

Analisi Teorica: Schemi di Ordinamento

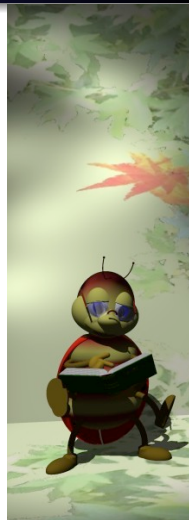
Claudio Mirolò

Dipartimento di Scienze Matematiche, Informatiche e Fisiche
Università di Udine

Laboratorio PLS

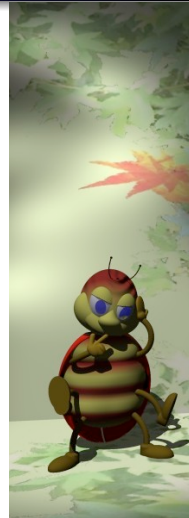
Sommario

- 1 **Problema**
 - in generale
- 2 **Informazioni**
 - analisi
 - cogliere il significato
- 3 **Algoritmi**
 - insertion sort
 - quick sort
- 4 **Epilogo**



Trama

- 1 **Problema**
 - in generale
- 2 **Informazioni**
 - analisi
 - cogliere il significato
- 3 **Algoritmi**
 - insertion sort
 - quick sort
- 4 **Epilogo**





Ordinamento in generale

- Ordinare n oggetti per *chiave* crescente
- unicamente sulla base del confronto (delle chiavi) di coppie di oggetti



Ordinamento in generale

- Ordinare n oggetti per *chiave* crescente
- unicamente sulla base del confronto (delle chiavi) di coppie di oggetti

Ordinamento in generale

- Ordinare n oggetti per *chiave* crescente
- unicamente sulla base del confronto (delle chiavi) di coppie di oggetti





Facciamo ordine. . .

- Quale strategia?
- Quanti confronti sono necessari nel peggiore dei casi?
- Ci sono strategie migliori da questo punto di vista?
- Si potrebbe fare ancora meglio?
- C'è un limite ineludibile?



Facciamo ordine. . .

- Quale strategia?
- Quanti confronti sono necessari nel peggiore dei casi?
- Ci sono strategie migliori da questo punto di vista?
- Si potrebbe fare ancora meglio?
- C'è un limite ineludibile?



Facciamo ordine...

- Quale strategia?
- Quanti confronti sono necessari nel peggiore dei casi?
- Ci sono strategie migliori da questo punto di vista?
- Si potrebbe fare ancora meglio?
- C'è un limite ineludibile?



Facciamo ordine...

- Quale strategia?
- Quanti confronti sono necessari nel peggiore dei casi?
- Ci sono strategie migliori da questo punto di vista?
- Si potrebbe fare ancora meglio?
- C'è un limite ineludibile?



Facciamo ordine. . .

- Quale strategia?
- Quanti confronti sono necessari nel peggiore dei casi?
- Ci sono strategie migliori da questo punto di vista?
- Si potrebbe fare ancora meglio?
- C'è un limite ineludibile?

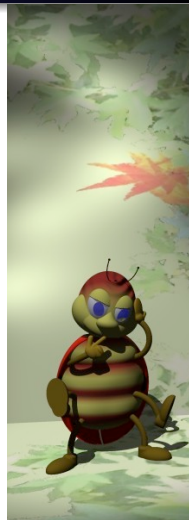


Facciamo ordine...

- Quale strategia?
- Quanti confronti sono necessari nel peggiore dei casi?
- Ci sono strategie migliori da questo punto di vista?
- Si potrebbe fare ancora meglio?
- C'è un limite ineludibile?

Trama

- 1 Problema
 - in generale
- 2 **Informazioni**
 - **analisi**
 - **cogliere il significato**
- 3 Algoritmi
 - insertion sort
 - quick sort
- 4 Epilogo





Informazioni

- In quanti modi diversi si possono mescolare n oggetti?



Informazioni

- In quanti modi diversi si possono mescolare n oggetti?
- Quanti sono i potenziali “percorsi” di ordinamento diversi?



Informazioni

- In quanti modi diversi si possono mescolare n oggetti?
- Quanti sono i potenziali “percorsi” di ordinamento diversi?
- Che *informazione* rivela un singolo confronto?



Informazioni

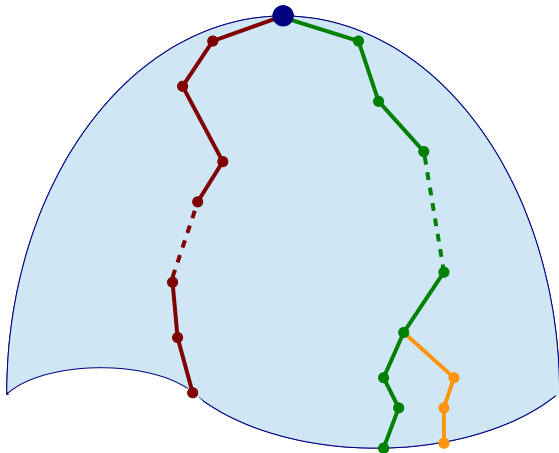
- In quanti modi diversi si possono mescolare n oggetti?
- Quanti sono i potenziali “percorsi” di ordinamento diversi?
- Che *informazione* rivela un singolo confronto?
- Che relazione fra *informazione* e numero di casi possibili?



