4.2 Limiti inferiori

Scoprire un algoritmo per la risoluzione di un problema equivale a stabilire un limite superiore di complessità, nel senso che risulta automaticamente provato come il problema sia risolubile entro i limiti di tempo e spazio stabiliti dall'algoritmo. Il quale può d'altra parte essere inefficientissimo, come l'algoritmo $\mathcal{A}_3$ del paragrafo precedente, fissando così limiti superiori di interesse assai scarso, se non è possibile in pari tempo fissare corrispondenti limiti inferiori che indichino quali margini di miglioramento siano ancora possibili.

La ricerca dei limiti inferiori di complessità risponde cioè alla domanda se non si possano determinare algoritmi più efficienti di quelli noti.

Lo studio è spesso limitato all'ordine di grandezza, cui si applica la notazione Ω, per la già discussa poca significatività di determinare le costanti, se non in relazione alle caratteristiche fisiche dell'elaboratore impiegato.

Il problema è quindi in genere quello di stabilire che la complessità asintotica $g(n)$ di *qualunque* algoritmo che risolva un dato problema è limitata inferiormente da una funzione $f(n)$, cioè $g(n) \in \Omega(f(n))$.

Diciamo subito che non esiste una teoria generale per determinare i limiti inferiori, ma che alcune linee di studio possono condurre a risultati soddisfacenti, cioè sufficientemente vicini ai valori di complessità dei più efficienti algoritmi noti. Si deve infatti notare che ragionamenti banali permettono in genere di determinare una funzione limitatrice $f(n)$ di ordine molto basso, ma totalmente irragiungibile dalla complessità di algoritmi costruibili, e quindi non interessante. In questi casi uno studio più raffinato può condurre a un limite inferiore più alto, che sostituisce più realisticamente il precedente. Per esempio, per ordinare n elementi è necessario esaminarli tutti, e quindi la complessità in tempo $g(n)$ di qualsiasi algoritmo di ordinamento è banalmente limitata da $g(n) \in \Omega(n)$. Tuttavia, come vedremo tra breve, ragionamenti più profondi permettono di determinare un interessante, assai più alto limite inferiore.

Quando la complessità di un algoritmo è pari al limite inferiore di complessità determinato per il problema, l'algoritmo si dice *ottimo*. E' sovente possibile dimostrare che un algoritmo è ottimo *in ordine di grandezza*, assai raro che se ne possa dimostrare l'ottimalità *in assoluto*, cioè provare che la funzione di complessità coincide esattamente con quella che esprime il limite inferiore.

Indichiamo ora alcune linee generali per la determinazione dei limiti inferiori, ragionando, salvo diversa specificazione, sulla complessità in tempo. Il lettore diffidi però dell'apparente semplicità dei discorsi che seguono: trovare limiti inferiori significativi è in genere assai più difficile che trovare buoni algoritmi.

4.2.1 Dimensioni dei dati. Abbiamo già osservato come $\Omega(n)$ sia un banale limite inferiore per operazioni che richiedano l'esame di tutti gli n elementi di un insieme. Se questo risultato non è interessante per l'ordinamento, lo è per esempio nel problema della determinazione del massimo in un insieme: l'algoritmo 3.3 già discusso esegue infatti $n-1$ confronti, e in virtù della sua struttura, consistente in un ciclo principale che contiene un confronto in ogni passata, più un numero costante di operazioni al di fuori del ciclo, ha complessità: $T(n) = a(n-1) + b \in O(n)$. L'identità tra

questo limite superiore e il limite inferiore precedentemente indicato qualifica l'algoritmo 3.3 come ottimo in ordine di grandezza.

Questa situazione è comune a tutti i problemi che richiedono di eseguire un numero limitato di operazioni su molti dati: la semplice necessità di esaminare tutti i dati in ingresso, o generare tutti i dati in uscita, fissa un limite inferiore alla complessità di ordine pari al numero di tali dati, che in casi fortunati può essere significativo.

