Algoritmi e Principi dell'Informatica

Davide Martinenghi
Email: davide.martinenghi@polimi.it
https://martinenghi.faculty.polimi.it/courses/api
Ed 20, piano 1; tel: 3669

ricevimento
Venerdì 10:30 previo appuntamento via email

Struttura del corso

Articolato in due parti:

1. “Informatica teorica”
2. “Algoritmi e strutture dati”

- Il primo modulo di API comprende i modelli dell’informatica (automi, logica) e la teoria della computazione (decidibilità…)
- Il secondo modulo parte dallo studio della complessità per poi occuparsi di strutture dati ed algoritmi
  - Algoritmi presentati in *pseudo-codice*
  - Non c’è una parte sui linguaggi di programmazione

Prerequisiti

- Le basi essenziali di Informatica (Fondamenti di Informatica)
- Elementi di matematica non continua (Logica e Algebra)

Il materiale didattico

- Testi ufficiali:
  - Mandrioli/Spoletini: *Informatica teorica* (II edizione), De Agostini, 2011
  - Cormen et al.: introduzione agli algoritmi e strutture dati Terza edizione, McGraw-Hill
    - Versione «create» e e-book su VitalSource Bookshelf
- Eserciziario: Mandrioli et al.: *Esercizi di informatica teorica* (III edizione), Città Studi
- Temi d'esame risolti, inclusi quelli del vecchio corso di Informatica Teorica a disposizione sul sito del corso
- Slide disponibili da subito

L'esame

- Basato sulla *capacità di applicare, non di ripetere*:
  - Scritto con possibilità di consultare testi
- Prima prova in itinere su informatica teorica
- Seconda prova in itinere su algoritmi e strutture dati
- Chi non ha partecipato alla prima p.i.i. o non l'ha superata, al posto della seconda p.i.i. sosterrà, nella stessa data, l'appello completo su entrambe le parti

L’esame (continua)

- Gli appelli sono su entrambe le parti, tuttavia **solo** chi è
  - complessivamente insufficiente
  - ma sufficiente in una delle due parti

  può rifare solo la parte in cui è insufficiente
- Un compito gravemente insufficiente, ad esempio <= 6 punti, comporta la valutazione «riprovato»
  - Se pensate di essere gravemente insufficienti, non consegnate!
  - Potrete comunque vedere le soluzioni, che verranno rese visibili sul sito del corso subito dopo l’esame

Prova finale

- Solo per chi l'ha nel piano di studi!
- Sarà presentata verso la fine di maggio e valutata:
  - Per i soli laureandi che ne facessero domanda, a luglio
  - Per tutti gli altri, in settembre
  - Per chi sarà laureando a febbraio dell'anno prossimo, si aprirà una finestra di valutazione in quel frangente
- Realizzazione di un progetto secondo specifiche date
- Valutazione automatica basata sul superamento di test con vincoli di efficienza… e **controlli anti-plagio molto severi**
  - NB: Il codice sviluppato non dev'essere reso pubblico (es., GitHub), se no gli altri possono copiarlo
  - Chi copia o fa copiare non supererà l'esame
  - Ogni anno capita di annullare qualche prova, anche a distanza di mesi

Difficoltà del corso

- Alcuni degli argomenti trattati nella parte di informatica teorica sono complessi
- Richiedono una capacità di astrazione e formalizzazione tipicamente non usata fino a questo punto degli studi
- Gli argomenti della seconda metà sono apparentemente più familiari, ma da non sottovalutare

- Il corso è complessivamente difficile!

Obiettivi e motivazioni

Perché l'informatica "teorica"?

La teoria stimolata dalla pratica:

generalità, rigore, "controllo"

- Comprendere a fondo e in maniera critica i principi dell'informatica (riesame approfondito di concetti elementari)
- Costruire basi solide per comprendere e dominare l'innovazione futura

Il programma

- **La modellizzazione informatica**
  (Come descrivo un problema e la sua soluzione):
  non tanto singoli modelli specialistici
  quanto creare la capacità di capire e "inventare"
  modelli vecchi e nuovi
- **La teoria della computazione**:
  che cosa so/posso fare con lo strumento informatico
  (quali problemi so risolvere)?

