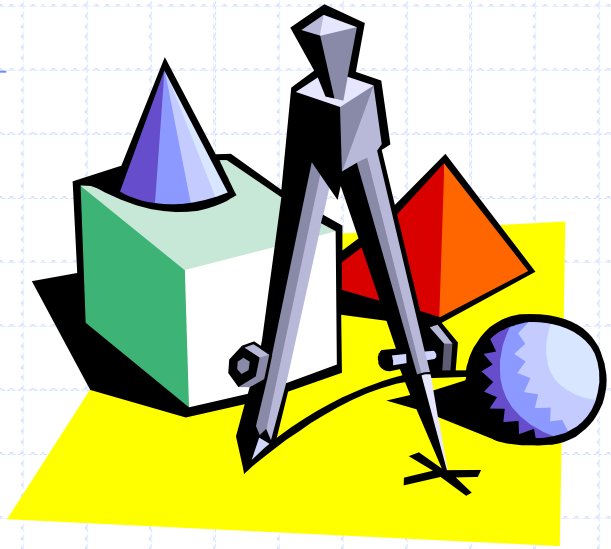
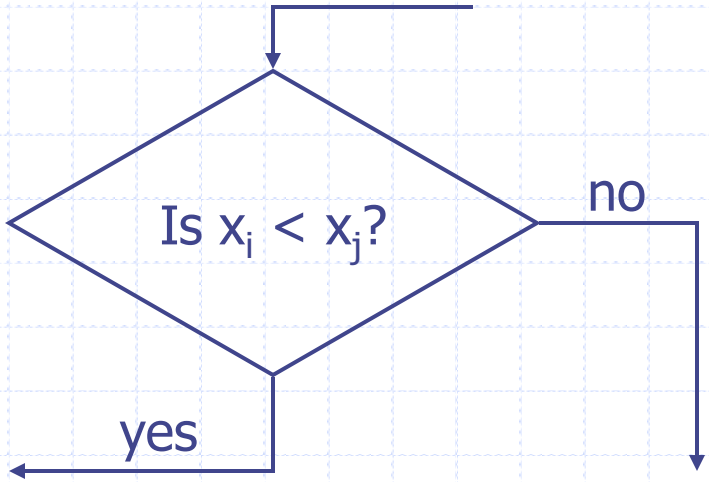
Lower Bound per l'ordinamento



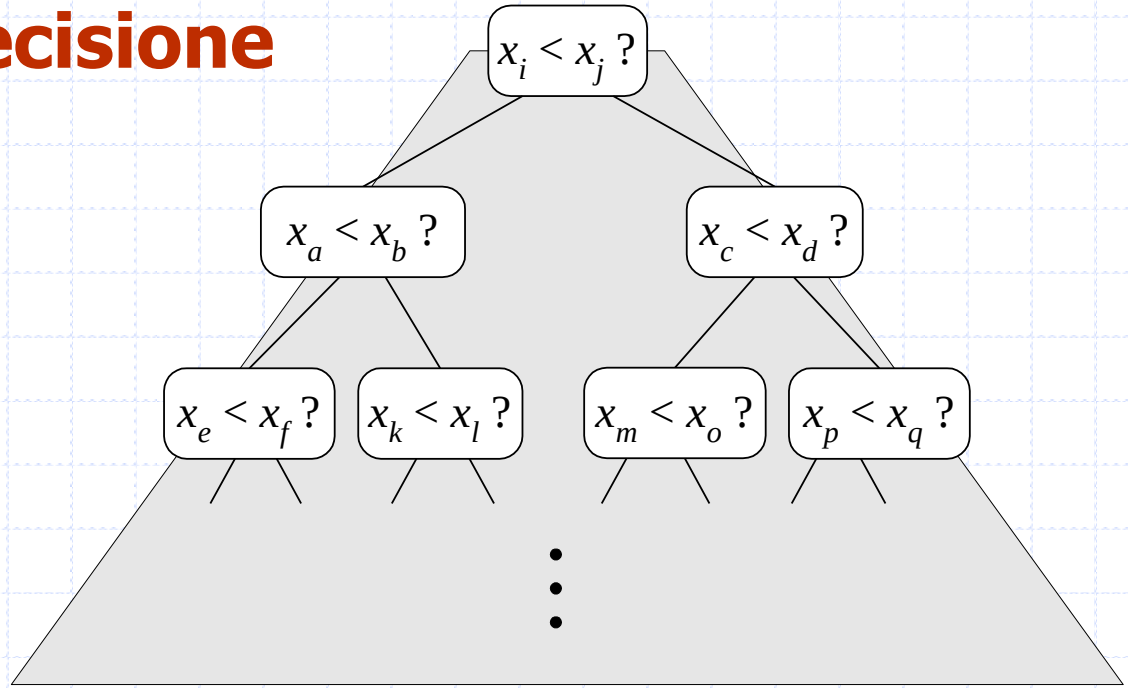
Ordinamento basato su confronti (§ 11.3)

