Scuola di Ingegneria Industriale

Laurea in Ingegneria Energetica

Laurea in Ingegneria Meccanica

POLITECNICO DI MILANO

Dipartimento di Elettronica, Informazione e Bioingegneria





Informatica B

Sezione D

Franchi Alessio Mauro, PhD *alessiomauro.franchi@polimi.it*

Un algoritmo...

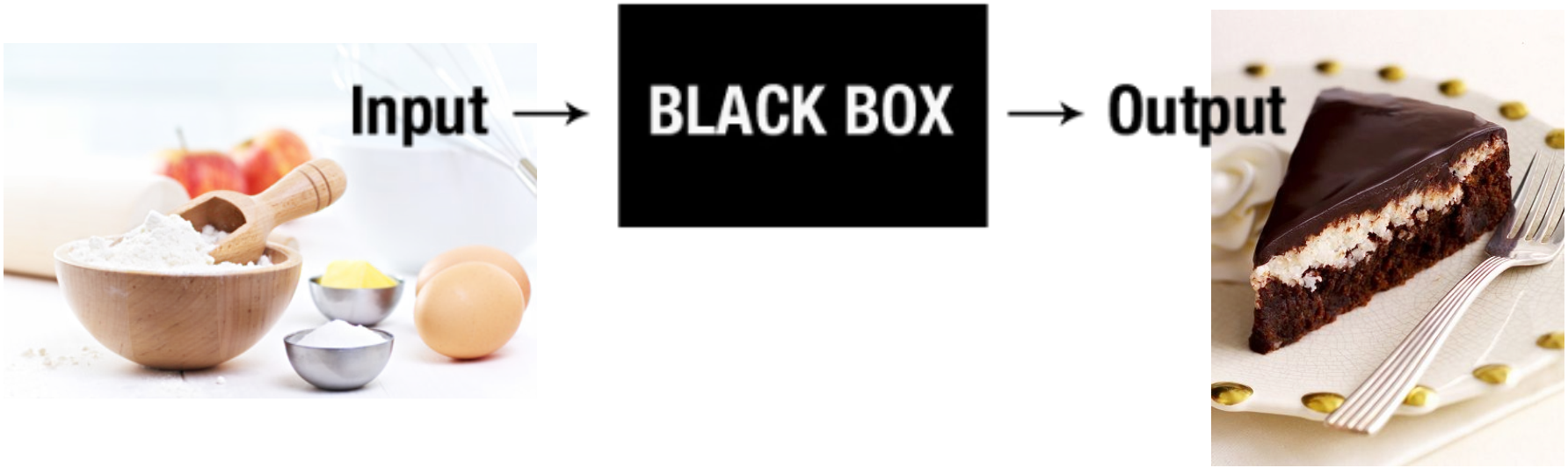
E' una **sequenza finita di passi** univocamente interpretabili che servono per raggiungere un determinato scopo (**output**) a partire da una precisa situazione (**input**).

Importante:

- Lista finita
- Istruzioni ben definite
- Istruzioni univoche
- Input, stato iniziale
- Output, stato finale
- Chi è l'esecutore?

Ad esempio...

Prepariamo una torta: prima di metterci all'opera è bene definire come vogliamo procedere!



Salviamo capra e cavoli: devo trasportare da una riva all'altra di un fiume un lupo, una capra e dei cavoli su una barchetta. Sapendo che la barca può trasportare solo un oggetto alla volta, come salvo capra e cavoli? Attenzione: il lupo mangia la capra, la capra mangia i cavoli!

In 7 mosse

Salviamo capra e cavoli:

1. Traghettare la capra da A a B (nel frattempo sulla sponda A restano il lupo e i cavoli)
2. Tornare indietro
3. Traghettare i cavoli da A a B
4. Riportare indietro la capra da B ad A (per evitare che mangi i cavoli, che ora si trovano sulla riva B)
5. Traghettare il lupo da A a B (per evitare che mangi la capra, che è tornata sulla sponda A)
6. Tornare indietro
7. Traghettare la capra da A a B (mentre sulla sponda B restano il lupo e i cavoli)

1. Traghettare la capra da A a B (nel frattempo sulla sponda A restano il lupo e i cavoli)
2. Tornare indietro
3. Traghettare il lupo da A a B
4. Riportare indietro la capra da B ad A (per evitare che venga mangiata dal lupo, che ora si trova sulla riva B)
5. Traghettare i cavoli da A a B (per evitare che vengano mangiati dalla capra, che è tornata sulla sponda A)
6. Tornare indietro
7. Traghettare la capra da A a B (mentre sulla sponda B restano il lupo e i cavoli)

Nel dettaglio

Prepariamo una torta:

- Il mio algoritmo è già abbastanza “raffinato”? E' bene descrivere nel dettaglio ogni passo!
- Adesso non ho più gli ingredienti a disposizione: il mio algoritmo deve cambiare? SI!

Un algoritmo dipende dallo stato iniziale/input del problema

Salviamo capra e cavoli:

- Immaginate di poter trasportare due oggetti alla volta: il mio algoritmo cambia? SI!

