Lezione 1

Algoritmi

Introduzione

- Un tipo di azioni compiute dall'uomo: soluzione di problemi
- un tipo di problema: elaborazione di informazione
  - Es: calcolare l'area di un cerchio, riconoscere il volto di una persona

Elaborazione dell'informazione

- Un problema di elaborazione dell'informazione è caratterizzato da:
  - insieme di dati di partenza
  - un risultato cercato
  - una soluzione: una procedura che genera il risultato a partire dai dati di partenza

La soluzione

- La conoscenza di come si risolve un problema e la capacità di risolverlo sono competenze distinte
- Es: ognuno è capace di riconoscere un volto, ma come avviene questo riconoscimento? Come descrivere la procedura per riconoscere uno specifico volto?

La procedura di soluzione

- Può capitare di trovarsi di fronte ad un problema la cui soluzione debba essere attuata non da noi, ma da una altro soggetto
- il soggetto può non sapere come risolvere il problema, sebbene possa dichiarare la sua disponibilità ad attuare la soluzione nel momento in cui gli venisse insegnata

La procedura di soluzione

- La procedura di soluzione deve allora essere realizzata in fasi distinte e successive:
  - analisi del problema e identificazione di una soluzione da parte del primo soggetto
  - descrizione della soluzione da parte del primo soggetto in termini comprensibili al secondo soggetto
  - interpretazione della soluzione da parte del secondo soggetto
  - attuazione della soluzione da parte del secondo soggetto

L'esecutore

- La procedura di soluzione deve essere descritta in una forma che l'esecutore sia in grado di di interpretare in modo corretto
- la soluzione deve specificare delle azioni che l'esecutore è in grado di attuare

- ogni esecutore è caratterizzato dalle sue capacità di interpretazione e di attuazione

Il calcolatore

- I calcolatori sono degli esecutori di soluzioni che esseri umani hanno precedentemente identificato e descritto.

- I calcolatori hanno una notevole velocità di esecuzione e possono ripetere la stessa operazione producendo sempre lo stesso risultato un numero elevato di volte

Il calcolatore

- Il calcolatore in quanto esecutore è caratterizzato da:
  - un linguaggio che è in grado di interpretare, con il quale devono essere descritte le soluzioni che vuole che esso attui
  - istruzioni che è in grado di eseguire

Problemi e algoritmi

- Se un problema è particolarmente semplice, l'esecutore potrebbe essere in grado di eseguire la soluzione direttamente
- Es: determinare la superficie di un cerchio di raggio r
- ma se il risolutore non conosce la formula risolutiva la si deve indicare esplicitamente: $s = \pi r^2$
- tuttavia se l'esecutore non conosce come elevare un numero al quadrato, si ha che la soluzione contiene a sua volta un problema la cui soluzione deve essere descritta in modo esplicito

Gerarchia di problemi

- In generale per giungere alla descrizione della soluzione di un problema si scompone il problema in sotto-problemi, e questi in sotto-sottoproblemi
- ci si ferma quando si giunge ad un problema elementare o primitivo la cui soluzione corrisponda ad una azione elementare che può essere direttamente compiuta dall'esecutore
- risolvere un problema equivale a risolvere una opportuna successione di problemi più semplici

Soluzione effettiva

- Definiamo effettiva per un esecutore la soluzione di un problema quando:
  - l'esecutore è in grado di interpretare la descrizione di tale soluzione e associare ad essa le azioni che deve compiere per eseguirla
  - l'esecutore è in grado di compiere tali azioni completandone l'esecuzione in tempo finito

Soluzione mediante scomposizione

- La soluzione di un problema mediante la sua scomposizione in sottoproblemi è strutturata nel seguente modo:
  - se la soluzione del problema è effettiva, allora l'esecutore la attua
  - altrimenti il problema viene scomposto in sottoproblemi, e per ognuno di questi si applica nuovamente la procedura

Procedura effettiva

- L'insieme dei sotto- problemi viene risolto da una procedura effettiva quando:
  - tutti i problemi sono elementari
  - è fissato l'ordine di soluzione dei problemi
  - è specificato il modo in cui un problema utilizza i risultati dei problemi che lo precedono

Problemi e procedure effettive

- A ogni scomposizione di un problema in sotto-problemi può essere associata una procedura effettiva quando vengano considerati come elementari tutti i sotto-problemi
- i concetti di problema elementare e azione elementare sono strettamente associati ed evidenziano l'aspetto descrittivo e esecutivo delle procedure risolutive.