- Molti algoritmi di ordinamento sono basati su confronti .
  - Ordinano attraverso confronti tra coppie di oggetti
  - Esempi: selection-sort, insertion-sort, heap-sort, merge-sort, quick-sort, ...
- Deriviamo un lower bound sul tempo di esecuzione di ogni algoritmo che usa confronti tra n elementi, x_1, x_2, ..., x_n.



Conteggio dei confronti

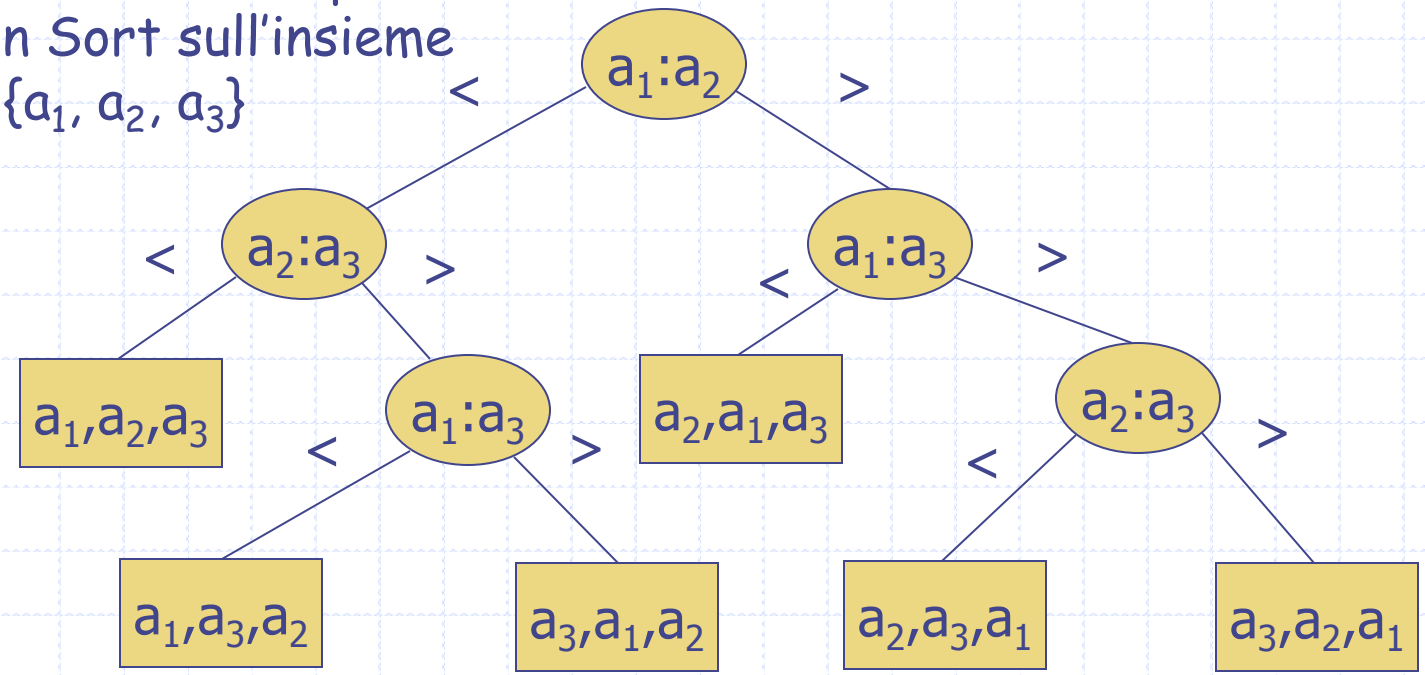
- Contiamo quindi il numero di confronti
- Ogni possibile esecuzione di un algoritmo corrisponde ad un cammino radice-foglia in un **albero di decisione**



