UNIVERSITÀ
degli STUDI
di CATANIA

Problemi, algoritmi, diagrammi di flusso

Corso di programmazione I AA 2020/21

Corso di Laurea Triennale in Informatica

Prof. Giovanni Maria Farinella
Web: http://www.dmi.unict.it/farinella
Email: gfarinella@dmi.unict.it

Dipartimento di Matematica e Informatica

Overview

Algoritmi

Algoritmo

Dato un problema, un **algoritmo** è una sequenza di passi concepita per essere eseguita automaticamente da una macchina in modo da risolvere il problema dato.

Un problema risolvibile mediante un algoritmo si dice **computabile**.

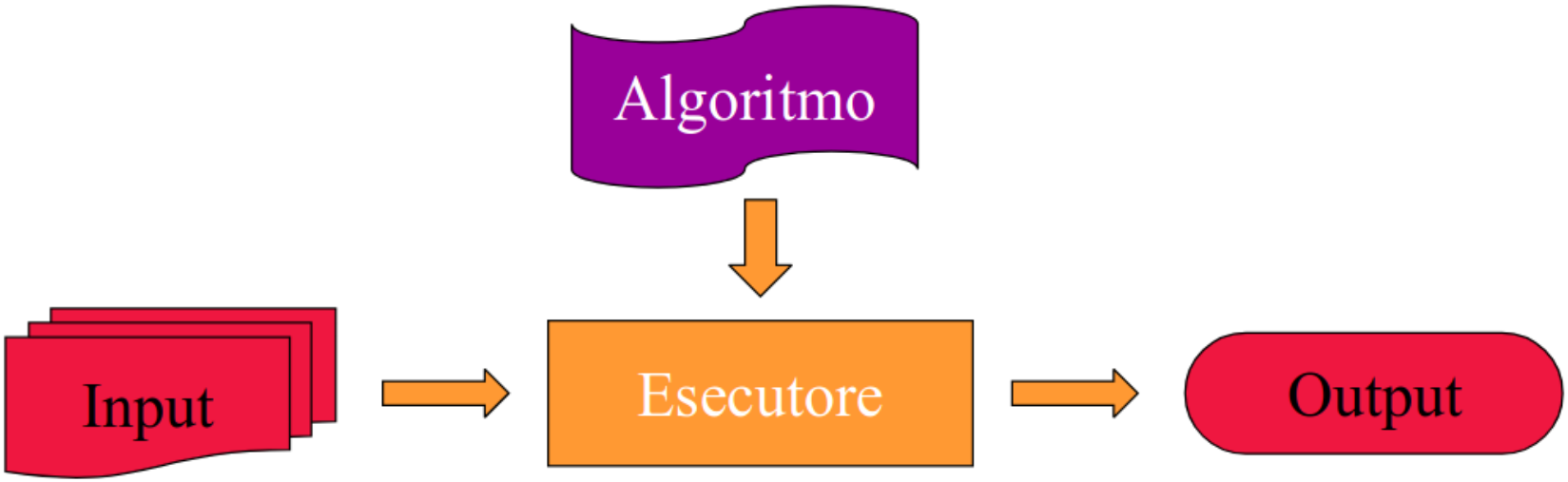
Algoritmo

Esempio

Preparazione del risotto ai funghi:

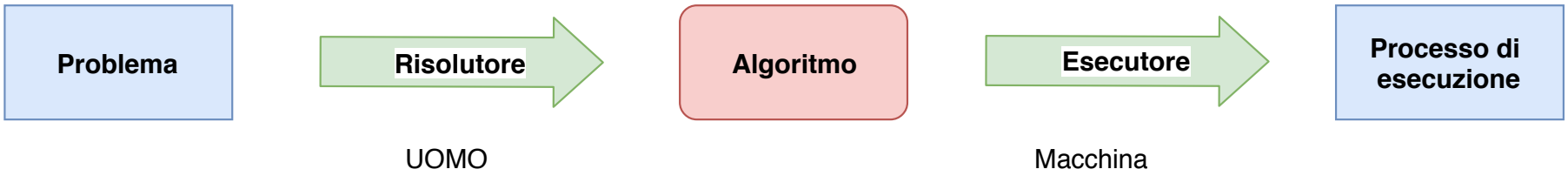
- Dati gli Ingredienti (riso, funghi, cipolla, pepe, ...)
- Seguire la ricetta (preparare i funghi, tostare il riso, procedere con la cottura, etc)
- Servire il piatto a tavola

Risoluzione di un problema



1. Prendere i dati iniziali (**input**)
2. Concepire l'algoritmo e codificarlo affinche' sia **interpretabile** da uno opportuno risolutore.
3. Avviare un **esecutore**.
4. Attendere la **fine del lavoro dell'esecutore** ed infine prelevare lo **output**.

