Informatica - Algoritmi

Mirko Cesarini - Dario Pescini
nome.cognome@unimib.it

Università di Milano Bicocca

Percorso: dagli algoritmi alla programmazione

- Obiettivo: imparare la programmazione (coding)
- Perché?
  - Come statistici avrete a che fare con grosse moli di dati
  - Alternativa: fare i conti a mano . . . o impare a far lavorare i computer
- Che cosa richiede?
  - Acquisire le competenze necessarie per:
    - progettare e implementare (piccole) procedure di elaborazione automatica di dati e informazioni
    - essere in grado di leggere, capire ed eventualmente modificare codice scritto da altre persone
  - Imparare ad analizzare i problemi con un approccio analitico
    - Distinguere la patologia dai sintomi
    - Distinguere i problemi principali dai problemi secondari

Suggerimenti per l'apprendimento

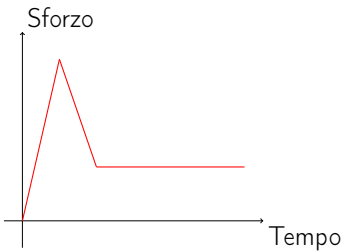
- La programmazione richiede di imparare a pensare in modo diverso da quello a cui siete abituati ...
- ... ciò non vale solo per l'informatica (ma anche per tante altre discipline)
- Relazione tra teoria e pratica
  - Test: conosco un argomento se sono in grado di metterlo in pratica
  - L'applicazione a casi reali mi aiuta a capire meglio gli aspetti teorici
  - Miglioramento continuo: mettete in discussione le vostre idee, verificate se stanno in piedi, miglioratele nel tempo
- Le esercitazioni / laboratori sono parte integrante delle lezioni
  - Per imparare la programmazione dovete esercitarvi (tanto)
  - Le parti teoriche non sono opzionali: ripassate gli argomenti delle lezioni prima dei laboratori (altrimenti buttate via il tempo)

Suggerimenti 2

- Il metodo è fondamentale (chi ben inizia è già a metà dell'opera)
- Non pretendete di saper fare tutto e subito
  - "...non ho mai studiato programmazione..."
  - "...il mio vicino di banco va alla velocità della luce, io invece sono fermo"
- Esempio (patente automobilistica)
  - La vs prima volta al volante di un'auto, come è andata?
  - Ora guidate bene (spero)! Come ci siete riusciti?
- Il corso (e i laboratori) sono tarati per chi non ha mai studiato programmazione
- Chiedo scusa (in anticipo) a chi invece la ha già studiata (le prime lezioni si annoierà un po')

Sulla base dell'esperienza passata

- La curva di apprendimento può essere descritta dal grafico:



- All'inizio farete fatica
  - La cosa importante è . . . non mollare (soprattutto all'inizio)
  - Ce la fanno tutti . . . (statisticamente parlando) ☺

Alcuni consigli su come studiare

- Date la precedenza ai vostri appunti
- Usate il materiale presentato a lezione per integrare i vostri appunti e per tener traccia del programma svolto
- Usate i libri suggeriti per approfondire gli argomenti trattati a lezione
- Se nonostante gli appunti, le slide, il materiale aggiuntivo e i libri non sarete soddisfatti del vostro livello di comprensione, fate domande al docente
  - In aula (prima o dopo una lezione, durante la lezione)
  - Approfittate dei ricevimenti
- Sintesi: **dovete avere un ruolo attivo**
- Cercate di non far passare inutilmente il tempo

Algoritmi

- Per risolvere un problema connesso con l'elaborazione delle informazioni, spesso occorre creare un algoritmo

Algoritmo

Intuitivamente, si può definire un algoritmo come un **procedimento** che a partire da uno **stato iniziale**, consente di ottenere in un **tempo finito** un **risultato atteso** eseguendo un insieme di operazioni descritte in maniera **completa** e **non ambigua**

