INTRODUZIONE ALL'(INGEGNERIA) INFORMATICA

INGEGNERIA: macchine che lavorano. Reattori chimici promuovono reazioni chimiche che trasformano materia in altra materia, macchine termiche trasformano energia termica (calore) in energia di altra natura, macchine meccaniche trasformano energia di qualche natura in energia meccanica (es. cinetica) ecc.

INGEGNERIA INFORMATICA: macchine che lavorano sull'INFORMAZIONE. Elaborazione dell'informazione. Trasformano informazione in altra informazione. Es: due numeri naturali nella loro somma (altro numero). Oppure due numeri naturali nel loro MCD, oppure una lista di studenti in una percentuale di promossi all'esame di Fondamenti di informatica, oppure un segnale digitale (ECG) in una frequenza cardiaca o in una diagnosi proposta, oppure un'immagine in una serie di nomi di proprietari di facce presenti.

Peculiarità: all'uscita da una fase di elaborazione di informazione troviamo di nuovo informazione: questa può nuovamente essere elaborata dalla stessa macchina, producendo elaborazioni di grande complessità.

INFORMATICA: scienza (e arte) della rappresentazione dell'informazione e dell'elaborazione dell'informazione.

È importante il modo in cui si rappresenta l'informazione. Diversi modi di rappresentare l'informazione possono agevolare oppure rallentare o rendere difficile l'elaborazione (esempio nel seguito, Run Lenght Code).

PROBLEMA informatico: Date informazioni in ingresso (DATI), produrre informazioni in uscita legate ai dati in ingresso da una certa legge (requisito) o relazione ben specificata.

Esempio: Somma di due numeri

INGRESSO (DATI): due numeri naturali

USCITA: un numero naturale

LEGGE (o relazione tra loro): il numero in uscita coincide con la somma dei due numeri in ingresso

ESECUTORE: un sistema (es. una macchina) in grado di eseguire un certo insieme (finito) di operazioni elementari.

ALGORITMO: Dato un problema (e un insieme di dati validi) e un esecutore, l'algoritmo è una combinazione di passi elementari, ben definiti ed eseguibili dall'esecutore, atta a risolvere il problema. Per qualunque insieme valido di dati di ingresso, l'algoritmo deve terminare e realizzare la corretta relazione tra dati in ingresso e informazione prodotta in uscita.

COMPUTAZIONE: l'esecuzione dei passi elementari dell'algoritmo

Passi elementari ben definiti = dall'effetto prevedibile: ovvero eseguendo il passo a partire da due situazioni identiche si ottiene lo stesso effetto.

VOI DOVE VI COLLOCATE IN QUESTO QUADRO?

Inventare algoritmi corretti (e possibilmente efficienti) per risolvere problemi importanti. Progettare esecutori adatti ad agevolare la computazione. Applicare algoritmi per risolvere problemi complessi.

ABILITA': ridurre un problema che sapreste risolvere (es. sommare, o ordinare alfabeticamente un insieme di nomi) in un algoritmo, ovvero progettare i passi per un esecutore limitato, ovvero scomporre il problema in operazioni elementari, facenti parte di un insieme ridotto. Entrare in "empatia" con l'esecutore. Questo processo mentale chiarisce e approfondisce la propria comprensione del problema.

ESEMPI DI ALGORITMI:

1. Problema: "Doppio"

-ingresso: un numero naturale

–uscita: un numero naturale.

-relazione: il numero in uscita è il doppio del numero in ingresso

Esecutore: sistema in grado di compiere le seguenti operazioni: Addizione di due numeri.

