

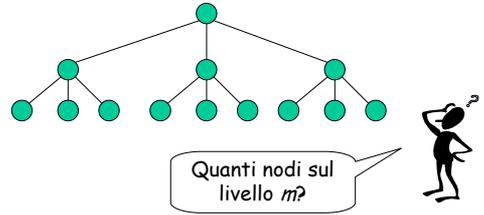
Algoritmi e Laboratorio a.a. 2005-06  
 Lezioni

prof. Elio Giovannetti

Parte 7 - Delimitazione inferiore per il problema dell'ordinamento  
 basato su confronti.

Una nozione preliminare:  
 albero  $k$ -ario completo

Un albero  $k$ -ario è **completo** se tutti i nodi interni hanno  $k$  figli, e tutte le foglie sono sullo stesso livello



Un albero  $k$ -ario è **completo** se tutti i nodi interni hanno  $k$  figli, e tutte le foglie sono sullo stesso livello

- Livello 0: 1 (la radice)
- Livello 1:  $k$  (i figli della radice)
- Livello 2:  $k \cdot k = k^2$
- ...
- Livello  $m$ :  $k^m$

La prova è per induzione su  $m$

Un albero  $k$ -ario è **completo** se tutti i nodi interni hanno  $k$  figli, e tutte le foglie sono sullo stesso livello

- Proprietà:
- le foglie di un albero  $k$ -ario completo di altezza  $h$  sono  $k^h$
  - se un albero  $k$ -ario completo ha  $n$  foglie, allora ha altezza  $h = \log_k n$

Quanti sono i nodi interni se l'altezza è  $h$ ?

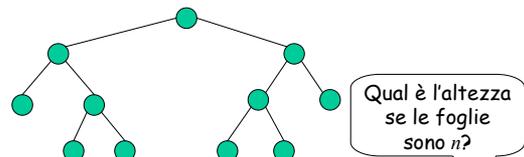
Un albero  $k$ -ario è **completo** se tutti i nodi interni hanno  $k$  figli, e tutte le foglie sono sullo stesso livello

$$\text{Nodi interni} = 1 + k + k^2 + \dots + k^{h-1} = \sum_{i=0}^{h-1} k^i = \frac{k^h - 1}{k - 1}$$

Allora la cardinalità di un albero  $k$ -ario completo di altezza  $h$  è

$$\frac{k^{h+1} - 1}{k - 1}$$

Un'altra nozione preliminare: albero binario quasi completo



Un albero binario **quasi completo** è completo sino al penultimo livello

$$2^{h-1} < n \leq 2^h$$

quindi  $h = \lceil \log_2 n \rceil$

## Il problema dell'ordinamento.

Osservazione: Per ordinare una sequenza di  $n$  elementi bisogna almeno esaminare tutti gli elementi (anche solo per verificare che la sequenza è già ordinata). La complessità del problema ha quindi una delimitazione inferiore (in ogni caso)  $\Omega(n)$ .

Abbiamo finora studiato quattro metodi di ordinamento:

- selection sort (ordinamento per selezione):  $T(n) = \Theta(n^2)$
- insertion sort (ord. per inserimento):  $T_{\text{medio}}(n) = \Theta(n^2)$
- mergesort (ord. per fusione):  $T(n) = \Theta(n \log n)$
- quicksort (ord. rapido di Hoare):  $T_{\text{medio}}(n) = \mathcal{O}(n \log n)$

Domanda: Esistono o possono esistere algoritmi di ordinamento di complessità (temporale) inferiore a  $n \log n$ ?

08/11/2005

E. Giovannetti - AlgELab-05-06 - Lez.07

7

## Il problema dell'ordinamento in generale.

- Immaginiamo di voler ordinare una sequenza di  $n$  elementi di cui sappiamo in anticipo:
  - esattamente quali sono i valori identificativi (chiavi) di tali elementi;
  - qual è il loro ordine.
- Esempio: Un array di 50 oggetti "Articolo" distinti, contenenti ciascuno un numero di codice da 0 a 49 (articoli distinti hanno codici distinti). Per ordinarlo è sufficiente:
  - disporre di un array ausiliario  $b$  della stessa lunghezza;
  - percorrere l'array di partenza andando a mettere nell'array ausiliario ogni elemento al suo posto, cioè l'articolo di codice  $h$  nell'elemento  $b[h]$ ;
  - ovviamente ciò richiede un tempo  $\Theta(n)$ ;
  - l'ordinamento è stato effettuato senza operare nessun confronto.

08/11/2005

E. Giovannetti - AlgELab-05-06 - Lez.07

8

## Il problema dell'ordinamento per confronti.

- Un algoritmo come il precedente non è utilizzabile in generale, quando non è noto a priori quali siano i singoli elementi della sequenza da ordinare (presi da un insieme potenzialmente infinito, quale quello degli interi, o dei reali, ecc.).
- Consideriamo quindi il problema dell'ordinamento assumendo che l'informazione sull'ordine degli elementi possa essere ottenuta solo mediante confronti.
- Ci chiediamo se tale problema (ordinamento per confronti) può essere risolto da un algoritmo di complessità inferiore a  $n \log n$  nel caso peggiore.

08/11/2005

E. Giovannetti - AlgELab-05-06 - Lez.07

9

## Delimitazione inferiore della complessità del caso peggiore.

- Il problema: data una sequenza di  $n$  elementi, trovarne una permutazione che sia ordinata.
- Trascuriamo il tempo necessario per gli spostamenti di elementi, gli incrementi di indici, ecc., e contiamo solamente il minimo numero di confronti necessari per trovare (cioè per determinare quale sia) la permutazione ordinata, nel caso peggiore.
- Il numero di istruzioni elementari di un qualunque algoritmo di ordinamento non può essere inferiore a tale numero.

