In questa lezione

- **Alberi di decisione nella determinazione di limiti inferiori ai problemi**

CLRS10 cap. 8 par. 8.1

Ordinamenti quadratici

Abbiamo visto tre algoritmi per ordinare una sequenza

- **insertionSort,**
- **SelectionSort**
- **BubleSort**

Senza ipotesi sul tipo degli elementi della sequenza, le uniche operazioni permesse sono confronti e assegnazioni.

Tutti con una complessità di tempo nel caso peggiore di $\Theta(n^2)$, tutti basati su confronti e tutti "in loco" cioè "operanti sul posto"

Non si usa memoria aggiuntiva

Dom. 1 Allora quale scegliere?

Dom. 2 Si può fare di meglio?

Allora quale scegliere?

	Complessità Nel caso peggiore	Complessità Nel caso migliore
• insertionSort	• $\Theta(n^2)$	• $\Theta(n)$
• SelectionSort	• $\Theta(n^2)$	• $\Theta(n^2)$
• BubbleSort	• $\Theta(n^2)$	• $\Theta(n^2)$ o $\Theta(n)$

Quindi contano le costanti! Nel BubbleSort il numero di scambi è il più delle volte proibitivo (tranne quando è già ordinato…), esistono varianti che migliorano leggermente, ma l'insertionSort è il più efficiente tra questi.

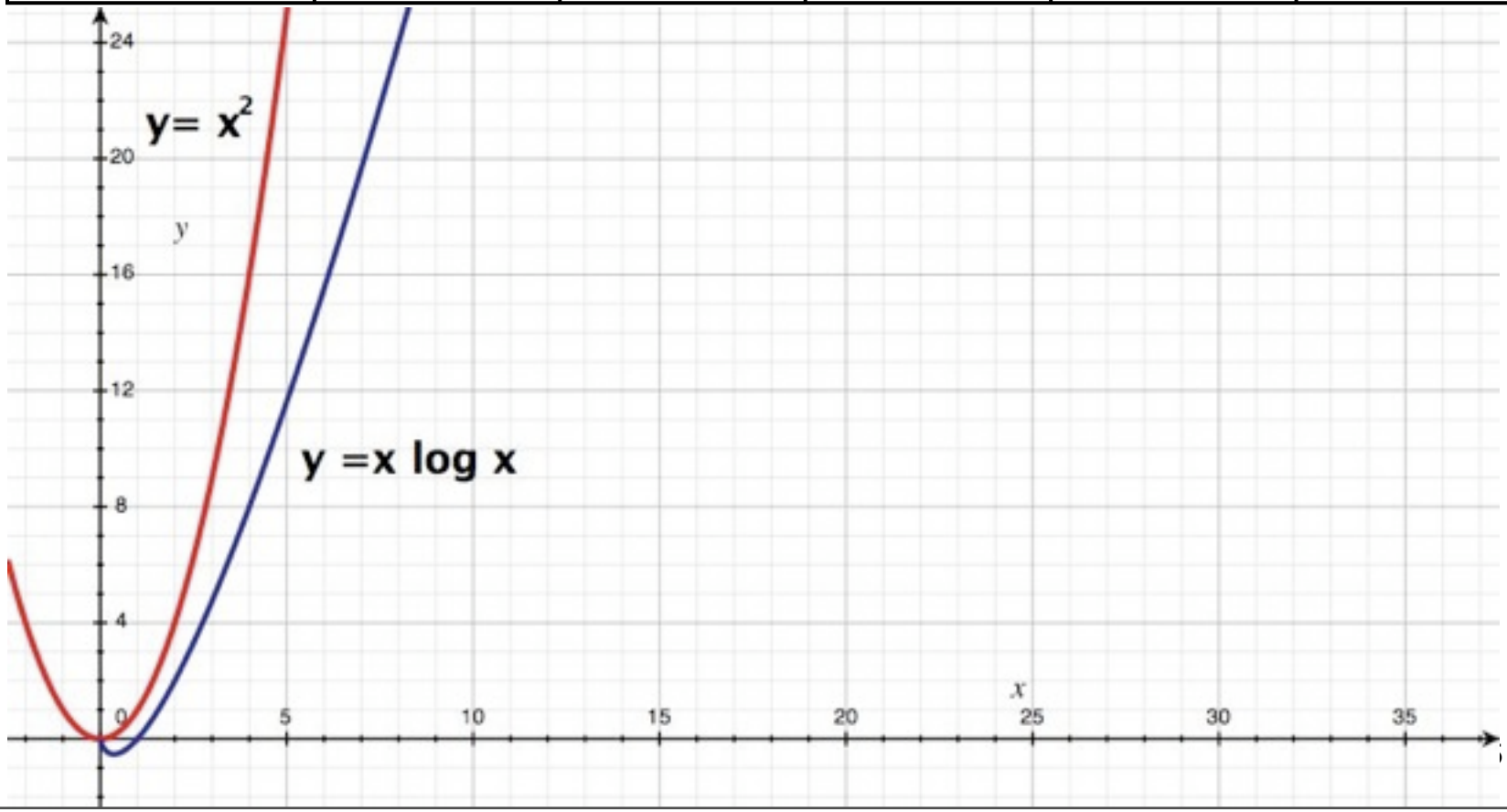
Algoritmi in O(nlg n)