Algoritmo:

1. Addiziona(dato, dato);
2. Produci in uscita il risultato di questa operazione;
3. Stop

2. Problema: Triplo (dati, uscita, relazione)

Esecutore: Addizione di due numeri; memorizzazione di un numero in un "contenitore"

Algoritmo:

1. Addiziona(dato ,dato);
2. Memorizza il risultato in un contenitore;
3. Addiziona(dato, contenuto del contenitore);
4. Produci in uscita il risultato di questa operazione;
5. Stop

3. Problema: Massimo (dati, uscita, relazione: uno dei due numeri in ingresso, tale che non sia più piccolo dell'altro)

Esecutore: Confronto tra due numeri

Algoritmo:

1. Confronta i due numeri in ingresso;
2. Se il risultato del confronto è maggiore allora produci in uscita il primo dei due dati
   altrimenti
   produci in uscita il secondo dei due dati;
3. Stop

NOTA: in quest'algoritmo è stato usato un modo di combinare passi diverso dalla semplice sequenza: se... altrimenti… Tale modalità è chiamata DECISIONE o PASSO CONDIZIONALE

4: Problema: Massimo tra una sequenza di numeri. Ingresso: una sequenza di numeri in quantità imprecisata; Uscita: un numero; Relazione: il numero in uscita fa parte della sequenza di numeri in ingresso ma non è minore di alcuno di quelli.

Esecutore: Confronto tra due numeri, memorizzazione di un numero in un contenitore

Algoritmo:

1. Memorizza il primo numero della sequenza in ingresso in un contenitore;
2. Se ci sono ancora numeri nella sequenza esegui i seguenti passi:
   2.1 Confronta il numero memorizzato con il successivo numero della sequenza;
   2.2 se il nuovo numero è più grande di quello memorizzato, memorizza nel contenitore il nuovo numero (altrimenti non fare nulla);
   2.3 Vai al passo 2;
3. Produci in uscita il numero nel contenitore;
4. Stop

NOTA: quando memorizzo un nuovo numero nel contenitore, poiché il contenitore è lo stesso, perdo il vecchio numero memorizzato

NOTA: avete introdotto un nuovo modo di combinare passi elementari: L'ITERAZIONE o CICLO. Qui realizzato con la combinazione di Se… e Vai al passo 2. Per chiarezza di lettura in algoritmi complessi lo realizzeremo come:

1. Memorizza il primo numero della sequenza di ingresso nel contenitore;
2. Mentre ci sono nuovi numeri nella sequenza di ingresso esegui i seguenti passi:
   2.1 confronta il prossimo numero della sequenza in ingresso con il numero ne contenitore
   2.2 se questo numero è maggiore di quello nel contenitore allora memorizza il nuovo numero nel contenitore (altrimenti non fare nulla)
3. Produci in uscita il numero nel contenitore
4. Stop

La parte di algoritmo che va sotto il numero 2. è un CICLO.

Ausilio all'intuizione: che ruolo ha o di che proprietà gode il numero memorizzato nel contenitore? E' il massimo parziale, cioè il massimo tra i numeri finora analizzati. All'inizio coincide con il primo numero (il solo analizzato) alla fine coincide con il massimo tra tutti i numeri in ingresso.

TEOREMA di BOEHM-JACOPINI: Bastano questi tre modi di combinare passi elementari (sequenza, decisione, ciclo) per formulare qualsiasi algoritmo.

PROGRAMMA: combinazione di passi elementari eseguibili e COMPRENSIBILI da parte dell'esecutore "Somma" oppure "add" oppure "… + …" ? L'esecutore potrà effettivamente eseguire i passi specificati nell'algoritmo solo se saranno stati formulati in modo comprensibile per lui. In tal caso questi passi sono chiamate istruzioni.

L'insieme delle frasi comprensibili da parte di un esecutore è detto Linguaggio di Programmazione di quell'esecutore. Noi studieremo un linguaggio di programmazione chiamato C.

Esempio: Moltiplicazione per somme ripetute

Problema: Moltiplicazione tra due numeri naturali (Ingresso, Uscita, Relazione)

Esecutore; confronto tra due numeri, addizione di due numeri, sottrazione di due numeri, memorizzazione di un numero in un contenitore.

