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.

4.2.4 L'oracolo. La teoria della complessità riserva il nome di *oracolo* a un avversario dispettoso e sleale, che divina senza sosta la situazione più sfavorevole in cui può operare l'algoritmo, e gliela presenta ad ogni passo. Combattendo contro l'oracolo si segue il percorso risolutivo più lungo, e si può quindi definire il limite inferiore alla complessità di caso pessimo. Il problema, naturalmente, è scoprire l'oracolo.

Poiché per individuare l'oracolo non esistono criteri generali, discuteremo come operare in un caso specifico piuttosto interessante, lasciando all'intuito del cercatore di oracoli la scelta del comportamento da adottare in altre situazioni. Studiamo dunque il *problema del torneo*, che ha interessato diversi matematici, primo tra tutti Lewis Carroll, che intese proporre le sue argomentazioni, non si sa con quale successo, nella stagione tennistica inglese del 1883. (Il suo saggio apparve il primo agosto di quell'anno sulla "St. James' Gazette".) In un torneo a eliminazione diretta n giocatori si affrontano: chi perde un incontro è subito eliminato. Quale che sia il gioco, si postula la transitività della bravura: così, se a perde un incontro con b ($a<b$), e b perde un incontro con c ($b<c$), si conclude che a perderebbe comunque con c, e non è quindi necessario disputare l'incontro (cioè per transitività si pone $a<c$, risultato che si ha comunque se a e c si affrontano direttamente).

Il primo, semplice quesito è trovare il vincitore. E' immediato costatare che questo è il problema della determinazione del massimo di un insieme, già risolto nell'algoritmo 3.3. Sappiamo che questo algoritmo esegue $n-1$ confronti, e che pertanto è ottimo in assoluto (§ 4.2.2). Tuttavia nei tornei gli incontri si organizzano in modo diverso da quello dettato dall'algoritmo 3.3, ovvero si esegue un diverso algoritmo. Posto per semplicità $n=2^k$, i giocatori si affrontano a coppie in una prima serie di $n/2$ incontri; eliminati i perdenti, si accoppiano i vincitori per una nuova serie di $n/4$ incontri; si procede similmente con un numero di vincitori di volta in volta dimezzato, fino ai quarti di finale (8 giocatori), alle semifinali (4 giocatori) e alla finale (2 giocatori) da cui emerge il vincitore del torneo.

L'organizzazione degli incontri è rappresentata nel noto "tabellone" del torneo (fig. 9), che è di fatto un albero binario perfettamente bilanciato $\mathscr{B}_k$ ($k=\log_2 n$), costruito a partire dalle foglie ove sono allocati tutti i giocatori. Ciascuna coppia di foglie è connessa a un nodo al livello precedente, ove si alloca il vincitore dell'incontro tra le due foglie, e si procede similmente fino alla radice che contiene il vincitore della finale, ovvero il campione.

Osserviamo che l'algoritmo implicitamente contenuto nel tabellone richiede tanti incontri quanti sono i nodi interni di $\mathscr{B}_k$. I quali, per una proprietà provata nel paragrafo precedente, sono $2^k-1=n-1$ (vedi fig. 9). Anche questo algoritmo è dunque ottimo in assoluto, e ha il pregio di impegnare ogni giocatore per non più di k incontri, mentre nell'algoritmo 3.3 il vincitore fa in genere $\Omega(n)$ incontri.

Il vincitore della finale è proclamato a buon diritto vincitore del torneo, in virtù della transitività della bravura. Ma chi è il secondo? Si suole assegnare questo titolo all'altro finalista ma, come osservò Lewis Carroll, vi è un'alta probabilità di commettere una ingiustizia. Se infatti colui che è veramente il secondo in bravura capita nella stessa "metà" del tabellone del primo, sarà da questi eliminato prima di giungere alla finale: è il caso dei giocatori n, p, k della figura 9, che sono sconfitti da m nei turni iniziali e

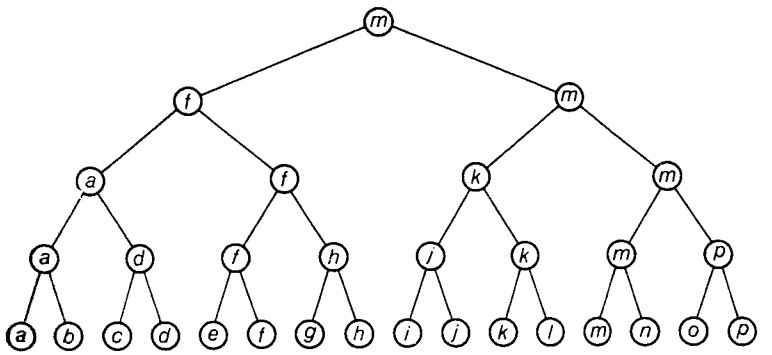

Figura 9
Tabellone di un torneo in forma $\mathscr{B}_4$ tra $2^4=16$ giocatori. m è il migliore; chi è il **secondo**?