L'esecutore (raffinamento della definizione)

- Un esecutore è caratterizzato da:
  - il linguaggio che è in grado di comprendere
  - l'insieme delle azioni che è in grado di compiere
  - l'insieme delle regole che a ogni costrutto linguistico sintatticamente corretto associano le rispettive azioni da compiere

Ambiguità

- Finché la soluzione di un problema viene descritta in termini informali (come ad es. tra gli esseri umani) può rimanere l'ambiguità circa l'attuabilità della soluzione da parte dell'esecutore (la sua effettività).
- Si ha ambiguità quando due soggetti giudicano come effettiva la stessa soluzione di un problema ma poi compiono azioni che producono risultati differenti
- per rimuovere tale ambiguità si deve formalizzare la definizione di un esecutore

L'esecutore (formalizzazione della definizione)

- Caratterizzazione sintattica dell'esecutore: il linguaggio che l'esecutore è in grado di interpretare deve essere definito in termini formali
- l'insieme delle azioni elementari che l'esecutore è in grado di compiere deve essere unicamente definito, e tali azioni devono essere deterministiche, cioè l'esecuzione di una stessa azione deve sempre produrre lo stesso risultato
- Caratterizzazione semantica dell'esecutore: l'insieme delle regole di associazione tra costrutti del linguaggio e azioni deve essere univocamente definito

Algoritmi e programmi

- Le soluzioni effettive per esecutori caratterizzati formalmente sono chiamate algoritmi
- quando l'esecutore è un calcolatore, gli algoritmi vengono detti programmi
- il linguaggio formale per la loro descrizione è detto linguaggio di programmazione

Sviluppo di un programma

- Il processo di sviluppo di un programma è organizzato in:
  - analisi del problema e identificazione di una soluzione
  - formalizzazione della soluzione e definizione dell'algoritmo risolutivo
  - programmazione, cioè scrittura dell'algoritmo in un linguaggio di programmazione di "alto livello"
  - traduzione del programma in un "linguaggio macchina" direttamente interpretabile dalla macchina

Linguaggi di alto livello e linguaggio macchina

- I linguaggi di alto livello sono più facilmente comprensibili dagli esseri umani ma sono sempre linguaggi formali
- il linguaggio macchina è un linguaggio formale comprensibile direttamente da uno specifico calcolatore
- la traduzione da quello di alto livello a quello macchina può essere fatta automaticamente in virtù delle proprietà formali di entrambi

La macchina universale

- Un elaboratore o computer è una macchina *digitale, elettronica, automatica* capace di effettuare trasformazioni o *elaborazioni* su i dati
  - digitale= l'informazione è rappresentata in forma numerica discreta
  - elettronica= la logica di manipolazione e la memorizzazione sono implementate con tecnologie di tipo elettronico (piuttosto che di tipo meccanico)
  - automatica= è in grado di eseguire una successione di operazioni in modo autonomo (cioè senza intervento di un operatore umano)

La macchina universale

- Le operazioni sono descritte sotto forma di un *programma*
- Il programma e i dati su cui deve operare sono registrate in un *dispositivo di memoria*
- Un dispositivo detto *unità di controllo* legge il programma e lo esegue su i dati
- Questo modo di operare è detto *architettura di Von Neumann*

La macchina universale

- Il programma permette di risolvere un problema in funzione dei dati
- Se i dati possono cambiare e il programma risolve sempre il problema, allora si dice che il programma risolve una classe di problemi

- Es. l'algoritmo per la somma di due numeri funziona per qualsiasi coppia di numeri; è indipendente dai due numeri dati in ingresso

La macchina universale

- L'elaboratore si dice universale perché può essere usato per risolvere qualsiasi problema la cui soluzione può essere descritta con un programma
- Per ogni classe di problema è necessario fornire un programma adeguato

