Corso di Introduzione agli Algoritmi
Prof.ssa Tiziana Calamoneri

Syllabus e Introduzione

Cos'è un algoritmo? (1)

Un algoritmo è una sequenza di comandi "elementari" ed "univoci"

possono essere interpretati in un solo modo

non possono essere scomposti in comandi più semplici

Cos'è un algoritmo? (2)

Facciamo un esempio:

Ricetta della frittata:

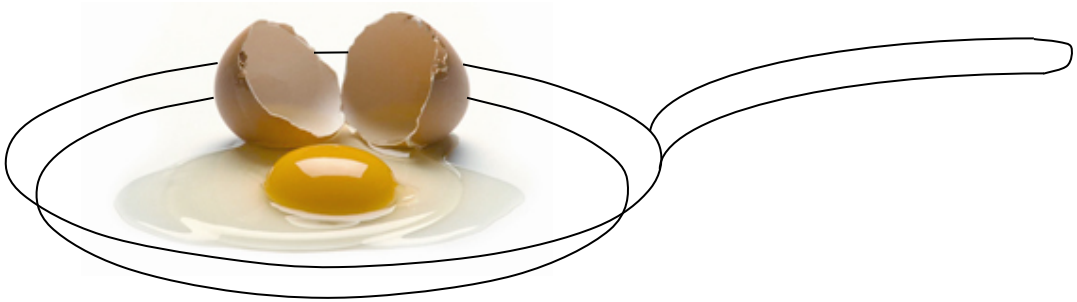
- rompere un uovo in una padella
- cuocere

Questo è un algoritmo???

Cos'è un algoritmo? (3)

- rompere un uovo in una padella
- cuocere

rompere un uovo in padella, oltre a non essere un'azione semplice, non è nemmeno univoca...



Potremmo riscrivere in questo modo:

- rompere il guscio dell'uovo
- buttare l'interno dell'uovo in padella

Cos'è un algoritmo? (4)

- rompere un uovo in una padella
- cuocere

cuocere NON è un'azione semplice perché si può scomporre in:

- accendere un fornello
- posizionarvi sopra la padella
- mettere un po' d'olio
- quando l'olio è caldo, inserire il cibo da cuocere
- attendere che il contenuto della padella sia cotto

Cos'è un algoritmo? (5)

Allora:

se l'algoritmo è ben fatto, chi lo esegue NON ha bisogno di pensare, ma solo di essere preciso come un... calcolatore!

ATTENZIONE: il calcolatore non pensa: esegue gli algoritmi pensati da una persona

Quindi: se il risultato finale è sbagliato, non ha sbagliato il calcolatore ma la persona che ha scritto l'algoritmo

...ed ora parliamo di questo corso...

Informazioni generali

Oltre al materiale che si trova sulla pagina web del corso, potete trovare informazioni utili qui:

- http://twiki.di.uniroma1.it/twiki/view/Intro_algo/EO/WebHome
- http://twiki.di.uniroma1.it/twiki/view/Intro_algo/AD/WebHome
- http://twiki.di.uniroma1.it/twiki/view/Info_gen/WebHome

Libri di testo

Un qualunque manuale di algoritmi e strutture dati fondamentali va bene, ma se dovete comprarne uno…

- Cormen, Leiserson, Rivest, Stein: Introduzione agli algoritmi e strutture dati, Mc Graw Hill

oppure

- Demetrescu, Finocchi, Italiano: Algoritmi e strutture dati, Mc Graw Hill

Sono anche a disposizione in rete delle dispense che trattano degli argomenti di questo corso.

In questo corso… (1)

UNITELMA SAPIENZA

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

…descriveremo:

- Alcuni algoritmi “classici” per risolvere problemi di base (ricerca e ordinamento)
- Esploreremo le strutture dati più adatte da usare
- Valuteremo l’efficienza, calcolandone il costo computazionale.

