Università degli Studi di Palermo
Dipartimento di Ingegneria Informatica

Elaborazione di Immagini e Suoni / Riconoscimento e Visioni Artificiali
12 c.f.u.

Anno Accademico 2008/2009

Docente: ing. Salvatore Sorce

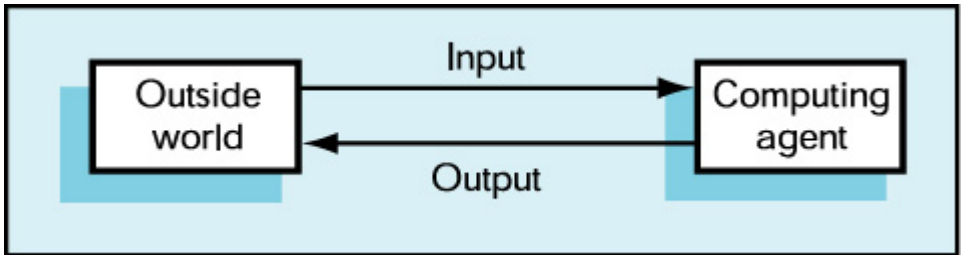
Progettazione di algoritmi e attributi

Rappresentazione di un algoritmo

- Linguaggio naturale
- Pseudocodice
- Diagrammi di flusso
- Linguaggio di programmazione formale

Operazioni sequenziali

- Operazioni sequenziali
  - Elaborazione, Ingresso, Uscita
- Operazioni di ingresso
  - Acquisisci il valore per “variabile”
- Operazioni di uscita
  - Stampa il valore per “variabile”, descrizione della variabile
- Operazioni di Elaborazione
  - Eseguire un calcolo per il valore di “variabile”



Operazioni condizionali

- Se “condizione” è vera/falsa allora
  - Prima serie di istruzioni
- Altrimenti
  - Seconda serie di istruzioni

Operazioni iterative

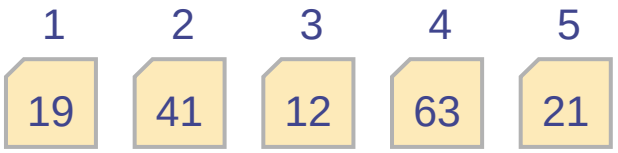
- Ciclo a condizione iniziale:
  - Finchè “condizione” è vera/falsa ripeti:
    - Istruzione 1
    - Istruzione 2
    - ...
    - Istruzione N

- Ciclo a condizione finale:
  - Ripeti:
    - Istruzione 1
    - Istruzione 2
    - ...
    - Istruzione N
  - Finchè “condizione” è vera/falsa

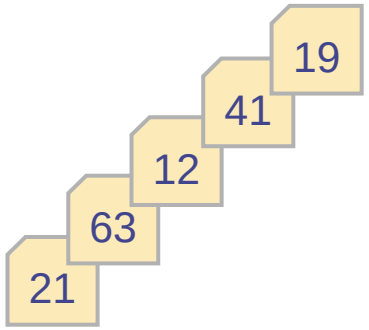
Progettazione di un algoritmo

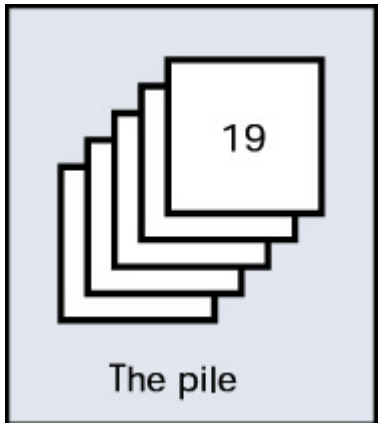
- Determinare il maggiore di un elenco di numeri
  - Dato un valore *n>1* e un elenco contenente esattamente *n* valori unici detti A_1, A_2, ..., A_n, trovare e stampare il valore maggiore nell'elenco e la posizione nell'elenco in cui è stato trovato.