Algoritmo:

1. Memorizza il valore 0 in un primo contenitore
2. Memorizza il secondo dei due dati in ingresso in un secondo contenitore
3. Mentre il numero nel secondo contenitore è maggiore di zero (confronto) esegui i seguenti passi:
   3.1 somma il primo dei due numeri in ingresso con il contenuto del primo contenitore e memorizza il risultato nello stesso contenitore
   3.2 decrementa di uno il secondo contenitore (ovvero sottrai 1 al numero contenuto nel secondo contenitore e memorizza il risultato nello stesso contenitore)
4. Produci in uscita il valore contenuto nel primo contenitore
5. Stop

Ausilio all'intuizione: Cosa contiene il secondo contenitore (quello che viene decrementato a ogni iterazione)? Il numero di volte in cui devo sommare il primo numero in ingresso al primo contenitore. All'inizio vale quanto il moltiplicatore (numero di somme da effettuare) alla fine vale 0: non ci sono più somme da effettuare.

Esercizio. Adattate l'algoritmo per il seguente problema: elevamento a potenza di un numero. Ingresso due numeri naturali, uscita un numero naturale, relazione il numero in uscita coincide con il primo numero elevato al secondo. Esecutore: confronto, memorizzazione, moltiplicazione, differenza.

Altro esempio: Divisione intera per differenze ripetute

Problema: Ingresso: due numeri naturali; Uscita: due numeri naturali; Relazione: il primo numero in uscita è il quoziente della divisione tra il primo numero in ingresso e il secondo numero in ingresso; il secondo numero in uscita è il resto della divisione intera.

Esecutore: Addizione, sottrazione, memorizzazione, confronto.

Algoritmo:

1. Se il primo numero in ingresso è maggiore del secondo allora esegui i seguenti passi:
   1.1 memorizza il primo numero in ingresso in un primo contenitore
   1.2 memorizza il secondo numero in ingresso in un secondo contenitore,

   altrimenti esegui i seguenti passi:

   1.3 memorizza il secondo numero in ingresso nel primo contenitore
   1.4 memorizza il primo numero in ingresso nel secondo contenitore

2. Memorizza il valore 0 in un terzo contenitore
3. Mentre il valore nel primo contenitore è non minore del valore nel secondo contenitore esegui i seguenti passi
   3.1 sottrai dal valore del primo contenitore il valore del secondo contenitore e memorizza il risultato sempre nel primo contenitore
   3.2 incrementa di uno il terzo contenitore
4. Produci in uscita il valore nel terzo contenitore come quoziente
5. Produci in uscita il valore nel primo contenitore come resto
6. Stop

Osservazione: il quoziente della divisione intera è il massimo numero di volte in cui si può sottrarre il divisore dal dividendo senza che il risultato diventi negativo. Il resto della divisione è il risultato dell'ultima sottrazione effettuata: tale resto è positivo o nullo, e comunque più piccolo del divisore.

Altro esempio.

Problema: Massimo comun divisore tra due numeri naturali (ingresso, uscita, relazione).

Esecutore: (Addizione), Sottrazione, (Moltiplicazione), Divisione intera (quoziente e resto).

Proprietà: Se un numero è comun divisore di due numeri M e N, allora è divisore anche della differenza tra M e N, e anche della differenza tra M e 2 volte N … e anche della differenza tra M e il più grande multiplo di N tale che la differenza sia ancora positiva. Che cosa è questo valore? E' il resto della divisione intera!

