

Complessita' intrinseca di un problema

Continuiamo a discutere il problema dell'ordinamento:

Ordinamento (Sorting)

INPUT: Sequenza di n numeri $\langle a_1, a_2, \dots, a_n \rangle$

OUTPUT: Permutazione

$\pi \langle a_1, a_2, \dots, a_n \rangle = \langle a'_1, a'_2, \dots, a'_n \rangle$
tale che $a'_1 \leq a'_2 \leq \dots \leq a'_n$

Abbiamo introdotto 3 algoritmi (IS, SS, BS) che risolvono il problema dell'ordinamento:

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

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

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

Ci chiediamo - Si puo' fare di meglio?

Piu' precisamente - E' possibile risolvere il problema dell'ordinamento mediante algoritmi aventi un comportamento asintotico migliore?

Notare - Per rispondere alla domanda, dobbiamo "alzare" il livello della nostra analisi. Dobbiamo discutere le **proprieta' generali** del problema, indipendentemente dagli algoritmi specifici utilizzati per risolverlo.

Prima di continuare

Quando forniamo limiti asintotici superiori/inferiori alla complessita' di un algoritmo senza specificare le proprieta' dell' input, ci riferiamo implicitamente al caso peggiore:

$T(n) = O(f(n))$ → Per **tutti** gli di input $T(n) = O(f(n))$
→ $T_{\text{peggiore}}(n) = O(f(n))$

$T(n) = \Omega(f(n))$ → Esiste almeno **un** insieme di input per cui $T(n) = \Omega(f(n))$ → $T_{\text{peggiore}}(n) = \Omega(f(n))$

Complessita' intrinseca di un problema \neq
Complessita' di un algoritmo

Un problema computazionale ha **delimitazione superiore alla complessita' $O(f(n))$** se esiste **un** algoritmo per la sua risoluzione con delimitazione superiore $O(f(n))$.

Un problema computazionale ha **delimitazione inferiore alla complessita' $\Omega(f(n))$** se **tutti** gli algoritmi per la sua risoluzione hanno delimitazione inferiore $\Omega(f(n))$.

Se dimostro che un problema ha delimitazione inferiore $\Omega(f(n))$ e trovo un algoritmo avente complessita' $O(f(n))$ allora:

A meno di costanti, ho un algoritmo **ottimale** per risolvere il problema

Esempio di algoritmo ottimale \rightarrow Algoritmo per la ricerca del minimo

Problema dell'ordinamento

Sappiamo per ora che:

Lower bound - $\Omega(n)$ perche'?

Upper bound - $O(n^2)$ IS, BS, SS

Abbiamo un Gap lineare tra upper bound e lower bound.
Possiamo fare meglio

Limite inferiore per il problema dell'ordinamento

Ordinamento per confronti

Dati due elementi a_i ed a_j , per determinarne l'ordinamento relativo effettuiamo una delle seguenti operazioni di confronto:

$$a_i < a_j ; a_i \leq a_j ; a_i = a_j ; a_i \geq a_j ; a_i > a_j$$

Non si possono esaminare i valori degli elementi o ottenere informazioni sul loro ordine in altro modo.

Notare - Tutti gli algoritmi di ordinamento considerati fino ad ora sono algoritmi di ordinamento per confronto.