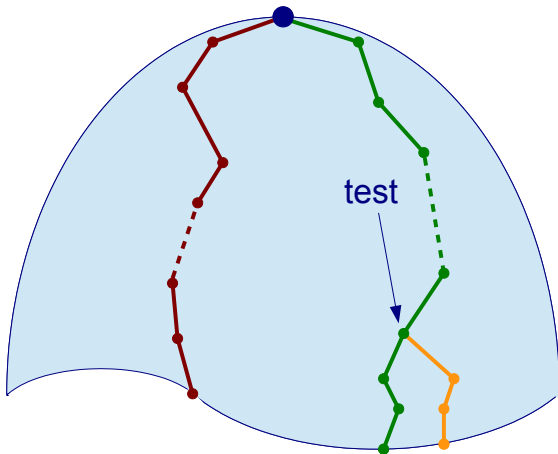
Informazioni

- In quanti modi diversi si possono mescolare n oggetti?
- Quanti sono i potenziali “percorsi” di ordinamento diversi?
- Che *informazione* rivela un singolo confronto?
- Che relazione fra *informazione* e numero di casi possibili?
- Nel peggiore dei casi, qual è il numero minimo di confronti?

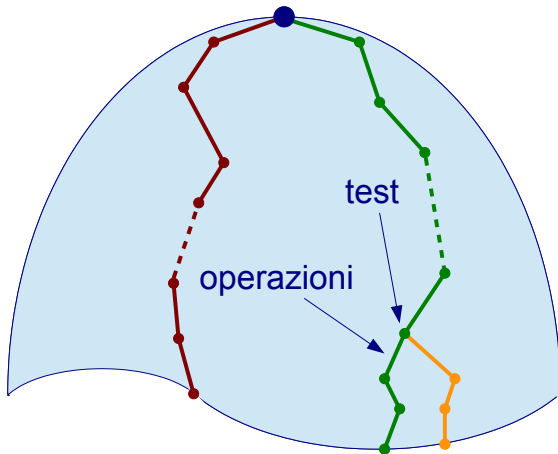
Confronti e operazioni



Confronti e operazioni



Confronti e operazioni





Informazioni. . . per districarsi fra casi possibili

Numero minimo di confronti che una strategia ideale deve effettuare per riuscire a ordinare n oggetti nella situazione più sfavorevole. . .

- $n! = n \cdot (n - 1) \cdot (n - 2) \cdot \dots \cdot 2 \cdot 1$ casi possibili
- Ogni confronto determina una bipartizione di casi
- Per bipartizioni *bilanciate* occorrono almeno k confronti, dove: $2^k \geq n!$
- Ovvero: $k \approx \log_2(n!)$



Informazioni... per districarsi fra casi possibili

Numero minimo di confronti che una strategia ideale deve effettuare per riuscire a ordinare n oggetti nella situazione più sfavorevole...

- $n! = n \cdot (n - 1) \cdot (n - 2) \cdot \dots \cdot 2 \cdot 1$ casi possibili
- Ogni confronto determina una bipartizione di casi
- Per bipartizioni *bilanciate* occorrono almeno k confronti, dove: $2^k \geq n!$
- Ovvero: $k \approx \log_2(n!)$



Informazioni... per districarsi fra casi possibili

Numero minimo di confronti che una strategia ideale deve effettuare per riuscire a ordinare n oggetti nella situazione più sfavorevole...

- $n! = n \cdot (n - 1) \cdot (n - 2) \cdot \dots \cdot 2 \cdot 1$ casi possibili
- Ogni confronto determina una bipartizione di casi
- Per bipartizioni *bilanciate* occorrono almeno k confronti, dove: $2^k \geq n!$
- Ovvero: $k \approx \log_2(n!)$



Informazioni. . . per districarsi fra casi possibili

Numero minimo di confronti che una strategia ideale deve effettuare per riuscire a ordinare n oggetti nella situazione più sfavorevole. . .

- $n! = n \cdot (n - 1) \cdot (n - 2) \cdot \dots \cdot 2 \cdot 1$ casi possibili
- Ogni confronto determina una bipartizione di casi
- Per bipartizioni *bilanciate* occorrono almeno k confronti, dove: $2^k \geq n!$
- Ovvero: $k \approx \log_2(n!)$



Informazioni... per districarsi fra casi possibili

Numero minimo di confronti che una strategia ideale deve effettuare per riuscire a ordinare n oggetti nella situazione più sfavorevole...

- $n! = n \cdot (n - 1) \cdot (n - 2) \cdot \dots \cdot 2 \cdot 1$ casi possibili
- Ogni confronto determina una bipartizione di casi
- Per bipartizioni *bilanciate* occorrono almeno k confronti, dove: $2^k \geq n! > 2^{k-1}$
- Ovvero: $k \approx \log_2(n!)$



Informazioni... per districarsi fra casi possibili

Numero minimo di confronti che una strategia ideale deve effettuare per riuscire a ordinare n oggetti nella situazione più sfavorevole...

