In questa lezione

Alberi di decisione nella determinazione di limiti inferiori ai problemi

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 per un problema di taglia n è molto più semplice: basta esibire un algoritmo di complessità O(f(n)).

Un limite superiore per il problema dell'ordinamento, nel modello basato su confronti, è dato da $O(n^2)$, perché abbiamo visto algoritmi di ordinamento con tempo di esecuzione $\Theta(n^2)$, nel caso peggiore.

Alberi di decisione

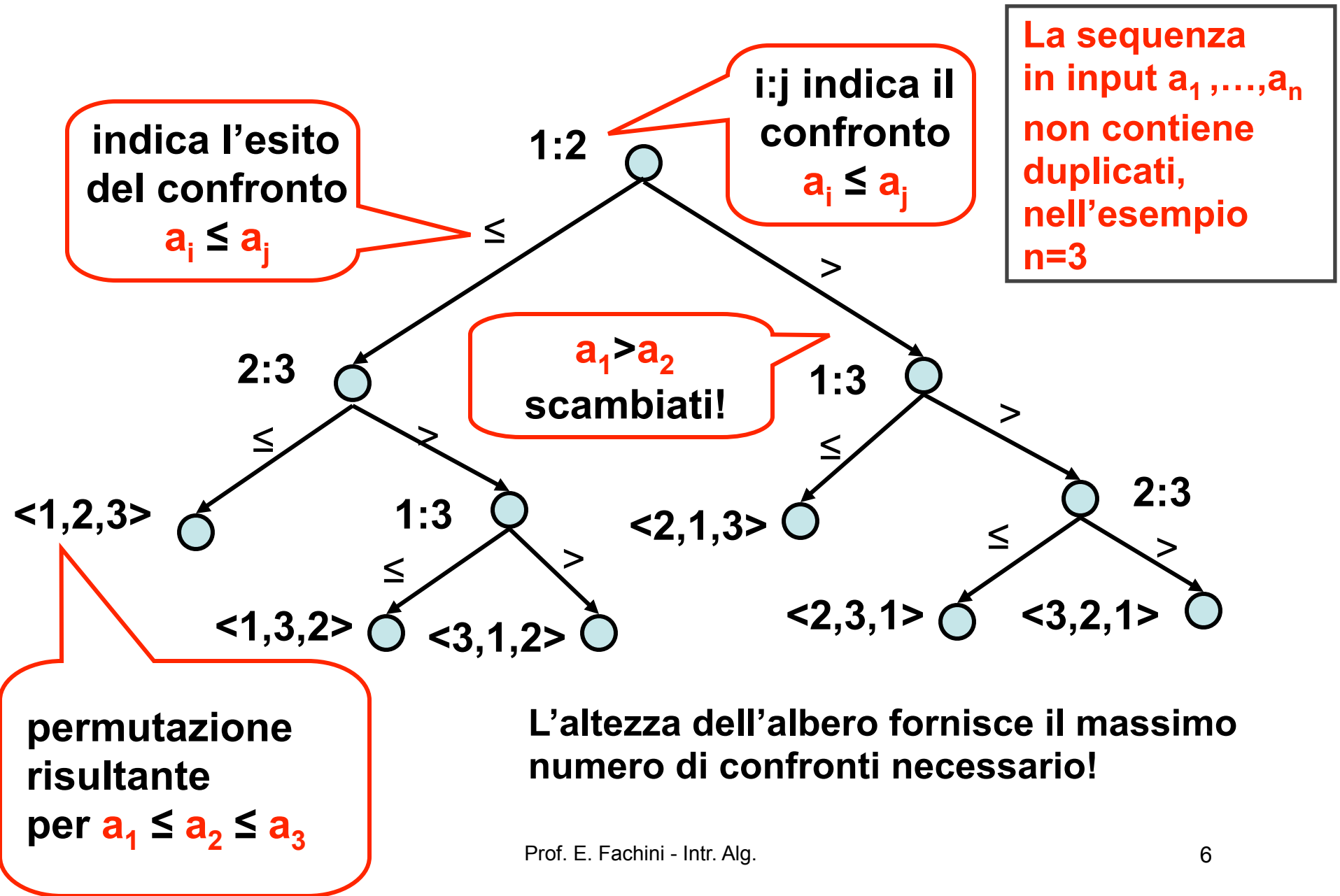
Dimostreremo che $\Omega(n \lg n)$ è un limite inferiore al numero di confronti eseguiti da un algoritmo di ordinamento, nel modello basato sui confronti, utilizzando il metodo degli alberi di decisione.

