ROMA TRE
UNIVERSITÀ DEGLI STUDI

Dipartimento di Ingegneria
Industriale, Elettronica e Meccanica

Collegio Didattico di Ingegneria Elettronica

Elementi di Informatica

Problemi e algoritmi – Parte I

Giordano Da Lozzo e Giuseppe Sansonetti

L'informatica permette di risolvere problemi

hai un problema
che vuoi risolvere

trovi una procedura che ti
permette di risolvere il problema
(su ogni possibile insieme di dati)

algoritmo

trasformi la procedura in una
sequenza di istruzioni che possono
essere eseguite dal calcolatore

programma

se vuoi risolvere
il problema su un certo
insieme di dati

istanza

rappresenti i dati nel calcolatore e fai eseguire
al calcolatore la sequenza di istruzioni che
permettono di risolvere il problema

esecuzione del programma

Algoritmo

- Una sequenza di istruzioni la cui applicazione permette di risolvere un problema.

- Una procedura computazionale che prende alcuni valori (input) e ne restituisce altri (output).

Progettazione ed esecuzione

- L'esecuzione di un algoritmo e un'attività che richiede solo di saper svolgere un certo tipo di operazioni. E' così semplice che persino un calcolatore la sa svolgere!

- La progettazione di un algoritmo è un'attività creativa, per la quale non esiste una procedura precisa!

- In questo corso vedremo metodologie per la progettazione di algoritmi che sono utili per la risoluzione di "molti" problemi.

Problemi

Problemi di interesse sono quelli per i quali è possibile una formalizzazione matematica.

- trovare il massimo comune divisore fra due interi

  find $z = \max \{w \in N \mid x\%w = 0 \wedge y\%w = 0\}$

- calcolare il massimo in un insieme di interi
- disporre una sequenza di interi in ordine crescente
- …

Non manca qualcosa?

- trovare il massimo comune divisore fra due interi

  quali due interi?

  find $z = \max \{w \in N \mid x\%w = 0 \wedge y\%w = 0\}$

  quale insieme di interi?

- calcolare il massimo in un insieme di interi
- disporre una sequenza di interi in ordine crescente

  quale sequenza di interi?

- …

Problemi parametrici

Problemi in cui alcuni parametri (ad es. il valore di due interi, il valore degli elementi in un insieme di interi, ...) sono incogniti.

Un algoritmo che risolve un problema parametrico, lo deve risolvere per qualsiasi valore dei parametri,

- ovvero per ogni insieme di dati iniziali o dati di ingresso,
- ovvero per ogni possibile input,
- ovvero deve risolvere ogni possibile istanza del problema.

Dati di ingresso e pre-condizioni

- Trovare il massimo comun divisore fra due interi x e y.

Un algoritmo deve funzionare se i dati in ingresso sono x=5 e y= ‘’giacomo’’ ?

No! Un algoritmo per risolvere un problema deve funzionare solo se i dati di ingresso soddisfano certe proprietà.

- L'insieme di queste proprietà prende il nome di pre-condizione dell'algoritmo.

Esempi

- Trovare il massimo comune divisore fra due interi
  - dati di ingresso: due interi X e Y
  - pre-condizione: X e Y sono interi maggiori di zero

- Calcolare il massimo in un insieme di interi
  - dati di ingresso: un insieme $\{x_1, \ldots, x_n\}$ di interi
  - pre-condizione: $\{x_1, \ldots, x_n\}$ sono interi e $n \geq 1$

Dati di uscita e post-condizioni

Un algoritmo, molto spesso, deve fornire un output, ovvero dei dati di uscita, che rappresentano la soluzione dell'istanza considerata del problema.

- Trovare il massimo comun divisore fra due interi
  - dati di ingresso: due interi X e Y
  - pre-condizione: X e Y sono interi maggiori di zero

- Può l'algoritmo fornire come dati di uscita Z=3 se X=4 e Y=6?

- NO! Un algoritmo per risolvere un problema a cui sono stati dati dei dati di ingresso che soddisfano le pre-condizioni deve fornire dei dati di uscita che soddisfano delle proprietà, dette post-condizioni.

Esempi

- Trovare il massimo comune divisore fra due interi
  - dati di ingresso: due interi X e Y
  - pre-condizione: X e Y sono interi maggiori di zero
  - dati di uscita: un intero Z
  - post-condizione: Z è un intero uguale al m.c.d. di X e Y

- Calcolare il massimo in un insieme di interi
  - dati di ingresso: un insieme $\{x_1, \dots, x_n\}$ di interi
  - pre-condizione: $x_1, \dots, x_n$ sono interi e $n \geq 1$.
  - dati di uscita: un intero Z
  - post-condizione: Z è pari al massimo fra $\{x_1, \dots, x_n\}$

Specifica di un problema di ingresso-uscita

Descrizione formale di un problema parametrico in termini di:

- dati di ingresso
- pre-condizioni
- dati di uscita
- post-condizioni

Descrivere la specifica di un problema è un passo utile per comprendere il problema e poter quindi procedere alla progettazione di un algoritmo per risolvere il problema.