- L'algoritmo può essere eseguito da un elaboratore poiché
  - è una descrizione completa e non ambigua di un procedimento
  - produce un risultato in un tempo finito

Caso: l'importanza degli algoritmi

- Immaginate di dover ordinare in ordine crescente i numeri seguenti:

10	5	8	1	4	3

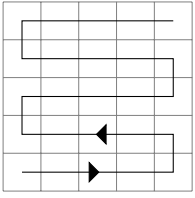
- Facile, giusto? Ecco il risultato:

1	3	4	5	8	10

Esperimento

- Immaginate di avere un'aula didattica, con 100 studenti disposti a forma di scacchiera rettangolare.
- Immaginate che ogni studente riceva (casualmente) un numero intero
- Supponete che i numeri debbano essere ordinati (in ordine crescente) come nell'esempio seguente

91	92	93	...	99	100
...	...	...	...	...	...
21	22	...	28	29	30
20	19	18	...	12	11
1	2	3	...	9	10



- Come si può fare?

. . .

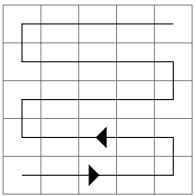
- Proviamo ad eseguire l'esperimento in aula (il docente vi distribuirà dei fogli con dei numeri sopra)
- Seguite le indicazioni del docente

. . .

- Proviamo ora invece ad utilizzare il seguente algoritmo. Immaginate di essere uno degli studenti seduti in aula.

Bubble Sort Umano Distribuito

1. Individuate la direzione che si snoda su ogni riga della scacchiera (vedi a lato)
2. Confrontate il vostro numero con quello del collega accanto (sulla linea precedentemente individuata)
3. Se i numeri non sono nell'ordine giusto, scambiate i fogli tra voi.
4. Se uno dei fogli del vostro vicino cambia, ripetete le operazioni dal punto 2
5. Ripetete le operazioni dal punto 2 fino a che non ci sarà più alcuno scambio.



Risultati

- Risultati dell'esperimento in aula
  - Quest'anno i numeri sono stati ordinati ...
  - Gli anni precedenti non sempre è stato raggiunto l'obiettivo
  - Indipendentemente dal risultato, è stato utile per comprendere che l'implementazione di un algoritmo richiede diversi raffinamenti successivi
- In generale: i docenti ringraziano per la collaborazione

Algoritmi, considerazioni

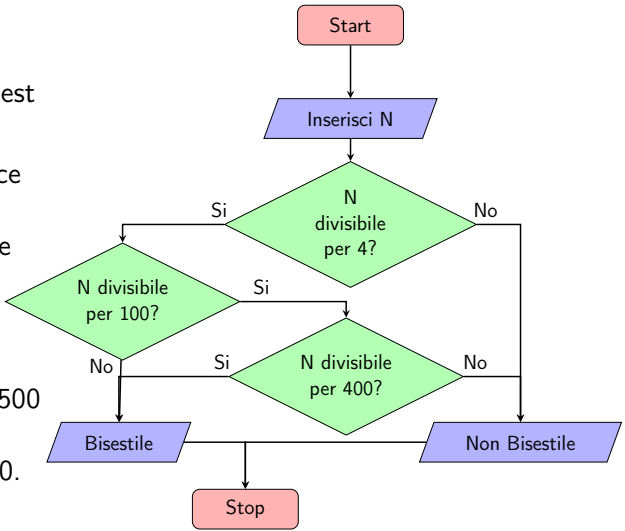
- Vedete, quello che appariva semplice (con piccoli numeri e con una sola persona coinvolta) ...
- ...si è rivelato molto più complicato in uno scenario con grandi numeri
- La soluzione è stata l'applicazione di un algoritmo
- Nella slide precedente abbiamo descritto in maniera (quasi ☺) **completa** e **non ambigua** un procedimento, che ha permesso a più persone di cooperare.