Per esempio l'addizione e la moltiplicazione di due matrici $n \times n$ richiedono di esaminare $2n^2$ dati di ingresso, e fissano quindi un limite inferiore di $\Omega(n^2)$. Nel caso dell'addizione tale limite è effettivamente raggiunto dall'algoritmo ovvio. Per la moltiplicazione tutti gli algoritmi noti appartengono a $O(n^k)$ con $k>2$ (vedi Bini e altri, 1979, autori del più interessante tra gli algoritmi recenti); vi è quindi ancora spazio per sperare in un miglioramento del metodo, o generare un nuovo più alto limite inferiore.

E' forse il caso di sottolineare che le argomentazioni basate sulla dimensione dei dati sono valide solo quando il problema richiede effettivamente l'esame di *tutti* i dati. Un esempio importante in cui ciò può non avvenire, che verrà esaminato tra breve, riguarda la ricerca di dati in un insieme.

4.2.2 Eventi contabili. Un altro criterio che permette a volte di stabilire un limite inferiore di complessità è legato al verificarsi di un evento per un numero contabile di volte tale che la sua ripetizione sia essenziale per la risoluzione del problema.

Prendiamo ad esempio la generazione di tutte le permutazioni di un insieme A, già largamente discussa. Un evento la cui ripetizione è richiesta dalla definizione stessa del problema è l'atto formale di nascita di una nuova permutazione. L'evento deve ovviamente ripetersi $n!$ volte, stabilendo così un limite inferiore di $\Omega(n!)$ al tempo richiesto da qualsiasi algoritmo di permutazione. La relazione [3.11], già derivata per l'algoritmo di permutazione 3.10, cioè $T(n) \in O(n!C(n))$, mostra una distanza ragionevole tra i due limiti se la funzione $C(n)$ si mantiene ragionevole.

Un esempio più interessante è nuovamente fornito dal problema di determinare il massimo tra n elementi mediante confronti (algoritmo 3.3). Volendo contare solo il numero di confronti, è lecito porre a tale numero un limite inferiore esattamente determinato, anziché specificato nel solo ordine di grandezza. Osserviamo anzitutto che ciascun elemento deve apparire in almeno un confronto, senza di che vi sarebbero elementi mai esaminati, e la determinazione del massimo sarebbe impossibile: ciò porta a stabilire un primo limite inferiore di $\lceil n/2 \rceil$ confronti. Argomentiamo poi,

come già facemmo a proposito dell'algoritmo 3.3, che ciascuno degli $n-1$ elementi non massimi deve uscire perdente da almeno un confronto, senza di che non può essere dichiarato non massimo. Poiché ogni confronto produce solo un perdente, stabiliamo il nuovo limite inferiore di $n-1$, che cancella il precedente $\lceil n/2 \rceil$, essendo a questo superiore. Ogni tentativo di ulteriori raffinamenti è ora inutile, poiché l'algoritmo 3.3 richiede per l'appunto $n-1$ confronti, ed è quindi, in tale rispetto, ottimo in assoluto.

4.2.3 L'albero di decisione. Nel primo capitolo abbiamo studiato un problema di pesate attraverso una peculiare struttura per la descrizione di algoritmi, nota come albero di decisione. Questa struttura è utilizzabile quando gli algoritmi sono basati su decisioni successive che superino in numero o in costo ogni altra azione, o che comunque diano una misura della complessità dell'algoritmo, poiché il loro numero è a questa proporzionale.

Definizioni e proprietà generali degli alberi saranno oggetto di un prossimo capitolo. Qui ricordiamo che l'albero di decisione rappresenta un algoritmo; ogni nodo *interno* rappresenta una decisione da prendere; ogni nodo terminale, o *foglia*, rappresenta la soluzione del problema per un particolare assetto iniziale dei dati; ogni percorso che dal nodo più alto, o *radice*, giunge a una foglia, indica la particolare esecuzione dell'algoritmo per giungere alla soluzione relativa alla foglia, e più in particolare la sequenza di decisioni prese nel percorso.

