In questa lezione

Dimostrazione di un limite inferiore al problema dell'ordinamento, con il metodo degli alberi di decisione.

CLRS10 cap. 8, par 1

Limiti inferiori asintotici

Dimostrare che f(n) è un limite inferiore per un problema di dimensione n vuol dire dimostrare che ogni algoritmo, all'interno di un certo modello di computazione, ha un tempo di esecuzione nel caso peggiore di almeno f(n).

(si potrebbe fare anche per gli altri casi: migliore e medio).

In questo caso diciamo che la complessità del problema è in Ω(f(n)).

Limiti superiori asintotici

Dimostrare che f(n) è un limite superiore, sempre nel caso peggiore, per un problema di taglia n è molto più semplice: basta esibire un algoritmo di complessità O(f(n)).

Un algoritmo il cui andamento nel caso peggiore eguaglia il limite inferiore del problema che risolve è detto ottimale.

Limite superiore ordinamento

Un limite superiore per il problema dell'ordinamento, nel modello basato su confronti, è dato da $O(n \lg n)$, perché abbiamo visto un algoritmo di ordinamento con tempo di esecuzione $\Theta(n \lg n)$, nel caso peggiore.

Nel modello basato sui confronti è possibile usare solo confronti tra gli oggetti da ordinare, operazioni aritmetiche (somme o prodotti), logiche (and e or), o altro (shift) sono proibite

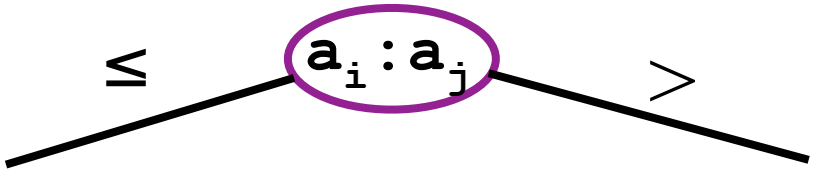
Alberi di decisione

Dimostreremo che $\Omega(n \lg n)$ è un limite inferiore al numero di confronti eseguiti da un algoritmo di ordinamento nel caso peggiore, nel modello basato sui confronti, utilizzando il metodo degli

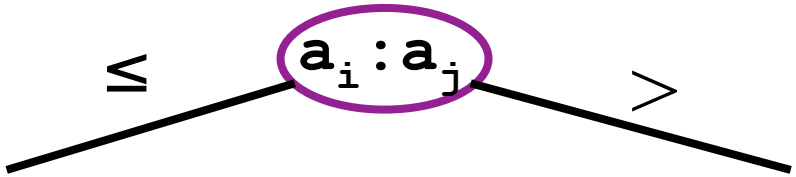
alberi di decisione.

Alberi di decisione: nodi interni

Un albero di decisione è un albero binario in cui ogni nodo interno corrisponde a un confronto tra due elementi, ogni nodo ha due figli corrispondenti ai due esiti del confronto: ≤ o >.