P.S. gli algoritmi verranno descritti tramite pseudocodice, per poterne dare una versione compatta senza perdersi in dettagli implementativi…

In questo corso... (2)

Per risolvere problemi avremo bisogno di usare dati e quindi useremo delle STRUTTURE DATI, strumenti per memorizzare e organizzare i dati e semplificarne l'accesso e la modifica.

Non esiste una struttura dati che vada bene per ogni problema, quindi dobbiamo conoscere proprietà, vantaggi e svantaggi delle principali strutture dati.

Esempi: vettore, lista, coda, pila...

Suggerimenti

Non basta essere solo “spettatori”:

- Alla fine di (quasi) ogni lezione troverete degli esercizi che vengono proposti.
- Provate a risolverli e, se non ci riuscite, approfondite ulteriormente l’argomento trattato.

Introduzione

Un algoritmo è… (1)

… una procedura di calcolo ben definita costituita da una sequenza finita di passi elementari che un esecutore deve seguire per generare un insieme di valori in OUTPUT a partire da un insieme di valori in INPUT.

Un algoritmo è uno strumento per risolvere un PROBLEMA COMPUTAZIONALE: ne descrive una specifica procedura computazionale per ottenere la relazione tra input ed output specificata dal problema.

Nell'esempio della frittata:

- Input: ingredienti
- Output: frittata
- Algoritmo: ricetta
- Esecutore: cuoco

Un algoritmo è… (2)

Esempio di problema computazionale: ordinamento

Input: sequenza di n numeri $a_1, a_2, \ldots, a_n$

(anche detto ISTANZA)

Output: permutazione $a_1', a_2', \ldots, a_n'$ della sequenza di input con la proprietà che $a_1' \leq a_2' \leq \ldots \leq a_n'$

Un algoritmo è CORRETTO se, per ogni istanza, termina con l'output corretto. In tal caso diremo che l'algoritmo RISOLVE il problema dato.

Tempo computazionale (1)

Un parametro fondamentale per lo studio degli algoritmi è l'EFFICIENZA, cioè l'ottimizzazione del tempo e dello spazio occupati da un algoritmo.

Questo perché i calcolatori possono essere veloci, ma non infinitamente, la memoria può essere economica, ma non gratuita.

Lungo tutto il corso parleremo di COSTO COMPUTAZIONALE in termini di numero di operazioni elementari, come funzione della dimensione dell'input.

Tempo computazionale (2)

Per comprendere l'importanza del costo computazionale di un algoritmo, in particolare per quanto riguarda il tempo di esecuzione richiesto, facciamo un semplice esempio:

calcolatore V (veloce): 10^9 istruzioni/sec

calcolatore L (lento): 10^7 istruzioni/sec

Ordinamento di $n=10^6$ interi

Algoritmo IS: $2n^2$ istruzioni

Algoritmo MS: 50 $n \log n$ istruzioni

Domanda: Eseguiamo IS sul calcolatore V ed MS sul calcolatore L.

La maggiore velocità del calcolatore V riesce a bilanciare la minore efficienza dell'algoritmo IS?

Risposta: No...

Tempo computazionale (3)

Infatti:

V(IS)= $\frac{2*(10^6)^2 istruz.}{10^9 istruz./\sec.} = 2000\sec. = 33\min.$

L(MS)=$\frac{50*10^6 \log 10^6 istruz.}{10^7 istruz.} = 100\sec. = 1.5\min.$

Addirittura, se $n=10^7$, il divario aumenta:

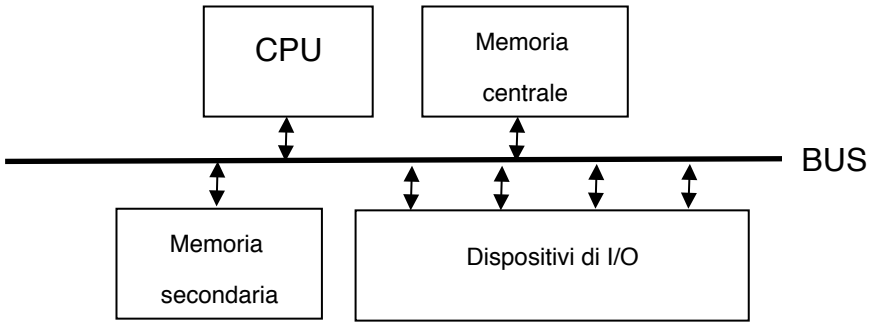
V(IS)=2 giorni e L(MS)=20 min.