Proprietà degli algoritmi

- L'informatica è lo studio sistematico degli algoritmi che descrivono e trasformano l'informazione
- Due proprietà fondamentali dell'algoritmo:
  - Correttezza: un algoritmo corretto perviene alla soluzione senza difettare di alcun passo fondamentale
    - capacità di risolvere il problema per ogni caso possibile o . . . (almeno) per ognuna delle situazioni iniziali previste
    - senza difettare di alcun passaggio fondamentale
  - Efficienza: l'algoritmo perviene alla risoluzione del problema nel modo più veloce possibile e/o usando la minima quantità di risorse, compatibilmente con la sua correttezza
- Risorse consumate per l'esecuzione di un algoritmo
  - tempo di esecuzione
  - memoria RAM
  - memoria di massa, . . .

Esempio di algoritmo

- Un esempio di algoritmo: test anno bisestile
- La regola? (l'algoritmo) dice che sono bisestili gli anni divisibili per quattro, tranne quelli di inizio secolo non divisibili per 400.
- Il 1944 è bisestile perché divisibile per quattro, ma 1500 non lo è perché è un inizio secolo non divisibile per 400.



Uovo al tegamino

Per produrre un algoritmo, occorre identificare chiaramente:

- il problema da risolvere
- input
- output
- la sequenza di passi da compiere (logica)

Algoritmo uovo al tegamino - tentativo 1

Elementi

1. **problema** Cuocere un uovo al tegamino
2. **input** uovo, sale
3. **output** uovo cotto
4. **logica**
   - prendi uovo
   - mettilo in padella
   - cuocilo
   - salalo
   - mettilo nel piatto

E sufficiente? riuscireste ad ottenere l'uovo cotto come vorreste?

Algoritmo uovo al tegamino - tentativo 2

...

4. logica

1. prendi padella
2. metti la padella sul fornello
3. accendi il fuoco sotto la padella
4. attendi temperatura corretta
5. rompi l'uovo **e se l'uovo è scaduto?**
6. metti l'uovo in padella **e se l'uovo si rompe?**
7. aggiungi sale
8. attendi 5 minuti **se è crudo o bruciato?**
9. prendi piatto
10. togli la padella dal fuoco
11. versa il contenuto della padella nel piatto

Il problema è concreto

Secondo voi il video è reale? Link per approfondimenti

Considerazione

- Creare un algoritmo non è semplice
- Occorre un lavoro (non banale) dell'intelletto per creare e perfezionare un algoritmo
- Dovete mettere in conto ...
  - tanti tentativi
  - (molto) tempo per individuare gli errori
  - pazienza per apportare le correzioni
- Dovrete esercitarvi per acquisire la capacità di creare algoritmi
- Ciò vi sarà utile per sviluppare un atteggiamento critico, che è alla base della Data Analysis

Algoritmi e linguaggi di programmazione

- Come si arriva a far eseguire un algoritmo ad un calcolatore?
- Scomponiamo il problema in 2 sotto-problemi
  1. Come rappresentare/descrivere un algoritmo?
     Utilizzando i linguaggi di programmazione
     - Un linguaggio di programmazione è un linguaggio artificiale che può essere usato per descrivere algoritmi
     - Un programma è una sequenza di istruzioni (scritte in uno specifico linguaggio di programmazione) che codifica un algoritmo
  2. Come fa un computer ad eseguire la "descrizione di un algoritmo" (un programma)

Dall'algoritmo al linguaggio macchina

Esempio di linguaggio macchina

- Un computer esegue soltanto programmi scritti in "linguaggio macchina"
- Come si passa da un algoritmo descritto con un linguaggio di programmazione al linguaggio macchina?