non possono confrontarsi né direttamente né indirettamente con il secondo finalista f. Più precisamente, nota nel tabellone la posizione iniziale del più bravo, vi sono $n-1$ posizioni iniziali dove può essere allocato (poniamo con pari probabilità) il secondo in bravura; di queste posizioni, $n/2$ gli consentono di accedere alla finale, mentre le altre $n/2-1$ non glielo consentono. Ne segue che la probabilità che il secondo finalista sia effettivamente il secondo in bravura è $(n/2)/(n-1)$, ovvero vi è circa il 50 per cento di probabilità di premiare un abusivo.

Nasce così il problema del torneo: come determinare, oltre al primo, anche il secondo, il terzo, ..., l'r-esimo giocatore con il minimo numero totale di confronti. (Determinare la posizione di *tutti* i giocatori equivale a determinare l'ordinamento, problema che trattiamo a parte.) Ci limitiamo qui a discutere la determinazione del primo e del secondo, problema per cui abbiamo del resto già fornito una soluzione come esempio di procedura ricorsiva (procedura PRIMSEC, algoritmo 3.5).

Ricapitoliamo ciò che abbiamo scoperto finora. L'algoritmo 3.5 esegue $3n/2-2$ confronti (soluzione delle relazioni [3.1]). Un limite inferiore banale e quasi certamente irraggiungibile è $n-1$, numero di confronti indispensabili a trovare solo il primo, mentre nulla di interessante si ottiene impiegando l'albero di decisione (§ 4.2.3). Si tratta dunque di trovare un algoritmo più efficiente dell'algoritmo 3.5, o di trovare un limite inferiore più realistico di $n-1$, o di fare l'uno e l'altro.

Riprendendo ragionamenti già fatti sul tabellone, possiamo facilmente concludere che il secondo deve essere ricercato tra tutti e soli coloro che hanno perduto in un incontro diretto con il primo. Si tratta di $k=\log_2 n$ giocatori, poiché tanti sono gli incontri disputati dal primo (in fig. 9 i candidati al secondo posto sono n, p, k, f). Possiamo allora determinare il secondo attraverso un nuovo torneo di consolazione riservato a questi $\log_2 n$ candidati, che con il medesimo algoritmo richiede $\log_2 n-1$ incontri (fig. 10). L'algoritmo complessivo, che chiameremo del *doppio torneo*, richiede un numero di incontri per determinare il primo e il secondo pari a

$$C(n)=n+\log_2 n-2, \qquad [4.14]$$

con notevole miglioramento rispetto all'algoritmo 3.5.

Dimostriamo ora che la relazione [4.14] fornisce anche un limite inferiore al numero di incontri nel caso peggiore; cioè l'algoritmo del doppio torneo è ottimo. In questa dimostrazione interverrà l'oracolo.

Come già notato, poiché non si può affermare che un giocatore non è il campione finché non viene sconfitto, qualsiasi algoritmo deve contenere

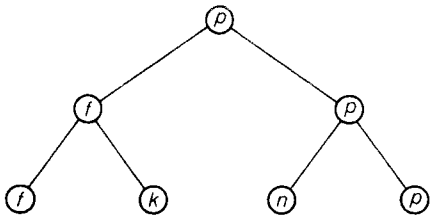

Figura 10
Girone di consolazione per il torneo di figura 9; p è il secondo dopo m.

almeno $n-1$ incontri in cui tutti i giocatori, tranne il campione, escono sconfitti. Dall'insieme $\mathcal{T}$ di tutti gli incontri disputati togliamo un sottoinsieme $\mathcal{I}$ di $n-1$ incontri con le caratteristiche suddette, includendo in $\mathcal{I}$ tutti gli incontri cui ha partecipato il campione. Se l'insieme $\mathcal{I}$ contiene gli incontri $a:c$ e $b:c$, con risultati $a<c$ e $b<c$, allora $\mathcal{I}$ non può contenere l'incontro $a:b$: infatti da ogni incontro di $\mathcal{I}$ deve uscire un diverso perdente, mentre a e b hanno già perso in $\mathcal{I}$. Quindi tutti gli incontri diretti tra giocatori che in $\mathcal{I}$ hanno perso contro il campione devono comparire in $\mathcal{T}-\mathcal{I}$. Detto y il numero di questi giocatori, occorrono almeno $y-1$ incontri diretti per determinare chi di essi sia il migliore, cioè il secondo in assoluto: quindi $\mathcal{T}-\mathcal{I}$ deve contenere almeno $y-1$ incontri, ovvero $\mathcal{T}$ deve contenerne

$$|\mathcal{T}|\geqslant n+y-2. \qquad [4.15]$$

Si tratta ora di determinare un limite inferiore per y, cioè il numero minimo di incontri che il campione deve necessariamente affrontare nel caso pessimo.