- Dati: n=5, {19,41,12,63,21}

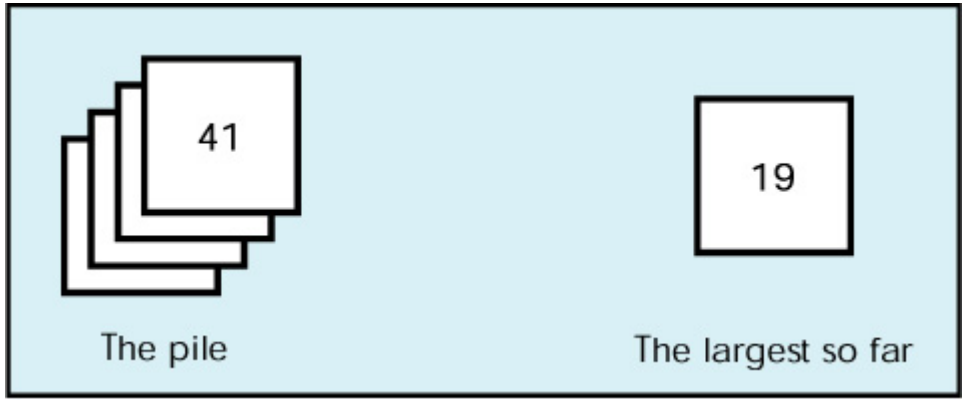


Procedimento manuale

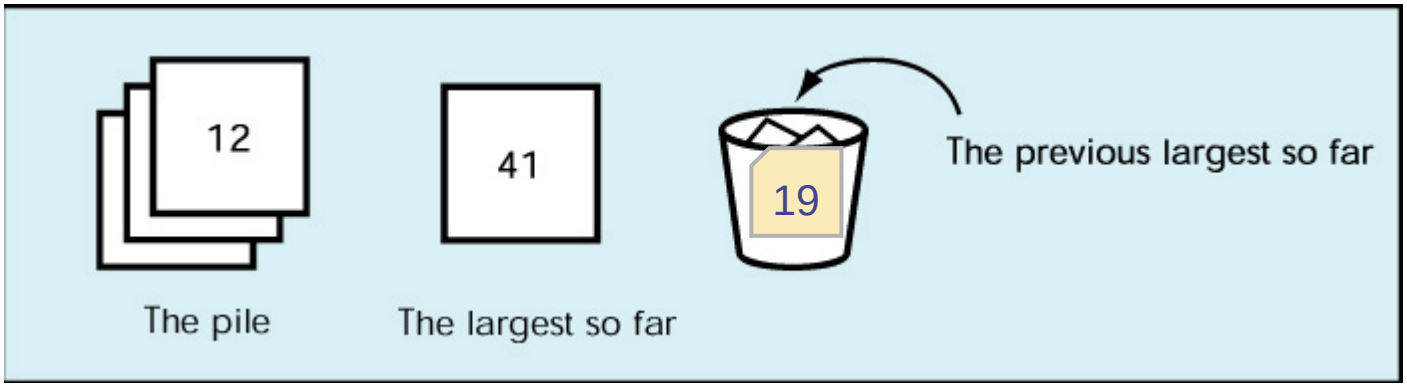




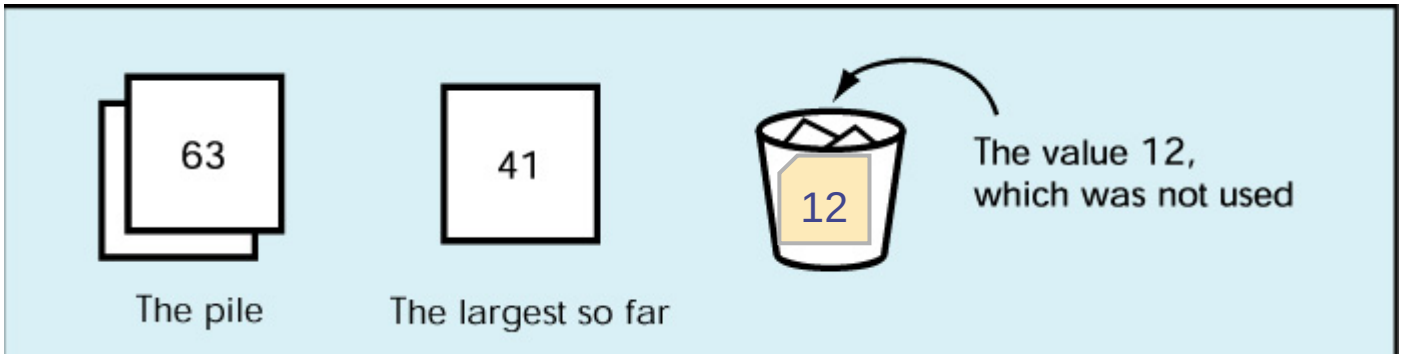
Procedimento manuale



Procedimento manuale



Procedimento manuale

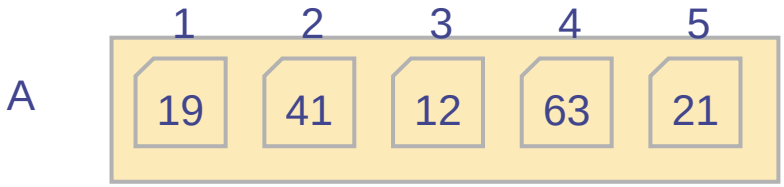


21

63

Acquisizione dei dati

- Acquisisci il valore *n* (il numero di dati in elenco)
- Acquisisci i valori per A_1, A_2, ..., A_n, (l'elenco in cui cercare)
  - i=1
  - Ripeti per i<=n
    - Acquisisci A(i)
    - i=i+1