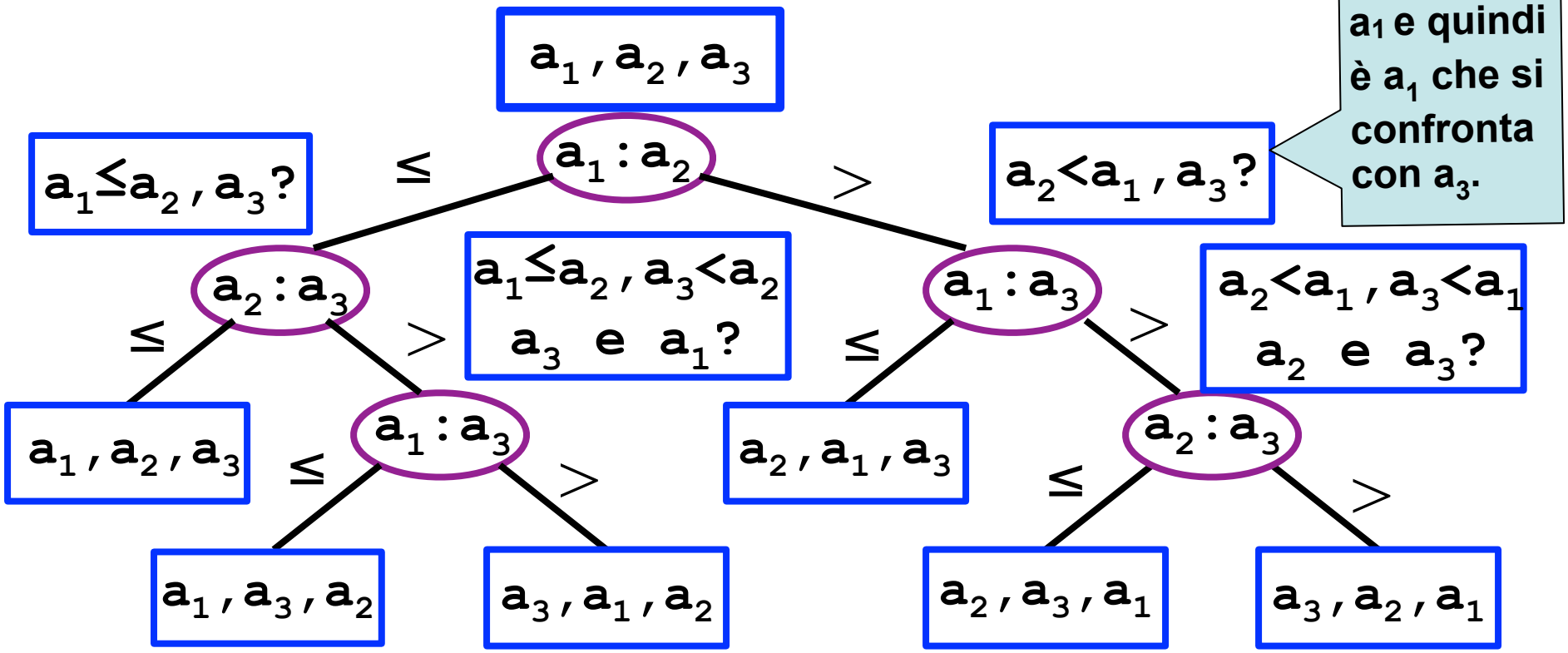
Alberi di decisione: solo confronti



un albero di decisione rappresenta un algoritmo di ordinamento che opera su un input di un data dimensione, astraendo da ogni altro aspetto dell'algoritmo come controlli, scambi,….

Esempio: InsertionSort

Albero delle decisioni su un array di 3 elementi. Confrontiamo gli elementi ignorando il fatto che sono stati spostati, nell'esecuzione dell'algoritmo



