

Algoritmi e Strutture Dati

A.A 2010/2011

Damiano Macedonio (mod A/B)

Nicola Rebagliati (mod A)

mace@dsi.unive.it

L'Informatica...

Informatica = Studio dei Computer (?)



«Computer science is no more about computers than astronomy is about telescopes»

*Edgser W. Dijkstra
(1930–2002)*

Informatica = Scrivere Programmi (?)

- La programmazione è importante, ma si tratta ancora di uno **strumento** (come il computer!).
- La programmazione viene utilizzata per **collaudare** le soluzioni proposte.

Informatica = Uso del PC e del software (?)

- « *Imparare l'utilizzo di un pacchetto software sta all'informatica come la patente di guida sta all'ingegneria meccanica* »
- Molte persone **usano** il software, ma l'informatico si occupa di **specificare, progettare, realizzare e collaudare** il software, oltre ai computer con i quali viene eseguito.

Tre convinzioni errate, ma non del tutto infondate

Sono semplicemente **incomplete**:

Computer, linguaggi di programmazione, software, e applicazioni fanno in effetti parte dell'informatica, ma nessuno di essi, e nemmeno tutti insieme, esauriscono la **ricchezza** e **varietà** di questa disciplina.

Definizione di Informatica

*L'informatica è
«la scienza dell'elaborazione (automatica)
dell'informazione»*

Dal francese: *Information automatique*
(P. Dreyfus, 1962)

Nozione centrale: **Algoritmo**

Definizione di Informatica

- L'informatica è *lo studio degli **algoritmi**, che comprende:*
 - Le loro **proprietà** formali e matematiche;
 - Le loro **implementazioni hardware**;
 - Le loro **implementazioni linguistiche**;
 - Le loro **applicazioni**.

(N. Gibbs e A. Tucker, 1986)

Algoritmo



«s.m. (dal nome del matematico persiano *al-Khwarizmi*), sistema di regole e procedure di calcolo ben definite che portano alla soluzione di un problema con un numero finito di operazioni.»

Operazioni

- **Sequenziali.** Eseguono una singola attività ben definita, terminata la quale si passa alla successiva.
- **Condizionali.** Pongono una domanda e l'operazione successiva è selezionata in base alla risposta.
- **Iterative.** Indicano di non proseguire con l'istruzione successiva, ma di ripetere un precedente blocco di istruzioni.

Operazioni sequenziali

- Aggiungi un cucchiaino di burro all'impasto nella scodella.
- Sottrai l'importo dell'assegno dal saldo del conto corrente.
- Imposta il valore di x a 1 .

Operazioni Condizionali

- Se l'impasto è troppo secco, aggiungi mezzo bicchiere d'acqua nella scodella.
- Se l'importo dell'assegno è minore o uguale al saldo del conto corrente, allora paga l'assegno; altrimenti informa la persona che l'assegno è scoperto.
- Se x diverso da 0 , allora imposta y a $1/x$, altrimenti stampa un messaggio di errore che informi dell'impossibilità di dividere per 0 .

Operazioni Iterative

- Ripeti le due operazioni precedenti finché l'impasto non si è ispessito.
- Finché vi sono ancora assegni da elaborare, esegui i seguenti cinque passaggi.
- Ripeti i passaggi *1,2* e *3* finché il valore di *y* è uguale a *+1*.

Esempio: videoregistratore

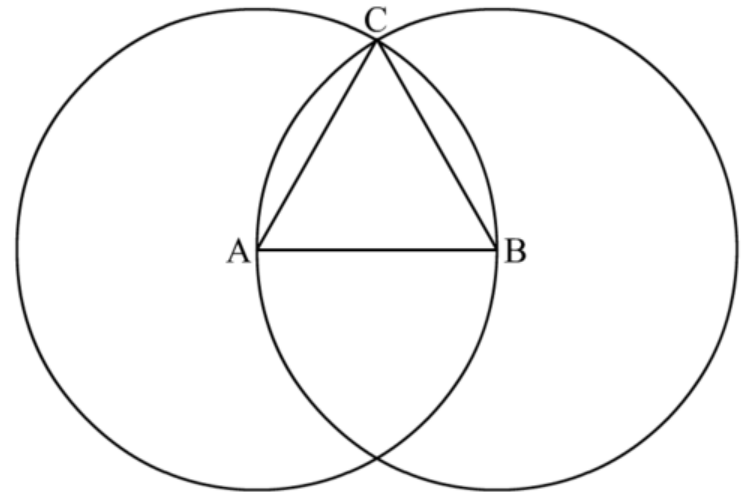
1. Se l'orologio e il calendario non sono regolati correttamente, vai a pag.9 del manuale, prima di passare a 2.
2. Inserisci un DVD vuoto nel videoregistratore.
3. Ripeti i passi da 4. a 7. per ogni programma che vuoi registrare.
4. Inserisci il numero del canale che vuoi registrare e premi il pulsante *CANALE*.
5. Inserisci l'ora a cui vuoi iniziare la registrazione e premi il pulsante *ORA-INIZIO*.
6. *Iserisci l'ora a cui vuoi interrompere la registrazione e premi il pulsante ORA-FINE.*
7. Se non vuoi registrare nient'altro, premi il pulsante *FINE-PROG*
8. Spegni il videoregistratore.