Questo ci fa capire come, indipendentemente dall'aumento di velocità dei calcolatori prodotto dagli avanzamenti tecnologici, l'efficienza degli algoritmi sia un fattore di importanza cruciale.

Random Access Machine (1)

- Per studiare l'efficienza di un algoritmo dobbiamo ANALIZZARLO, cioè prevedere le risorse che esso richiede.
- Per fare questo, dobbiamo avere un modello astratto di calcolatore, che sia indipendente dalla tecnologia usata.
- In questo corso consideriamo un modello astratto di calcolo detto RANDOM ACCESS MACHINE (RAM).

Random Access Machine (2)



Secondo questo modello, il calcolatore è costituito di quattro unità funzionali:

- processore (CPU, Central processing Unit);
- memoria centrale (RAM, Random Access Memory);
- memoria secondaria (o memoria di massa);
- dispositivi di input/output (tastiera, schermo, stampante, ecc.).

Random Access Machine (3)

- La memoria centrale (RAM) può essere vista come una lunga sequenza di componenti elementari, ognuna delle quali contiene una unità di informazione, il bit (abbreviazione di binary digit) che assume solo i valori zero e uno). Ciascuno di questi componenti è anche detto cella di memoria.
- Un gruppo di 8 bits è detto byte.
- I bytes sono aggregati fra loro in strutture, dette registri di memoria o parole di memoria, caratterizzate dal fatto che su di esse il processore è in grado di operare (in lettura o scrittura) con un'unica operazione. Nei calcolatori attuali una parola di memoria può essere costituita da 4 bytes (calcolatori a 32 bits) o 8 bytes (calcolatori a 64 bits).

Random Access Machine (4)

- Aspetti importanti sono i seguenti:
  - ciascuna parola di memoria ha un indirizzo;
  - gli indirizzi corrispondono alla posizione dei bytes nella sequenza;
  - gli indirizzi sono numeri interi e non sono memorizzati da nessuna parte, ma sono determinati dall'ordinamento consecutivo delle parole stesse;
  - solo i bytes e non i singoli bits sono indirizzabili (cioè accessibili da parte della CPU).

Random Access Machine (5)

- Il termine random access memory usato per la memoria centrale convoglia un concetto molto importante: il tempo di accesso ad una qualunque parola di memoria è sempre lo stesso, indipendentemente dalla posizione (e quindi dall'indirizzo) della parola.
- La parola di memoria è l'unità massima sulla quale è possibile operare mediante un'unica operazione (istruzione).
- Per eseguire l'operazione si deve specificare l'indirizzo della parola di memoria su cui si vuole operare, che coincide con l'indirizzo del primo byte facente parte della parola stessa.

Random Access Machine (6)

- La RAM (memoria centrale) è caratterizzata dal tempo di accesso, ossia il tempo necessario per leggere l'informazione contenuta in una parola di memoria o scrivere un'informazione su una parola di memoria. Oggi è dell'ordine dei nanosecondi (10^{-9} secondi).
- La memoria centrale è piuttosto costosa, e perde il suo contenuto quando viene a mancare l'alimentazione elettrica. Questa caratteristica si indica col termine volatilità.