Risoluzione di un problema: uomo vs macchina



Uomo è **reale risolutore** (concepisce algoritmo)

Macchina è solo **esecutore**.

- genera un **processo di esecuzione** e..
- alla fine di esso un output

Definizione meno informale di algoritmo

Algoritmo

Sequenza **ordinata** e **finita** di passi (azioni o istruzioni) che producono un ben determinato **risultato in un tempo finito**.

Caratteristiche di un algoritmo

1. Azioni **eseguibili** e **non ambigue**
   - "abbastanza", "a volontà" non sono espressioni ammissibili
2. **Determinismo**.
   - Fatto un passo, il **successivo** è uno ed uno solo, ben **determinato**.
   - **Alternative** sono ammesse, ma la scelta deve essere univoca.
3. **Numero finito di passi**.
4. **Terminazione**
   - NB: Numero finito di passi **non implica** terminazione.

Caratteristiche di un algoritmo

Ogni passo deve:

- **terminare** entro un intervallo **finito di tempo**;
- produrre un effetto **osservabile**;
- produrre lo **stesso effetto** ogni volta che viene eseguito a partire dalle **stesse condizioni iniziali** (input, valori iniziali delle varibili, etc)

Esempi

1. Algoritmo che non termina

1. Si consideri un numero N
2. Scrivere N.
3. Scrivere il numero successivo.
4. Ripetere il passo precedente.

Esempi

Quale algoritmo? Dipende dal tipo di input..

2. Ricerca di un nome in elenco

- Elenco non ordinato (lista di firme)
  - Ricerca sequenziale..
- Elenco ordinato (elenco telefonico)
  - Ricerca dicotomica..

Codifica degli algoritmi: linguaggi e programmi

Osservazione

La macchina deve essere in grado di **comprendere** l'algoritmo.

Ovvero attribuire la giusta **semantica** a tutti i passi dello algoritmo in modo che essi siano **eseguiti in modo corretto**.