- $n! = n \cdot (n - 1) \cdot (n - 2) \cdot \dots \cdot 2 \cdot 1$ casi possibili
- Ogni confronto determina una bipartizione di casi
- Per bipartizioni *bilanciate* occorrono almeno k confronti, dove: $2^k \geq n! > 2^{k-1}$
- Ovvero: $k \approx \log_2(n!)$



$\log(n!) ?$

$x!$ cresce molto rapidamente, ma...

$\log(x)$ cresce molto lentamente

- $n \cdot (n-1) \cdot \dots \cdot 1$
- $n \cdot (n-1) \cdot \dots \cdot 1$
- $\frac{n}{2} \cdot \frac{n}{2} \cdot \dots \cdot \frac{n}{2} < n \cdot (n-1) \cdot \dots \cdot 1 < n \cdot n \cdot \dots \cdot n$
- Cioè: $\left(\frac{n}{2}\right)^{\frac{n}{2}} < n! < n^n$



$\log(n!) ?$

$x!$ cresce molto rapidamente, ma...
 $\log(x)$ cresce molto lentamente

- $n \cdot (n-1) \cdot \dots \cdot 1$
- $n \cdot (n-1) \cdot \dots \cdot 1$
- $\frac{n}{2} \cdot \frac{n}{2} \cdot \dots \cdot \frac{n}{2} < n \cdot (n-1) \cdot \dots \cdot 1 < n \cdot n \cdot \dots \cdot n$
- Cioè: $\left(\frac{n}{2}\right)^{\frac{n}{2}} < n! < n^n$



$\log(n!) ?$

$x!$ cresce molto rapidamente, ma...
 $\log(x)$ cresce molto lentamente

- $n \cdot (n-1) \cdot \dots \cdot 1$
- $n \cdot (n-1) \cdot \dots \cdot 1$
- $\frac{n}{2} \cdot \frac{n}{2} \cdot \dots \cdot \frac{n}{2} < n \cdot (n-1) \cdot \dots \cdot 1 < n \cdot n \cdot \dots \cdot n$
- Cioè: $\left(\frac{n}{2}\right)^{\frac{n}{2}} < n! < n^n$



$\log(n!) ?$

$x!$ cresce molto rapidamente, ma...

$\log(x)$ cresce molto lentamente

- $n \cdot (n-1) \cdot \dots \cdot 1 < n \cdot n \cdot \dots \cdot n$ (n volte)
- $n \cdot (n-1) \cdot \dots \cdot 1$
- $\frac{n}{2} \cdot \frac{n}{2} \cdot \dots \cdot \frac{n}{2} < n \cdot (n-1) \cdot \dots \cdot 1 < n \cdot n \cdot \dots \cdot n$
- Cioè: $\left(\frac{n}{2}\right)^{\frac{n}{2}} < n! < n^n$



$\log(n!) ?$

$x!$ cresce molto rapidamente, ma...

$\log(x)$ cresce molto lentamente

- $n \cdot (n-1) \cdot \dots \cdot 1 < n \cdot n \cdot \dots \cdot n$ (n volte)
- $n \cdot (n-1) \cdot \dots \cdot 1$
- $\frac{n}{2} \cdot \frac{n}{2} \cdot \dots \cdot \frac{n}{2} < n \cdot (n-1) \cdot \dots \cdot 1 < n \cdot n \cdot \dots \cdot n$
- Cioè: $(\frac{n}{2})^{\frac{n}{2}} < n! < n^n$



$\log(n!) ?$

$x!$ cresce molto rapidamente, ma...

$\log(x)$ cresce molto lentamente

- $n \cdot (n-1) \cdot \dots \cdot 1 < n \cdot n \cdot \dots \cdot n$ (n volte)
- $n \cdot (n-1) \cdot \dots \cdot 1 > n \cdot (n-1) \cdot \dots \cdot \frac{n}{2}$
- $\frac{n}{2} \cdot \frac{n}{2} \cdot \dots \cdot \frac{n}{2} < n \cdot (n-1) \cdot \dots \cdot 1 < n \cdot n \cdot \dots \cdot n$
- Cioè: $(\frac{n}{2})^{\frac{n}{2}} < n! < n^n$



$\log(n!) ?$

$x!$ cresce molto rapidamente, ma...

$\log(x)$ cresce molto lentamente

- $n \cdot (n-1) \cdot \dots \cdot 1 < n \cdot n \cdot \dots \cdot n$ (n volte)
- $n \cdot (n-1) \cdot \dots \cdot 1 > \frac{n}{2} \cdot \frac{n}{2} \cdot \dots \cdot \frac{n}{2}$ ($\frac{n}{2}$ volte)
- $\frac{n}{2} \cdot \frac{n}{2} \cdot \dots \cdot \frac{n}{2} < n \cdot (n-1) \cdot \dots \cdot 1 < n \cdot n \cdot \dots \cdot n$
- Cioè: $(\frac{n}{2})^{\frac{n}{2}} < n! < n^n$



$\log(n!) ?$

$x!$ cresce molto rapidamente, ma...