Gli *Elementi* di Euclide (300 a.C.)

Libro I, Proposizione I.

1. Sia AB il segmento dato.
2. Si tracci la circonferenza di centro A e raggio AB
3. Si tracci la circonferenza di centro B e raggio AB
4. Sia C un punto di intersezione tra le due circonferenze.
5. Il triangolo ABC è quello cercato.

Costruire un triangolo equilatero su un segmento dato.



Algoritmo di Euclide

Trova il massimo comun divisore di due interi.

1. Prendi due numeri interi positivi in input. Chiama m il maggiore ed n il minore.
2. Dividi m per n e chiama r il resto.
3. Se $r \neq 0$, reimposta m al valore di n , reimposta n al valore di r e torna al passo 2.
4. Stampa il valore di n , che da il risultato.

Esempio: Somma di Interi

Dati:

- $m \geq 1$
- $a_{m-1} a_{m-2} \dots a_0$ intero positivo di m cifre
- $b_{m-1} b_{m-2} \dots b_0$ intero positivo di m cifre

Trovare:

- $c_m c_{m-1} c_{m-2} \dots c_0 = a_{m-1} a_{m-2} \dots a_0 + b_{m-1} b_{m-2} \dots b_0$

Esempio: Somma di Interi (ctd)

1. Imposta il valore di *riporto* a 0 .
2. Imposta il valore di i a 0 .
3. Finché il valore di $i \leq m-1$, ripeti le istruzioni dei passi da i . a iii .
 - i. Somma le cifre a_i e b_i al valore corrente di *riporto* per avere c_i .
 - ii. Se $c_i \geq 10$ allora riporta c_i a $(c_i - 10)$ e imposta il valore di *riporto* a 1 ; altrimenti, imposta il nuovo valore di *riporto* a 0 .
 - iii. Somma 1 a i .
4. Imposta c_m al valore di *riporto*.
5. Stampa la soluzione finale $c_m c_{m-1} c_{m-2} \dots c_0$.

Un esempio complicato?

Perché esprimere in modo così complicato un'attività semplice come sommare due numeri?

Gli algoritmi sono fondamentali nell'informatica perché se siamo in grado di **specificare un algoritmo per risolvere un problema**, allora possiamo automatizzare la risoluzione del problema.

*‘Il computer non è una macchina intelligente che aiuta le persone stupide, anzi è una macchina stupida che funziona solo nelle mani delle persone intelligenti.’
(Umberto Eco)*

Definizione formale

Algoritmo: insieme ordinato di operazioni non ambigue ed effettivamente computabili che, quando eseguito, produce un risultato e si arresta in un tempo finito.

...insieme ordinato...

Bisogna sapere quale operazione eseguire per prima e quale eseguire dopo averne completata una.

Es. Da un flacone di shampoo:

1. Bagnare i capelli;
2. Insaponare;
3. Sciacquare;
4. Ripetere.

...non ambigue...

Le operazioni devono essere **comprese** dall'agente di calcolo.

Es. Torta di ciliegie:

1. Prepara la base;
2. Prepara il ripieno di ciliegie;
3. Metti il ripieno nella base;
4. Cuoci nel forno a 200° per 45 minuti.

...non ambigue...

Meglio andare più in dettaglio!

1. Prepara la base;

- i. Prendi tre tazze di farina.
- ii. Passa al setaccio la farina.
- iii. Mescola la farina setacciata con un etto di burro e e mezza tazza d'acqua.
- iv. Impasta due dischi da *25 cm* di diametro.

2. Prepara il ripieno di ciliegie;

- i. Apri un vasetto di marmellata di ciliegie da *400 g* e versa il contenuto in una ciotola.
- ii. Aggiungi un pizzico di cannella e noce moscata e mescola.

...effettivamente computabili...

Le operazioni devono essere **eseguibili** dall'agente di calcolo.

Es. “Sbatti le ali velocemente e vola!”

Es. Per stampare il *100-esimo* numero primo:

1. Genera un elenco di tutti i numeri primi.
2. Ordina l'elenco in ordine crescente.
3. Stampa il *100-esimo* elemento nell'ordine.

Es. Scrivi il valore decimale esatto di π .

Es. Imposta *media* a $(SommaDeiValori / NumeroDeiValori)$

Es. Imposta il valore di *Risultato* a \sqrt{x}

Es. Somma *1* al valore attuale di *x*.

...che produce un risultato...

Per determinare se una operazione è corretta
l'algoritmo deve produrre un risultato osservabile
dall'utente.

N.B. “risultato” e non “risposta”: nel caso l'algoritmo non sia in grado di dare una risposta, deve produrre qualcos'altro come un messaggio di errore o di avvertimento.

