In questa lezione

•**Alberi di decisione nella determinazione di limiti inferiori alla complessità dei problemi**

- **CLRS cap 8, par 8.1**

Il problema dell'ordinamento

Problema dell'ordinamento.

***Input*: sequenza $a_1,a_2,\ldots,a_n$ di elementi su cui è definita una relazione d'ordine ≤.**

***Output*: la permutazione di $a_1,a_2,\ldots,a_n$, $a'_1,a'_2,\ldots,a'_n$ tale che $a'_1 \leq a'_2 \leq \ldots \leq a'_n$.**

Algoritmi per l'ordinamento

Abbiamo visto tre algoritmi quadratici per ordinare una sequenza

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

Modello basato sui confronti:
le uniche operazioni sono confronti e assegnamenti.

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

Abbiamo visto altri tre algoritmi, basati su confronti per ordinare una sequenza con complessità in media O(nlg n)

- **mergeSort,**
- **quickSort**
- **heapSort**

Mergesort e heapsort con una complessità di tempo$O$(nlg n) anche nel caso peggiore.

Confronto x^2 e $x \lg x$

n	$8=2^3$	$128=2^7$	$32.768 = 2^{15}$
$n \lg_2 n$	24	896	491.520
n^2	64	16.384	1.073.741.824

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 la complessità del problema dell'ordinamento, nel modello basato su confronti, è in $\Omega(n \lg n)$.

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

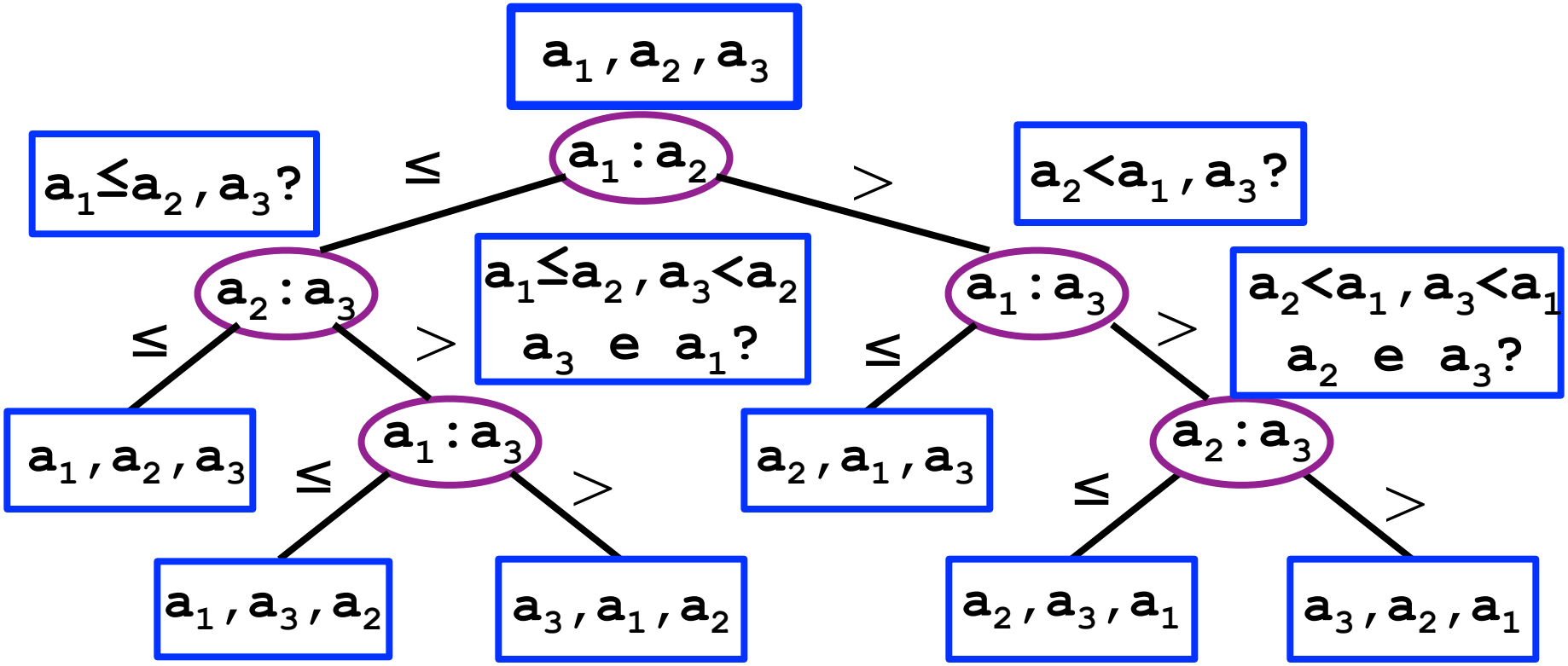
Un algoritmo di ordinamento basato sui confronti può essere rappresentato da un albero binario in cui ogni nodo interno corrisponde a un confronto tra due elementi, ogni nodo ha due figli corrispondenti ai due esiti del confronto: ≤ o >.