$\log(x)$ cresce molto lentamente

- $n \cdot (n-1) \cdot \dots \cdot 1 < n \cdot n \cdot \dots \cdot n$ (n volte)
- $n \cdot (n-1) \cdot \dots \cdot 1 > \frac{n}{2} \cdot \frac{n}{2} \cdot \dots \cdot \frac{n}{2}$ ($\frac{n}{2}$ volte)
- $\frac{n}{2} \cdot \frac{n}{2} \cdot \dots \cdot \frac{n}{2} < n \cdot (n-1) \cdot \dots \cdot 1 < n \cdot n \cdot \dots \cdot n$
- Cioè: $(\frac{n}{2})^{\frac{n}{2}} < n! < n^n$



$\log(n!) ?$

$x!$ cresce molto rapidamente, ma...

$\log(x)$ cresce molto lentamente

- $n \cdot (n-1) \cdot \dots \cdot 1 < n \cdot n \cdot \dots \cdot n$ (n volte)
- $n \cdot (n-1) \cdot \dots \cdot 1 > \frac{n}{2} \cdot \frac{n}{2} \cdot \dots \cdot \frac{n}{2}$ ($\frac{n}{2}$ volte)
- $\frac{n}{2} \cdot \frac{n}{2} \cdot \dots \cdot \frac{n}{2} < n \cdot (n-1) \cdot \dots \cdot 1 < n \cdot n \cdot \dots \cdot n$
- Cioè: $(\frac{n}{2})^{\frac{n}{2}} < n! < n^n$



$\log(n!) ?$

In realtà più del numero preciso di confronti interessa il *trend*:
 come peggiorano le prestazioni al crescere di n

- $(\frac{n}{2})^{\frac{n}{2}} < n! < n^n$
- $\log(\frac{n}{2})^{\frac{n}{2}} < \log(n!) < \log(n^n)$
- $\frac{n}{2} \log(\frac{n}{2}) < \log(n!) < n \log(n)$
- $k \approx \log(n!) \approx \gamma \cdot n \log(n)$



$\log(n!) ?$

In realtà più del numero preciso di confronti interessa il *trend*:
 come peggiorano le prestazioni al crescere di n

- $(\frac{n}{2})^{\frac{n}{2}} < n! < n^n$
- $\log(\frac{n}{2})^{\frac{n}{2}} < \log(n!) < \log(n^n)$
- $\frac{n}{2} \log(\frac{n}{2}) < \log(n!) < n \log(n)$
- $k \approx \log(n!) \approx \gamma \cdot n \log(n)$



$\log(n!) ?$

In realtà più del numero preciso di confronti interessa il *trend*:
 come peggiorano le prestazioni al crescere di n

- $(\frac{n}{2})^{\frac{n}{2}} < n! < n^n$
- $\log(\frac{n}{2})^{\frac{n}{2}} < \log(n!) < \log(n^n)$
- $\frac{n}{2} \log(\frac{n}{2}) < \log(n!) < n \log(n)$
- $k \approx \log(n!) \approx \gamma \cdot n \log(n)$



$\log(n!) ?$

In realtà più del numero preciso di confronti interessa il *trend*:
 come peggiorano le prestazioni al crescere di n

- $(\frac{n}{2})^{\frac{n}{2}} < n! < n^n$
- $\log(\frac{n}{2})^{\frac{n}{2}} < \log(n!) < \log(n^n)$
- $\frac{n}{2} \log(\frac{n}{2}) < \log(n!) < n \log(n)$
- $k \approx \log(n!) \approx \gamma \cdot n \log(n)$