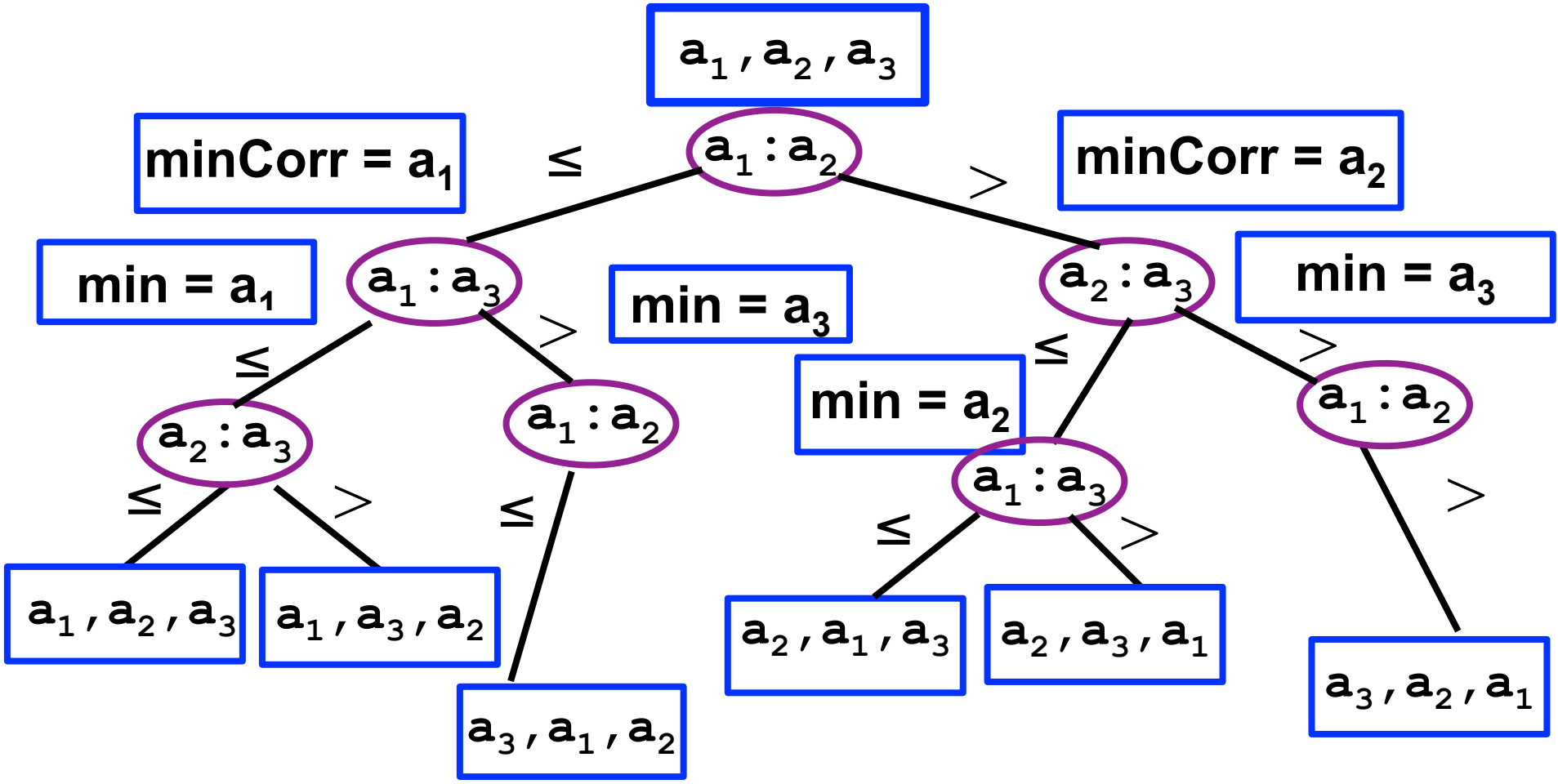
Alberi di decisione: cammini radice foglia

Un’esecuzione dell’algoritmo su una configurazione dei dati, della dimensione scelta, corrisponde a un cammino radice-foglia.
Quindi ogni cammino radice-foglia di profondità massima corrisponde a una configurazione di dati che dà il caso peggiore, in termini del numero dei confronti.

Alberi di decisione: le foglie

I possibili output di un algoritmo di ordinamento che opera su un input di dimensione n, sono le permutazioni della sequenza in input.
Quindi ogni permutazione è associata a una foglia dell'albero di decisione.

Esempio: SelectionSort



Alberi di decisione: proprietà

Ogni algoritmo di ordinamento A che opera su un input di una data dimensione può essere visto come un albero di decisione T_A.

In un albero di decisione T per un algoritmo di ordinamento su n elementi valgono due proprietà:

1. il numero delle foglie di T è maggiore o uguale a n!

Questo perchè le permutazioni di 1,2,...,*n* sono *n*!, e l'albero delle decisioni di un algoritmo qualunque di ordinamento deve avere almeno *n*! foglie per poter rappresentare tutte le possibili soluzioni.