Algoritmo di **Euclide**:

1. Se il primo numero in ingresso è maggiore del secondo allora esegui i seguenti passi:
   1.1 memorizza il primo numero in ingresso in un primo contenitore
   1.2 memorizza il secondo numero in ingresso in un secondo contenitore,

   altrimenti esegui i seguenti passi:

   1.3 memorizza il secondo numero in ingresso nel primo contenitore
   1.4 memorizza il primo numero in ingresso nel secondo contenitore
2. Mentre il resto della divisione intera tra il valore nel primo contenitore e il valore del secondo contenitore è diverso da 0 esegui i seguenti passi:
   2.1 memorizza tale resto in un terzo contenitore
   2.2 memorizza il valore del secondo contenitore nel primo contenitore
   2.3 memorizza il valore del terzo contenitore nel secondo contenitore
3. Produci in uscita il valore del secondo contenitore
4. Stop

Notazione più compatta:

Algoritmo di Euclide: M e N numeri interi

1. Se M > N
   1.1 A ← M
   1.2 B ← N

   Altrimenti

   1.3 A ← N
   1.4 B ← M
2. Mentre (A % B) diverso da 0
   2.1 R ← A % B
   2.2 A ← B
   2.3 B ← R
3. Produci B
4. Stop

Domanda: non avrei potuto semplicemente dire per il passo 3: memorizza il resto della divisione intera e il valore di B rispettivamente in B e in A?

NO: poiché se effettuo prima la memorizzazione del resto in B, il valore di questo contenitore cambierebbe alterando il valore da memorizzare in A. Se anche invertissi l'ordine delle memorizzazioni altererei il divisore B della divisione intera prima di effettuarla.

E allora non posso effettuare entrambe le memorizzazioni in un passo solo?

NO: l'esecutore, in un passo, può memorizzare UN numero in UN contenitore. Non due o più… questo riflette la realtà degli esecutori che troviamo in pratica.

Pertanto, per lo scambio di valori tra due contenitori A e B occorre il supporto di un contenitore supplementare C:

1. Memorizza il valore del primo contenitore in un terzo contenitore: C ← A
2. Memorizza il valore del secondo contenitore nel primo contenitore: A ← B
3. Memorizza il valore del terzo contenitore nel secondo contenitore: B ← C

Altro esempio: Ricerca in sequenza ordinata

Problema: Ingresso: una sequenza ordinata di numeri, un numero da cercare; Uscita: una posizione nella sequenza; Relazione: la posizione in uscita coincide con la posizione del numero cercato nella sequenza (se presente). In caso di assenza del numero dalla sequenza, la posizione è negativa.

Esecutore: accesso diretto qualunque posizione nella sequenza, confronto tra due numeri, addizione, sottrazione, divisione (per due)

Algoritmo:

1. Inizio ← prima posizione nella sequenza
2. Fine ← ultima posizione nella sequenza
3. Mentre (Inizio <= Fine) esegui i seguenti passi:
   3.1 Media = (Inizio + Fine) / 2
   3.2 Se (Numero in posizione Media == Numero da cercare) Produci posizione Media e Stop
   altrimenti esegui i seguenti passi
   3.2.1 Se (Numero da cercare < Numero in posizione Media) allora Fine ← Media -1
   Altrimenti Inizio ← Media +1
4. Produci posizione -1 (numero non presente nella sequenza)
5. Stop

RAPPRESENTAZIONE DELL'INFORMAZIONE

Consideriamo un segnale digitale binario. Segnale: una sequenza di valori. Digitale, una sequenza finita di valori ad esempio interi. Binario: i valori assumibili da ciascun valore della sequenza sono solamente due: 1 e 0.

Supponiamo che il segnale sia lungo 1000, cioè contenga 1000 valori, e che vogliamo calcolare il numero di "1" presente nel segnale.

Se il segnale è rappresentato come una sequenza di mille valori, un algoritmo per contare il numero di "1" è ad esempio:

1. Memorizza il valore 0 in un contenitore
2. Mentre ci sono ancora nuovi valori nella sequenza esegui il passo seguente:
   2.1 Se il nuovo valore è pari a 1 incrementa il contenitore
3. Produci il valore del contenitore in uscita
4. Stop

Per contare gli "1" occorrerebbe eseguire il ciclo presente nell'algoritmo 1000 volte.