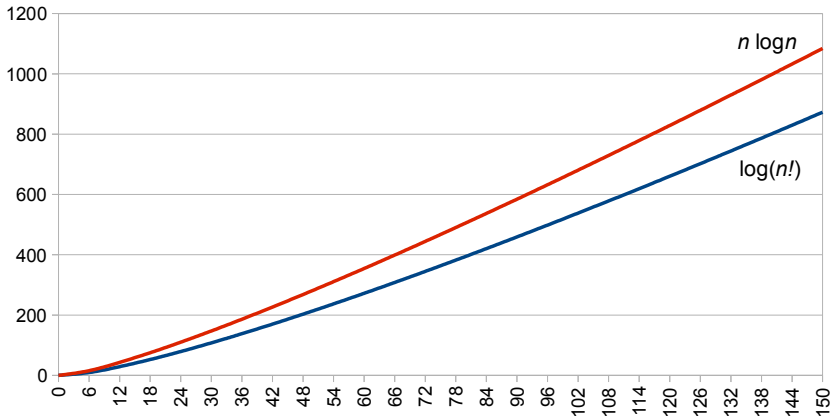


$\log(n!) ?$

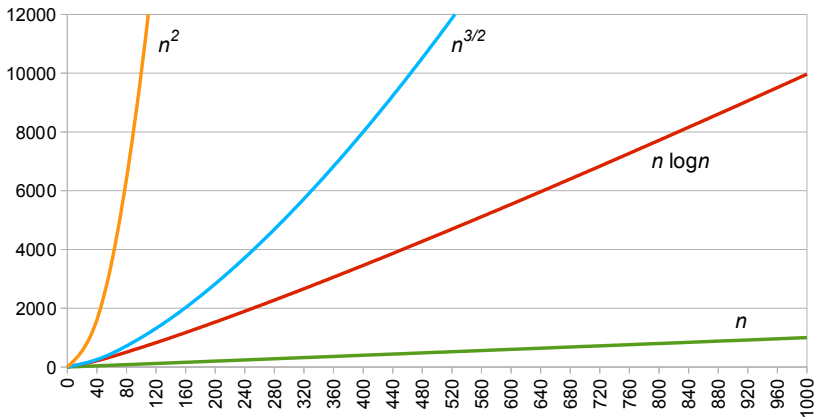
In realtà più del numero preciso di confronti interessa il *trend*:
 come peggiorano le prestazioni al crescere di n

- $(\frac{n}{2})^{\frac{n}{2}} < n! < n^n$
- $\log(\frac{n}{2})^{\frac{n}{2}} < \log(n!) < \log(n^n)$
- $\frac{n}{2} \log(\frac{n}{2}) < \log(n!) < n \log(n)$
- $k \approx \log(n!) \approx \gamma \cdot n \log(n)$

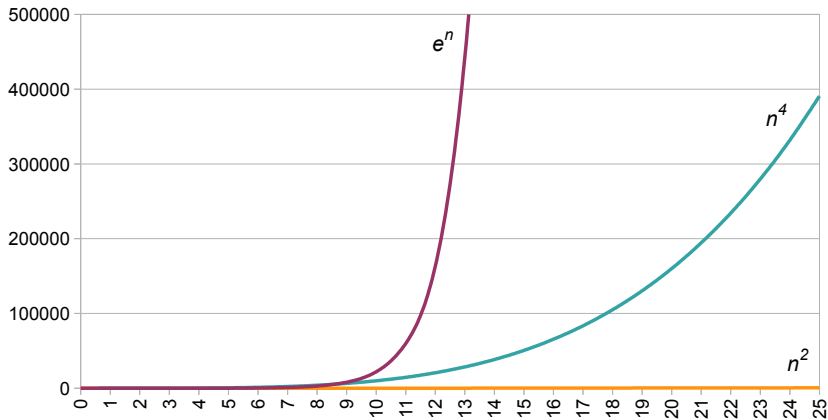
$\log(n!) ?$



Cogliere il significato

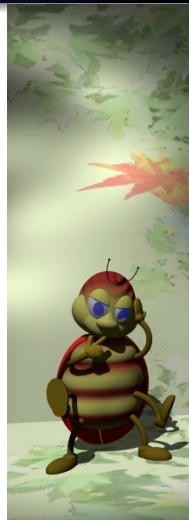


L'impresa impossibile della formichina. . .



Trama

- 1 Problema
 - in generale
- 2 Informazioni
 - analisi
 - cogliere il significato
- 3 **Algoritmi**
 - insertion sort
 - quick sort
- 4 Epilogo





Algoritmi...

Forse la scoperta più significativa determinata dall'avvento del computer sarà che gli algoritmi, come oggetti di studio, sono straordinariamente ricchi di proprietà interessanti [...].

D.E. Knuth, 1974



Algoritmi...

Forse la scoperta più significativa determinata dall'avvento del computer sarà che gli algoritmi, come oggetti di studio, sono straordinariamente ricchi di proprietà interessanti [...].

D.E. Knuth, 1974



Algoritmi

- Numero di confronti che non cresca più rapidamente di $n \log(n) \dots$
- Questo obiettivo è perseguibile?
- Esistono, cioè, strategie con *trend* ottimale?



Algoritmi

- Numero di confronti che non cresca più rapidamente di $n \log(n) \dots$
- Questo obiettivo è perseguibile?
- Esistono, cioè, strategie con *trend* ottimale?



Algoritmi

- Numero di confronti che non cresca più rapidamente di $n \log(n) \dots$
- Questo obiettivo è perseguibile?
- Esistono, cioè, strategie con *trend* ottimale?

Insertion Sort



Insertion Sort



Insertion Sort



Insertion Sort



Insertion Sort



Insertion Sort



Insertion Sort



Insertion Sort



Insertion Sort



Insertion Sort



Insertion Sort



Insertion Sort



Insertion Sort



Insertion Sort



Insertion Sort



Insertion Sort



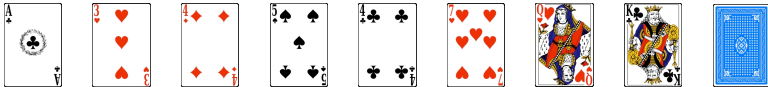
Insertion Sort



Insertion Sort



Insertion Sort



Insertion Sort



Insertion Sort



Insertion Sort



Insertion Sort





Programma

```
public void insertionSort( int[] seq ) {  
  
    int n = seq.length;  
  
    for ( int i=1; i<n; i=i+1 ) {  
  
        int x = seq[i], j = i - 1;  
  
        while ( (j >= 0) && (seq[j] > x) ) {  
            seq[j+1] = seq[j]; j = j - 1;  
        }  
        seq[j+1] = x;  
    }  
}
```



Algoritmo



Visualizzazione

- È un algoritmo con trend ottimale?
- Quanto ci si può aspettare aumenti il tempo di calcolo
 - se la lunghezza della sequenza raddoppia ($\times 2$)?
 - e se viene decuplicata ($\times 10$)?