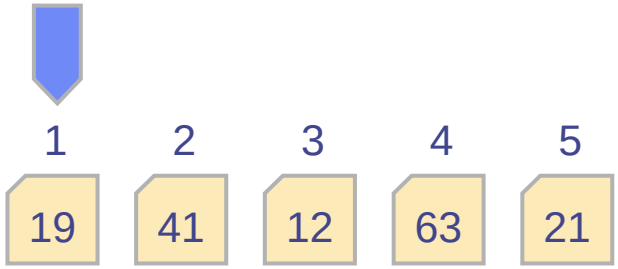


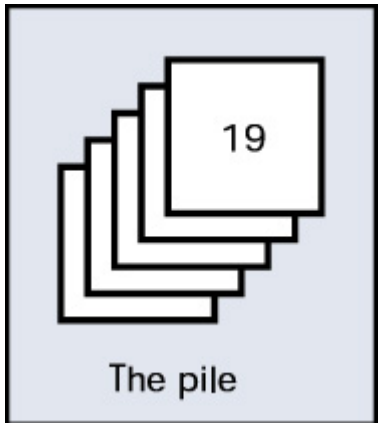
- i=1
- Ripeti per i<=n
  - Acquisisci A(i)
  - i = i + 1

Qual è il valore di i alla fine di questa porzione di pseudocodice?

Costruzione dell'algoritmo

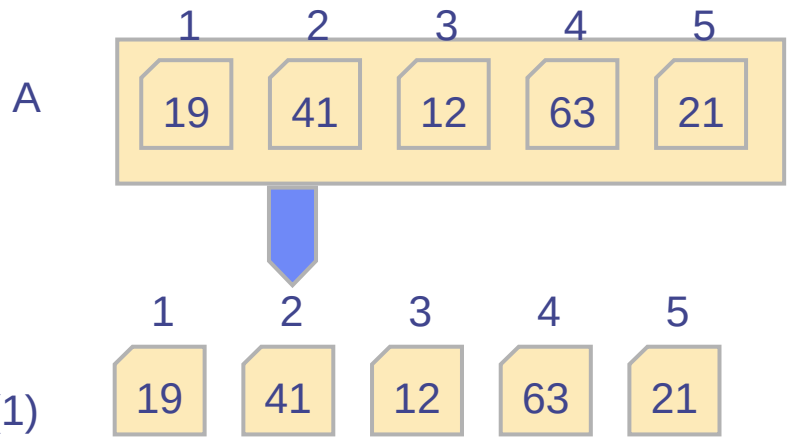
- Organizzare i dati in una pila





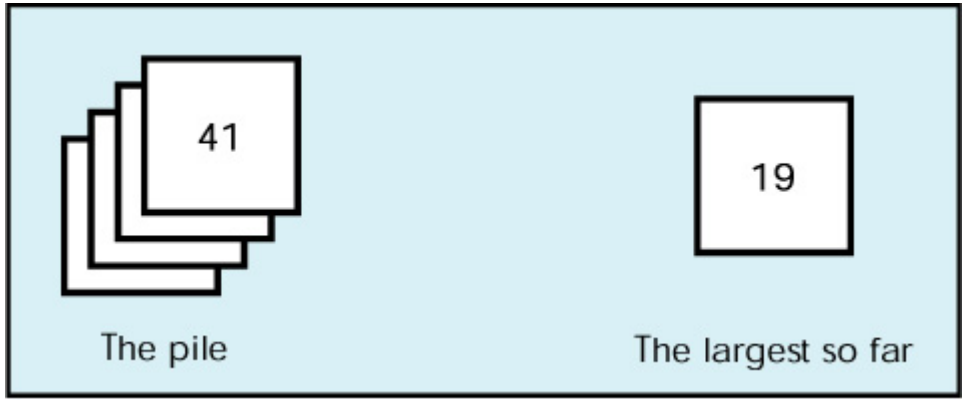
Costruzione dell'algoritmo

- Il primo elemento è il *maggiore attuale*.
- Lo salviamo in una posizione a parte



19
Maggiore attuale
1
locazione
2
i

1. i=1
2. locazione=i
3. Maggiore attuale=A(1)
4. i=i+1



Costruzione dell'algoritmo

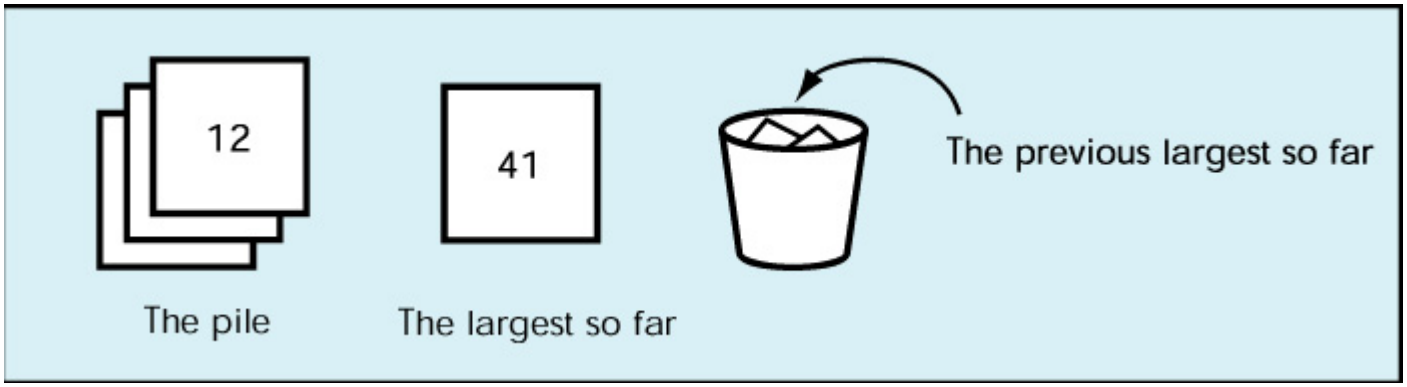
	1	2	3	4	5
A	19	41	12	63	21

41
Maggiore attuale
2
locazione
3
i

- Confronta il prossimo elemento con il *maggiore attuale*
- Se l'elemento considerato è maggiore del *maggiore attuale* allora esso diventa il nuovo *maggiore attuale* altrimenti esso viene scartato