I modelli in campo ingegneristico

- In ingegneria la fase di progetto si basa sull'uso di *modelli*
- Infatti è spesso impossibile o impraticabile la verifica di soluzioni nel mondo reale
- Talvolta modelli *fisici* (es. ponte o diga in miniatura)
- Spesso modelli *formali*: oggetti matematici che fungono da rappresentazioni astratte di entità reali complesse

Uso dei modelli formali

1. Formalizzazione del problema: da entità reali ad astrazione matematica
2. Risoluzione del problema formalizzato
3. Interpretazione dei risultati ottenuti dal modello, ossia valutazione/deduzione delle scelte di progetto

Il modello è *adeguato* se i risultati ottenuti riflettono le proprietà che ci interessano del sistema fisico

(entro limiti di approssimazione accettabili)

I modelli per l'informatica

Fasi principali dell'ingegneria del software classica:

1. Analisi dei requisiti → *documento di specifica*
2. Progetto → *architettura sw*
3. Implementazione → *codice*

Verso i modelli formali

- Tradizionalmente si usa il linguaggio naturale
  - o semplici schemi con semantica informale

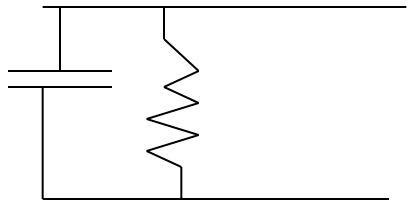
- La tendenza attuale è verso un uso sempre maggiore di linguaggi formali (o semi-formali) in tutte le fasi di produzione del software
  - Migliore affidabilità e facilità di manutenzione, ma soprattutto permette l'uso di **strumenti automatici**

I modelli dell'informatica: qualche nota

- Non (sol)tanto discreti anziché continui (bit e byte e non numeri reali), quanto:
  - *Generali*: il sistema informatico nel contesto (spesso) di un sistema più ampio: impianto, organizzazione, sistema "embedded", …
  - *Flessibili*: spesso non esiste il "modello già pronto": occorre saper adattare modelli esistenti ad esigenze specifiche e impreviste
    - esistono molti (troppi) modelli specialistici: occorre saper studiare/inventare nuovi modelli

Differenza con i modelli di altre discipline

- Occorre (maggiore) attitudine dinamica e critica:
  - confronto modello-realtà
  - analisi e sintesi del modello/progetto
- Approccio diverso rispetto a:



oppure

$$\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} + \frac{\partial^2 f}{\partial z^2} = g(x, y, z)$$

Formulazione del problema

- Spesso la vera difficoltà di un problema sta nel …. formularlo!

  Che cosa significa:
  - "automatizzare una procedura d'ufficio"
  - "evitare incidenti ferroviari/aerei/…"
  - …

    ?

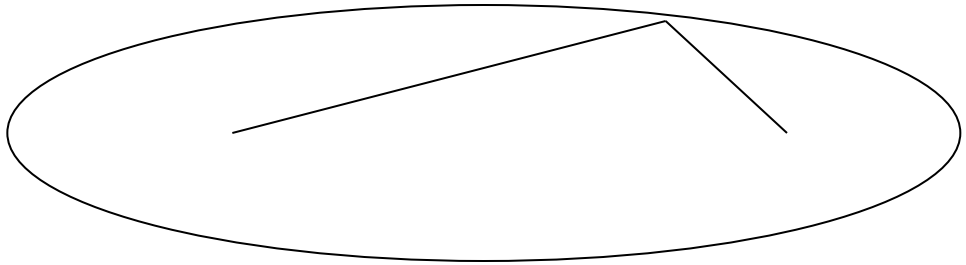
Tipi di modello

- Modelli *operazionali*
  (macchine astratte, sistemi dinamici, …)
  basati sul concetto di stato e di meccanismi
  (operazioni) per la sua evoluzione

- Modelli *descrittivi*
  tesi a formulare le proprietà desiderate o temute del
  sistema piuttosto che del suo funzionamento

Esempi di modello

- Modello operazionale dell'ellisse:



- Modello descrittivo dell'ellisse:

$$ax^2 + by^2 = c$$

Altri esempi di modello

- Descrizione operazionale dell'ordinamento:
  - Calcola il minimo e mettilo al primo posto;
  - Calcola il minimo degli elementi rimasti e mettilo al secondo posto;
  - …
- Descrizione non-operazionale dell'ordinamento:
  - Individua una permutazione della sequenza data tale che $\forall i \; (a[i] \leq a[i+1])$

Conclusioni

- In realtà le differenze tra modellizzazione operazionale e modellizzazione descrittiva non sono così nette: più che altro si tratta di un utile riferimento nel classificare un tipo di modello