Algoritmo



Visualizzazione

- È un algoritmo con trend ottimale?
- Quanto ci si può aspettare aumenti il tempo di calcolo
 - se la lunghezza della sequenza raddoppia ($\times 2$)?
 - e se viene decuplicata ($\times 10$)?



Algoritmo



Visualizzazione

- È un algoritmo con trend ottimale?
- Quanto ci si può aspettare aumenti il tempo di calcolo
 - se la lunghezza della sequenza raddoppia ($\times 2$)?
 - e se viene decuplicata ($\times 10$)?



Algoritmo



Visualizzazione

- È un algoritmo con trend ottimale?
- Quanto ci si può aspettare aumenti il tempo di calcolo
 - se la lunghezza della sequenza raddoppia ($\times 2$)?
 - e se viene decuplicata ($\times 10$)?



Algoritmo



Visualizzazione

- È un algoritmo con trend ottimale?
- Quanto ci si può aspettare aumenti il tempo di calcolo
 - se la lunghezza della sequenza raddoppia ($\times 2$)?
 - e se viene decuplicata ($\times 10$)?



Stima dei costi computazionali

InsertionSort: Stima del numero di confronti

- Caso peggiore:

$$(n-1) + (n-2) + \dots + 2 + 1 = \frac{n(n-1)}{2} \approx \alpha \cdot n^2$$

- Caso migliore (già ordinato), estremamente improbabile:

$$\approx \beta \cdot n$$

- Caso medio (in “media” $\frac{1}{2}k$ confronti per l’inserimento fra k oggetti già ordinati):

$$\frac{1}{2} \cdot \frac{n(n-1)}{2} \text{ (circa)} \approx \gamma \cdot n^2$$



Stima dei costi computazionali

InsertionSort: Stima del numero di confronti

- Caso peggiore:

$$(n - 1) + (n - 2) + \dots + 2 + 1 = \frac{n(n-1)}{2} \approx \alpha \cdot n^2$$

- Caso migliore (già ordinato), estremamente improbabile:

$$\approx \beta \cdot n$$

- Caso medio (in “media” $\frac{1}{2}k$ confronti per l’inserimento fra k oggetti già ordinati):

$$\frac{1}{2} \cdot \frac{n(n-1)}{2} \text{ (circa)} \approx \gamma \cdot n^2$$



Stima dei costi computazionali

InsertionSort: Stima del numero di confronti

- Caso peggiore:

$$(n - 1) + (n - 2) + \dots + 2 + 1 = \frac{n(n-1)}{2} \approx \alpha \cdot n^2$$

- Caso migliore (già ordinato), estremamente improbabile:

$$\approx \beta \cdot n$$

- Caso medio (in “media” $\frac{1}{2}k$ confronti per l’inserimento fra k oggetti già ordinati):

$$\frac{1}{2} \cdot \frac{n(n-1)}{2} \text{ (circa)} \approx \gamma \cdot n^2$$



Stima dei costi computazionali

InsertionSort: Stima del numero di confronti

- Caso peggiore:

$$(n-1) + (n-2) + \dots + 2 + 1 = \frac{n(n-1)}{2} \approx \alpha \cdot n^2$$

- Caso migliore (già ordinato), estremamente improbabile:

$$\approx \beta \cdot n$$

- Caso medio (in “media” $\frac{1}{2}k$ confronti per l’inserimento fra k oggetti già ordinati):

$$\frac{1}{2} \cdot \frac{n(n-1)}{2} \text{ (circa)} \approx \gamma \cdot n^2$$

Quick Sort



Quick Sort



Quick Sort



Quick Sort



Quick Sort



Quick Sort



Quick Sort



Quick Sort



Quick Sort



Quick Sort



Quick Sort



Quick Sort



Quick Sort



Quick Sort



Quick Sort



Quick Sort



Quick Sort



Quick Sort



Quick Sort



Quick Sort



Quick Sort



Quick Sort



Quick Sort



Quick Sort





Programma

```
private void quickSort( int l, int u, int[] seq ) {  
  
    if ( l < u ) {  
        int m = seq[l], i = l, j = u;  
        do {  
            while ( seq[i] < m ) { i = i + 1; }  
            while ( seq[j] > m ) { j = j - 1; }  
            if ( i < j ) {  
                int x = seq[i]; seq[i] = seq[j]; seq[j] = x;  
                i = i + 1; j = j - 1;  
            }  
        } while ( i < j );  
        if ( seq[j] > m ) { j = j - 1; }  
        quickSort( l, j, seq );  
        quickSort( j+1, u, seq );  
    }  
}
```



Algoritmo



Visualizzazione

- È un algoritmo con trend ottimale?
- Quanto ci si può aspettare aumenti il tempo di calcolo
 - se la lunghezza della sequenza raddoppia ($\times 2$)?
 - e se viene decuplicata ($\times 10$)?