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.

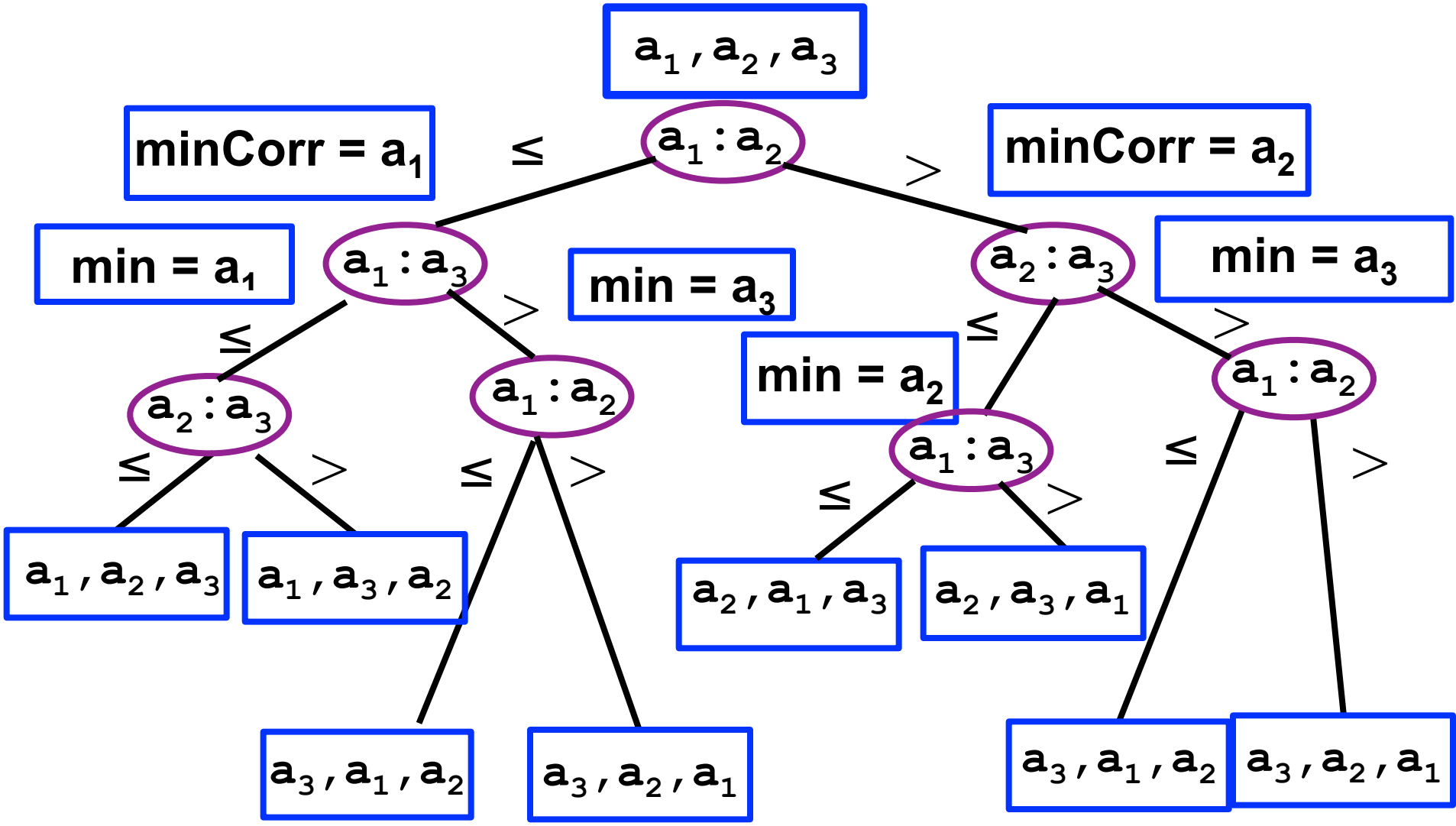
Esempio

Albero delle decisioni di un algoritmo che ordina un array di 3 elementi.



