is}(n) \approx a n^2$ (asintot.)
Merge $\rightarrow T_m(n) = \Theta(n) \rightarrow T_m(n) \approx b n$ (asintot.)

Quando dividiamo la sequenza in due sottosequenze, dal punto di vista del tempo di esecuzione dell'algoritmo:

- **Miglioriamo** : Per ordinare, utilizzando l'Insertion Sort, 2 sequenze di $n/2$ elementi e' necessario meno tempo che per ordinare 1 sequenza di n elementi:

$$\Delta T_1 = 2 T_{is}(n/2) - T_{is}(n) = 2 a (n/2)^2 - a n^2 = - (a/2) n^2$$

- **Peggioriamo**: E' necessario utilizzare l'algoritmo merge per combinare le due sotto-sequenze ordinate:

$$\Delta T_2 = b n$$

Chi vince?

$$\Delta T = \Delta T_1 + \Delta T_2 = - (a/2) n^2 + b n$$

→ Se n e' sufficientemente grande l'algoritmo **Divide1** e' piu' vantaggioso dell'algoritmo di **IS**.

Ma allora, perche' non continuare?

Divide2(A)

```
n = length[A]
```

$$q = \lfloor (1+n)/2 \rfloor$$

```

Insertion-sort(A,1,q)    →    q2 =  $\lfloor (1+q)/2 \rfloor$ 
                           Insertion-sort(A,1,q2)
                           Insertion-sort(A,q2,q)
                           Merge(A,1,q2,q)
Insertion-sort(A,q+1,n) →    q3 =  $\lfloor (q+1+n)/2 \rfloor$ 
                           Insertion-sort(A,q+1,q3)
                           Insertion-sort(A,q3,n)
                           Merge(A,q+1,q3,n)

```

Merge(A,1,q,n)

Notare

I sottoproblemi sono identici al problema iniziale (cambia solo la dimensione dell'input).

L'approccio divide-et-impera puo' essere iterato.

Cio' puo' essere fatto efficacemente mediante l'uso di procedure ricorsive:

Ricorsione

Una procedura che chiama se stessa, direttamente o indirettamente e' detta ricorsiva.

Algoritmo merge-sort

Merge-Sort(A, p, r)

If ($p < r$)

then {
 $q = \lfloor (p+r)/2 \rfloor$
 Merge-Sort(A, p, q)
 Merge-Sort($A, q+1, r$)
 Merge(A, p, q, r)

Merge(A, p, q, r) \rightarrow

Assume che:

$A[p \dots q]$ ordinata

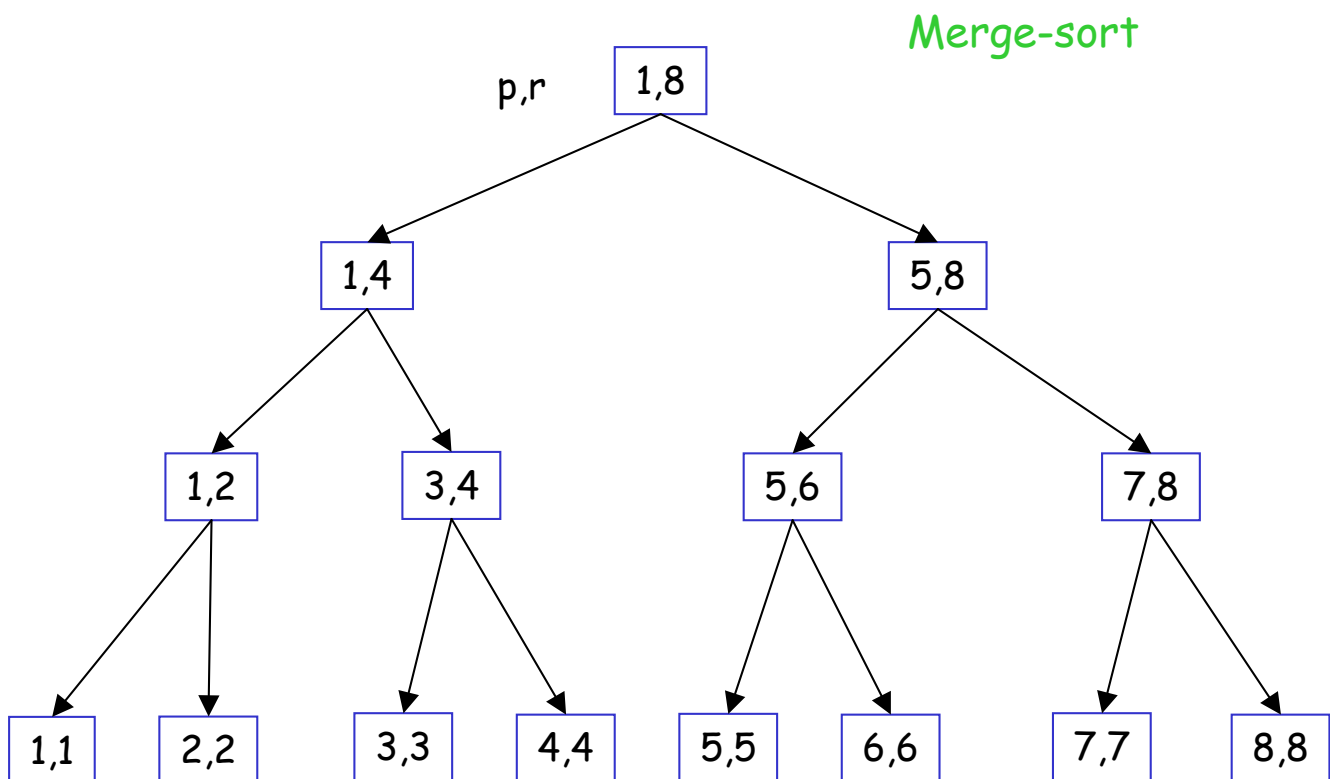
$A[q+1 \dots r]$ ordinata

Genera:

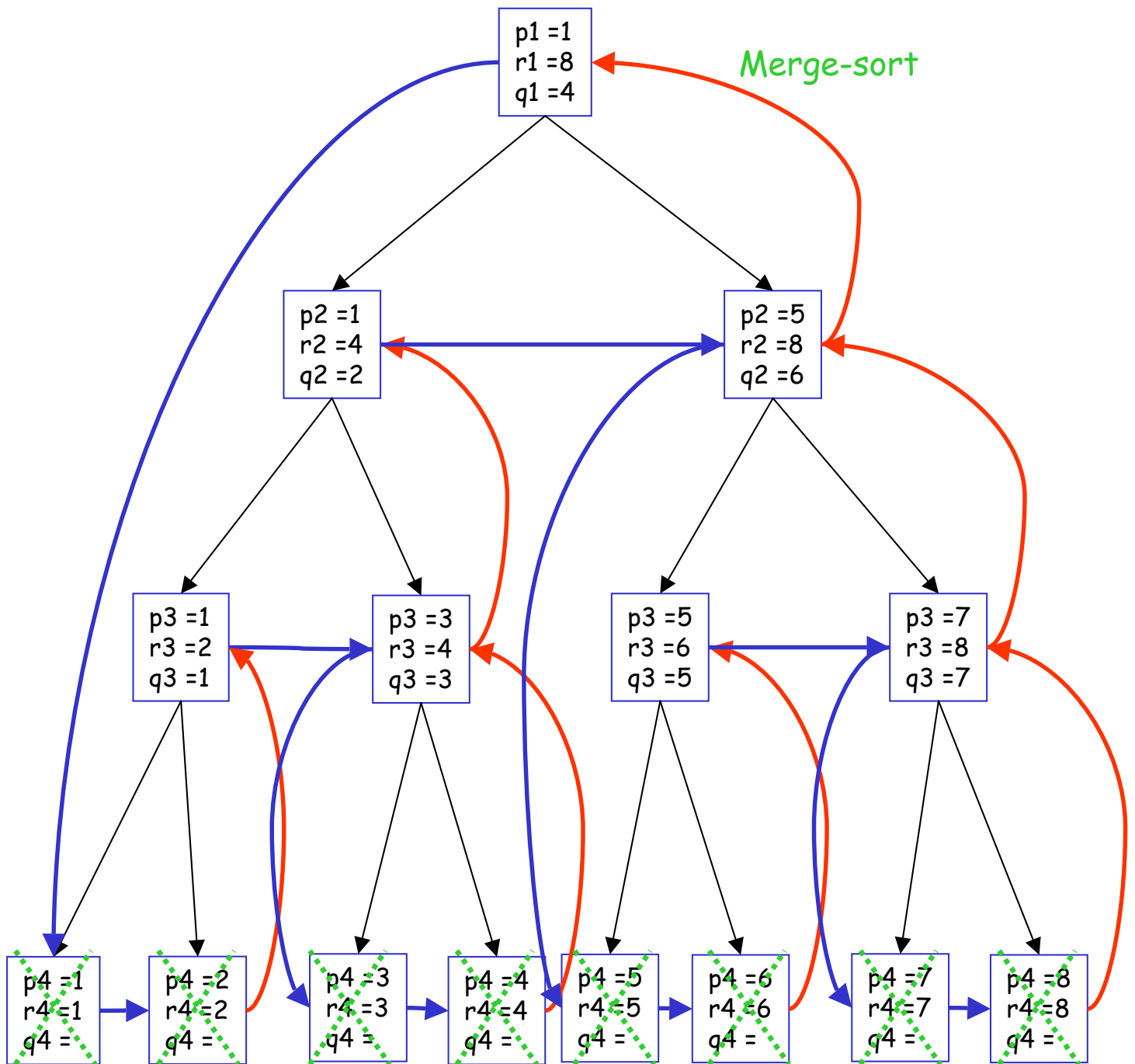
$A[p \dots r]$ ordinata

Per ordinare A si lancia Merge-Sort($A, 1, n$)

Funzionamento del merge-sort - Valore dei parametri p, r



Funzionamento del merge-sort - Progressione delle chiamate



Algoritmo merge-sort

Merge-Sort(A, p, r)

If ($p < r$)

then {
 $q = \lfloor (p+r)/2 \rfloor$
 Merge-Sort(A, p, q)
 Merge-Sort($A, q+1, r$)
 Merge(A, p, q, r)

Merge(A, p, q, r) \rightarrow

Assume che:

$A[p \dots q]$ ordinata

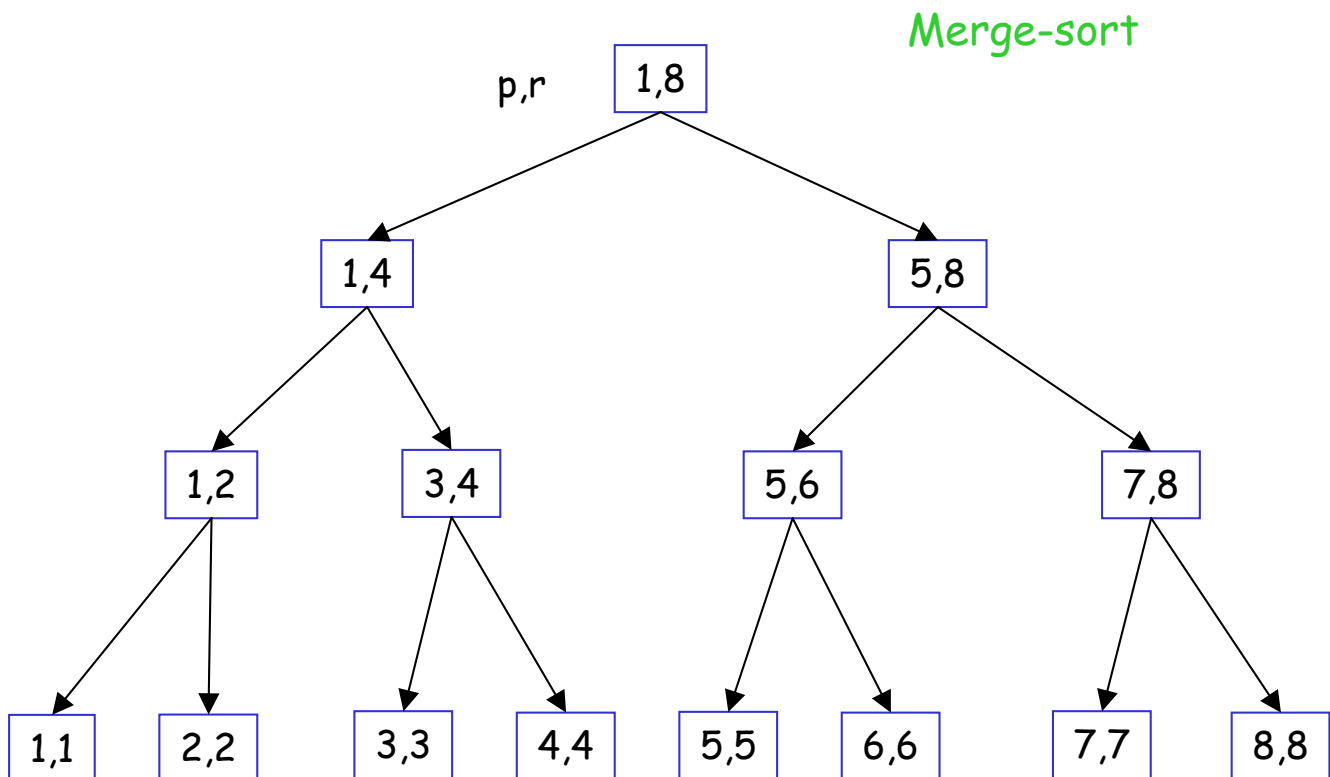
$A[q+1 \dots r]$ ordinata

Genera:

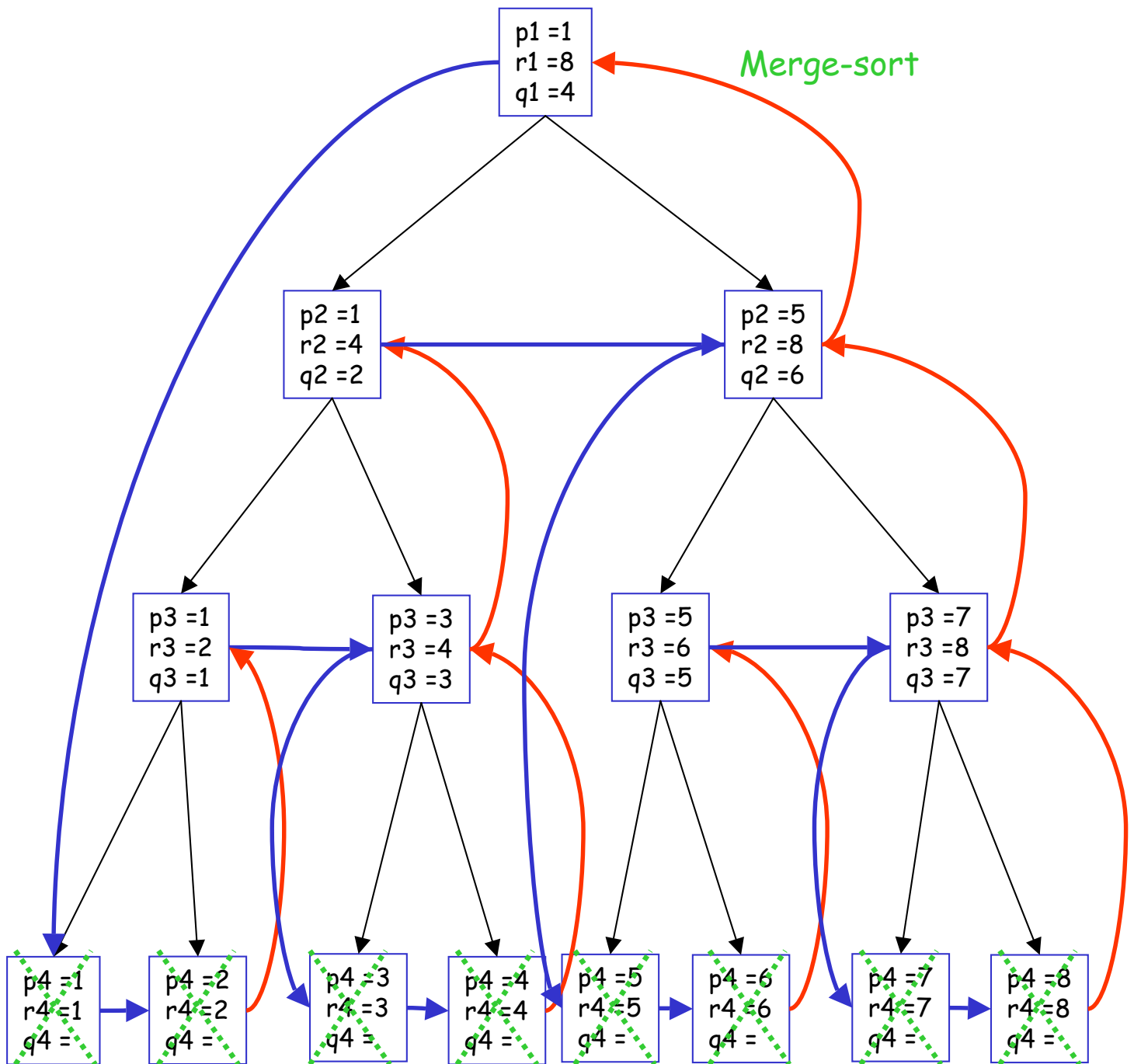
$A[p \dots r]$ ordinata

Per ordinare A si lancia Merge-Sort($A, 1, n$)

Funzionamento del merge-sort - Valore dei parametri p, r



Funzionamento del merge-sort - Progressione delle chiamate



Funzionamento del merge sort - Un esempio

$n = 8$
 $A = \langle 5, 2, 4, 6, 1, 3, 8, 7 \rangle$

Merge-sort