Esempi

- Diamo un esempio di algoritmo semplice e di un algoritmo che usi il risultato di un altro algoritmo per risolvere un problema più complesso

Determinazione del maggiore di due numeri interi

- Occorre definire quali problemi sono elementari, cioè quali problemi hanno una soluzione che può essere eseguita direttamente senza dover ricorrere ad altre scomposizioni.
- Supponiamo che la differenza tra due interi e la valutazione del segno positivo o negativo di un numero siano problemi elementari

Determinazione del maggiore di due numeri

- P1 leggi un valore dall'esterno e inseriscilo nella variabile x
- P2 leggi un valore dall'esterno e inseriscilo nella variabile y
- P3 calcola la differenza d ← x - y
- P4 se d ha segno positivo vai al passo P5 altrimenti al passo P6
- P5 stampa "Il massimo e'" e il valore di x; vai al passo P7
- P6 stampa "Il massimo e'" e il valore di y
- P7 termina

Scomposizione in sottoproblemi

- Per problemi più complessi il numero di passi cresce notevolmente
- per semplificare la scrittura di un algoritmo lo si può scrivere in funzione di sottoproblemi non elementari perché di essi sia nota la scomposizione in problemi elementari
- questi problemi dalla soluzione nota sono detti problemi terminali

Scomposizione in sottoproblemi

- L'esecuzione di un algoritmo può essere pensata in termini di soluzione per un insieme di problemi terminali
- in un linguaggio di programmazione
  - alla soluzione dei problemi terminali elementari corrisponde il concetto di istruzione
  - alla soluzione dei problemi terminali non elementari corrisponde il concetto di sottoprogramma (procedura o funzione)

Determinazione del maggiore tra tre numeri

- Possiamo considerare terminale l'algoritmo per la soluzione del problema del massimo tra due numeri.
- Il problema può essere dunque scomposto come:
- P1 se x è maggiore di y allora esegui P2 altrimenti esegui P3
- P2 la soluzione è il maggiore tra x e z
- P3 la soluzione è il maggiore tra y e z

Determinazione del maggiore di n numeri

- Si può generalizzare il procedimento ottenendo:

- P1 trova il maggiore tra i primi due numeri
- P2 trova il maggiore tra il terzo ed il risultato del passo precedente
- P3 trova il maggiore tra il quarto ed il risultato del passo precedente
- ...

Determinazione del maggiore di n numeri

- Più elegantemente:
- P1 trova il maggiore tra i primi due numeri
- P2 finché ci sono numeri esegui P3 altrimenti esegui P4
- P3 trova il maggiore tra il nuovo numero e quello trovato al passo precedente
- P4 la soluzione è l'ultimo numero trovato al passo P3

Nota

- Il passo P2 mostra una struttura usata spesso nella descrizione dei problemi ripetitivi: "finché condizione ripeti azione".
- Tale struttura indica che l'azione deve essere eseguita ripetutamente valutando ogni volta la condizione
- In questo modo si ottiene una formulazione molto concisa e indipendente da ogni specifico valore di n
- Un problema che ammette una soluzione di questo tipo si dice che ha una soluzione di tipo iterativa

Nota

- I linguaggi di programmazione hanno modi molto compatti per esprimere diverse strutture iterative per controllare il flusso dell'elaborazione

Esempi

- Caratterizziamo ulteriormente le proprietà delle proposizioni usate nel linguaggio formale usato per descrivere un algoritmo

Algoritmo per il calcolo delle radici di una equazione di 2°grado

- Problema: data l'equazione $ax^2+bx+c=0$ trovare i valori di x che la soddisfano una volta assegnati i coefficienti a,b,c.