1. Se maggiore attuale > A(i) allora 2 altrimenti 1.1
   1.1 Maggiore attuale=A(i)
   1.2 Locazione=i
2. i=i+1

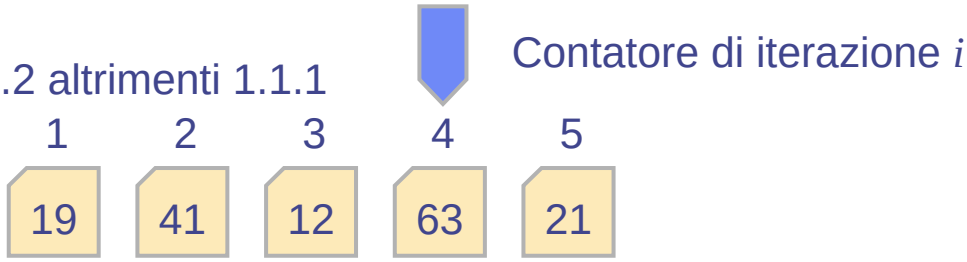
1	2	3	4	5
19	41	12	63	21

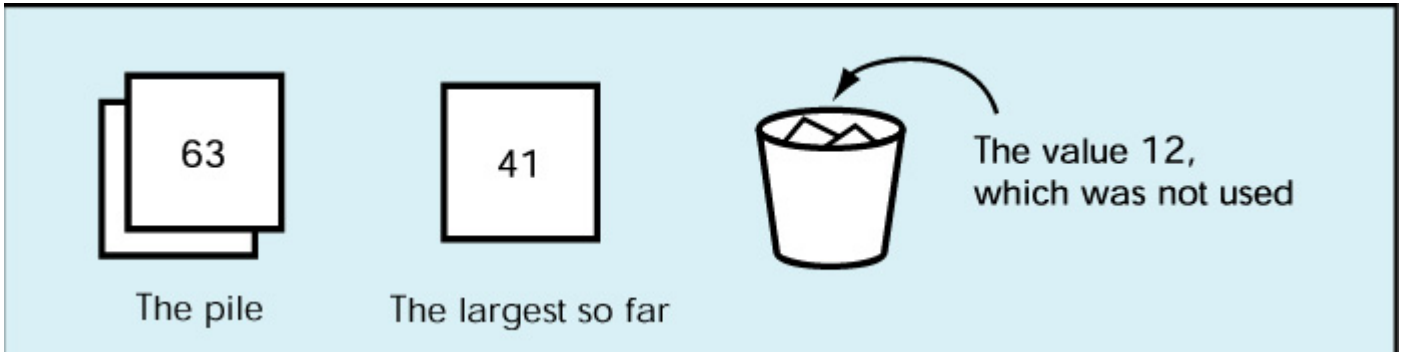


Costruzione dell'algoritmo

- Ripeti finché ci sono dati in elenco
  - Confronta il prossimo elemento con il *maggiore attuale*
  - Se l'elemento considerato è maggiore del *maggiore attuale* allora esso diventa il nuovo *maggiore attuale* altrimenti esso viene scartato

1. Ripeti per i<=n
1.1 Se maggiore attuale > A(i) allora 1.2 altrimenti 1.1.1
1.1.1 Maggiore attuale=A(i)
1.1.2 Locazione=i
1.2 i=i+1





Algoritmo “Trova il maggiore”

1. Acquisisci un valore per *n*, il numero di dati in elenco
2. i = 1
3. Ripeti i passi 4 e 5 mentre i<=n
4. Acquisisci A(i)
5. i=i+1
6. i=1
7. Poni il valore di *maggiore attuale* = *A(i)*
8. Poni il valore di *locazione* = i
9. Ripeti i passi 10 e 11 fino mentre i<=n
10. Se A(i) > maggiore attuale allora:
    Poni il valore di *maggiore attuale* = *A(i)*
    Poni il valore di *locazione* = *i*
11. Aggiungi 1 al valore di *i*
12. Stampa i valori di *maggiore attuale* e *locazione*
13. Stop

Prosegui e moltiplica

- Moltiplicazione di due interi mediante la somma

  prodotto = A x B = A + A + A + ... + A (B volte)

Prosegui e moltiplica

- Moltiplicazione di due interi mediante la somma

  prodotto = A x B = A + A + A + ... + A (B volte)