Verso la progettazione

Quali sono le caratteristiche che sono richieste al mio algoritmo ? Quali proprietà deve soddisfare ?

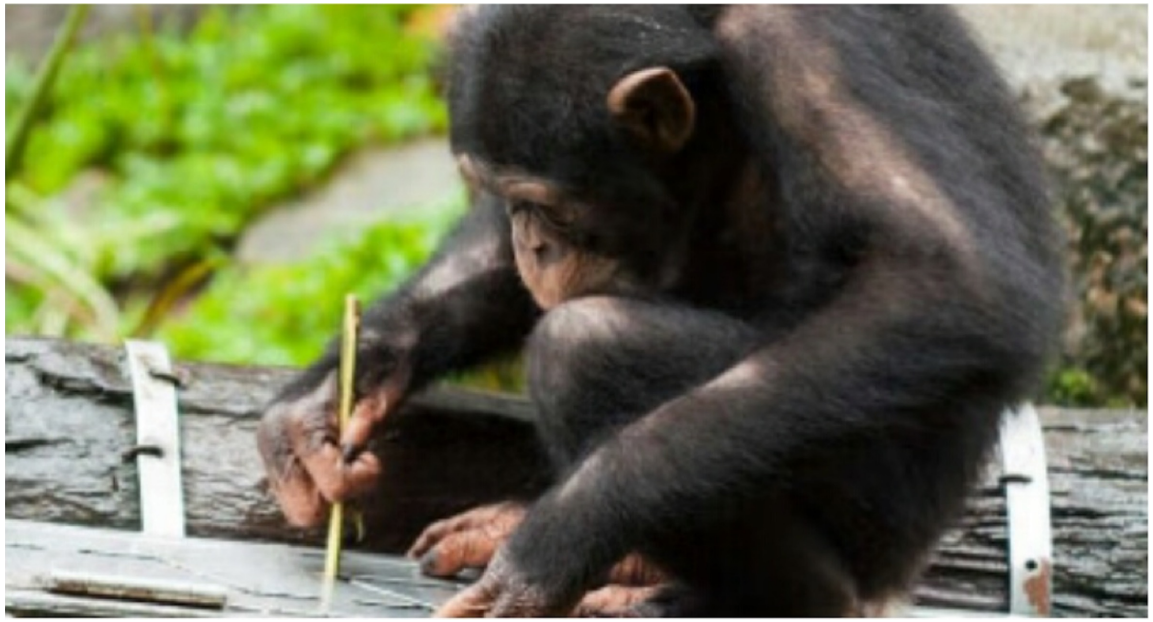
Problema: MCD

- Trovare il massimo comune divisore fra due interi
  - dati di ingresso: due interi X e Y
  - pre-condizione: X e Y sono interi maggiori di zero
  - dati di uscita: un intero Z
  - post- condizione: Z è un interno uguale al m.c.d. di X e Y

Un algoritmo

1. input (x,y)
2. output (x-y-2)

Q: quale naturale proprietà di un algoritmo non soddisfa?



Un algoritmo

1. input (x,y)
2. output (x-y-2)

Q: quale naturale proprietà di un algoritmo non soddisfa?

L'algoritmo non è corretto, ovvero esistono istanze del problema per le quali l'algoritmo fornisce una risposta diversa da quella attesa.

Correttezza: per ogni insieme di dati di ingresso che soddisfano le pre-condizioni, l'algoritmo fornisce un insieme di dati di uscita che soddisfano le post-condizioni.

Osserva: l'algoritmo qui sopra si comporta correttamente se $x = 10$ e $y = 6$, ma non si comporta correttamente per ogni insieme di dati di ingresso che soddisfano le pre-condizioni.

Un altro algoritmo

1. input (x,y)
2. inizializza risultato = 1 e contatore = 2
3. se contatore divide x e contatore divide y,
   allora aggiorna risultato = contatore
4. incrementa contatore di 1 e torna all'istruzione 3.
5. output (risultato)

Q: quale naturale proprietà di un algoritmo non soddisfa?

Un altro algoritmo

1. input (x,y)
2. inizializza risultato = 1 e contatore = 2
3. se contatore divide x e contatore divide y,
   allora aggiorna risultato = contatore
4. incrementa contatore di 1 e torna all'istruzione 2.
5. output (risultato)

Q: quale naturale proprietà di un algoritmo non soddisfa?