Alberi di decisione

Un algoritmo di ordinamento basato sui confronti può essere rappresentato da un albero binario in cui ogni nodo interno corrisponde a un confronto tra due elementi.

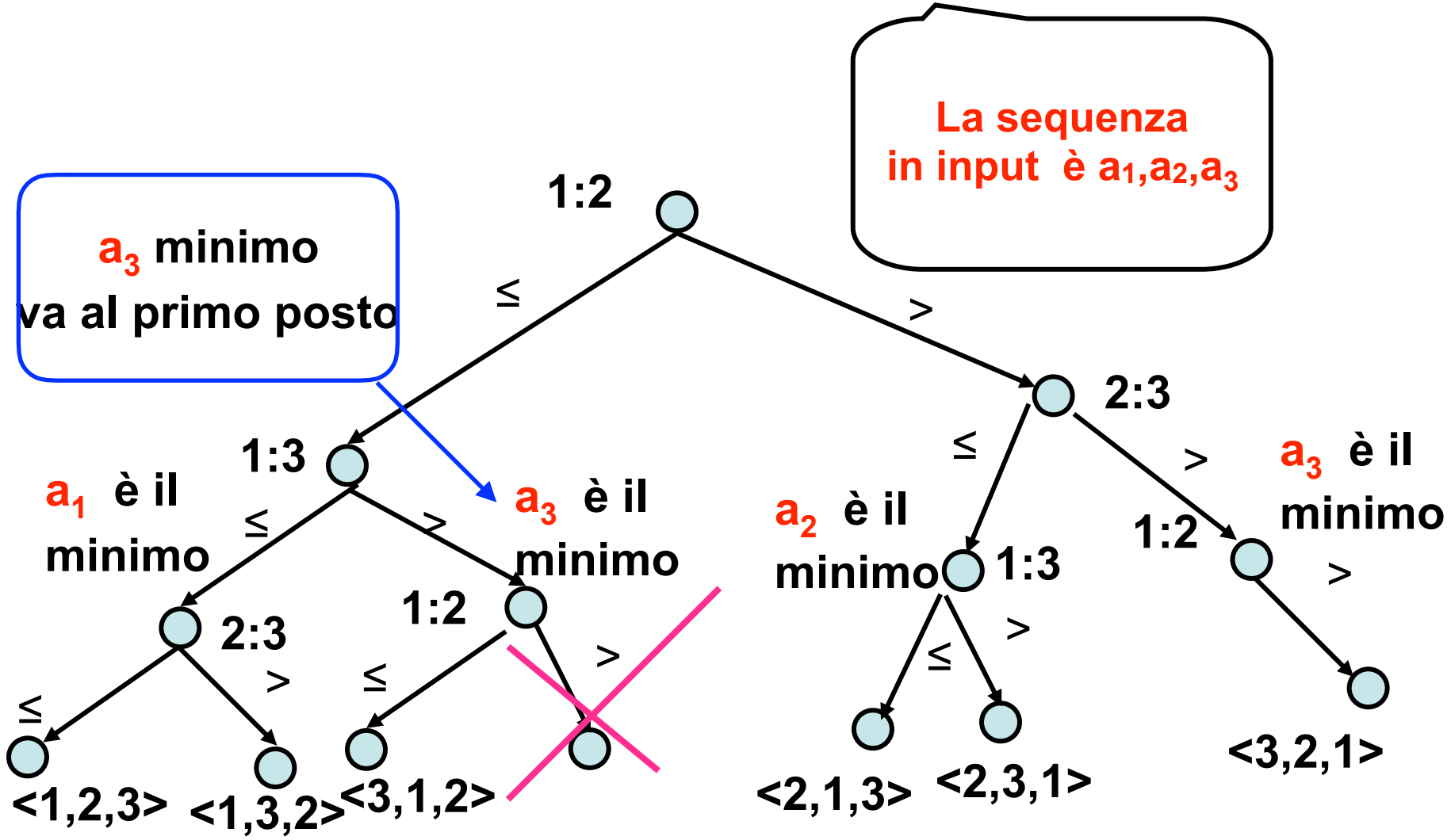
Si astrae da ogni altro aspetto dell'algoritmo come controllo, scambi,…, vediamo solo i confronti. Ogni cammino radice-foglia descrive una possibile computazione.
I possibili output di un algoritmo di ordinamento sono le permutazioni della sequenza in input che sono quindi associate alle foglie dell'albero di decisione.