	Complessità Nel caso peggiore	Complessità Nel caso medio
• MergeSort	• $\Theta(n \lg n)$	
• QuickSort	• $\Theta(n^2)$	• $\Theta(n \lg n)$
• HeapSort	• $O(n \lg n)$	

Si può fare di meglio?

Confronto x^2 e $x \lg x$

n	10	100	1000	10^6	10^9
$n \lg_2 n$	~ 33	~ 665	$\sim 10^4$	$\sim 2 \cdot 10^7$	$\sim 3 \cdot 10^{10}$
n^2	100	10^4	10^6	10^{12}	10^{18}



Si può fare di meglio?

In altri termini quanti confronti sono necessari (o qual è il limite inferiore al numero di confronti da eseguire) per ordinare una sequenza di n elementi?

Un limite inferiore f(n) per il problema dell'ordinamento, nel modello basato su confronti, stabilisce che ogni algoritmo di ordinamento basato sui confronti deve fare almeno f(n) di confronti.

Limite inferiore per l'ordinamento

Dimostreremo che ogni algoritmo di ordinamento basato su confronti deve eseguire nel caso peggiore $\Omega(n \lg n)$ confronti.

In altri termini dimostreremo che nlg n è un limite inferiore alla complessità del problema dell'ordinamento, nel modello basato su confronti.

Limiti inferiori asintotici

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

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

Questo non garantisce che riusciamo a trovare una algoritmo con quella complessità.

Se per il mio problema fosse necessario un tempo di calcolo inferiore al limite inferiore, dovrò cambiare il modello di calcolo o porre dei vincoli al problema...

Taglia di un problema

È una misura dell'input (in bit, parole, ecc.) che dipende dalla rappresentazione dei dati (struttura dati)

Esempi

- **ordinamento: numero di oggetti da ordinare**
- **gestione dati: numero dei dati da gestire**
- **algoritmi e problemi su alberi: numero di nodi o altezza**

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 O(nlgn) perché abbiamo algoritmi di ordinamento di questa complessità nel caso peggiore .

Algoritmi ottimali

Se abbiamo dimostrato che un problema ha complessità, in un determinato modello di calcolo, nel caso peggiore

$$\Omega(f(n))$$

(cioè che f(n) è un limite inferiore alla sua complessità)

e troviamo un algoritmo A che ha complessità

$$O(f(n))$$

(cioè che f(n) è un limite superiore alla sua complessità)

allora possiamo dire il problema ha complessià

$$\Theta(f(n))$$

(cioè che f(n) è un limite stretto alla sua complessità)

e che l'algoritmo A è asintoticamente ottimale

Alberi di decisione

Per dimostrare che $\Omega(n \lg n)$ è un limite inferiore al numero di confronti eseguiti da un algoritmo di ordinamento, basato sui confronti, utilizziamo gli alberi di decisione.