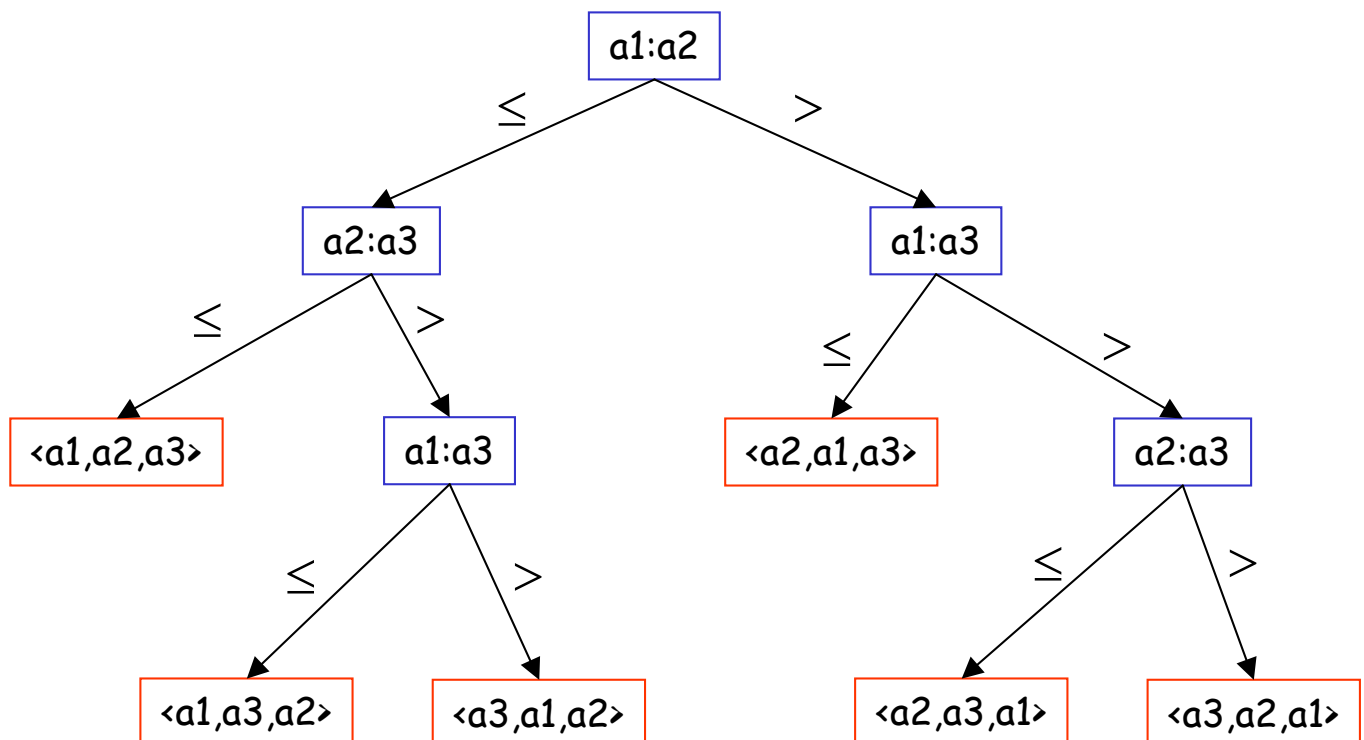
Gli algoritmi di ordinamento per confronto possono essere descritti in modo astratto in termini di **ALBERI DI DECISIONE**.

Un generico algoritmo di ordinamento per confronto lavora nel modo seguente:

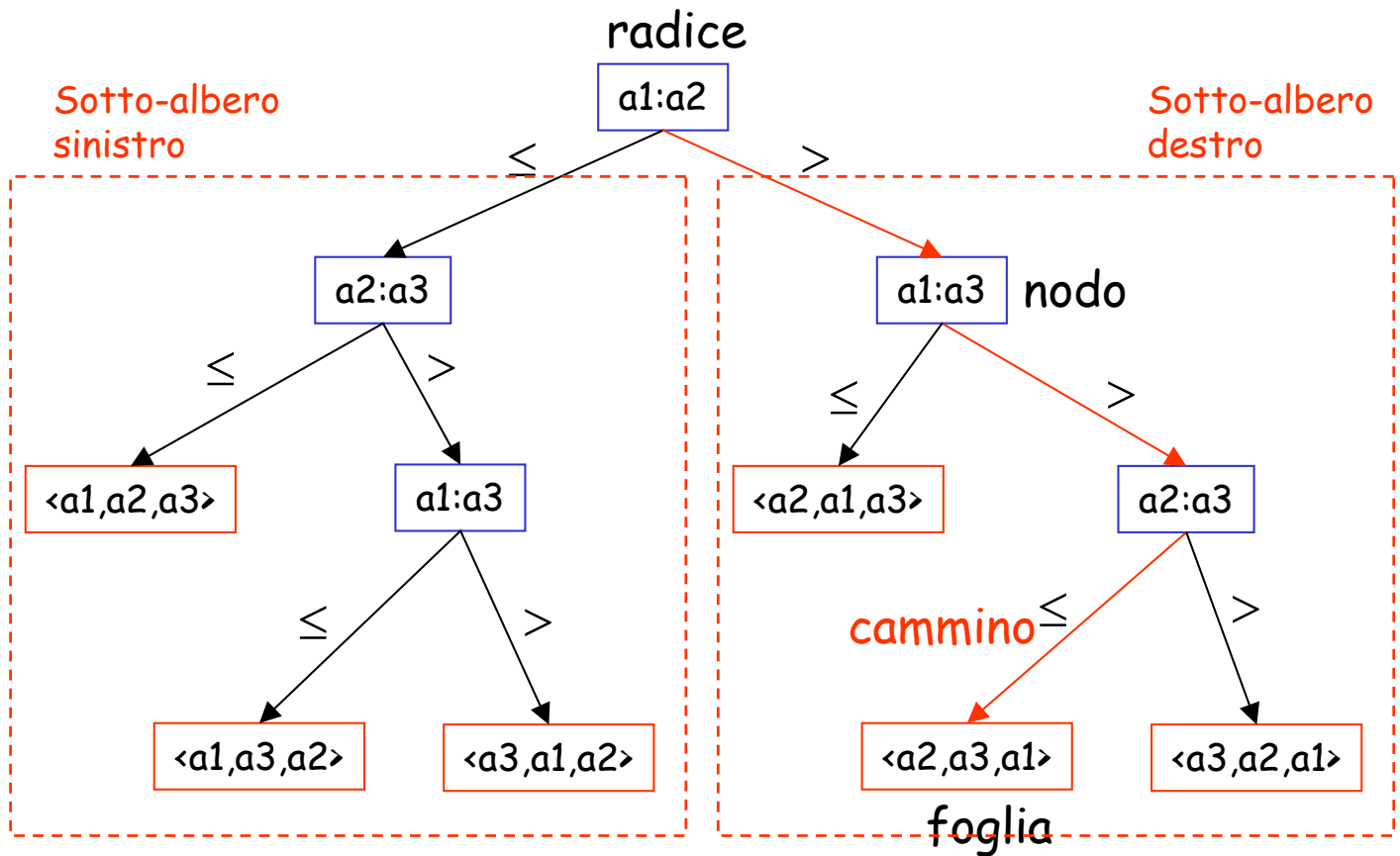
- Confronta due elementi a_i ed a_j (ad esempio effettua il test $a_i \leq a_j$);
- A seconda del risultato - riordina e/o decide il confronto successivo da eseguire.

Albero di decisione - Descrive i confronti che l'algoritmo esegue quando opera su un input di una determinata dimensione. I movimenti dei dati e tutti gli altri aspetti dell'algoritmo vengono ignorati