Si riesamini ad esempio l'albero di figura 4 nel capitolo 1. Già da allora avevamo sviluppato ragionamenti embrionali, che ci avevano permesso di trarre certe conclusioni sul minimo numero di operazioni necessarie a risolvere il problema delle dodici monete. E' nostro scopo portare ora questi ragionamenti a grande generalità.

Il caso pessimo nell'esecuzione di un algoritmo è quello che corrisponde al percorso radice-foglia di massima lunghezza o, il che è lo stesso, alla soluzione allocata nella foglia a distanza dalla radice, o *livello*, massima. L'albero di decisione che, fra tutti i possibili alberi relativi a un problema, minimizza la lunghezza massima dei percorsi, *fornisce con tale lunghezza un limite inferiore al numero di decisioni necessarie a risolvere il problema nel caso pessimo.*

Similmente il valor medio del numero di decisioni prese da un algoritmo è la media delle lunghezze di tutti i percorsi radice-foglia nel relativo albero. L'albero ove è minima tale lunghezza media *fornisce un limite inferiore al numero medio di decisioni* per risolvere il problema assegnato.

Gli alberi di decisione trovano naturale impiego nella rappresentazione di algoritmi basati su confronti. Spesso accade che il confronto $a : b$ sia sempre eseguito tra elementi distinti su cui è definita una relazione di ordinamento $<$, dando così luogo a non più di due risultati, che indicheremo con $a<b$ e $b<a$. L'albero che consegue è *binario*, cioè da ogni nodo si dipartono al più due rami che hanno distinte identità di ramo *sinistro* ($a<b$) e ramo *destro* ($b<a$). Nello studio dei limiti inferiori hanno particolare interesse gli alberi binari *perfettamente bilanciati*, cioè tali che da ogni nodo interno si dipartono sempre due rami, e che hanno tutte le foglie al medesimo livello (fig. 5). Indicando con 0 il livello della radice, e con successivi interi i livelli dei nodi via via più lontani dalla radice, e indicando con $\mathscr{B}_k$ l'albero perfettamente bilanciato che raggiunge il livello massimo k, si può con facile argomento induttivo provare la seguente

Proprietà *$\mathscr{B}_k$ ha $2^{k+1}-1$ nodi, di cui 2^k sono foglie.*

Per dimostrare la proprietà enunciata, verifichiamo la base dell'induzione notando che $\mathscr{B}_0$ ha $2^1-1=1$ nodi, di cui $2^0=1$ foglie. Costruiamo ora $\mathscr{B}_{k+1}$ aggiungendo a $\mathscr{B}_k$ uno strato di foglie, che discendono a coppie dalle foglie di $\mathscr{B}_k$: le foglie di $\mathscr{B}_{k+1}$ risultano cioè il doppio di quelle di $\mathscr{B}_k$, e ammettendo vera l'ipotesi induttiva per k, le foglie di $\mathscr{B}_{k+1}$ risultano $2\times 2^k=2^{k+1}$; i nodi di $\mathscr{B}_{k+1}$ sono pari in totale ai nodi di $\mathscr{B}_k$ più le foglie di $\mathscr{B}_{k+1}$, e cioè sono $2^{k+1}-1+2^{k+1}=2^{k+2}-1$. L'ipotesi induttiva è cioè vera per $k+1$, ed è quindi vera in genere.

Costruiamo ora un albero $\mathscr{H}$ che abbia 2^k foglie come $\mathscr{B}_k$, ma che