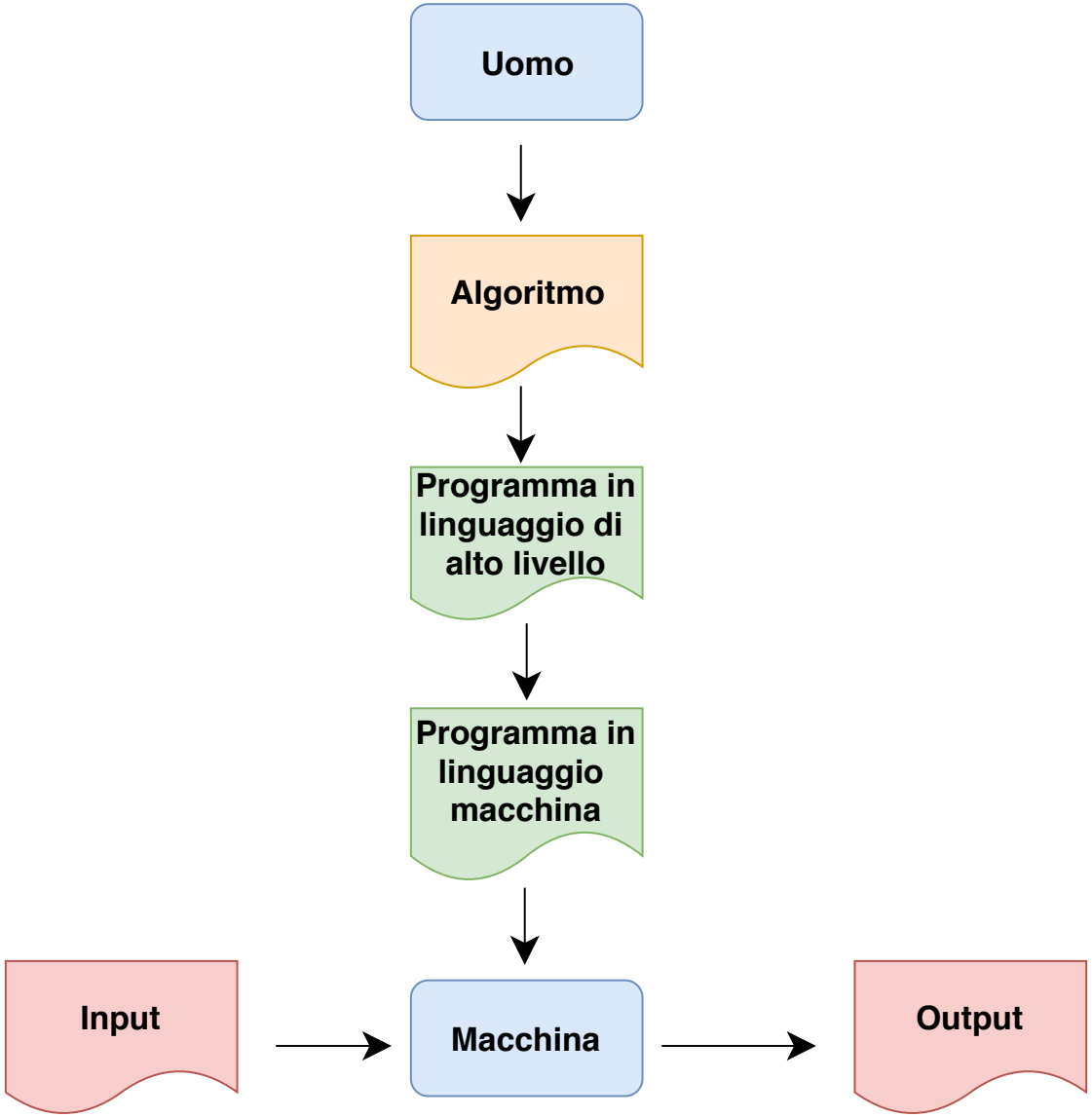
Codifica dello algoritmo

Programma

Lo algoritmo va **codificato** in un uno opportuno **linguaggio** di "alto livello".

Il risultato di tale codifica viene chiamato **programma**

Codifica dello algoritmo



Progettazione di algoritmi

Progettare un algoritmo

Risotto ai funghi

1. Preparare il brodo vegetale (*)
2. Far bollire i funghi **per 3 minuti circa**
3. Trifolare i funghi (*)
4. Tostare il riso **fino a quando sarà ben caldo** (*).
5. Sfumare con vino bianco (*)
6. Aggiungere brodo vegetale
7. Cuocere **per 12 minuti** circa aggiungendo brodo vegetale **quanto basta**
8. Aggiungere i funghi trifolati e cuocere **per altri 5 minuti**
9. Lasciare riposare **per 5 minuti**

NB: I passi con asterisco rappresentano sottoproblemi.

Progettare un algoritmo

Un sottoproblema: Preparare il brodo vegetale

- Ingredienti: 2 gambi di Sedano, 4 carote medie, 1 ciuffo di prezzemolo, 1 cipolla grande.
- Riempire una casseruola di acqua e portare velocemente a ebollizione.
- Versare gli ingredienti nella casseruola e fare sobbollire per circa un'ora e trenta.
- A cottura completata filtrare il brodo con un canovaccio da cucina.

Progettare un algoritmo

Un altro sottoproblema: sfumare con il vino bianco

- versare mezzo bicchiere di vino bianco
- far sobbollire a fuoco lento **fino a quando tutto l'alcool sarà evaporato**

La ricetta precedente è un esempio di cosa significa **scomporre un problema in sottoproblemi**.

Ogni sottoproblema può essere scomposto in **problemi via via più elementari**.

Approccio Top-down

1

Si costruisce una **visione generale del problema**, senza scendere nel dettaglio delle sue parti

- ES: preparare il brodo vegetale.

2

Ogni parte del sistema è *successivamente rifinita* per **decomposizione** aggiungendo dettagli.

- ES: Lista di ingredienti per il brodo, come prepararli, tempo di cottura e filtraggio

Approccio Top-down

3

Si opera, se necessario, mediante **successive decomposizioni**, che permetteranno di specificare ulteriori dettagli.

4

Il processo di decomposizione potrà **concludersi** quando la specifica avrà fornito **sufficienti dettagli** da poter validare il modello.

Approccio Bottom Up

1

Parti individuali del sistema sono *specificate* in dettaglio.

2

La parti vengono **connesse** tra loro per formare **componenti** più grandi.

Approccio Bottom Up

3

Successive connessioni/composizioni permetteranno di realizzare un sistema più completo.

Bottom up (puro) si usa spesso quando

- si hanno a disposizione **svariate componenti pronte** per essere utilizzate. Queste possono essere collegate insieme a formare componenti più grandi.
- si dispone di una certa **esperienza** nella realizzazione di un sistema che risolve lo **stesso problema o problemi simili**.

Top Down vs Bottom Up

Spesso si adotta un **approccio ibrido**

Esempio

Stampare tutti i nomi di persona presenti in un testo.