InsertionSort

SelectionSort



Albero di decisione ordinamenti

Se l'algoritmo è corretto a permutazioni distinte in input corrispondono cammini radice-foglia distinti nell'albero.

Un qualsiasi 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, raggiungibili dalla radice.

Poiché l'altezza dell'albero dà la complessità del problema nel caso peggiore ci chiediamo qual' è la minima altezza per un albero di decisione per il problema dell'ordinamento?

In un albero binario di altezza h si ha
$$nFoglie(t) \leq 2^h$$

Da cui
$$\lg(nFoglie(t)) \leq h$$

in un albero di decisione per l'ordinamento
$$n! \leq nFoglie(t)$$

e quindi
$$\lg(n!) \leq \lg(nFoglie(t)) \leq h$$

Dunque nel caso peggiore l'algoritmo deve eseguire almeno $\lg(n!)$ confronti.

Limite inferiore

Sappiamo che

$$\lg n! = \lg (n(n-1) \dots 2) =$$

$$\lg (n(n-1) \dots n/2(n/2+1) \dots 2)$$

I fattori $n(n-1) \dots n/2$ sono maggiori o uguali a $n/2$ e eliminando gli altri fattori $(n/2+1) \dots 2$ si ha

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

$$n/2 \lg n/2$$

Quindi $\lg n! = \Omega(n \lg n)$

$\Omega(n \lg n)$ 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.

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

Θ(*n* lg *n*) per il problema dell' ordinamento.

Quindi il limite superiore al problema dell'ordinamento è O(*n lg n*), mentre l'inferiore è Ω(*n* lg *n*), allora possiamo concludere che Θ(*n* lg *n*) è un limite stretto, nel caso peggiore, per il 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)

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 calcolo, 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!

Dimensione di un problema

È una misura dell'input (in bit, parole, ecc.).

Esempi

- **ordinamento: numero di oggetti da ordinare**
- **gestione dati: numero dei dati da gestire**
- **algoritmi e problemi su alberi: numero di nodi o altezza**
- **dipende dalla rappresentazione dei dati (struttura dati)**

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, O(nlg n), perché abbiamo algoritmi di ordinamento di complessità nel caso peggiore Θ(nlgn).

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 complessità

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

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

e che l'algoritmo A è asintoticamente ottimale

Il metodo degli alberi di decisione

Il metodo dell'albero di decisione per trovare un limite inferiore a un problema si applica ogni qual volta il processo risolutivo fa uso essenzialmente di confronti.

Per esempio il problema della ricerca di un elemento in una sequenza ordinata.

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

Conosciamo due algoritmi che risolvono il problema:

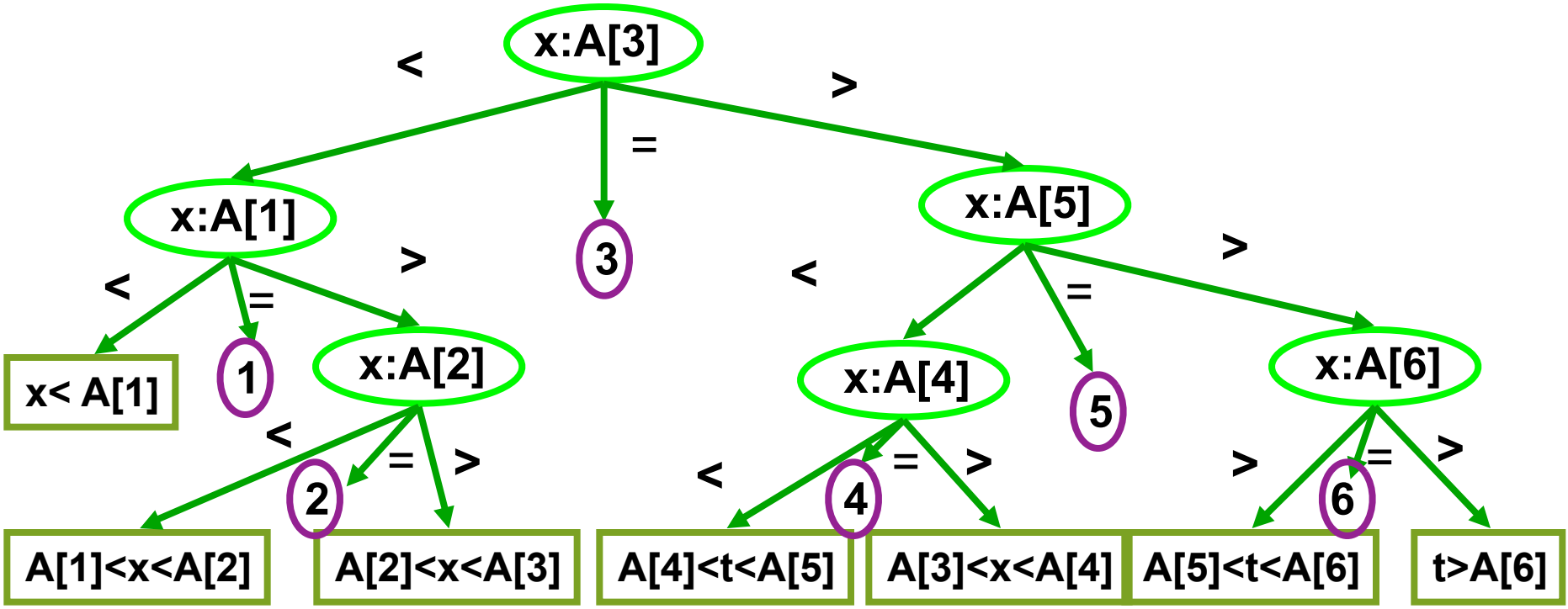
1. la ricerca sequenziale, con complessità $\Theta(n)$,
2. la ricerca binaria, con complessità $\Theta(\lg n)$ sempre nel caso peggiore