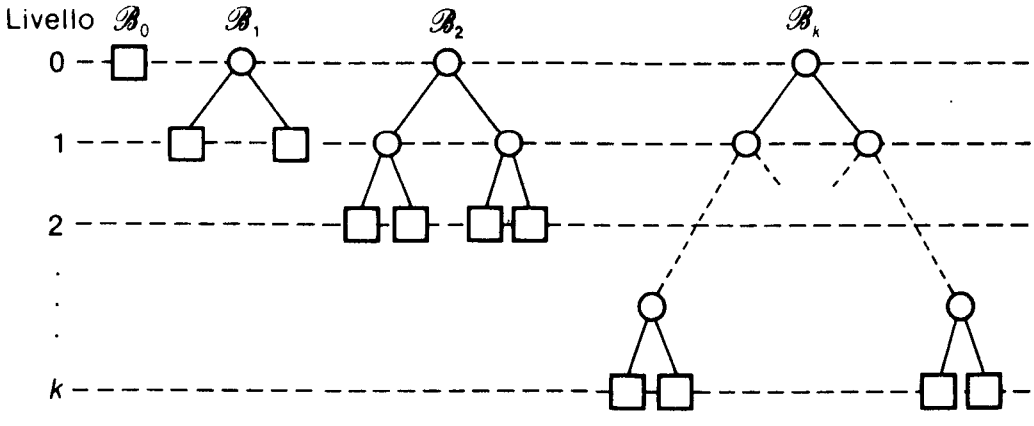

Figura 5
Alberi binari perfettamente bilanciati, ordinati per valori crescenti del livello delle foglie; nodi interni e foglie sono rappresentati rispettivamente con ○ e □.

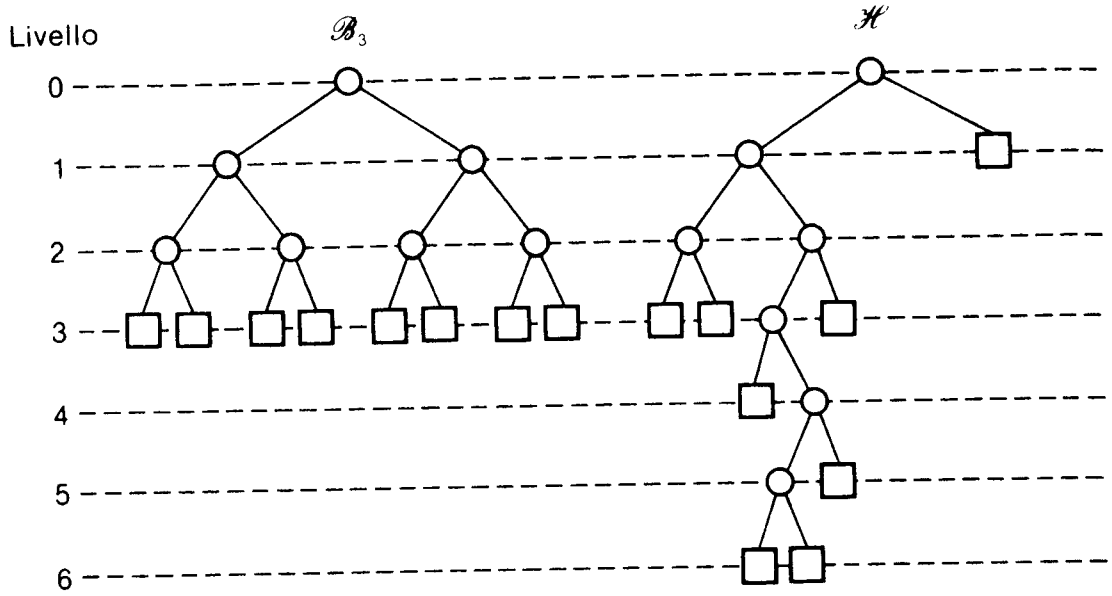

Figura 6
Un albero sbilanciato $\mathscr{H}$ con lo stesso numero di foglie di $\mathscr{B}_3$.

non sia perfettamente bilanciato. $\mathscr{H}$ contiene necessariamente foglie a livello minore di k e foglie a livello maggiore di k (fig. 6); se $\mathscr{B}_k$ e $\mathscr{H}$ sono alberi di decisione, cioè algoritmi, per lo stesso problema, è chiaro che sarà sempre il primo a minimizzare il livello massimo, quindi a stabilire un limite inferiore alla complessità in tempo nel caso pessimo.