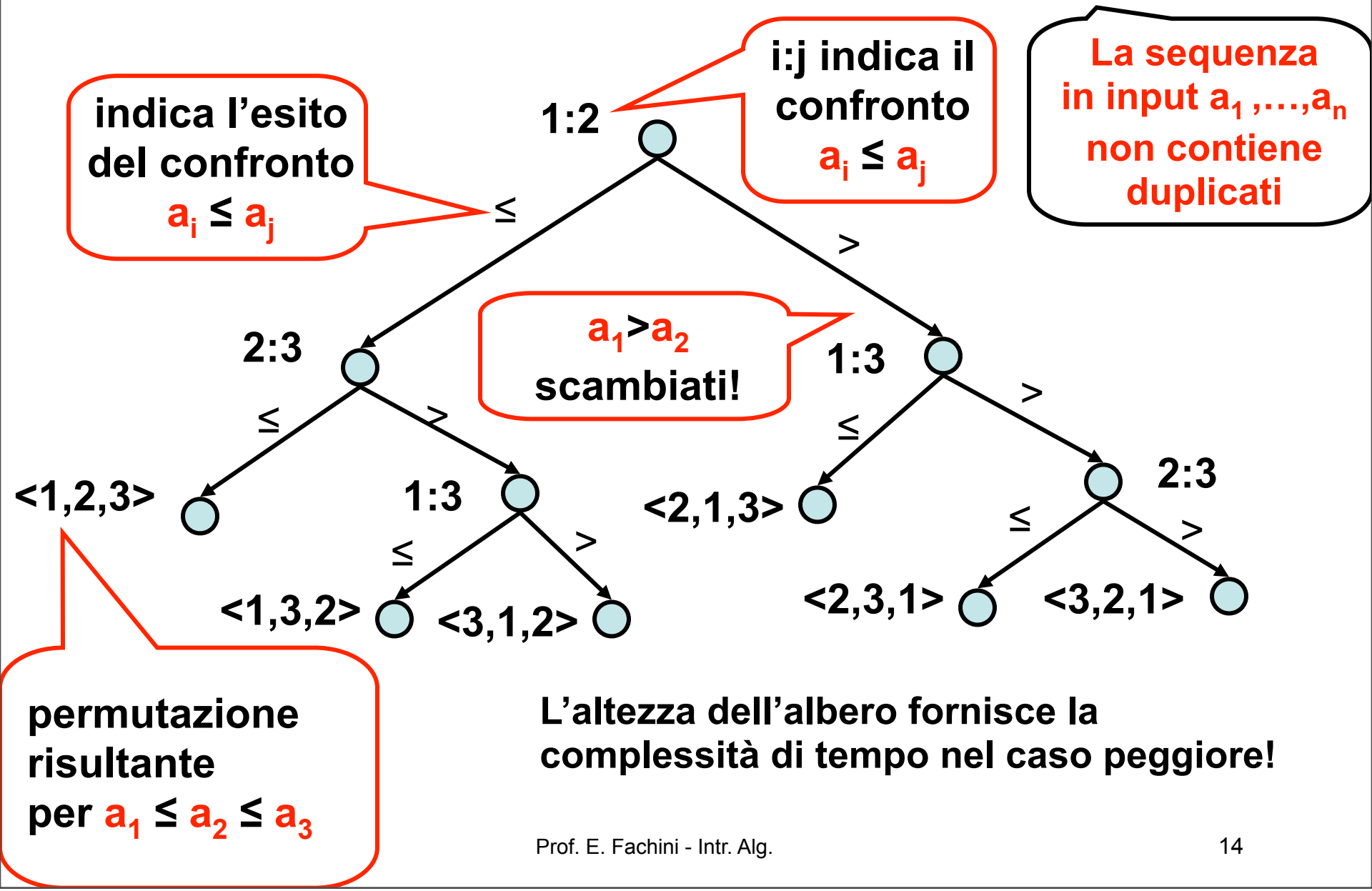
Alberi di decisione

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.

Astraiamo da ogni altro aspetto dell'algoritmo come controllo, scambi,…, vediamo solo i confronti.

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 la complessità di tempo nel caso peggiore!