© 2004 Goodrich, Tamassia

Alberi di decisione

- Albero di decisione per Insertion Sort sull'insieme $\{a_1, a_2, a_3\}$

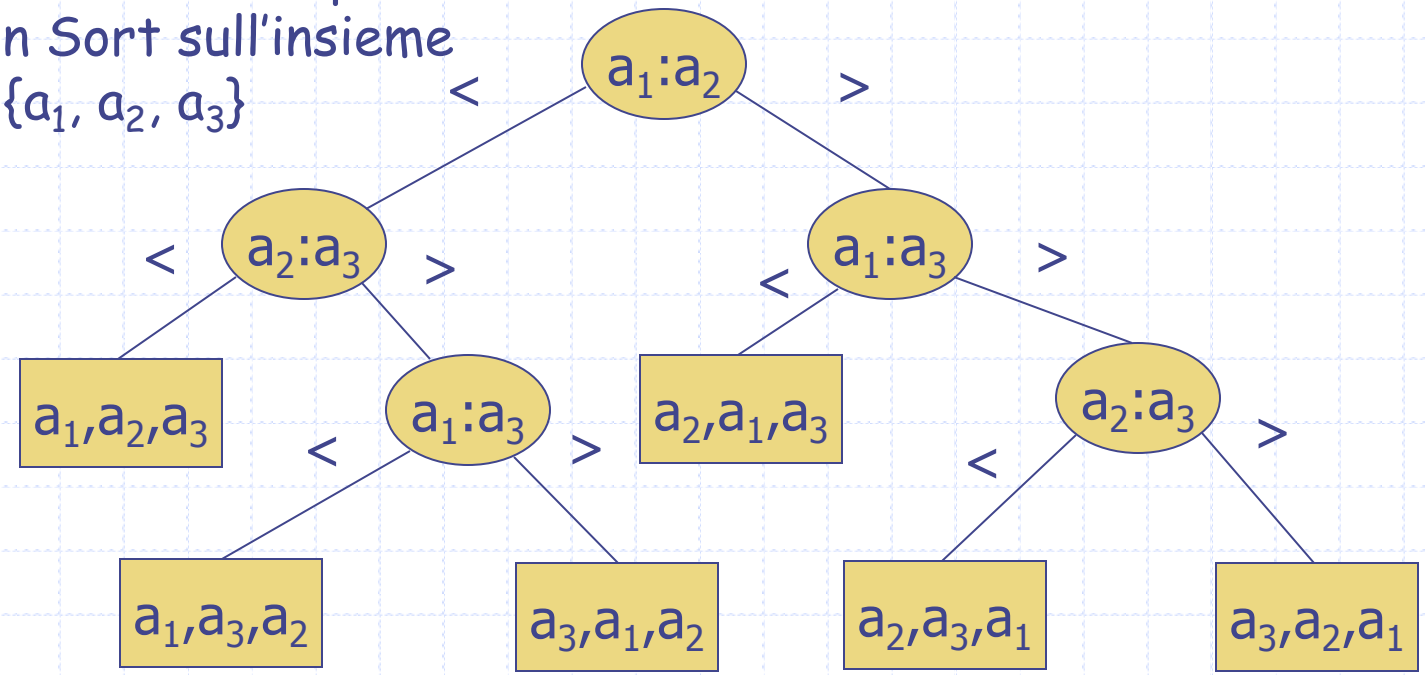


- Un albero di decisione rappresenta i confronti eseguiti da un algoritmo su un dato input
- Ogni foglia corrisponde ad una delle possibili permutazioni