E' inoltre possibile dimostrare che $\mathscr{B}_k$ minimizza anche la lunghezza media dei cammini tra radice e foglie, e su di esso si stabilisce quindi il limite inferiore alla complessità media.[3] Il lettore potrà convincersi di questo fatto osservando che per ogni percorso in $\mathscr{H}$ che termina a livello $h<k$, si paga il vantaggio di aver diminuito il livello di una foglia con la necessità di allocare 2^{k-h} foglie a livelli maggiori di k. Per esempio, nell'albero $\mathscr{H}$ di figura 6, la presenza di una foglia a livello 1 ha spinto $2^{3-1}=4$ foglie a livelli maggiori di 3.

Naturalmente non avviene in genere che un problema abbia un numero di soluzioni $s=2^k$ per qualche k, e quindi non esiste un albero di decisione nella forma $\mathscr{B}_k$. Si considera allora una struttura immediatamente derivata da $\mathscr{B}_k$: l'albero binario *quasi perfettamente bilanciato* $\mathscr{Q}_k$, che fino al livello $k-1$ contiene il numero massimo di nodi (2^k-1 in totale) e che al livello k contiene un numero di nodi (foglie) compreso tra 1 e

[3] Ciò è vero nell'ipotesi, che ammettiamo sempre valida, che le probabilità di seguire i diversi percorsi siano tutte uguali.

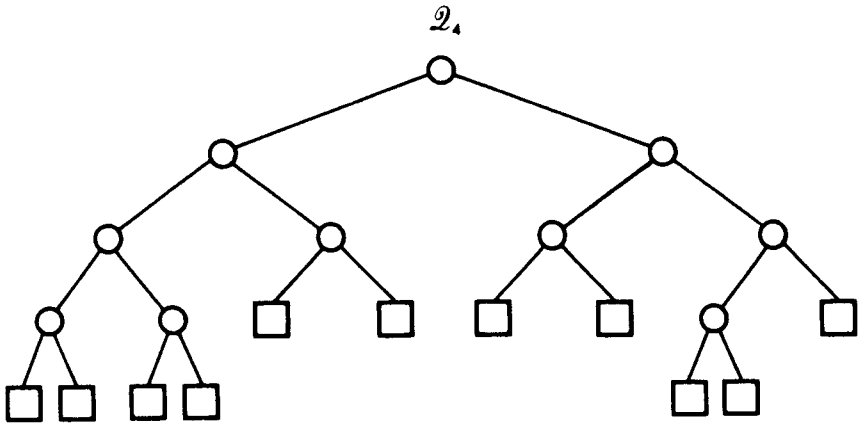

Figura 7
Un albero quasi perfettamente bilanciato per $k=4$: il numero di foglie s è compreso tra 2^3 e 2^4.

2^k (fig. 7). Un tale albero può essere costruito in molti modi diversi per qualunque numero di foglie s, con $2^{k-1} \leqslant s \leqslant 2^k$: se vale l'eguaglianza superiore, l'unico $\mathcal{Q}_k$ coincide con $\mathcal{B}_k$. Sarà così $\mathcal{Q}_k$ a rappresentare l'algoritmo ideale che minimizza le altezze massima e media, e per esso vale la relazione

$$\log_2 s \leqslant k \qquad [4.9]$$

che lega il numero s di soluzioni distinte (foglie) al numero massimo k di nodi di decisione incontrati nei percorsi radice-foglia.