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 decisione deve avere almeno *n*! foglie, cioè n! ≤ nFoglie(t)

Tra tutti gli alberi di decisione, che corrispondono ad altrettanti algoritmi di ordinamento, cerchiamo quello di altezza minima. Questo mi darà un limite inferiore.

In un albero binario di altezza h si ha nFoglie(t) ≤ 2^h
Da cui lg (n!) ≤ lg(*n*Foglie(t)) ≤ h, per la monotonicità della funzione logaritmo.

Dunque come minimo l'albero deve avere altezza lg(*n*!) o equivalentemente, nel caso peggiore l'algoritmo deve eseguire almeno lg(*n*!) confronti.

Il limite inferiore

Stirling ha dimostrato che $n! > (n/e)^n$

e = costante di Nepero

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

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

$$n \lg (n/e) =$$

$$n \lg n - n \lg e$$

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

Senza usare la formula di Stirling

Notiamo che $n! = n(n-1) \dots 2 =$

$n(n-1) \dots \lfloor n/2 \rfloor (\lfloor n/2 \rfloor -1) \dots 2$

$\geq n(n-1)\dots \lfloor n/2 \rfloor$

$\geq \lfloor n/2 \rfloor^{\lfloor n/2 \rfloor}$

perchè i fattori sono tutti maggiori o uguali a $\lfloor n/2 \rfloor$

quindi otteniamo che

$\lg n! > \lg (\lfloor n/2 \rfloor^{\lfloor n/2 \rfloor}) = \lfloor n/2 \rfloor \lg \lfloor n/2 \rfloor$

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

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

Quindi il limite superiore al problema dell'ordinamento è $O(n \lg n)$, mentre l'inferiore è $\Omega(n \lg n)$, allora possiamo concludere che $\Theta(n \lg n)$ è un limite stretto, nel caso peggiore, per i problema dell'ordinamento, quando ci restringiamo ad algoritmi basati sul confronto.

Gli algoritmi di ordinamento con complessità $O(n \lg n)$ nel caso peggiore, mergeSort e heapSort, sono asintoticamente ottimali (il loro albero di decisione ha altezza minima)

$\Omega(n \lg n)$ 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.

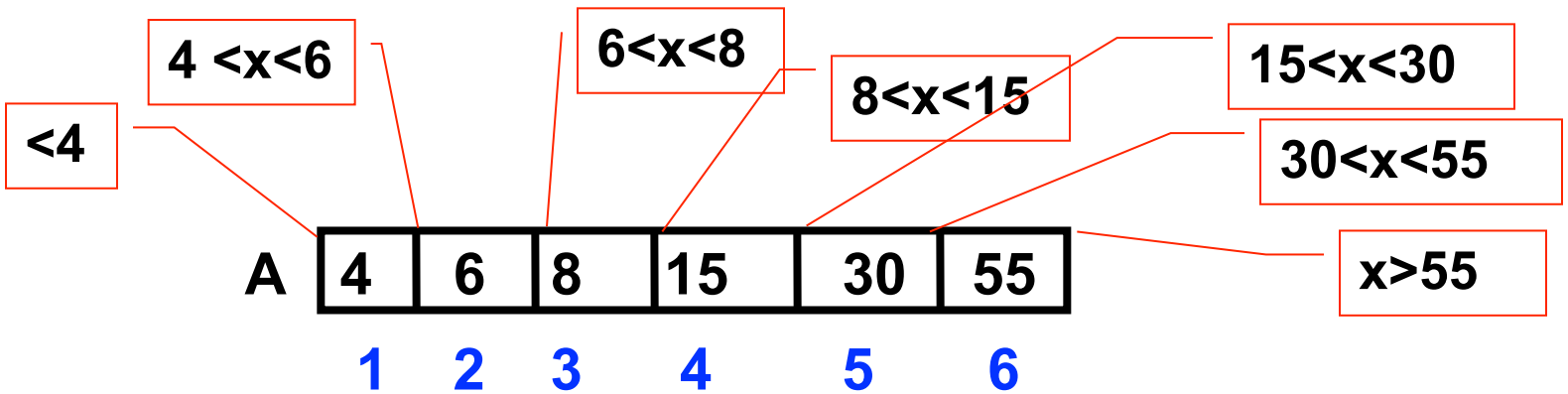
Sfruttando conoscenze sull'input possiamo avere ordinamenti più veloci, ma in $\Omega(n)$!

