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 $\lceil \log_3(n+1) \rceil$ 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

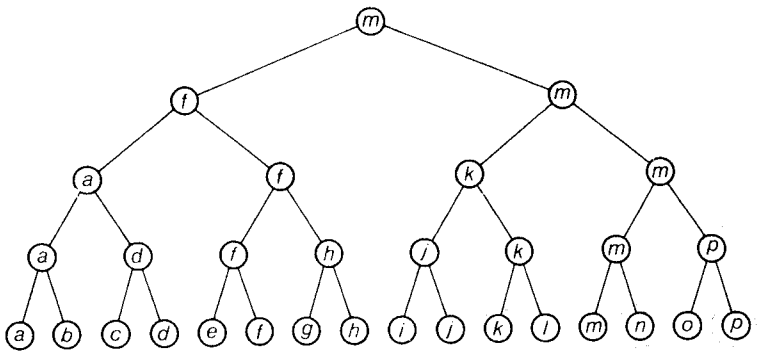


**Figura 9**
Tabellone di un torneo in forma $\mathscr{B}_4$ tra $2^4=16$ giocatori. $m$ è il migliore; chi è il secondo?

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 in-