Albero di decisione dell' insSort



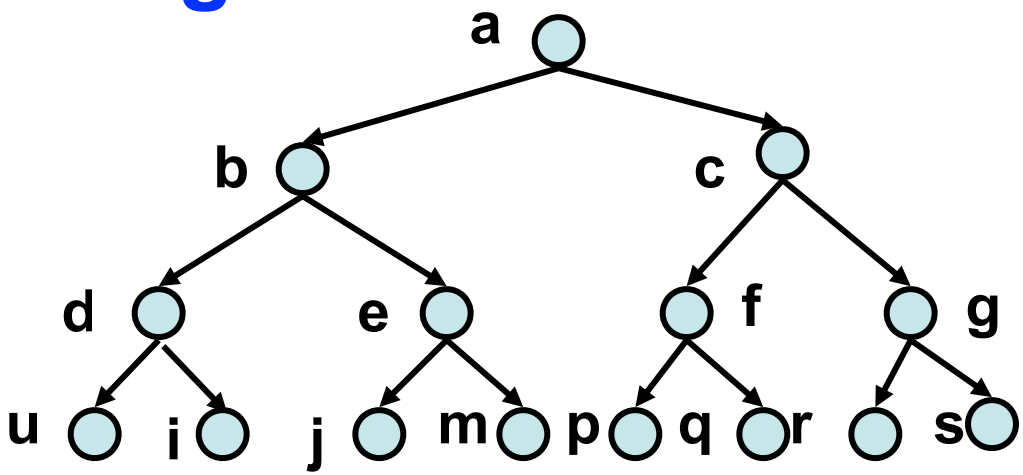
L'altezza dell'albero fornisce il massimo numero di confronti necessario!

Albero di decisione del selection Sort



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

Un algoritmo di ordinamento deve poter produrre tutti i possibili output e poiché le permutazioni di 1,2,...,*n* sono *n*!, l'albero delle decisioni di un algoritmo qualunque di ordinamento deve avere almeno *n*! foglie, cioè

$$n! \leq nFoglie(t)$$

Sappiamo che in un albero binario di altezza h si ha

$$nFoglie(t)) \leq 2^h \text{ quindi } \lg(nFoglie(t)) \leq h$$

Dunque nel caso peggiore l'algoritmo deve eseguire almeno lg(*n*!) confronti, perchè

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

Calcolo di lg(n!)

Nota che $n/2 = \lfloor n/2 \rfloor$

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)
Ogni fattore tra n(n-1)…n/2 è maggiore o uguale a n/2 quindi si può dire che
lg (n(n-1)…n/2) ≥ lg $((n/2)^{n/2})$ = n/2 lg n/2