...e termina in tempo finito...

Evitare cicli infiniti!

1. Bagna i capelli
2. Insapona i capelli
3. Risciacqua i capelli
4. Insapona i capelli
5. Risciacqua i capelli

1. Bagna i capelli
2. Imposta il valore di *ContaLavaggi* a *0*.
3. Ripeti i seguenti passaggi finché *ContaLavaggi* vale *2*
 - i. Insapona i capelli
 - ii. Risciacqua i capelli
 - iii. Somma *1* al valore di *ContaLavaggi*

Obiettivo del Corso

- Acquisire competenze di base per progettare algoritmi:
 - Corretti.
 - Efficienti in termini di tempo.
 - Efficienti in termini di spazio.
 - Facili di modificare, estendere.

Esercizio

Un venditore vuole visitare 25 città riducendo al minimo il numero totale di chilometri da percorrere. Decide di progettare un algoritmo per determinare l'ordine ottimale in cui visitare le città per (1) mantenere minima la distanza percorsa e (2) visitare ogni città esattamente una volta.

L'algoritmo è il seguente: il calcolatore prima elenca tutti i possibili modi per visitare le 25 città e poi, per ciascuno, determina la distanza totale associata. Tra tutti i modi possibili sceglie quello la cui distanza associata è minore.

Se un calcolatore potesse analizzare 10 000 000 percorsi al secondo, quanto tempo servirebbe per determinare il percorso ottimale?

Pensate che l'algoritmo sia realizzabile?

Riuscite a escogitare una soluzione ragionevole al problema?

Programma (Mod A, 9 CFU)

Algoritmi, modelli di calcolo e metodologie di analisi. Introduzione informale agli algoritmi. Modelli di calcolo. Notazione asintotica. Ricorrenze.

Strutture dati elementari. Array, liste. Pile, code. Alberi.

Ordinamento. Heapsort. Mergesort. Quicksort. Mediane e statistiche d'ordine.

Alberi di ricerca. Alberi binari. Alberi rosso-neri. Alberi AVL.

Hashing. Tavole ad accesso diretto. Tavole hash. Risoluzione delle collisioni.

Tecniche fondamentali per il progetto di algoritmi. Tecnica *divide et impera*. Programmazione dinamica. Algoritmi golosi (*o greedy*).

Grafi e visite di grafi. Rappresentazione di grafi. Visita in ampiezza. Visita in profondità. Ordinamento topologico. Componenti fortemente connesse.

Alberi di copertura minimi. Definizioni e proprietà. Algoritmi di Kruskal e Prim.

Cammini minimi. Definizioni e proprietà. Algoritmi di Dijkstra, Bellman-Ford e Floyd-Warshall.

Problemi NP-completi. Classi di complessità P e NP. Riducibilità polinomiale e NP-completezza. Esempi di problemi NP-completi.

Programma (Mod B, 6 CFU)

Reti di flusso. Definizione e principali proprietà. Metodo di Ford-Fulkerson. Algoritmo di Edmonds-Karp. Applicazione al problema dell'abbinamento massimo nei grafi bipartiti.

Sperimentazione: Problema della segmentazione di immagini.

String matching. Formalizzazione del problema. Algoritmo di Rabin-Karp. Automa di sting-matching. Algoritmo di Knuth-Morris-Pratt.

Sperimentazione: Calcolo della struttura secondaria dell'RNA.

Geometria computazionale. Introduzione e risoluzione di problemi di geometria elementare. Involucro convesso. Algoritmi di Graham e Jarvis. Ricerca della copia di punti più vicini.

Sperimentazione: Applicazioni dell'involucro convesso.

Ogni argomento prevede inoltre esercitazioni individuali per verificare e applicare le competenze fornite durante il corso.

Testi

T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein

Introduction to Algorithms

MIT Press, 2001

(Traduzione italiana: *Introduzione agli algoritmi e strutture dati*, McGraw-Hill, 2005)

C. Demetrescu, I. Finocchi, G. F. Italiano

Algoritmi e strutture dati

McGraw-Hill, 2008 (seconda edizione)

Ricevimento

In aula dopo ogni lezione (magari mandate una e-mail per avvertire...)

... domande, commenti, suggerimenti via e-mail... sono benvenuti!

Domande durante la lezione...
possono essere di chiarimento per tutti!

Esame

- **Prova scritta:** esercizi, risoluzione di problemi, teoremi
- **Prova orale:** discussione su argomenti svolti a lezione



Domande?

Questo è un esame *facile*?

... **No.** Ma non è nemmeno impossibile, basta studiare. È fondamentale esercitarsi il più possibile. A tale proposito cerco di assegnare periodicamente degli esercizi che vi invito vivamente a svolgere. Gli argomenti trattati nel corso non possono essere padroneggiati in modo passivo, per cui è fondamentale affiancare lo studio della teoria con lo svolgimento degli esercizi.