Infatti per ordinare n elementi almeno li dobbiamo leggere!

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

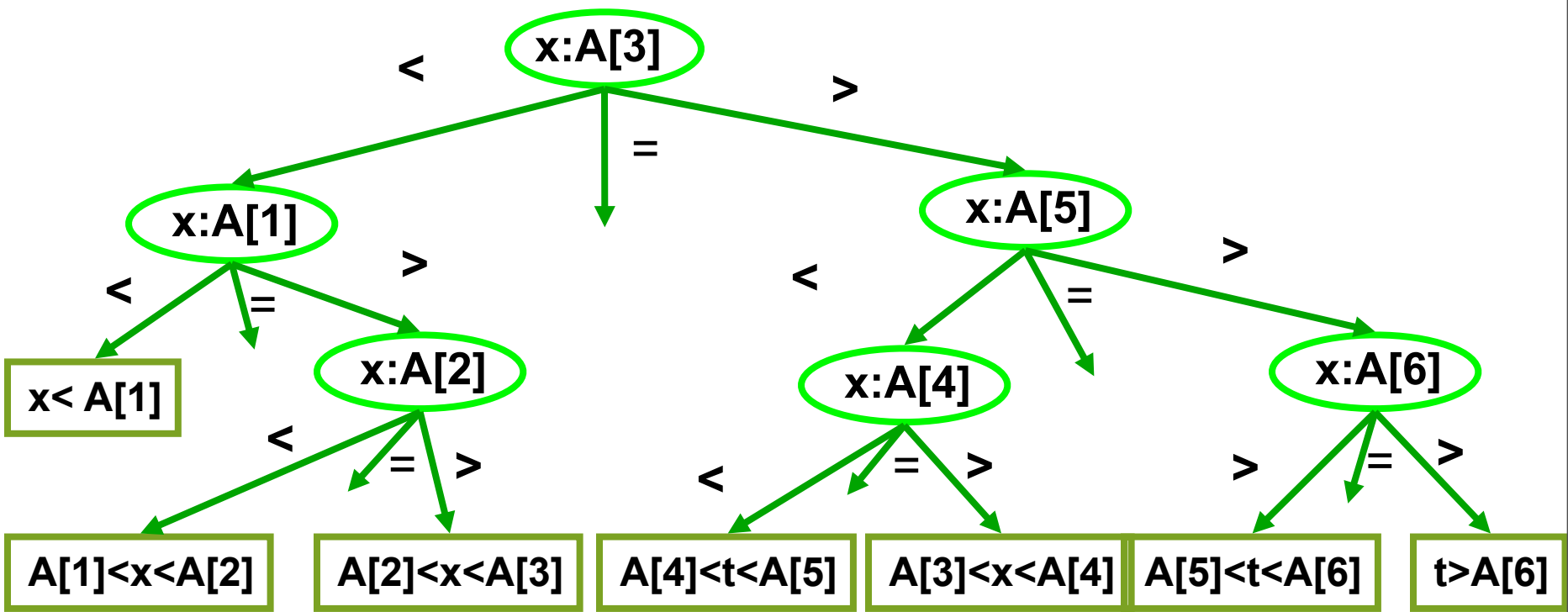
OUTPUT: la posizione di x in A se x è presente in A e NIL altrimenti



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: <,= >.

Per esempio per la ricerca binaria:



Limite inferiore

Il numero delle foglie dell'albero di decisione è almeno 2n+1

Analgamente al caso binario per un albero ternario vale

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

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

Quindi

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

da cui

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

Limite superiore

Conosciamo due algoritmi:

1. **la ricerca sequenziale, con complessità $\Theta(n)$, nel caso peggiore e**

2. **la ricerca binaria, con complessità $\Theta(\lg n)$**

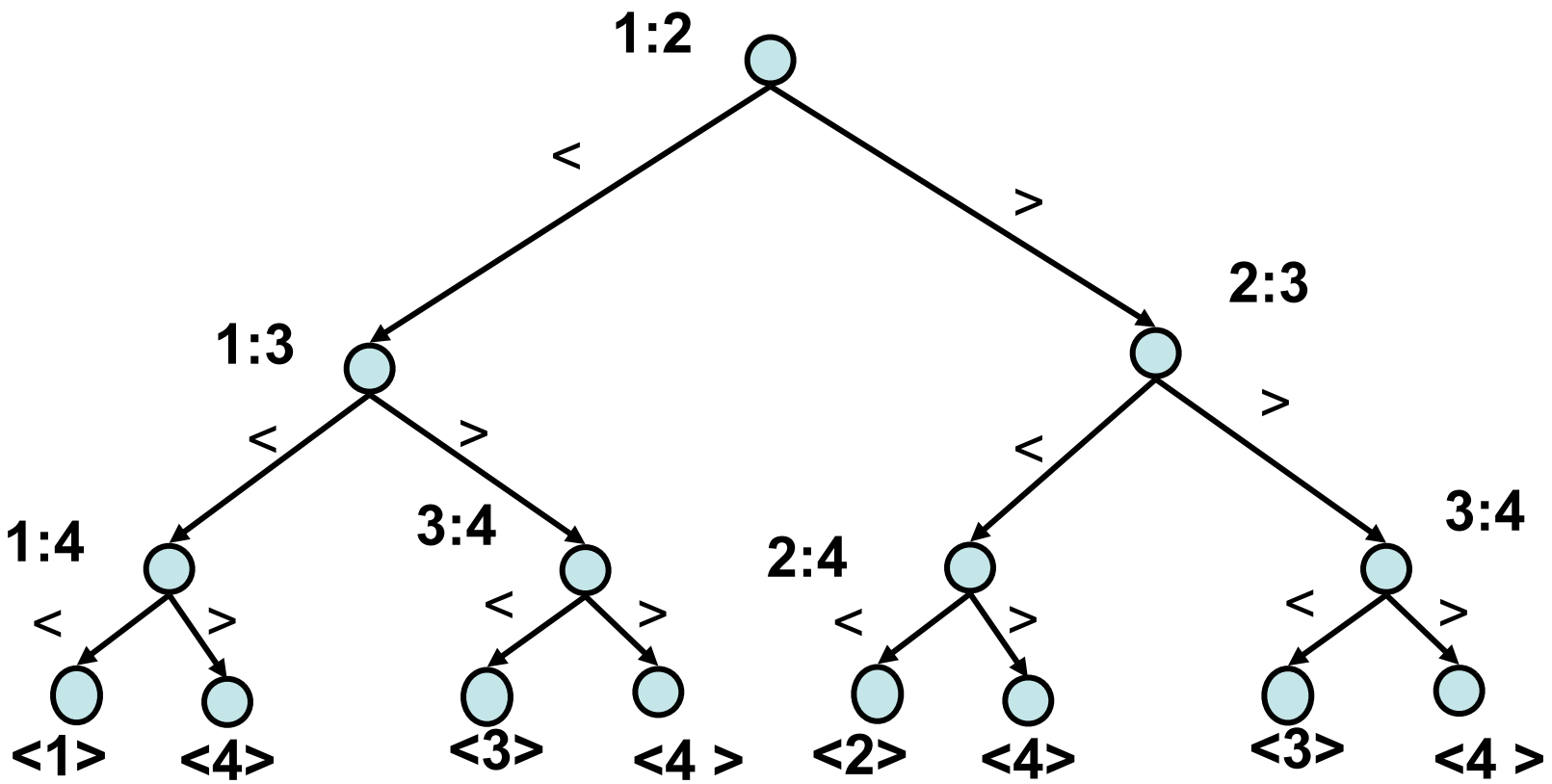
sempre nel caso peggiore

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?

Albero di decisione dell'algoritmo per la ricerca del minimo



Limite inferiore per la 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. $\Omega(n)$ è un limite inferiore migliore che si ottiene 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**

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

L'algoritmo ha complessità nel caso peggiore $\Theta(n)$.

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

Conclusione

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 per il problema della ricerca in una sequenza ordinata .

Inoltre l'algoritmo della ricerca binaria è ottimale.