1. Ottieni il valore di A
2. Ottieni il valore di B
3. Poni *i* = 1
4. Poni *prodotto* = 0
5. Se A = 0 allora vai al passo 7
6. Mentre *i* <= B ripeti:
   1. *prodotto* = *prodotto* + A
   2. *i* = *i* + 1
7. Stampa il valore di *prodotto*
8. Fine

Attributi

- Correttezza
  - Un algoritmo non deve soltanto produrre un risultato, ma deve produrre un risultato corretto
  - Un algoritmo deve produrre un risultato utile
- Facilità di comprensione
  - Necessità di adattare un algoritmo per una varietà di scenari possibili.
  - Importante per garantire la manutenibilità dei programmi
- Eleganza
  - Spesso in antitesi con facilità di comprensione
  - Es.: Somma dei primi 100 numeri

Università degli Studi di Palermo

Attributi

- Efficienza
  - Tempo di calcolo e spazio in memoria sono quantità limitate
  - L'efficienza di una algoritmo misura la sua capacità di utilizzare bene le risorse del calcolatore sui cui gira, in termini di tempo di calcolo e memoria utilizzata
- Efficienza nell'uso dello spazio
  - Quantità di informazioni da memorizzare per svolgere il compito in aggiunta ai dati di ingresso
  - Tanto più inefficiente quanto più memoria aggiuntiva è richiesta
- Efficienza nell'uso del tempo di calcolo
  - Benchmarking: fissare i parametri della misura, ovvero l'insieme dei dati di ingresso, la macchina specifica, il particolare profilo di uso dell'algoritmo

Università degli Studi di Palermo

Attributi

- Efficienza nell'uso del tempo di calcolo
  - Indicazione della ***quantità di lavoro*** richiesta dalla natura dell'algoritmo
  - Tale quantità di lavoro dipende dal numero di passi richiesti per eseguire il compito
  - Il confronto tra due algoritmi va operato sulla base del numero di passi e non del tempo di esecuzione su una particolare macchina.

Università degli Studi di Palermo

Misura dell'efficienza di un algoritmo

Ricerca Sequenziale

1. Acquisisci i valori di *NOME*, N_1, ..., N_n, T_1, ..., T_n
2. Poni il valore di *i* a 1 ed il valore di *Trovato* a NO
3. Ripeti fino a che (*Trovato* = SI) ovvero (*i* > *n*)
   4. Se NOME = N_i
      Stampa il numero di telefono T_i
      Poni il valore di Trovato a SI
   5. Altrimenti aggiungi 1 al valore di i
6. Se Trovato = NO allora stampa il messaggio "Mi dispiace: numero non in elenco"
7. Fermati

Misura dell'efficienza di un algoritmo

Ricerca Sequenziale

1. Acquisisci i valori di *NOME*, N_1, ..., N_n, T_1, ..., T_n
2. Poni il valore di *i* a 1 ed il valore di *Trovato* a NO
3. Ripeti fino a che *Trovato* = SI ovvero *i* > *n*
   4. Se NOME = N_i
      Stampa il numero di telefono T_i
      Poni il valore di Trovato a SI
   5. Altrimenti aggiungi 1 al valore di i
6. Se Trovato = NO allora stampa il messaggio "Mi dispiace: numero non in elenco"
7. Fermati

OPERAZIONI MARGINALI

Hanno un peso costante ***c***, che non varia col numero di unità di lavoro *n*

Misura dell'efficienza di un algoritmo

Ricerca Sequenziale

1. Acquisisci i valori di *NOME*, N_1, ..., N_n, T_1, ..., T_n
2. Poni il valore di *i* a 1 ed il valore di *Trovato* a NO
3. Ripeti fino a che *Trovato* = SI ovvero $i > n$
   4. Se NOME = N_i
      Stampa il numero di telefono T_i
      Poni il valore di Trovato a SI
   5. Altrimenti aggiungi 1 al valore di i
6. Se Trovato = NO allora stampa il messaggio "Mi dispiace: numero non in elenco"
7. Fermati