Quindi il limite superiore è O(lg n)
e quello inferiore?

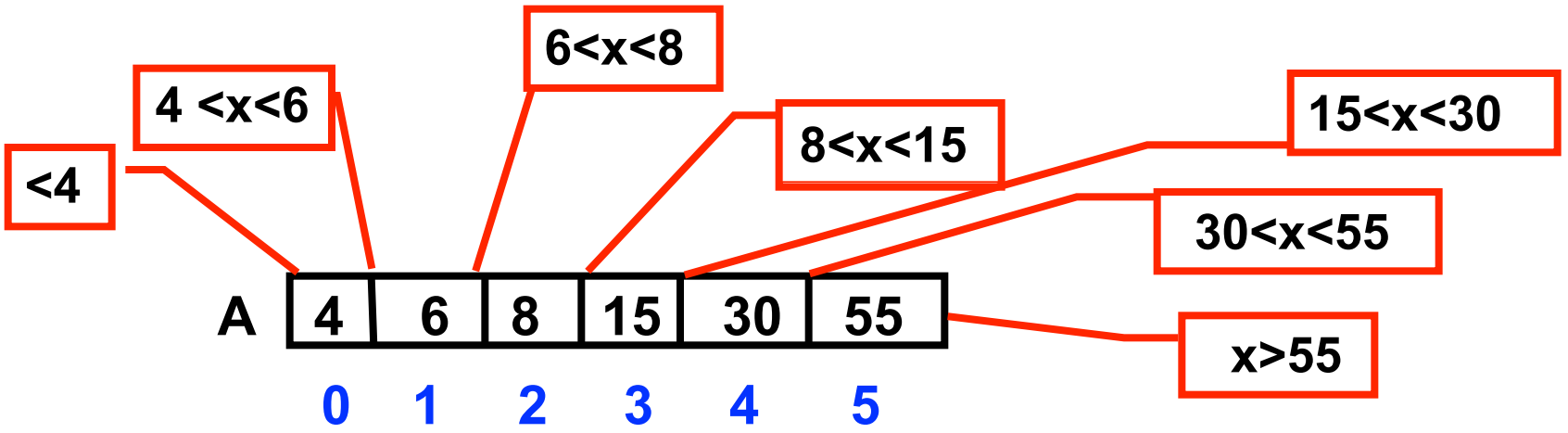
Albero di decisione ternario

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 su 6 elementi:



Esiti della ricerca



Poiché l'elemento cercato può essere uguale a uno degli elementi si hanno n possibili risposte positive, ma se non è presente può esserlo in n +1 modi diversi, corrispondenti agli intervalli tra gli elementi (che sono n-1) più i due intervalli all'inizio e alla fine.

Quindi ogni algoritmo corretto deve avere un albero di decisione con almeno 2n+1 foglie.

Limite inferiore

Il numero delle foglie di un albero di decisione per il problema della ricerca in un insieme ordinato è almeno 2n+1

Analgamente al caso binario per un albero ternario t 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

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

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 .