	Comando	Param.	Traduz. comando in assembly	Traduz. param. in assembly
0	0100	00000000	(READ)	
1	0010	00100000	(STORE)	32
2	0001	00000000	(LOAD=)	0
3	0010	00100001	(STORE)	33
4	0000	00100000	(LOAD)	32
5	1100	00001101	(BEQ)	13
6	0100	00000000	(READ)	
7	0110	00100001	(ADD)	33
8	0010	00100001	(STORE)	33
9	0000	00100000	(LOAD)	32
10	0011	00000001	(SUB=)	1
11	0010	00100000	(STORE)	32
12	1000	00000100	(BR)	4
13	0000	00100001	(LOAD)	33
14	0101	00000000	(WRITE)	
15	1111	00000000	(END)	

Un po' di terminologia ...

- **Programma sorgente**: sequenza di istruzioni espresse attraverso un linguaggio di programmazione
  - Non è direttamente eseguibile da un computer
  - (Abbastanza) semplice da interpretare per una persona umana
- **Eseguibile** (chiamato anche "programma in linguaggio macchina"): sequenza di istruzioni in linguaggio macchina ...
  - ... direttamente eseguibili dal calcolatore
  - ... difficilmente interpretabile da una persona umana

Strumenti per passare da programma sorgente ad un programma eseguibile

- **Compilatore**
  - Riceve in ingresso un programma sorgente
  - Genera un eseguibile
    - traduce **tutte** le istruzioni in un unico passaggio (detto processo di compilazione)
    - L'eseguibile, una volta generato, può essere eseguito (tutte le volte che si vuole) senza richiedere di effettuare un'altra compilazione
- **Interprete**
  - Riceve in ingresso un programma sorgente
  - Interpreta ed esegue le istruzioni **una alla volta**
  - Non genera un eseguibile vero e proprio, ma è uno strumento che permette di eseguire il programma sorgente
- Sottolineatura
  - Una volta generato l'eseguibile, il compilatore **non è più** necessario per eseguire il programma
  - L'interprete, poiché non genera alcun eseguibile, è **sempre** necessario per eseguire un programma,

Linguaggi di alto e basso livello

Esistono diverse tipologie di linguaggi di programmazione

- linguaggi (di programmazione) di alto livello:
  - più simili al linguaggio naturale
  - istruzioni astratte
  - indipendenti dai dettagli dell'architettura della macchina
- linguaggi (di programmazione) di basso livello es., assembly (chiamato anche assembler):
  - istruzioni codificate in sequenze di caratteri alfa-numerici
  - dipendente dall'architettura della macchina
- linguaggi macchina:
  - i più lontani dal linguaggio naturale
  - istruzioni codificate in sequenze di bit
  - fortemente dipendente dai dettagli dell'architettura della macchina

Sintassi e semantica dei linguaggi

- **Sintassi**. L'insieme di regole formali che dettano le modalità per costruire frasi corrette nel linguaggio stesso.
- **Semantica**. l'insieme dei significati da attribuire alle frasi (sintatticamente corrette) costruite nel linguaggio.
- Una frase può essere sintatticamente corretta e tuttavia non avere significato.
  - semanticamente corretta $\rightarrow$ sintatticamente corretta
  - sintatticamente corretta $\not\rightarrow$ semanticamente corretta

Sintassi vs Semantica

L'utenza potenziale porta avanti la verifica critica degli obbiettivi istituzionali e l'individuazione di fini qualificanti in una visione organica e ricondotta a unità recuperando ovvero rivalutando nella misura in cui ciò sia fattibile un indispensabile salto di qualità.

- Che cosa vuol dire?
- Dal punto di vista sintattico la frase è corretta
- Dal punto di vista semantico, alla frase non è associabile un significato (non vuol dire niente)

Processi

- Processo: un'istanza di un programma in esecuzione
- Nei computer moderni possono esserci diversi processi in esecuzione ...
  - ... dovuti a più istanze dello stesso programma
  - ... dovuti a istanze di diversi programmi
  - o una combinazione delle precedenti
- Processore: la componente hardware che si occupa di eseguire le istruzioni
- Vedremo più avanti come un unico processore può eseguire più processi contemporaneamente