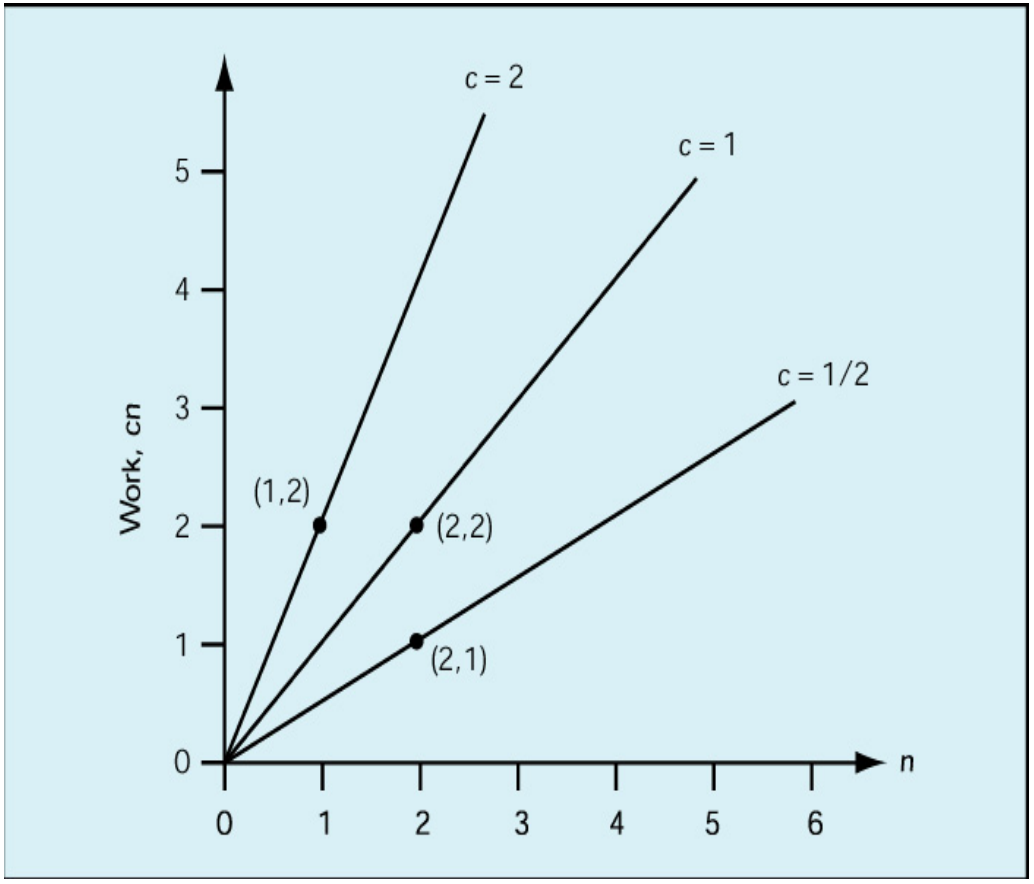
CASO MIGLIORE **1**
CASO PEGGIORE **n**
CASO MEDIO **n/2**

Misura dell'efficienza di un algoritmo

Ordine di grandezza

L'algoritmo di ricerca sequenziale ha un ordine di grandezza $\Theta(n)$

La notazione indica che il lavoro svolto, varia con lo stesso **andamento** di n



Misura dell'efficienza di un algoritmo

Ordine di grandezza $\Theta(n^2)$

- Si vuole stampare il contenuto di una tabella di registrazioni

Per ogni riga da 1 a 4 ripeti
per ogni colonna da 1 a 4 ripeti
Stampa il valore in tabella

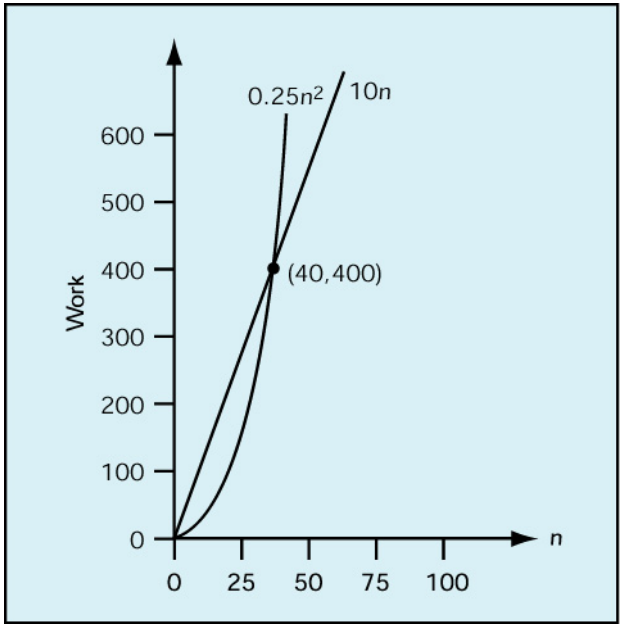
	1	2	3	4
1	243	187	314	244
2	215	420	345	172
3	197	352	385	261
4	340	135	217	344

- L'operazione stampa sarà ripetuta per ogni colonna in ogni riga
  - $4 \times 4 = 16 = 4^2$
- Con *n* righe e colonne, il lavoro svolto è proporzionale ad n^2

Misura dell'efficienza di un algoritmo

- Confronto tra ordini di grandezza $\Theta(n)$ e $\Theta(n^2)$
- $\Theta(n^2)$ è sempre più lento di $\Theta(n)$ nel medio-lungo termine
- Esempio:
  - Il lavoro marginale per l'algoritmo B è 1 milione di volte più pesante dell'algoritmo A
  - $100/0.00001 = 10^6$

	Number of Work Units Required	
n	Algorithm A $0.0001n^2$	Algorithm B $100n$
1,000	100	100,000
10,000	10,000	1,000,000
100,000	1,000,000	10,000,000
1,000,000	100,000,000	100,000,000
10,000,000	10,000,000,000	1,000,000,000



Misura dell'efficienza di un algoritmo

Ricerca Binaria (su elenco ordinato)