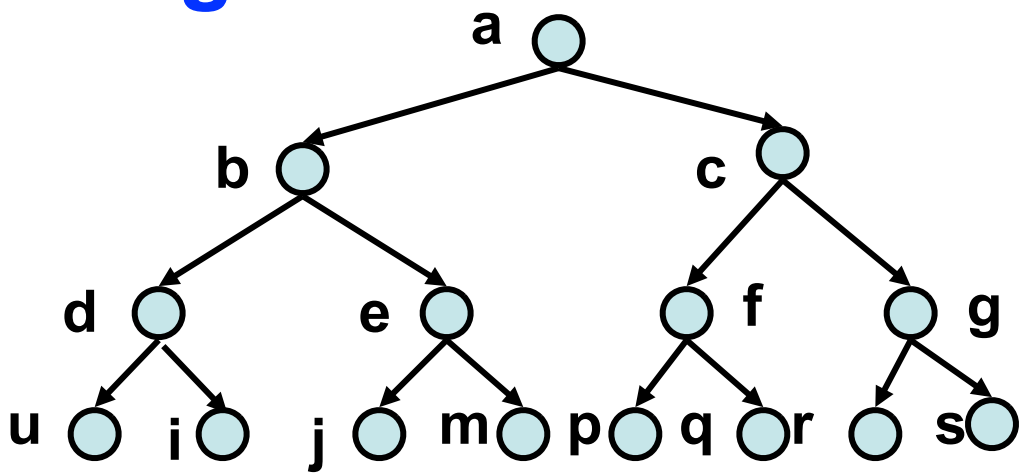
Alberi di decisione: proprietà

In un albero di decisione T per un algoritmo di ordinamento su n elementi valgono due proprietà:

2. l'altezza di T rappresenta il numero dei confronti in una computazione nel caso peggiore (ogni cammino radice foglia rappresenta un'esecuzione dell'algoritmo)

Altezza e numero di foglie

Un albero binario completo di altezza h ha 2^h foglie.



Un qualsiasi albero binario t di altezza h ha nFoglie(t) ≤ 2^h.

Albero di decisione ordinamenti

Sappiamo che in un albero binario di altezza h si ha

$$n! \leq n\text{Foglie}(t)) \leq 2^h \text{ quindi } n! \leq 2^h$$

Passando ai logaritmi otteniamo che nel caso peggiore l'algoritmo, su n elementi, deve eseguire almeno lg(n!) confronti, perchè

$$\lg (n!) \leq h,$$

Limite inferiore per lg(n!)

lg n! =
lg (n(n-1)…2) = lg (n(n-1)...n/2(n/2 -1)…2)

Eliminando la seconda metà, circa, dei termini del prodotto si ottiene un numero più piccolo
lg n! > lg (n(n-1)…n/2+1)
Ogni fattore tra n(n-1)…n/2+1 è maggiore di n/2 quindi
lg (n(n-1)…n/2+1) ≥ lg ((n/2)$^{n/2}$) = n/2 lg n/2

Quindi possiamo concludere che
lg n! = Ω(nlg n) e quindi h = Ω(nlg n)

Limite superiore per lg(n!)

lg n! =
lg (n(n-1)…2) ≤ lg (n^n) = nlg n

Quindi possiamo concludere che

$$\lg n! = \Theta(n \lg n)$$

Usando Stirling

In base all'approssimazione di Stirling possiamo dire che $n! > (n/e)^n$

Quindi $\lg n! > \lg (n/e)^n$

quindi $\lg n! > \lg (n/e)^n = n\lg (n/e) = n(\lg n - \lg e)$

e = 2,71828 …costante di Nepero

Quindi possiamo concludere che
$\lg n! = \Omega(n\lg n)$ e quindi $h = \Omega(n\lg n)$

$\Omega(n \lg n)$ per gli ordinamenti

Il limite inferiore $\Omega(n \lg n)$ vale per tutti gli algoritmi di ordinamento *generali*, cioè quelli per i quali non si fa alcuna ipotesi sugli elementi da ordinare e nei quali le uniche operazioni ammesse sono i confronti.

Riassumendo:

L'esecuzione di un qualsiasi algoritmo di ordinamento A basato su confronti su n elementi può essere vista come un albero di decisione T_A in cui vale che:

1. **il numero delle foglie di T_A è maggiore o uguale al numero dei possibili output, n!**
2. **l'altezza di T_A, h, appresenta il numero dei confronti in una computazione nel caso peggiore**
3. **l'altezza di T_A è limitata inferiormente dal $\lg(n)! = \Theta(n \lg n)$**
4. **$h = \Omega(n \lg n)$**

Complessità ordinamenti

1. **Il problema ha complessità $\Omega(n \lg n)$**

3. Esiste un algoritmo che risolve il problema in $\theta(n \lg n)$, nel caso peggiore?

Sì, abbiamo visto il mergeSort iterativo. Vedremo altri ordinamenti ottimali.

L'InsertionSort, il SelectionSort e il BubbleSort non sono ottimali.