Anzitutto notiamo che $\mathcal{T}$ non conterrà mai un incontro $a:b$ se la relazione tra a e b è derivabile per transitività da altri incontri già effettuati in $\mathcal{T}$. Nel rispetto di questa regola, sarebbe auspicabile che nell'incontro tra a, che ha vinto molto, e b, che ha molto perduto, risultasse $a<b$, onde stabilire per transitività, cioè senza altri incontri, una lunga catena di relazioni tra il primo dei giocatori migliori di b e l'ultimo dei giocatori peggiori di a. A questo si oppone l'oracolo che prevede il risultato di ogni confronto in modo da minimizzare il numero delle addizionali relazioni derivabili per transitività.

L'oracolo stabilisce che nell'incontro $a:b$ risulti $a<b$ se e solo se il numero di giocatori che sono già stati riconosciuti peggiori di b è maggiore

o uguale al numero di quelli peggiori di a. I più forti, cioè, seguitano a vincere aggiungendo il minimo di informazione al sistema. In queste condizioni il campione c effettua il primo incontro con un giocatore a che non ha ancora mai vinto (in caso contrario c perderebbe con a). Ora c ha un elemento peggiore, $a<c$, e può incontrare un nuovo giocatore b che ha al più un elemento peggiore, $f<b$. Risulta così $b<c$ dall'incontro e $f<c$ per transitività, quindi c ha adesso al più tre elementi peggiori a, b, f. Al prossimo incontro c batterà un giocatore che ne ha al più tre peggiori, cosicché gli elementi peggiori di c diverranno al più sette. Proseguendo nel ragionamento è facile vedere che il numero massimo $M(j)$ di giocatori riconosciuti peggiori del campione, dopo che questo ha effettuato j incontri, è determinato dalle relazioni di ricorrenza

$$M(j)=0, \qquad \text{per} \quad j=0,$$
$$M(j)=2M(j-1)+1, \qquad \text{per} \quad j>0,$$

la cui soluzione, determinabile per successive sostituzioni, risulta

$$M(j)=2^j-1.$$

Per stabilire che gli altri $n-1$ giocatori sono a lui inferiori il campione dovrà quindi effettuare un numero y di incontri tale che $M(y)\geqslant n-1$, ovvero $2^y-1\geqslant n-1$, da cui

$$y\geqslant \log_2 n.$$

Questo valore, sostituito nella relazione [4.15], determina il limite inferiore al numero totale di incontri:

$$|\mathscr{T}|\geqslant n+\log_2 n-2, \qquad [4.16]$$

che è pari al valore [4.14] richiesto dall'algoritmo del doppio torneo. Questo algoritmo è quindi ottimo in assoluto.

Capitolo 5

Valutazione della complessità

In diverse occasioni abbiamo calcolato il numero di operazioni eseguite da un algoritmo, in funzione della dimensione n del problema trattato. Gli argomenti adottati sono stati vari e sempre contingenti; in particolare abbiamo costruito relazioni di ricorrenza per esprimere il numero di operazioni $T(n)$ in funzione di $T(n')$, valutato su dimensioni $n'<n$.

Ci occuperemo ora di calcolare la complessità di un algoritmo seguendo metodi il più possibile generali. Gran parte del nostro studio sarà focalizzato sulle relazioni di ricorrenza. Esse nascono infatti quando l'elemento generico di una sequenza può essere espresso in funzione di elementi precedenti, e costituiscono quindi un mezzo naturale per formulare la complessità di programmi ricorsivi, intervenendo in generale ovunque il concetto di ricorsività consente di descrivere in modo compatto la formazione di una sequenza.

In mancanza di metodi generali le relazioni di ricorrenza sono state fin qui risolte per sostituzioni successive finché, risultata chiara la struttura della sequenza, è stato possibile individuarne la somma. Illustreremo qui nel seguito alcune famiglie di relazioni particolarmente importanti nell'analisi di algoritmi e in problemi a questa correlati, e mostreremo come possano essere sistematicamente risolte.

5.1 Relazioni di ricorrenza lineari di ordine costante

La più famosa e forse la più antica relazione di ricorrenza fu posta nel *Liber abaci* di Fibonacci in relazione a un problema ideale sulla riproduzione dei conigli.[1] La successione che ne derivò, detta oggi dei *numeri di*

[1] Leonardo di Bonaccio da Pisa, più noto come Fibonacci, fu il più grande mate-