Algoritmo



Visualizzazione

- È un algoritmo con trend ottimale?
- Quanto ci si può aspettare aumenti il tempo di calcolo
 - se la lunghezza della sequenza raddoppia ($\times 2$)?
 - e se viene decuplicata ($\times 10$)?



Algoritmo



Visualizzazione

- È un algoritmo con trend ottimale?
- Quanto ci si può aspettare aumenti il tempo di calcolo
 - se la lunghezza della sequenza raddoppia ($\times 2$)?
 - e se viene decuplicata ($\times 10$)?



Algoritmo



Visualizzazione

- È un algoritmo con trend ottimale?
- Quanto ci si può aspettare aumenti il tempo di calcolo
 - se la lunghezza della sequenza raddoppia ($\times 2$)?
 - e se viene decuplicata ($\times 10$)?



Algoritmo

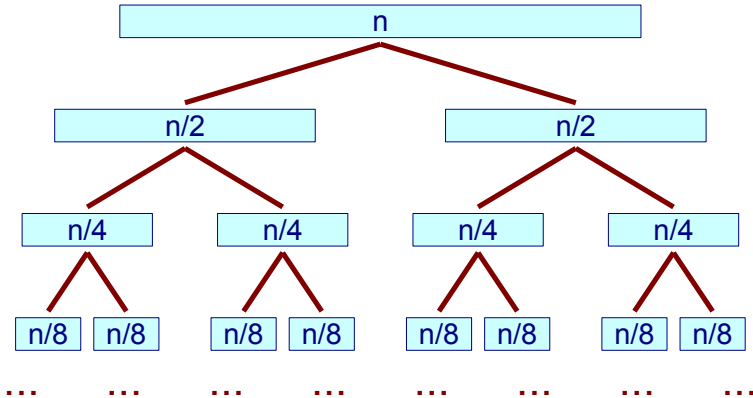


Visualizzazione

- È un algoritmo con trend ottimale?
- Quanto ci si può aspettare aumenti il tempo di calcolo
 - se la lunghezza della sequenza raddoppia ($\times 2$)?
 - e se viene decuplicata ($\times 10$)?

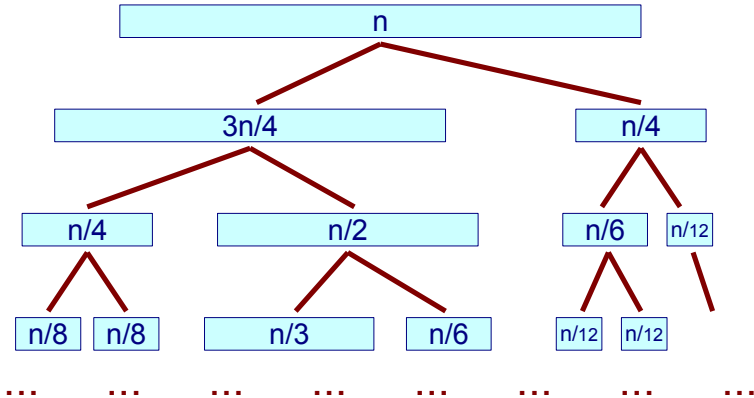


Bipartizioni perfettamente bilanciate





Bipartizioni più o meno bilanciate





Stima dei costi computazionali

QuickSort: Stima del numero di confronti

- Caso peggiore, estremamente improbabile:

$$(n-1) + (n-2) + \dots + 2 + 1 = \frac{n(n-1)}{2} \approx \alpha \cdot n^2$$

- Caso migliore, bipartizioni perfettamente bilanciate:

$$\log n \text{ (livelli)} \cdot n \text{ (confronti/livello)} \approx \beta \cdot n \log n$$

- Caso medio, bipartizioni in proporzione $\frac{1}{\rho}$ e $1 - \frac{1}{\rho}$ (ripartizione “media”, dove $1 < \rho < 2$):

$$\log_{\rho} n \text{ (livelli)} \cdot n \text{ (confronti/livello)} \approx \gamma \cdot n \log n$$



Stima dei costi computazionali

QuickSort: Stima del numero di confronti

- Caso peggiore, estremamente improbabile:

$$(n-1) + (n-2) + \dots + 2 + 1 = \frac{n(n-1)}{2} \approx \alpha \cdot n^2$$

- Caso migliore, bipartizioni perfettamente bilanciate:

$$\log n \text{ (livelli)} \cdot n \text{ (confronti/livello)} \approx \beta \cdot n \log n$$

- Caso medio, bipartizioni in proporzione $\frac{1}{\rho}$ e $1 - \frac{1}{\rho}$ (ripartizione “media”, dove $1 < \rho < 2$):

$$\log_{\rho} n \text{ (livelli)} \cdot n \text{ (confronti/livello)} \approx \gamma \cdot n \log n$$