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

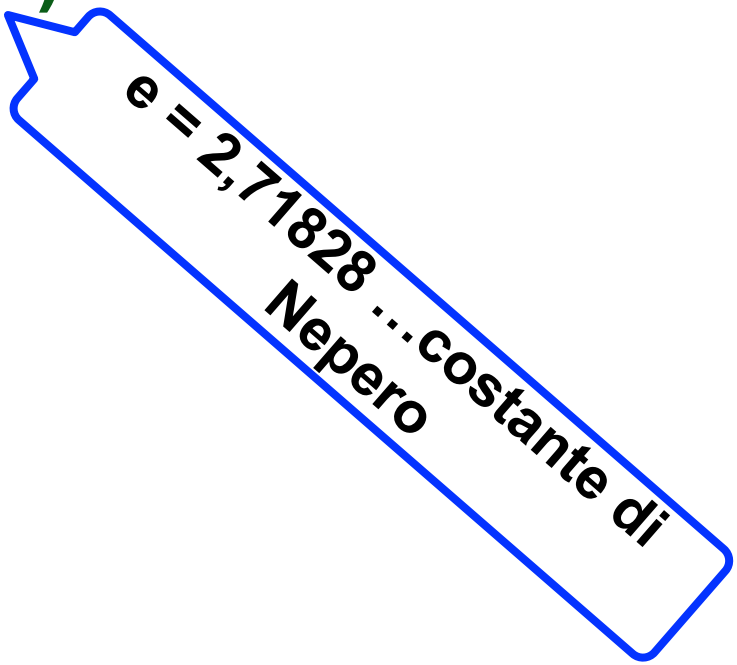
Usando Stirling

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

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

Avevamo che $h \geq \lg n!$
quindi $h > \lg (n/e)^n = n\lg (n/e) = n(\lg n - \lg e)$

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



$\Omega(n \lg n)$ 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 quindi le uniche operazioni ammesse sono confronti e assegnamenti.

Complessità ordinamenti

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

2. Gli algoritmi che conosciamo su n elementi risolvono il problema in $\theta(n^2)$, nel caso peggiore.

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

Sì, lo vedremo nelle prossime lezioni.

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

<u>Sfruttando ulteriori conoscenze sull'input</u> possiamo avere ordinamenti più veloci, ma in $\Omega(n)$!

Infatti per ordinare n elementi si dovranno fare almeno n-1 confronti, quelli necessari per verificare che sono già ordinati.

La ricerca del minimo

Possiamo utilizzare il modello degli alberi di decisione anche per dimostrare limiti inferiori di altri problemi, che si risolvono mediante confronti.

Quanti confronti sono necessari (limite inferiore) e sufficienti (limite superiore) per trovare il minimo tra n elementi?