Poiché un algoritmo di forma $\mathcal{Q}_k$ non necessariamente esiste per ogni problema, k costituisce un limite inferiore, non necessariamente raggiungibile, al massimo numero $D(n)$ di decisioni necessarie nel caso pessimo. La relazione [4.9] permette di esprimere tale limite in funzione del numero di soluzioni $s(n)$ del problema:

$$D(n) \geqslant \lceil \log_2 s(n) \rceil. \qquad [4.10]$$

Similmente il numero medio $D_M(n)$ di decisioni è limitato inferiormente da $k-1$, cioè:

$$D_M(n) \geqslant \lceil \log_2 s(n) \rceil - 1. \qquad [4.11]$$

Nei problemi ove $s(n)$ è funzione polinomiale di n, o, più precisamente, $s(n) \in \Omega(n^h)$ per $h \geqslant 0$, i limiti [4.10] e [4.11] sono nell'ordine di $\Omega(h \log_2 n)$, evidentemente molto bassi e sovente non significativi. Per esempio il problema, già tante volte discusso, della determinazione del massimo di un

insieme A per confronti tra elementi dell'insieme, ha $s(n)=n$ soluzioni: infatti il massimo può coincidere con $A(1), A(2), ..., A(n)$. La relazione [4.10] genera il limite inferiore $\lceil \log_2 n \rceil$ al numero di confronti, chiaramente non interessante, poiché largamente superato dal limite $n-1$ stabilito per altra via (§ 4.2.2). Ciò significa che qualunque algoritmo per la risoluzione di questo problema ha un corrispondente albero di decisione sostanzialmente sbilanciato, ben lungi quindi da $\mathcal{Q}_k$.

Una situazione simile si riscontra nella determinazione contemporanea del primo e secondo elemento di A (algoritmo 3.5). Il problema ha $s(n)=n(n-1)$ soluzioni, tante quante sono le coppie ordinate di n elementi, e la relazione [4.10] genera il limite inferiore $\lceil \log_2 n + \log_2 (n-1) \rceil$, nuovamente non interessante, poiché superato persino dal limite $n-1$ che si applica alla determinazione del solo massimo.

Ragionamenti analoghi valgono per i valori medi prodotti dalla [4.11].

L'albero di decisione trova in genere applicazioni interessanti nei casi in cui $s(n) \in \Omega(f(n)^{g(n)})$, ove $f(n)$ può tipicamente ridursi a una costante: la riduzione logaritmica contenuta nelle relazioni [4.10] e [4.11] limita il numero di decisioni con il valore $\Omega(g(n) \log_2 f(n))$. Un esempio importante è quello dell'ordinamento degli elementi di un insieme A, già affrontato nell'algoritmo 3.8 QUICKSORT. In questo problema le soluzioni sono $s(n)=n!$: infatti qualsiasi permutazione di A può essere la permutazione ordinata, e quindi la soluzione, in relazione all'arrangiamento iniziale degli elementi di A. Dalla relazione [4.10], sostituendo alla funzione fattoriale il valore [4.6] della formula di Stirling, si ha

$$D(n) \geqslant \lceil \log_2 n! \rceil$$
$$\cong \lceil \log_2 \sqrt{2\pi n} + n \log_2 (n/e) \rceil,$$

da cui

$$D(n) \in \Omega(n \log_2 n). \qquad [4.12]$$

Similmente dalla [4.11] si ottiene

$$D_M(n) \in \Omega(n \log_2 n). \qquad [4.13]$$

I risultati [4.12] e [4.13] sono importantissimi, poiché l'ordinamento è problema importante per sé, e appare come parte integrante di molti altri problemi. Tali risultati affermano che non si possono ordinare n elementi con meno di $\Omega(n \log_2 n)$ confronti, e questo limite vale anche in media. E' stato mostrato a suo tempo come l'algoritmo 3.8 QUICKSORT richieda un numero di confronti limitato superiormente da $O(n \log_2 n)$ nel caso medio (relazione [3.7]), e da $O(n^2)$ nel caso pessimo (relazione [3.9]).

Ciò significa che QUICKSORT è un algoritmo ottimo nel caso medio, ma non è necessariamente efficiente nel caso pessimo. In effetti vedremo in un prossimo paragrafo come possa costruirsi un algoritmo di ordinamento che richiede $O(n \log_2 n)$ confronti anche nel caso pessimo, e che pertanto è ottimo in tale caso in virtù della [4.12].