- Algoritmo
  - 1. Inizio algoritmo
  - 2. Acquisire i coefficienti a,b,c
  - 3. Calcolare il valore $\Delta = b^2-4ac$
  - 4. Se $\Delta<0$ non esistono radici, eseguire l'istruzione 8
  - 5. Se $\Delta=0$ allora $x_1 = x_2 = -b/2a$
  - 6. Se $\Delta>0$ allora $x_1=(-b+\sqrt{\Delta})/2a$, $x_2=(-b-\sqrt{\Delta})/2a$
  - 7. Comunicare all'esterno i valori x_1 e x_2
  - 8. Fine dell'algoritmo

Gioco dell'11

- Ci sono 11 oggetti e due giocatori A e B. I giocatori a turno prendono fino a 3 oggetti. Perde chi prende l'ultimo oggetto. Problema: quale è l'algoritmo che permette al giocatore A di vincere?
- Algoritmo:
  - 1. Inizio dell'algoritmo
  - 2. A preleva 2 oggetti
  - 3. B preleva k oggetti
  - 4. A preleva $4-k$ oggetti
  - 5. Se gli oggetti non sono finiti, allora ritorna a istruzione 3
  - 6. Fine dell'algoritmo

Proprietà degli algoritmi

- Perché una descrizione formale di una soluzione sia un algoritmo devono essere soddisfatti i seguenti requisiti:
  - Finitezza
  - Generalità
  - Non ambiguità

Proprietà degli algoritmi

- Finitezza
  - Il numero di istruzioni è finito
  - Ogni istruzione è eseguita in un intervallo finito di tempo
  - Ogni istruzione è eseguita un numero finito di volte

Proprietà degli algoritmi

- Generalità
  - un algoritmo fornisce la soluzione ad una classe di problemi
  - cioè:
    - dato un insieme di definizione o *dominio*
    - dato un insieme di arrivo o *codominio*
    - l'algoritmo può operare su tutti i dati appartenenti al dominio per fornire una soluzione all'interno del codominio

Proprietà degli algoritmi

- Non ambiguità
  - le istruzioni sono definite in modo univoco
  - non ci sono paradossi, contraddizioni, ambiguità
  - a parità di dati su cui lavorare il risultato dell'algoritmo è identico indipendentemente da chi lo sta eseguendo

Esempio

- Testiamo l'Alg. Delle Radici per le proprietà di algoritmo:
  - *Finitezza*: ci sono 8 istruzioni. Tutte sono eseguite al più una volta. Tutte impiegano un tempo finito per essere valutate o eseguite
  - *Generalità*: in ingresso è ammissibile una qualsiasi tripla di numeri reali e l'uscita è un numero reale.
  - *Non ambiguità*: le istruzioni sono ben definite: operazioni aritmetiche o confronti fra reali

Descrizione degli algoritmi

- Le proposizioni usate da un linguaggio formale descrivono due entità:
  - le operazioni che devono essere eseguite (*istruzioni*)
  - gli oggetti (*dati*) sui quali si devono eseguire le operazioni

Costanti e variabili

- I dati possono essere *costanti* o *variabili*
- I dati costanti sono dati che rimangono inalterati durante l'esecuzione dell'algoritmo da quando ha inizio a quando termina

- Es. Nell'Alg. dell'11 è una costante il numero di oggetti (2) preso dal primo giocatore.

Variabili

- I dati variabili o semplicemente *variabili* sono coppie *<nome, valore>*
- possono essere immaginati come scatole che hanno come etichetta il *nome* e come contenuto il *valore*
- Alle variabili **deve** essere assegnato esplicitamente un valore.
- Al momento di inizio di un algoritmo le variabili hanno un valore **indeterminato**.

- Es: in A1 sono variabili *a,b,c,* x_1 e x_2

Assegnazione

- L'istruzione di assegnazione è quella particolare istruzione che permette di *definire* il valore attuale di una variabile.
- Il valore rimane *inalterato* fino a una nuova assegnazione alla variabile.
- Forma generale:

  nome ← espressione

- L'assegnazione viene eseguita nei seguenti passi:
  - si valuta l'espressione di destra
  - si attribuisce il valore determinato alla variabile

Assegnazione

- **Regola**
  - Ogni volta che una variabile appare a destra di dell'istruzione di assegnazione ← , è necessario che un valore sia già stato assegnato a quella variabile.
- Es:

  nel caso a ← 2, b ← 3, c ← a+b, allora si ha c=5

  nel caso x ← 2, x ← x+3, allora si ha x=5

Istruzioni

- Le istruzioni possono essere divise in 5 categorie
  - *istruzioni operative*
  - *istruzioni di controllo*
  - *istruzioni di salto*
  - *istruzioni di inizio/fine esecuzione*
  - *istruzioni di ingresso/uscita*