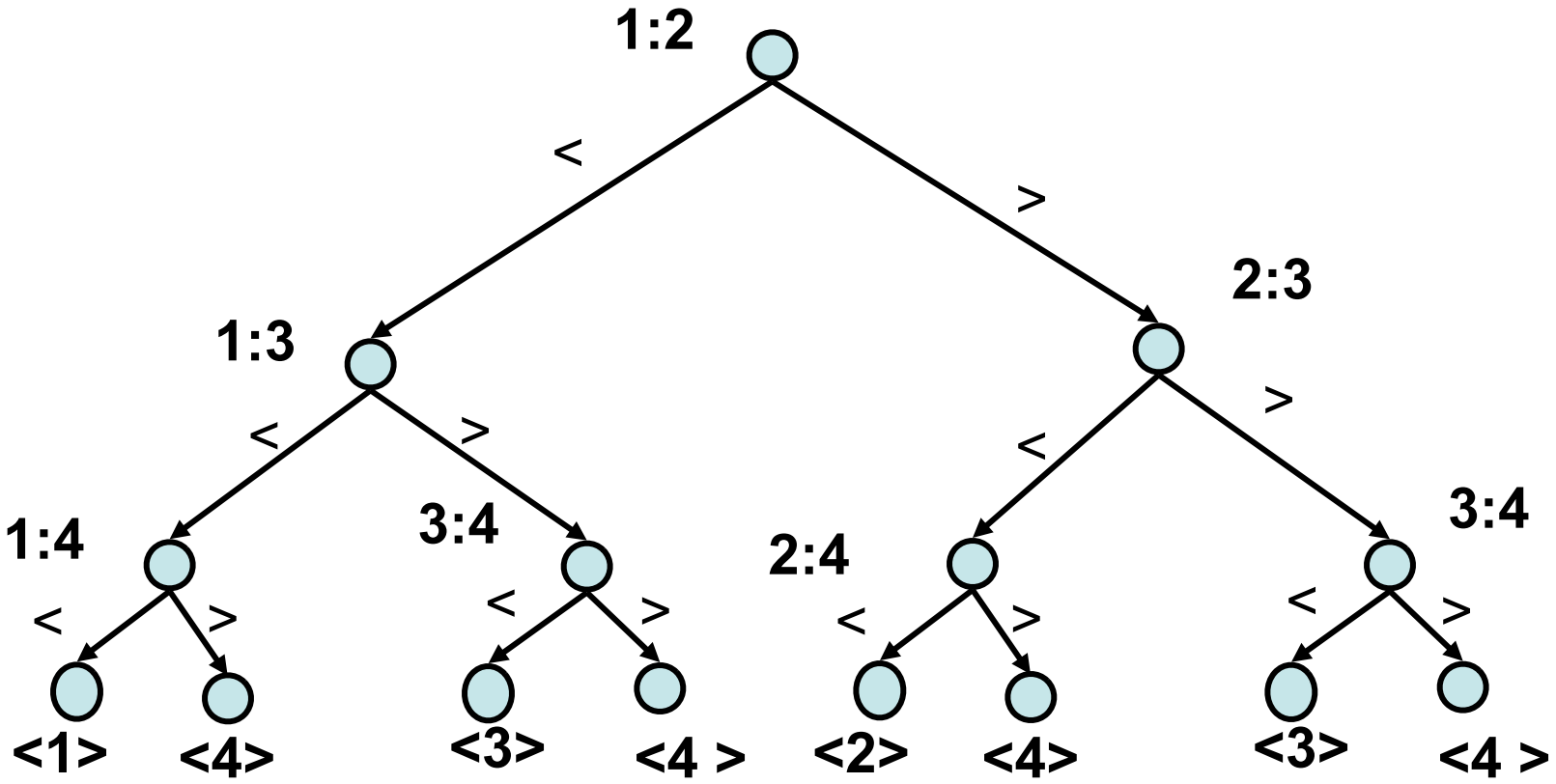
Il metodo

Preso un problema del quale vogliamo stabilire il limite inferiore:

1. stabiliamo il numero dei possibili diversi esiti dell'algoritmo

2. consideriamo questo numero come limite inferiore al numero delle foglie, a questo punto il suo logaritmo fornisce un limite inferiore all'altezza di un qualsiasi albero di decisione per il problema.

Albero di decisione dell'algoritmo per la ricerca del minimo su 4 elementi



Ricerca del minimo

1. L'albero di decisione deve avere almeno n foglie e dunque altezza almeno lg n, quindi il problema ha complessità $\Omega(\lg n)$

2. L'algoritmo di cui disponiamo su n elementi risolve il problema in $\theta(n)$ in tutti i casi.

3. Esiste un algoritmo che risolve il problema in $\theta(\lg n)$, nel caso peggiore? No!

Limite inferiore per la ricerca del minimo

Non sempre il metodo dell'albero delle decisioni fornisce il miglior limite inferiore possibile. Nel caso del minimo:

$\Omega(n)$ è un limite inferiore migliore che si ottiene ragionando come segue:

- **se gli n elementi sono tutti distinti allora n-1 elementi non possono essere il minimo**
- **in ogni confronto c'è sempre un solo "perdente" (il maggiore)**
- **quindi sono necessari almeno n-1 confronti**

Algoritmi ottimali

Se abbiamo dimostrato che un problema ha complessità, in un determinato modello di calcolo, in $\Omega(f(n))$ (cioè che f(n) è un limite inferiore alla sua complessità)

e troviamo un algoritmo A che ha complessità O(f(n)) nel caso peggiore (cioè che f(n) è un limite superiore al suo tempo di esecuzione nel caso peggiore) allora possiamo dire il problema ha complessità $\Theta(f(n))$ (cioè che f(n) è un limite asintotico stretto alla sua complessità).

Inoltre l'algoritmo A è asintoticamente ottimale.