08/11/2005

E. Giovannetti - AlgELab-05-06 - Lez.07

10

## Delimitazione inferiore della complessità del caso peggiore.

- Il numero delle possibili permutazioni di  $n$  elementi è:  
 $n(n-1)(n-2) \dots 2 \cdot 1 = n!$
- L'esito di ogni successivo confronto in generale restringe l'insieme delle permutazioni fra cui si trova la soluzione.
- Esempio:  
dato l'insieme di tre elementi distinti  $\{a, b, c\}$ , a seconda del risultato del confronto fra  $a$  e  $b$  si ha:  
 $a < b$  soluzioni possibili le sequenze:  $c a b \quad a c b \quad a b c$   
 $a > b$  soluzioni possibili le sequenze:  $c b a \quad b c a \quad b a c$
- Ogni confronto ha due possibili risultati: un insieme di possibili sequenze di confronti può quindi essere rappresentato come un albero binario (l'albero delle decisioni).

08/11/2005

E. Giovannetti - AlgELab-05-06 - Lez.07

11

## L'albero delle decisioni (o di decisione).

È un albero binario in cui:

- Ogni nodo interno rappresenta un confronto.
  - Ogni foglia rappresenta una permutazione-risultato.
- Fissata la lunghezza  $n$  della sequenza:
- a ogni algoritmo di ordinamento per confronti corrisponde un particolare albero di decisione;
  - a ciascuna possibile esecuzione di un algoritmo corrisponde un particolare cammino nel corrispondente albero, dalla radice ad una foglia.
- Attenzione: l'albero di decisione **non** è un albero di ricorsione!

L'albero di decisione che minimizza l'altezza fornisce un confine inferiore al numero di decisioni necessarie nel caso peggiore.

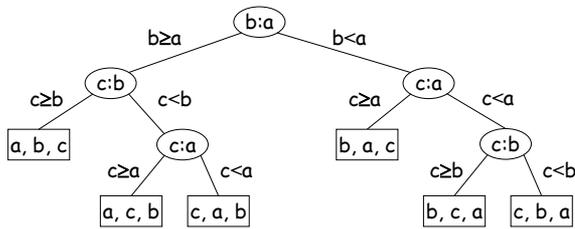
08/11/2005

E. Giovannetti - AlgELab-05-06 - Lez.07

12

Esempio: l'albero di decisione per l'insertion sort, su una sequenza di 3 elementi.

Indichiamo con  $x:y$  l'operazione di confronto fra  $x$  e  $y$ .  
Sequenza di ingresso:  $a, b, c$ .



## Algoritmi e alberi di decisione

- Data la lunghezza  $n$  della sequenza, algoritmi diversi determinano alberi di decisione diversi, ma sempre con almeno  $n!$  foglie (se l'algoritmo è inefficiente vi possono essere più foglie corrispondenti ad una stessa permutazione).
- L'esecuzione di un particolare algoritmo su una particolare sequenza di lunghezza  $n$  determina un cammino nell'albero dalla radice a una foglia.
- Il tempo nel caso peggiore per quell'algoritmo è quindi  $\geq$  alla lunghezza del cammino più lungo, cioè  $\geq$  all'altezza dell'albero
- Ma qualunque sia l'algoritmo, l'albero ha almeno  $n!$  foglie, quindi ha altezza non inferiore a  $\log(n!)$ .
- Il cammino **più lungo**, sull'albero **meno alto** (fra quelli relativi a tutti i possibili algoritmi di ordinamento), ha quindi lunghezza  $\log(n!)$ .

Una delimitazione inferiore per il problema dell'ordinamento.  
Riassumendo:

- Le permutazioni di  $n$  oggetti sono  $n!$
- L'albero delle decisioni per il problema dell'ordinamento ha allora  $n!$  foglie, ed ha altezza minima se è quasi completo
- Dunque una delimitazione inferiore per l'ordinamento è  $\Omega(\log n!)$



Che cos'è  $\log(n!)$  ?

Che cosa è  $\log(n!)$  ?

$$n! = n(n-1)(n-2) \dots 2 \cdot 1 \geq n(n-1)(n-2) \dots \lceil n/2 \rceil \geq (n/2)^{n/2}$$

quindi

$$\log(n!) \geq \log((n/2)^{n/2}) = (n/2) \log(n/2) = (n/2) \log(n/2) = (1/2) n \log n - (1/2)n \quad (\text{ricorda che } \log_2 \frac{1}{2} = -1)$$

d'altra parte:

$$n! = n(n-1)(n-2) \dots 2 \cdot 1 \leq n \cdot n \cdot n \dots n = n^n$$

quindi

$$\log(n!) \leq \log(n^n) = n \log n$$

$$\log(n!) = \Theta(n \log n)$$

Complessità del problema nel caso peggiore:

$$T_{\text{worst}}(n) = \Omega(n \log n)$$

Un altro modo di valutare  $\log(n!)$

La formula di Stirling

$$\log_2 n! \approx \log_2 (\sqrt{2\pi n} (n/e)^n) = \log_2 \sqrt{2\pi n} + n \log_2 (n/e)$$



Allora  $\log(n!) \in \Theta(n \log n)$

Ve la ricordavate la formula di Stirling?



È facile vedere che è anche  $T_{\text{medio}}(n) = \Omega(n \log n)$ , perché la lunghezza media dei cammini dalla radice a una foglia, sull'albero meno alto, è  $n \log n$ .

Grazie a quest'ultimo risultato, possiamo affermare che la complessità del caso medio per il quicksort, precedentemente valutata in  $O(n \log n)$ , è in realtà  $\Theta(n \log n)$ .

Conclusione:

- il quicksort è ottimale nel caso medio;
- il mergesort è ottimale nel caso medio e peggiore.