La ricerca del minimo

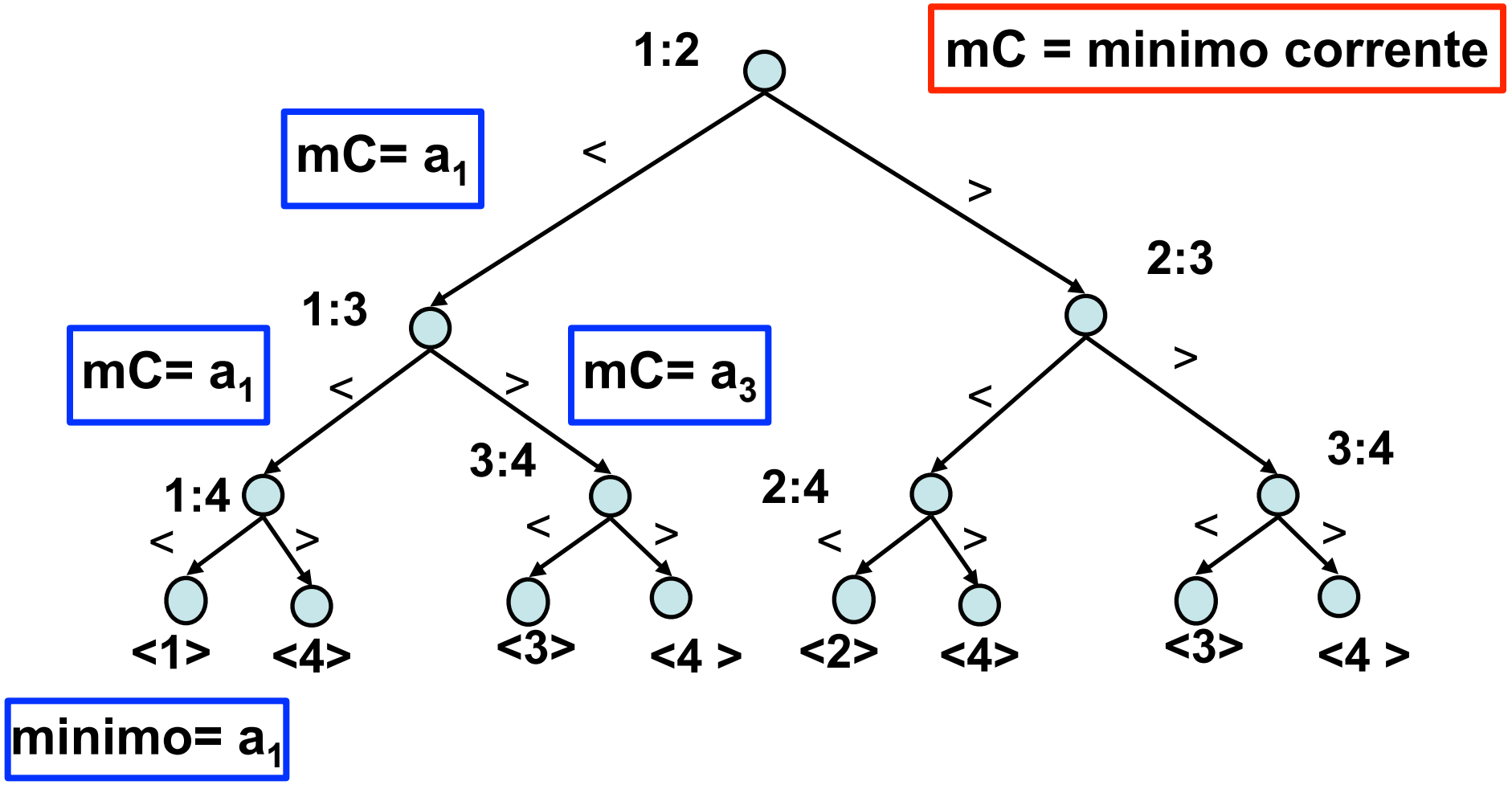
Quanti confronti sono necessari (limite inferiore) e sufficienti (limite superiore) per trovare il minimo tra n elementi?

Conosciamo un algoritmo di complessità nel caso peggiore (in realtà in tutti i casi) Θ(*n*), quindi il limite superiore per il problema della ricerca del minimo è O(n).

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 $\Omega(\lg n)$

Albero di decisione dell'algoritmo per la ricerca del minimo



Limite inferiore per la ricerca del minimo

Ma Ω(*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**

Poiché il limite superiore per il problema della ricerca del minimo è O(n) e il limite inferiore è Ω(*n*) anche questo limite è stretto e possiamo dire che il problema di individuare il minimo ha complessità Θ(*n*).

Esercizi

- **[CLRS] es. 8.1-1 e 8.1-4**
- **Come possiamo usare gli alberi di decisione per determinare un limite inferiore alla complessità della ricerca di un elemento in una sequenza non ordinata. Sugg.: l'esito del confronto sarà = o ≠.**