La raggiungibilità del limite [4.12] implica che sia possibile costruire un algoritmo di ordinamento sotto forma di albero di decisione quasi perfettamente bilanciato, o comunque così prossimo a questa forma da mantenere un legame logaritmico tra k e $s(n)$. Per esempio, per ordinare tre elementi si può usare l'algoritmo di figura 8, ove risulta: $s(n)=3!=6$; $D(n)=3$ (in accordo con la [4.10]); $D_M(n)=2,\overline{6}$ (in accordo con la ([4.11]).

Ricordiamo ora che le relazioni [4.10] e [4.11] sono state ricavate per alberi binari, e sono quindi valide quando ogni decisione genera al più due risultati. Nel caso generale in cui le decisioni possono aprire fino a r alternative distinte, $r \geqslant 2$, la determinazione dei limiti inferiori deve essere condotta su alberi r-ari, e le relazioni assumono la forma generale

$$D(n) \geqslant \lceil \log_r s(n) \rceil; \qquad [4.10']$$

$$D_M(n) \geqslant \lceil \log_r s(n) \rceil - 1. \qquad [4.11']$$

Oltre al valore $r=2$ è comune il valore $r=3$, che si presenta nel caso di confronti $a : b$ che possano dare i tre risultati: $a<b$, $a=b$, $b<a$. Abbiamo in effetti già impiegato un albero ternario di decisione nel problema delle dodici monete (cap. 1).

Per concludere prendiamo in esame il problema della ricerca di un ele-

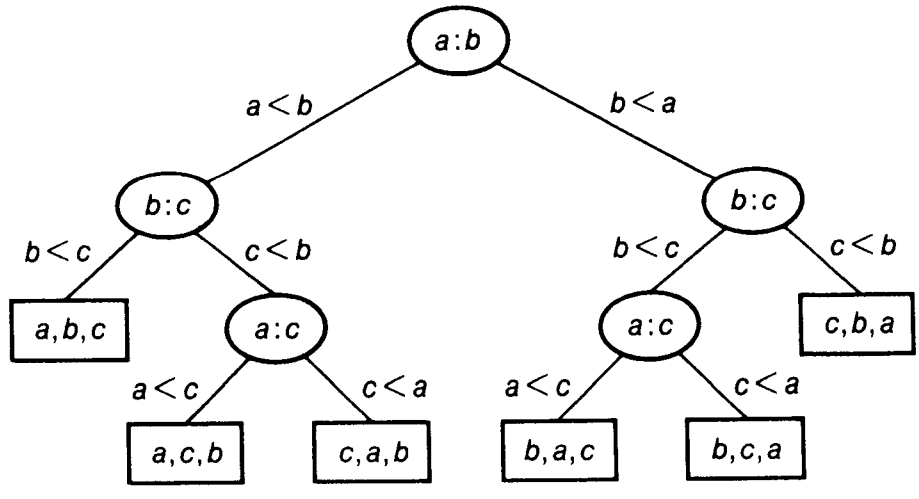

Figura 8
Albero di decisione per l'ordinamento di tre elementi a, b, c. Le foglie contengono le sei possibili soluzioni.

mento assegnato z in un insieme A. Per risolverlo abbiamo proposto vari algoritmi nel capitolo 2, che richiedono tutti una scansione più o meno intelligente dell'insieme, e hanno quindi una complessità limitata inferiormente da $\Omega(n)$, nel caso peggiore. Volendo generare un limite inferiore mediante l'albero di decisione, notiamo che le soluzioni del problema sono $s(n)=n+1$, corrispondenti agli n casi in cui z coincide con un elemento di A, più il caso in cui z non è contenuto nell'insieme. I confronti $z : A(i)$ danno tre possibili risultati, e si applica quindi per $r=3$ la relazione [4.10′] che stabilisce un limite inferiore di $[\log_3(n+1)]$ confronti.

Vedremo in effetti nel prossimo capitolo come il problema possa essere risolto in tempo logaritmico in n se l'insieme è stato preventivamente ordinato.