Random Access Machine (7)

Per superare il problema della volatilità si fa ricorso a un tipo diverso di memoria, la memoria secondaria (o memoria di massa), con le seguenti caratteristiche:

- conserva il contenuto (programmi e dati) anche in assenza di alimentazione elettrica;
- è molto più lenta della memoria centrale;
- è molto più abbondante della memoria centrale (arriva anche ai terabyte, 10^{12} byte);
- è meno costosa della memoria centrale.

Random Access Machine (8)

Caratteristiche del modello RAM (random access machine):

- esiste un singolo processore, che esegue le operazioni sequenzialmente, una dopo l'altra (no più operazioni contemporaneamente);
- esiste un insieme di operazioni elementari, l'esecuzione di ciascuna delle quali richiede per definizione un tempo costante.
  Esempi: operazioni aritmetiche, di lettura, di scrittura, salto condizionato, ecc.;
- esiste una relazione precisa tra il numero di bits usati per rappresentare l'informazione (cioè la dimensione della parola di memoria) e il massimo valore rappresentato:

$$\text{max val. rappresentato} = 2^{\text{dim. della parola}}$$

Misure di costo (1)

- Abbiamo fatto l'ipotesi che ogni dato in input sia minore di $2^{\text{dim. della parola}}$ per consentire che ogni operazione sia eseguita in tempo costante, indipendentemente dalla dimensione degli operandi.
- Questo criterio è detto MISURA DI COSTO UNIFORME.
- Sebbene utile, questo è un modello di costo non sempre realistico perché i dati potrebbero non essere rappresentabili in una sola parola di memoria, e quindi anche le operazioni aritmetiche non richiederebbero, in realtà, tempo costante, ma logaritmico.

Misure di costo (2)

- E' stato perciò proposto un altro criterio di misurazione, MISURA DI COSTO LOGARITMICO, che assume il costo delle istruzioni dipendente dalla dimensione degli operandi. Cioè, eseguire un'operazione su un numero n implica un tempo proporzionale a $\log n$ (=num. di bits necessari a memorizzare n).
- Tutte le volte in cui possiamo assumere che i dati di input siano rappresentati da un numero costante di bit, le due misure coincidono (a meno di una costante moltiplicativa).
- Poiché usare la misura di costo logaritmico potrebbe inserire delle complicazioni non necessarie, in questo corso (ed in letteratura!) si sceglie di usare la misura di costo uniforme, che è adatta alla maggior parte dei problemi che studieremo.

Misure di costo (3)

Esempio:

x=1;

for *i*=1 to *n* do

    x=*x***2*; /*calcola 2^n

C.U. tempo di esecuzione proporzionale ad *n*

C.L. *x* raddoppia ad ogni passo, quindi all'*i*-esimo passo $x=2^i$

Il tempo speso per incrementare *i* e raddoppiare *x* è, rispettivamente, *log i* e *log x*= *log* 2^i=*i*. Quindi il tempo di esecuzione è proporzionale a

$$\sum_{i=1}^{n}(i+\log i)$$

$$\sum_{i=1}^{n}(i+\log i) \geq \sum_{i=1}^{n} i = n(n+1)/2 \qquad \sum_{i=1}^{n}(i+\log i) \leq \sum_{i=1}^{n} 2i = n(n+1)$$

Pseudocodice (1)

Concludiamo con l'osservare che, affinché ogni esecutore sia in grado di comprendere un algoritmo, è necessario utilizzare una descrizione:

- il più formale possibile
- indipendente dal linguaggio che si intende usare

Pseudocodice

Pseudocodice (2)

Lo pseudocodice è una sorta di linguaggio di programmazione "informale" nell'ambito del quale:

- si impiegano, come nei linguaggi di programmazione, i costrutti di controllo (for, if then else, while, ecc.);
- si impiega il linguaggio naturale per specificare le operazioni;
- si ignorano i problemi di ingegneria del software;
- si omette la gestione degli errori, al fine di esprimere solo l'essenza della soluzione.

Non esiste una notazione universalmente accettata per lo pseudocodice.