Un algoritmo dipende dalle azioni che ho a disposizione

Rappresentare un algoritmo

Abbiamo appena rappresentato/descritto due algoritmi tramite "**pseudocodice**"

Per pseudocodice si intende un "linguaggio di programmazione" fittizio non interpretabile dall'esecutore. Non esiste uno pseudolinguaggio standard e convenzionalmente usato, ciascun programmatore può essere portato ad utilizzare una propria variante

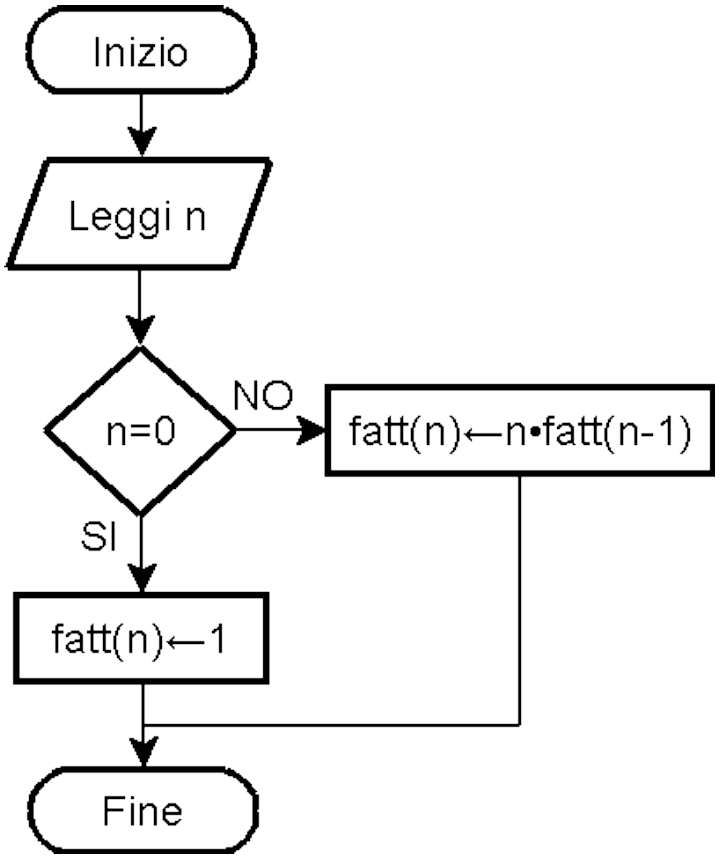
1. **Leggi** A e B
2. min= il minimo tra A e B
3. tmp = 1
4. MCD = 1
5. **Finche'** tmp < min
   1. tmp = tmp + 1
   2. **Se** tmp divide A e B
      1. **Allora** MCD = tmp
6. **Stampa** MCD

```
Begin
   input a
   input n
   s ← 0
   if (a != 0) AND (n != 0) then
    begin
      do
         s ← s + a
         n ← n - 1
      while n != 0
    end
   output s
End
```

```
Subgraph_Mining (GS,S,s)
{
  1: if s ≠ min(s)
  2:     return;
  3: 𝕊 ← 𝕊 ∪ {s};
  4: generate all s' poter
  5: Enumerate(s);
  6: for each c, c is s' c
  7:     if support(c) ≥ r
```

Rappresentare un algoritmo

I **diagrammi di flusso** sono una valida alternativa allo pseudocodice per la definizione di un algoritmo



Disegni/diagrammi che rappresentano graficamente tutti i passi del nostro algoritmo.

Occorre definire i “simboli” utilizzati all’interno dei diagrammi di flusso per identificare i vari tipi di azione.

Riepilogando

Abbiamo visto come si passa da un problema da risolvere ad un piano d'azione, l'algoritmo.

- Ho un input/stato iniziale del problema
- Devo raggiungere uno stato finale/produrre un output
- Decido una sequenza di passi che risolvono il problema
- Mano a mano raffino la sequenza
- L'esecutore esegue

Questa è solo una idea di come passare da un problema ad un algoritmo!

Ogni persona/programmatore ha il suo metodo e quindi trova/inventa il suo personale algoritmo!

Prendete confidenza con questa mentalità: **“scervellatevi”!!!**

Esercizi

1. Dato il raggio R di un cerchio calcolare circonferenza C e area A;
2. Trovare l'mcm e l'MCD tra due numeri in ingresso A e B;
3. Data una sequenza finita di numeri, copiare in una nuova lista i numeri pari e in un'altra quelli dispari;
4. Data una sequenza finita di numeri, verificare che ogni numero sia maggiore della somma dei precedenti;
5. Data una sequenza finita di numeri, trovare il maggiore e il minore;
6. Data una sequenza finita di numeri non ordinati, riordinarla in ordine crescente;
7. Data una parola (sequenza di lettere) verificare che questa sia palindroma;
8. Data una parola, calcolarne l'"istogramma"; l'istogramma di una parola associa ad ogni lettera dell'alfabeto il numero di occerrenze nella parola;
9. Date due frazioni in input (viste come numeratore N e denominatore D), calcolare somma e differenza.