© 2004 Goodrich, Tamassia

Alberi di decisione/2

- Albero di decisione per Insertion Sort sull'insieme $\{a_1, a_2, a_3\}$

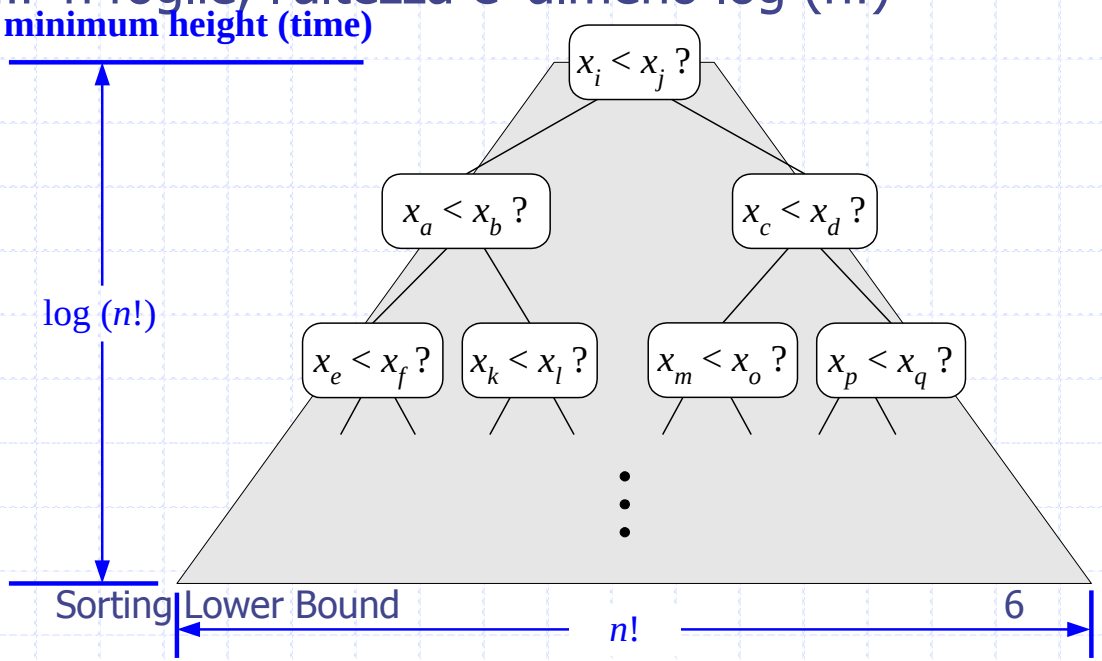


- Vi sono n! possibili permutazioni -> l'albero deve contenere n! foglie
- L'esecuzione di un algoritmo corrisponde ad un cammino sull'albero di decisione corrispondente all'input considerato

© 2004 Goodrich, Tamassia

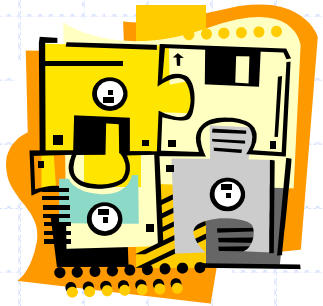
Altezza dell'albero di decisione

- L'altezza di un albero di decisione e' un limite inferiore sul tempo di esecuzione
- Ogni possibile paermutazione dell'input deve portare ad una diversa foglia di output.
  - Se no, un input …4…5… avrebbero lo stesso ordinamento di un altro input …5…4…, che sarebbe sbagliato.
- Poiche' vi sono n!=1*2*…*n foglie, l'altezza e' almeno log (n!)



© 2004 Goodrich, Tamassia

Il Lower Bound



- Qualunque algoritmo di ordinamento basato su confronti richiede almeno tempo log (n!)
- Quindi, qualunque algoritmo di questo tipo richiede tempo almeno

$$\log\ (n!) \geq \log \left(\frac{n}{2}\right)^{\frac{n}{2}} = (n/2)\log\ (n/2).$$

- Cioe', ogni algoritmo di ordinamento basato su confronti deve eseguire in tempo Ω(n log n).