1. Acquisisci i valori di *NOME*, N_1, ..., N_n, T_1, ..., T_n
2. Poni il valore di *inizio* a 1 ed il valore di *Trovato* a NO
3. Poni il valore di *fine* ad *n*
4. Ripeti fino a che *Trovato* = SI ovvero *fine < inizio*
   5. Poni il valore di **m** alla metà tra **inizio** e **fine**
   6. Se **NOME** = $\mathbf{N_m}$
      Stampa il numero di telefono T_i
      Poni il valore di Trovato a SI
   7. Altrimenti se **NOME** < $\mathbf{N_m}$ poni **fine** = **m** – 1
      Altrimenti poni ***inizio*** *=* ***m*** *+ 1*
8. Se Trovato = NO allora stampa il messaggio "Mi dispiace: numero non in elenco"
9. Fermati

Misura dell'efficienza di un algoritmo

Ricerca Binaria

Anna Bobo Cora Dino Gino Nino Susy

1 2 3 4 5 6 7

cerco *CORA*

inizio=1

fine=7

m = 4

Misura dell'efficienza di un algoritmo

Ricerca Binaria

Anna Bobo Cora Dino Gino Nino Susy — cerco *CORA*
1 2 3 4 5 6 7

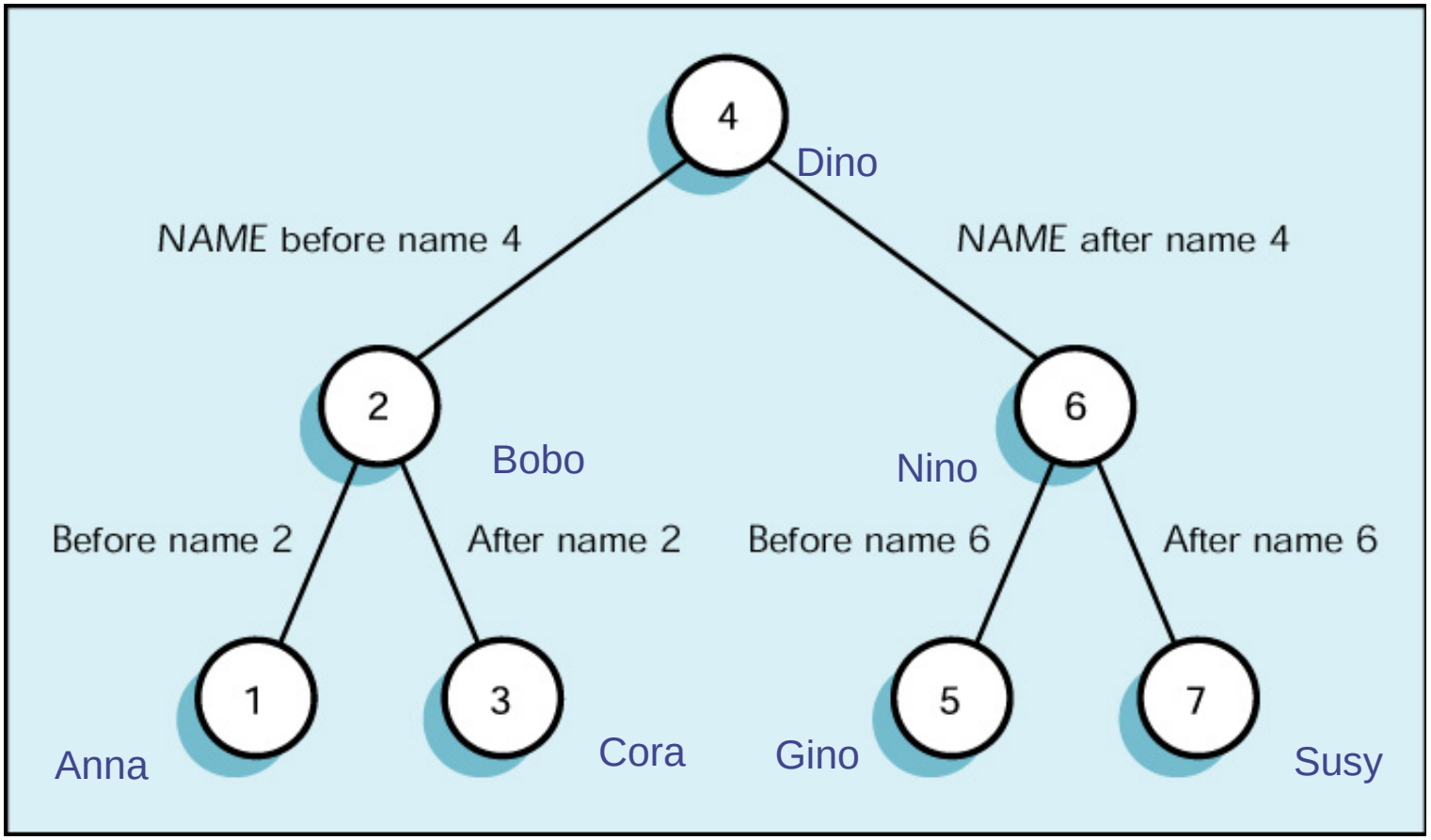
m = 4 — Anna Bobo Cora Dino — Cora < Dino
1 2 3 4