Istruzioni operative

- Istruzioni che *producono un risultato* se eseguite
- Ne fanno parte le operazioni aritmetiche e le assegnazioni

Istruzioni di controllo

- Istruzioni che controllano il verificarsi di condizioni specificate e che in base al risultato determinano quale istruzione eseguire
- Es.
  se ... allora ...altrimenti (*if ... then ... else*)

Istruzioni di salto

- Istruzioni che alterano il normale ordine di esecuzione delle istruzioni di un algoritmo, specificando esplicitamente quale sia la successiva istruzione da eseguire
- Si distinguono istruzioni di salto *condizionato* o *incondizionato*
- Per quelle condizionate il salto è subordinato al verificarsi di una condizione specificata, per quelle incondizionate il salto è eseguito ogni volta che l'istruzione viene eseguita.

Istruzioni di inizio/fine

- Indicano quale istruzione dell'algoritmo debba essere eseguita inizialmente e quale determini la fine dell'esecuzione

Istruzioni di ingresso/uscita

- Istruzioni che indicano una trasmissione di dati o messaggi fra l'algoritmo e tutto ciò che è esterno all'algoritmo
- si dicono di ingresso o *lettura* quando l'algoritmo riceve dati dall'esterno
- si dicono di uscita o *scrittura* quando i dati sono comunicati dall'algoritmo all'esterno

Esempio

- Nell'algoritmo delle Radici
  - Le istruzioni 1,8 sono di inizio/fine
  - Le istruzioni 2,7 sono di ingresso/uscita
  - La istruzione 3 è operativa
  - La istruzione 4 è di salto condizionato
  - Le istruzioni 5,6 sono condizionali

Proposizioni

- Un *proposizione* è un costrutto linguistico del quale si può dire se è vero o falso
- Il *valore di verità* di una proposizione è l'essere vera o falsa della proposizione

- Es:

"3 è un numero pari" è una proposizione

"Incrementa *x* di 10" non è una proposizione

Predicati

- E' un *predicato* una proposizione che contiene delle variabili e in cui il valore delle variabili determina il valore di verità del predicato
- Es. la variabile *età* è minore di 30

Predicati

- La valutazione di un predicato avviene tramite i seguenti *operatori relazionali*

  = uguale, ≠ diverso
  > maggiore, ≥ maggiore o uguale
  < minore, ≤ minore o uguale

Predicati semplici e composti

- Un predicato che contiene un solo operatore relazionale è detto predicato semplice
- Gli operatori relazionali possono essere combinati con i seguenti *operatori logici*

  NOT

  AND

  OR
- Un predicato che contiene più operatori relazionali combinati tramite uno o più operatori logici è detto composto

Operatori logici

- NOT: dato un predicato *p*, *NOT p* è vero quando *p* è falso e viceversa
- AND: dati due predicati *p* e *q*, *p AND q* è vero solo quando entrambi *p* e *q* sono veri e falso altrimenti
- OR: dati due predicati *p* e *q*, *p OR q* è falso solo quando entrambi *p* e *q* sono falsi e vero altrimenti

Vettori

- Le variabili considerate fino ad adesso sono dette *variabili scalari*
- Una *variabile vettore* (*array*) è una coppia

  <nome, insieme di valori>
- Può essere immaginata come una scatola che ha un nome e che è divisa in tanti comparti ognuno dei quali è numerato e può contenere un valore

Vettori

- Ogni valore è individuato dal nome della variabile seguito dal numero del comparto detto *indice*
- La notazione usata è:

  A(i)
- La *dimensione* di un vettore è il numero dei suoi elementi

Vettori

- Gli indici vanno da 1 ad un numero massimo rappresentato dalla dimensione del vettore
- In fase di dichiarazione di una variabile vettore si specifica la sua dimensione che non è più modificabile successivamente

Matrice

- E' l'estensione del concetto di vettore.
- Una matrice è un insieme di valori che sono indicizzati facendo ricorso a due o più indici
- La notazione usata è:

  M(i,j)

- Per una matrice a 2 dimensioni l'îndice *i* è detto indice *riga* e *j* indice *colonna*