$\Theta(n)$ per la ricerca del minimo

Il nostro algoritmo MinInd per il calcolo del minimo ha complessità nel caso peggiore $\Theta(n)$, questo fornisce un limite superiore O(n) per il problema della ricerca del minimo.

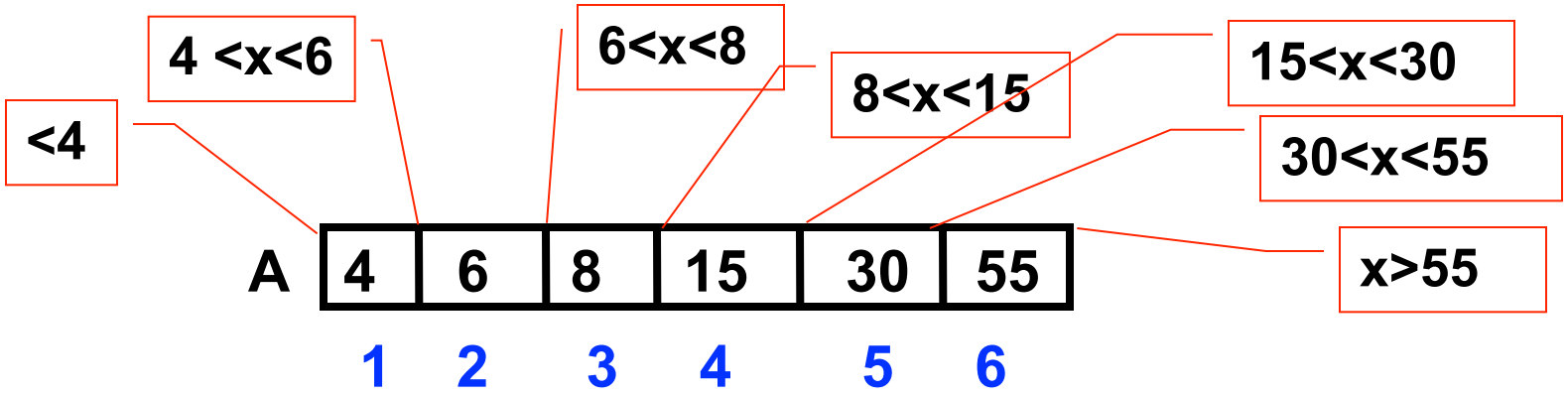
Poiché il limite inferiore è $\Omega(n)$ e quello superiore è O(n) anche questo limite è stretto e possiamo dire che il problema di individuare il minimo ha complessità $\Theta(n)$.

Inoltre l'algoritmo MinInd è asintoticamente ottimale.

La ricerca in una sequenza ordinata

INPUT: array A[1]...A[n] di elementi tali che A[i]<A[i+1], per 1≤i<n e un elemento x