fine = 3, m = 2 — Anna Bobo Cora — Cora > Bobo
1 2 3

fine = 3, inizio = m = 3 — Cora — Cora = Cora

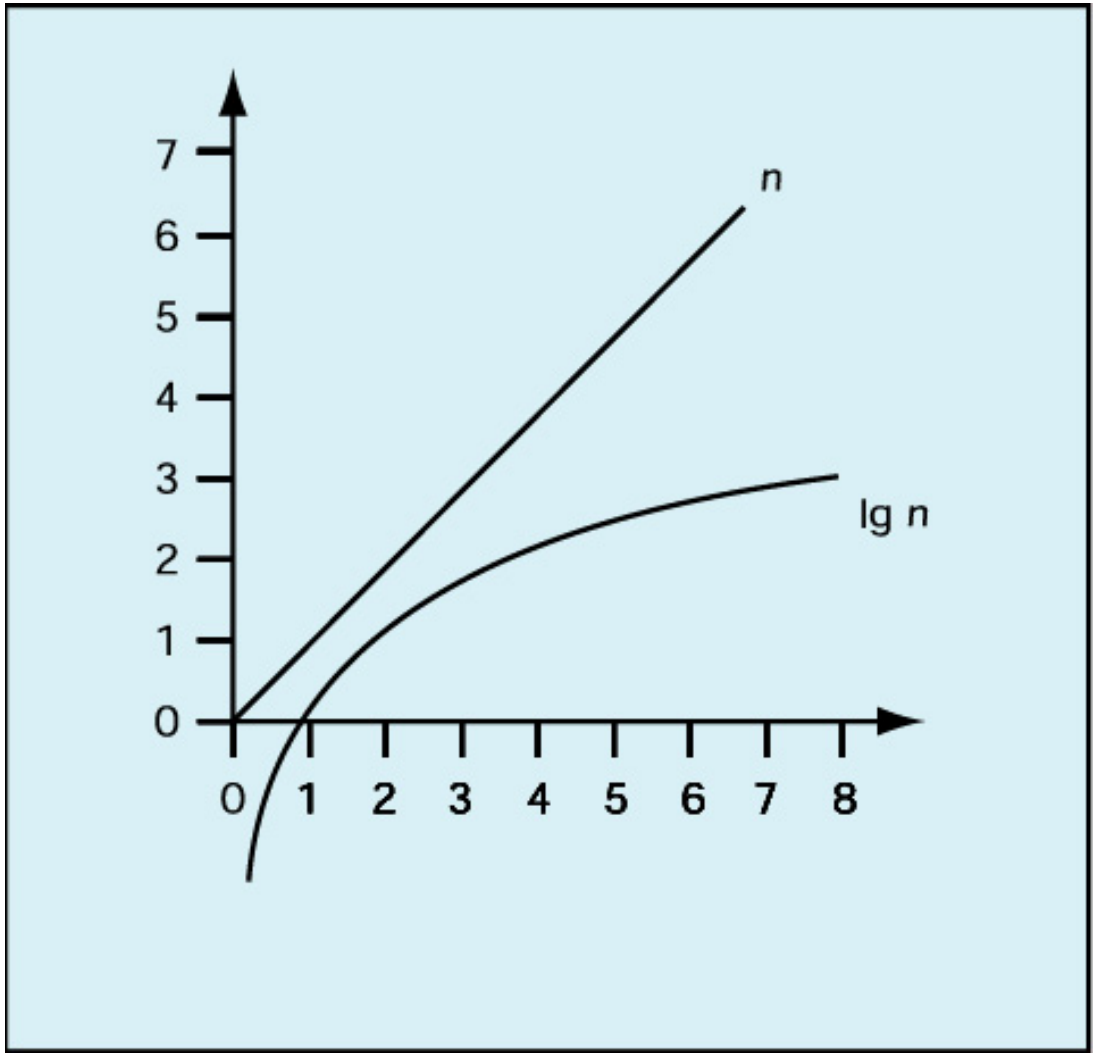
Trovato = SI — FINE

Misura dell'efficienza di un algoritmo



$\Theta(\log_2 n)$

Misura dell'efficienza di un algoritmo



Università degli Studi di Palermo

La tartaruga e la lepre

- Confronto tra un Pentium Pro 200 e un Cray T3E-900

	Pentium Pro 200	Cray T3E-900
N. proc	1	1320
Velocità	75 MFLOPS	670 GFLOPS
Costo	€ 2.000,00	€ 31.000.000,00
Rapporto Prestazioni	1	8933
Rapporto Costo	1	15500

- Al P-Pro viene assegnato un algoritmo di classe $\Theta(n)$, mentre al Cray un algoritmo di classe $\Theta(n^2)$

La tartaruga e la lepre

- Al P-Pro viene assegnato un algoritmo di classe $\Theta(n)$, mentre al Cray un algoritmo di classe $\Theta(n^2)$

n	Ppro O(n) sec	Cray O(n^2) sec	
750	0,00001	0,00000084	
7.500	0,0001	0,000084	
75.000	0,001	0,0084	
750.000	0,01	0,84	
7.500.000	0,1	84	
75.000.000	1	8396	2,3 ore
750.000.000	10	839552	9,7 giorni

Domande?