1. (TD) Leggere il testo, riga per riga, separando le singole parole.
   - (BU) Usare la funzione **getLine()**.
   - (BU) Usare la funzione **getWords()** sull'intera riga.
2. (TD) Memorizzare ogni nome di parola quando esso viene letto.
3. (TD) Stampare tutti i nomi di parola memorizzati.
   - (BU) Usare la funzione *print()* su tutte le parole.

Diagrammi di flusso

Descrizione di un algoritmo

La descrizione di un algoritmo è **propedeutica** alla sua successiva **codifica**.

Ma va usato un linguaggio generale, **indipendente** dalla codifica stessa:

1. diagrammi di flusso
2. pseudo-codice

Diagrammi di flusso

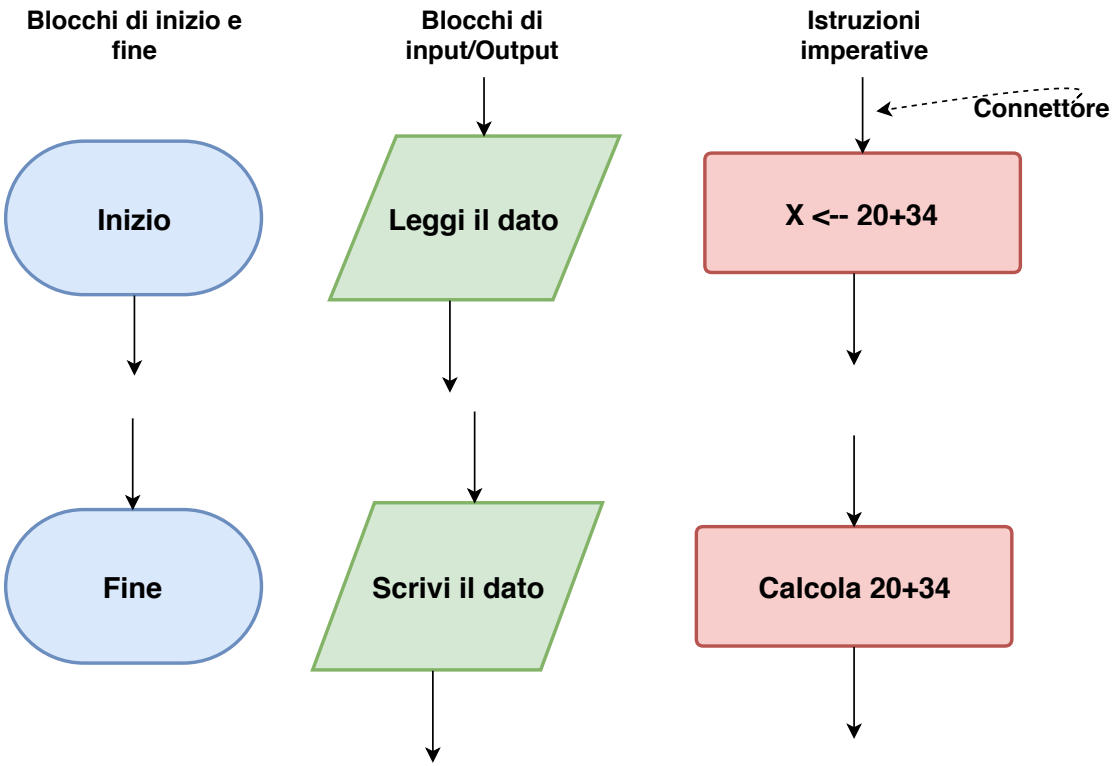
Un diagramma di flusso permette di descrivere in **modo grafico** le **azioni** che costituiscono un algoritmo nonché il loro **flusso di esecuzione**.

- I **Blocchi** rappresentano le azioni
- I **connettori** permettono di specificare in quale **ordine** vanno eseguite le azioni

Diagrammi di flusso

Blocchi e connettori

L'ordine di esecuzione delle istruzione avviene in base ai connettori.
La posizione dei connettori determina il **flusso di esecuzione**



Istruzioni di assegnamento

Variabile $\leftarrow$ Espressione

- Variabile: Entità caratterizzata da
  - **nome**
  - **valore** il quale puó cambiare nel tempo
  - ES: X, Y, Z, Pippo, ...
- Espressione: **combinazione** di operatori aritmetici, costanti e variabili che da luogo ad un **risultato** numerico.
  - ES: $X + 2$, $Y/2$, $Y\%2$, ...
- ES: $X \leftarrow Y + 2$, $Y \leftarrow Y/2 + Z$

Diagrammi di flusso

Esempio: Somma di due numeri letti in input