Se invece si utilizzasse una diversa rappresentazione dello stesso segnale si potrebbe rendere la computazione dell'algoritmo più veloce. Supponiamo di rappresentare il segnale tramite una codifica Run Length Code (codifica a lunghezza di corsa). Il segnale verrebbe rappresentato da una sequenza di interi (non più binari) in cui ogni intero indica il numero di valori uguali consecutivi nella sequenza del segnale prima che il valore cambi. Ad esempio 0001011110001 verrebbe rappresentata come 0311431. Il primo zero è dovuto al fatto che per convenzione si comincia con il numero di "1" consecutivi, poi il numero di "0" consecutivi, poi di nuovo il numero di "1" consecutivi e così via, fino a che la somma dei valori non raggiunge la lunghezza del segnale (1000).
Usando questa rappresentazione, per contare gli "1" basterebbe sommare tra loro i 0, 1, 4, 1 cioè i valori di posto dispari nella sequenza.

Esercizio: scrivere l'algoritmo di conteggio a partire dalla codifica Run Length Code; scrivere l'algoritmo che traduce la codifica per esteso di un segnale nella sua codifica RLC, e scrivere l'algoritmo "inverso", cioè quello che produce -a partire dalla codifica RLC di un segnale- la sua codifica per esteso.

ConteggioDaRLC

1. Memorizza il valore 0 in un contenitore "contatore"
2. Memorizza il valore 1 in un contenitore "di stato"
3. Mentre ci sono nuovi valori nella sequenza RLC esegui i seguenti passi
   3.1 se il contenitore di stato è uguale a 1 aggiungi al contatore il nuovo valore nella sequenza
   3.2 memorizza nel contenitore di stato il risultato (1 - valore del contenitore di stato)
4. Produci in uscita il valore del contatore
5. Stop

ConteggioDaRLC (notazione compatta)

1. C ← 0
2. S ← 1
3. Mentre ci sono nuovi valori nella sequenza RLC esegui i seguenti passi
   3.1 se S è uguale a 1 allora C ← C + nuovo valore nella sequenza
   3.2 S ← 1 - S
4. Produci in uscita C
5. Stop

Esteso2RLC

1. Memorizza 1 nel contenitore di stato
2. Memorizza 0 nel contatore
3. Mentre ci sono ancora nuovi valori nella sequenza del segnale esegui i seguenti passi:
   3.1 se il nuovo valore del segnale è uguale al valore del contenitore di stato incrementa il contatore
   altrimenti esegui i seguenti passi:

3.2 produci in uscita il valore del contatore
3.3 memorizza il valore 1 nel contatore
3.4 memorizza il valore (1 – contenitore di stato) nel contenitore di stato

4. Produci in uscita il valore del contatore
5. Stop

Esteso2RLC (notazione compatta)

4. S ← 1
5. C ← 0
6. Mentre ci sono ancora nuovi valori nella sequenza del segnale esegui i seguenti passi:
   5.1 se il nuovo valore del segnale è uguale a S allora C ← C + 1
   altrimenti esegui i seguenti passi:
   5.2 produci in uscita C
   5.3 C ← 1
   5.4 S ← 1 – S
6. Produci in uscita C
7. Stop

RLC2Esteso

1. Memorizza 1 nel contenitore di stato
2. Mentre vi sono nuovi valori nella sequenza RLC esegui i seguenti passi:
   2.1 Memorizza il nuovo valore in un contatore
   2.2 Mentre il contatore è maggiore di 0 esegui i seguenti passi
      2.2.1 Produci in uscita il contenitore di stato
      2.2.2 Decrementa di 1 il contatore

   Memorizza nel valore di stato il valore di (1- valore nel contenitore di stato)

3. Stop

RLC2Esteso (notazione compatta)

1. S ← 1
2. Mentre vi sono nuovi valori nella sequenza RLC esegui i seguenti passi:
   2.1 C ← nuovo valore della RLC
   2.2 Mentre C > 0 esegui i seguenti passi
      2.2.1 Produci in uscita S
      2.2.2 C ← C - 1

   S ← 1 - S

3. Stop