Albero di decisione dell'algoritmo Insertion Sort



Prima di discutere i dettagli, alcune definizioni



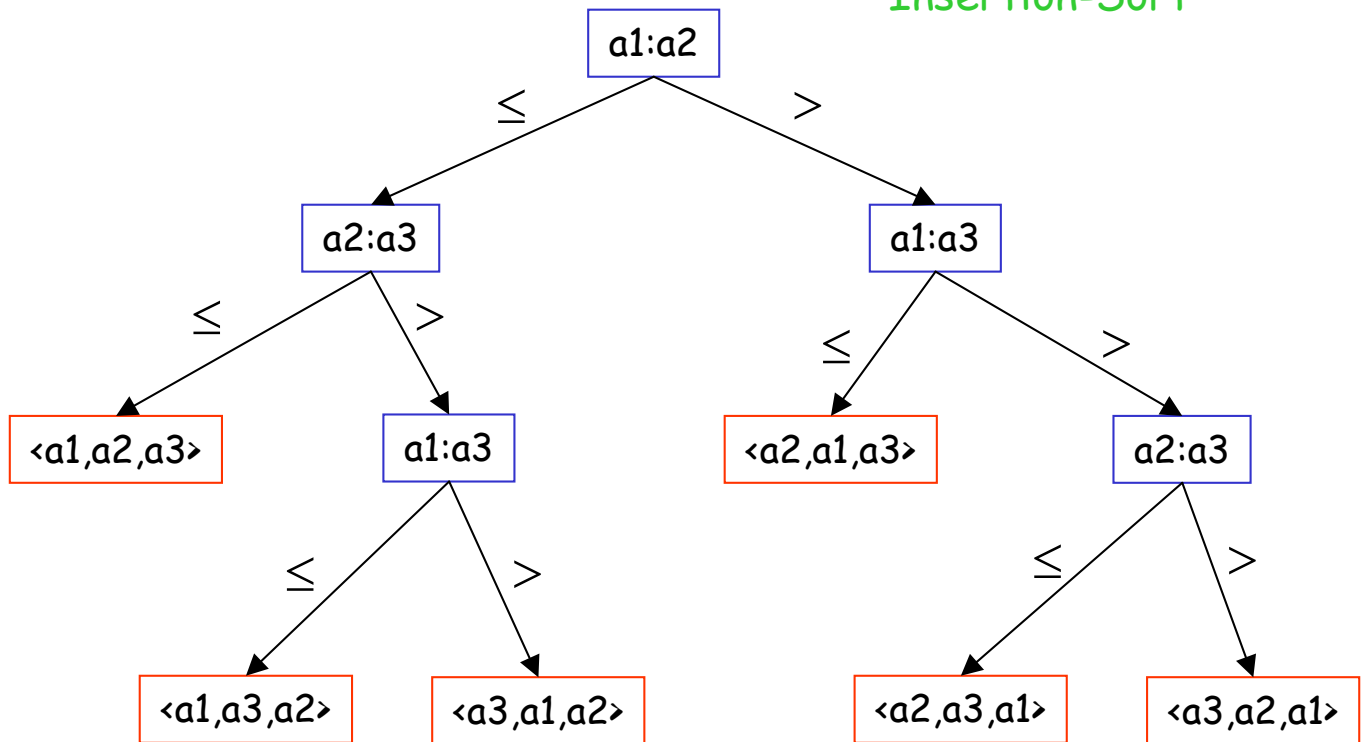
Profondita' di un nodo = lunghezza del cammino che lo congiunge alla radice.

Altezza di un albero = valore massimo della profondita' dei nodi.