OUTPUT: la posizione di x in A se x è presente in A e NIL altrimenti

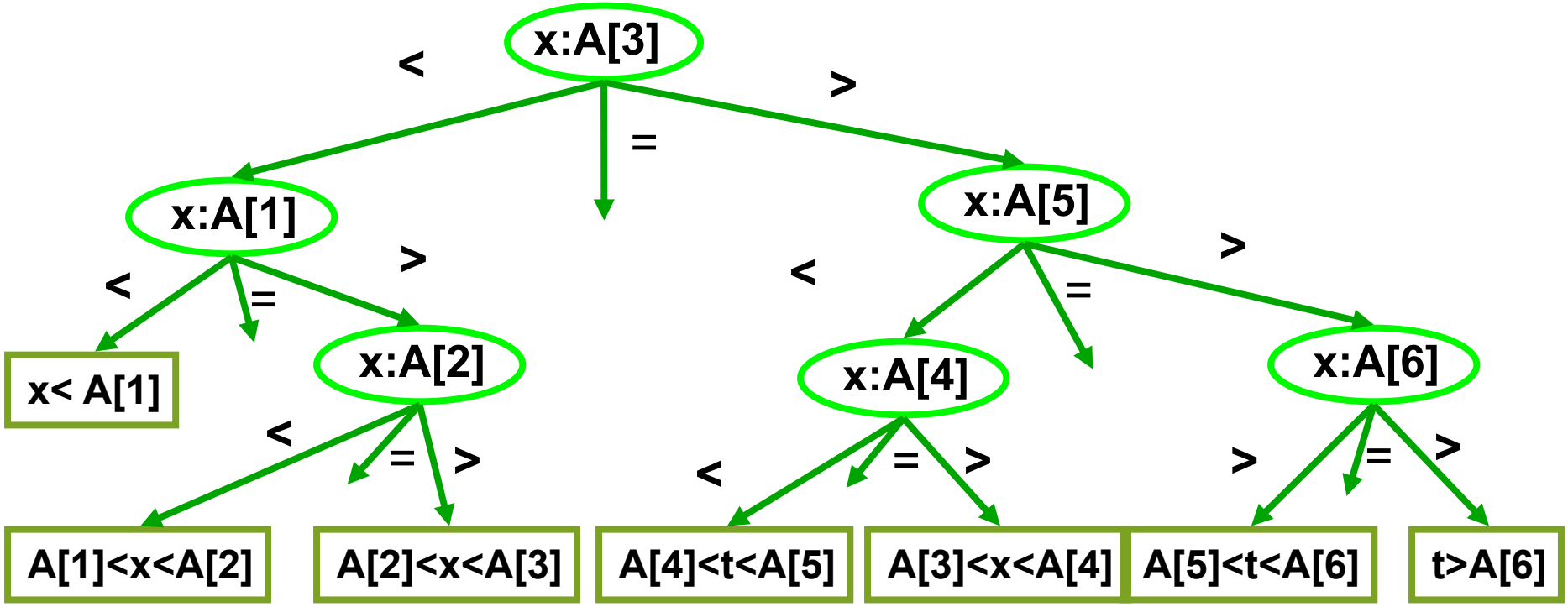


Albero di decisione ternario

Un albero di decisione per il problema avrà i nodi etichettati dal confronto tra l'elemento cercato e un elemento della sequenza e tre figli per i tre esiti del confronto: <,= >.

Albero di decisione ternario

Per esempio per la ricerca binaria:



Limite inferiore per la ricerca in una sequenza ordinata

Il numero delle foglie di un albero di decisione per la ricerca in una sequenza ordinata deve essere almeno 2n+1.

Analogamente al caso binario per un albero ternario vale

$$\text{nfoglie}(t) \leq 3^h$$

Dunque $\log_3 (2n+1) \leq \log_3 \text{nfoglie}(t) \leq h$.

Quindi

$h \geq \log_3(2n+1) \geq \log_3 n$ **per ogni** $n \geq 1$

da cui $h = \Omega(\lg n)$

Limite superiore per la ricerca in una sequenza ordinata

Conosciamo due algoritmi:

1. **la ricerca sequenziale, con tempo di esecuzione nel caso peggiore in $\Theta(n)$,**

2. **la ricerca binaria, con tempo di esecuzione nel caso peggiore in $\Theta(\lg n)$**

Quindi il miglior limite superiore per il problema della ricerca in una sequenza ordinata è O(lg n).

Problema ricerca in un sequenza ordinata

Poiché il limite inferiore e il superiore coincidono, anche questo limite è stretto e possiamo concludere che la complessità del problema della ricerca in una sequenza ordinata (nel caso peggiore) è $\Theta(\lg n)$, cioè che lg n è il numero necessario e sufficiente di confronti, nel caso peggiore, per il problema della ricerca in una sequenza ordinata .

Infine possiamo dire che l'algoritmo della ricerca binaria è ottimale.

Limiti inferiori asintotici

Se la complessità di un problema è in $\Omega(f(n))$ non è detto che si riesca a trovare una algoritmo che nel caso peggiore ha tempo di esecuzione in $O(f(n))$.

D'altra parte se si avesse necessità di un tempo di esecuzione migliore del limite inferiore, $\Omega(f(n))$, si dovrà modificare il problema in modo da poterlo risolvere con un algoritmo con un tempo di esecuzione nel caso peggiore migliore di $\Theta(f(n))$.