Stima dei costi computazionali

QuickSort: Stima del numero di confronti

- Caso peggiore, estremamente improbabile:

$$(n-1) + (n-2) + \dots + 2 + 1 = \frac{n(n-1)}{2} \approx \alpha \cdot n^2$$

- Caso migliore, bipartizioni perfettamente bilanciate:

$$\log n \text{ (livelli)} \cdot n \text{ (confronti/livello)} \approx \beta \cdot n \log n$$

- Caso medio, bipartizioni in proporzione $\frac{1}{\rho}$ e $1 - \frac{1}{\rho}$ (ripartizione “media”, dove $1 < \rho < 2$):

$$\log_{\rho} n \text{ (livelli)} \cdot n \text{ (confronti/livello)} \approx \gamma \cdot n \log n$$



Stima dei costi computazionali

QuickSort: Stima del numero di confronti

- Caso peggiore, estremamente improbabile:

$$(n-1) + (n-2) + \dots + 2 + 1 = \frac{n(n-1)}{2} \approx \alpha \cdot n^2$$

- Caso migliore, bipartizioni perfettamente bilanciate:

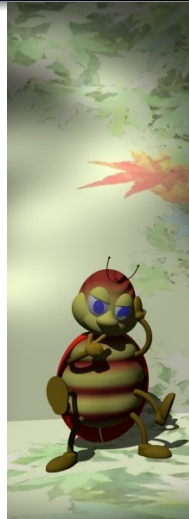
$$\log n \text{ (livelli)} \cdot n \text{ (confronti/livello)} \approx \beta \cdot n \log n$$

- Caso medio, bipartizioni in proporzione $\frac{1}{\rho}$ e $1 - \frac{1}{\rho}$ (ripartizione “media”, dove $1 < \rho < 2$):

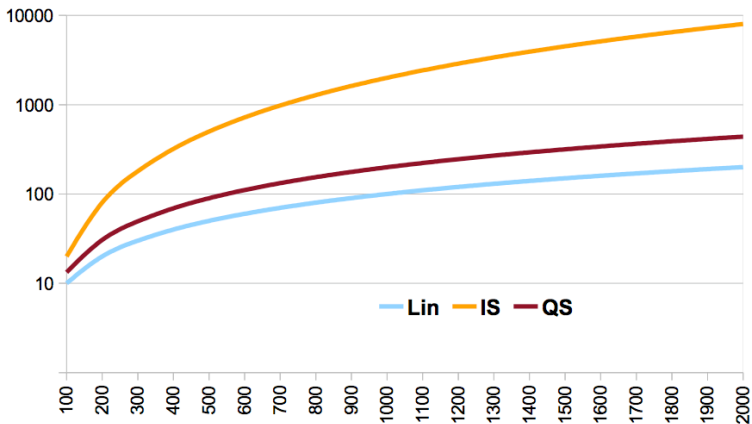
$$\log_{\rho} n \text{ (livelli)} \cdot n \text{ (confronti/livello)} \approx \gamma \cdot n \log n$$

Trama

- 1 Problema
 - in generale
- 2 Informazioni
 - analisi
 - cogliere il significato
- 3 Algoritmi
 - insertion sort
 - quick sort
- 4 Epilogo



Confronto dei trend



Altri algoritmi



Visualizzazione

- Quali algoritmi hanno trend dei costi ottimale?

Altri algoritmi



Visualizzazione

- Quali algoritmi hanno trend dei costi ottimale?

Domandina. . .

Che tipo di “informatica” abbiamo affrontato oggi?

Risposta:

Domandina. . .

Che tipo di “informatica” abbiamo affrontato oggi?

Risposta:

Analisi teorica con strumenti caratteristici della matematica

Nuove domande... da affrontare in laboratorio

- Come si può misurare sperimentalmente tempi di calcolo?
I risultati sono sempre attendibili?
- I tempi rilevati sperimentalmente confermano la teoria?
In che senso la confermano?
- *Quick Sort* è sistematicamente migliore di *Insertion Sort* ?
si osserva qualcosa di nuovo rispetto all'analisi teorica?

Nuove domande... da affrontare in laboratorio

- Come si può misurare sperimentalmente tempi di calcolo?
I risultati sono sempre attendibili?
- I tempi rilevati sperimentalmente confermano la teoria?
In che senso la confermano?
- *Quick Sort* è sistematicamente migliore di *Insertion Sort* ?
si osserva qualcosa di nuovo rispetto all'analisi teorica?

Nuove domande... da affrontare in laboratorio

- Come si può misurare sperimentalmente tempi di calcolo?
I risultati sono sempre attendibili?
- I tempi rilevati sperimentalmente confermano la teoria?
In che senso la confermano?
- *Quick Sort* è sistematicamente migliore di *Insertion Sort* ?
si osserva qualcosa di nuovo rispetto all'analisi teorica?

Apprendista scienziato

Messa a punto di un adeguato “strumento di misura”

- *Ordine di grandezza* e unità di misura dei tempi di calcolo
- *Risoluzione*, ripetibilità, interferenza di fattori estranei
- Misura dei tempi *medi* di ordinamento, generazione dei campioni, criteri
- Compensazione di *errori sistematici* della misurazione