Notare - $(\text{Num. Foglie}) \leq 2^h$
 h = altezza dell'albero

Torniamo al problema dell'ordinamento

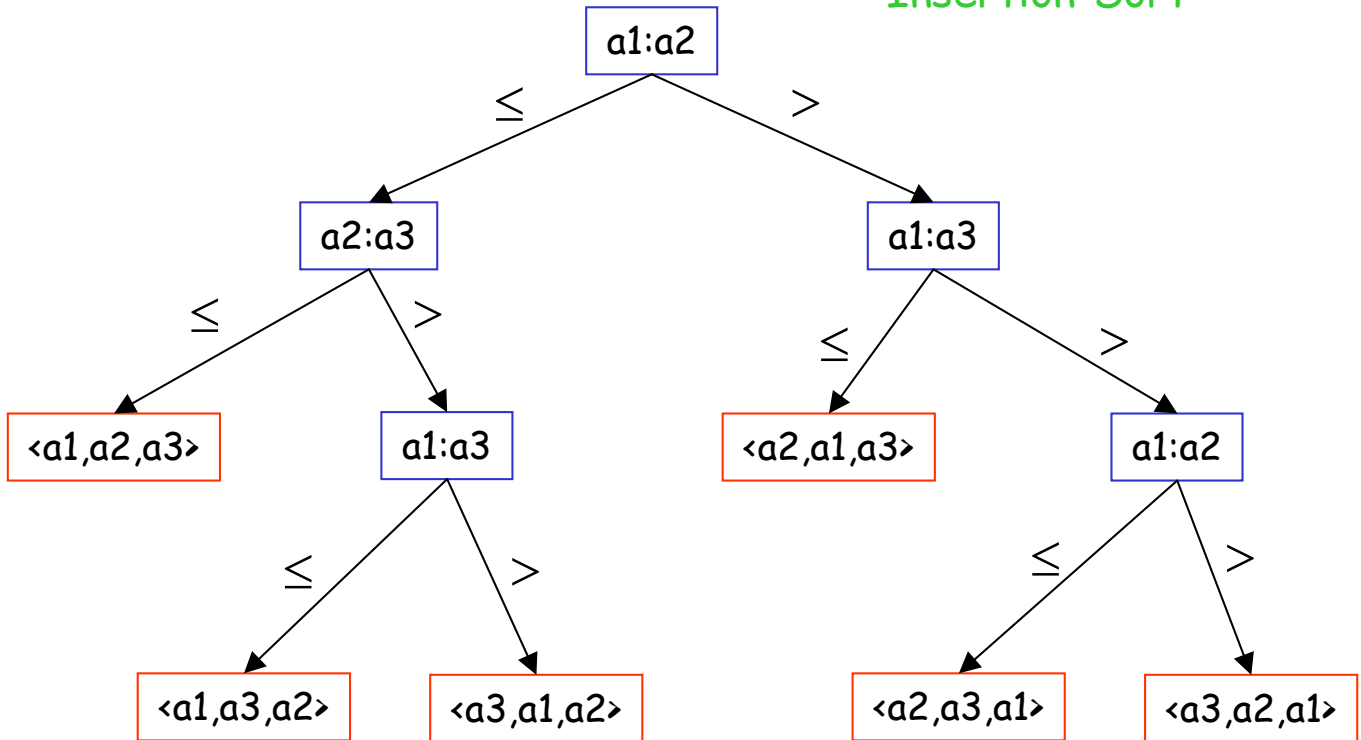
Insertion-Sort



- Ogni foglia e' etichettata con una **permutazione** della sequenza iniziale;
 - L'esecuzione dell' algoritmo corrisponde a tracciare un cammino dalla radice ad una foglia;
 - L'algoritmo segue un cammino diverso a seconda delle caratteristiche dell'input
- caso migliore?**
caso peggiore?
- L'**altezza** dell'albero fornisce il **numero di confronti** che l'algoritmo esegue nel **caso peggiore**.

Limite inferiore al problema dell'ordinamento per confronti

Insertion-Sort



Notiamo che:

- il numero di foglie dell'albero di decisione deve essere pari al **numero di possibili permutazioni** della sequenza iniziale:

$$(n.\text{foglie}) = n!$$

- Esiste una relazione tra il numero di foglie e l'altezza di un albero binario:

$$(n.\text{foglie}) \leq 2^h$$

Cio' implica:

$$2^h \geq n!$$

Da cio' segue: $h \geq \log_2(n!) > \log_2((n/e)^n) = \Omega(n \log_2(n))$

In conclusione:

Ricordando che l'altezza dell'albero di decisione indica il num.confronti che un generico algoritmo effettua nel caso peggiore, otteniamo:

$$\forall \text{ Algoritmo} \quad T(n) = \Omega(n \log_2(n))$$

Quindi -

Lower Bound = $\Omega(n \log_2(n))$
(problema dell'ordinamento per confronti)

Esercizi -

- 1) Alberi di decisione per SS e BS (3 elementi)
- 2) Ho scritto $T(n) = \Omega(n \log_2(n))$.
Implica $T(n) = \Omega(n \ln(n))$?
- 3) Considerare il problema della fusione di 2
sequenze ordinate di lunghezza $n/2$ in una
sequenza ordinata di lunghezza n . Scrivere un
algoritmo, analizzarne la complessita', valutare se
l'algoritmo scritto e' un algoritmo ottimale.