L'algoritmo non è finito, ovvero la sua esecuzione non termina mai.

Finitezza: per ogni insieme di dati di ingresso che soddisfano le pre-condizioni, l'algoritmo deve eseguire un numero finito di operazioni e terminare.

Altre proprietà di un algoritmo

- Non ambiguità: ciascuna istruzione che costituisce l'algoritmo deve poter essere univocamente interpretata.

- Eseguibilità: ogni istruzione che costituisce l'algoritmo deve poter essere eseguita in un tempo finito e con una memoria finita.

Proprietà di un algoritmo

- Correttezza
- Finitezza
- Non ambiguità
- Eseguibilità
- …
- Efficienza

Efficienza

Un algoritmo (un programma) è tanto più efficiente quanto minori sono le risorse di calcolo di cui necessita per essere eseguito.

Risorse:

- Tempo

- Memoria

- …

Tempo di esecuzione

Da cosa dipende il tempo di esecuzione di un programma?

- Dalla velocità del calcolatore.
- Dalla dimensione dei dati di input.
- Dal numero di operazioni che il programma deve eseguire.

Efficienza temporale di un algoritmo

Si valutano il numero di operazioni elementari svolte dall' algoritmo come funzione della dimensione dei dati di ingresso.

Operazioni elementari: operazioni aritmetiche o confronti che possono essere eseguite in tempo costante (ad esempio $1\mu s$).

Dimensione dei dati: un intero n.

Ad esempio per il problema della ricerca del massimo in un insieme $\{x_1, \dots, x_n\}$ la dimensione n è il numero di interi.

Osservazione: il numero di operazioni eseguite dall' algoritmo potrebbe cambiare per diversi dati di ingresso con dimensione n ⇒

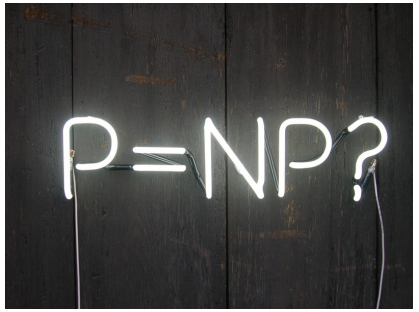
efficienza nel caso peggiore, nel caso medio, nel caso migliore...

Algoritmi efficienti per tutti i problemi?

- Esistono problemi indecidibili - ovvero per i quali non esiste un algoritmo risolutivo. Ad es. il problema della fermata [Turing ’36]

- Esistono problemi intrattabili - problemi decidibili per i quali ogni algoritmo (deterministico) risolutivo richiede una quantità di risorse insostenibile in pratica (esponenziale) [Hartamanis-Stearns ‘65]

- Per una moltitudine di problemi comuni, non sono noti algoritmi risolutivi con un numero polinomiale di istruzioni ed è credenza comune che tali algoritmi non esistano.



Problema: MCD

- Trovare il massimo comune divisore fra due interi
  - dati di ingresso: due interi X e Y
  - pre-condizione: X e Y sono interi maggiori di zero
  - dati di uscita: un intero Z
  - post- condizione: Z è un interno uguale al m.c.d. di X e Y

- Q: un algoritmo per questo problema?

Algoritmo per MCD

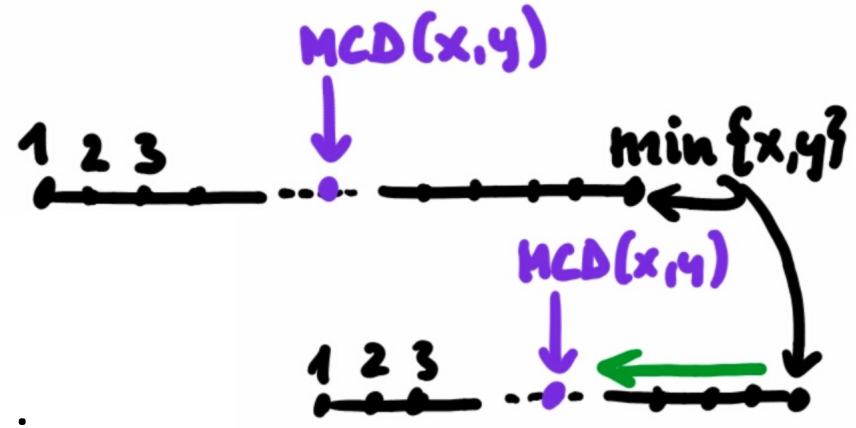
1. input (x, y)
2. inizializza risultato = min $\{x, y\}$
3. se risultato divide x e risultato divide y,
   allora vai all'istruzione 5
4. decrementa risultato di 1 e
   torna all'istruzione 3
5. output (risultato)

Q: E' corretto? E' finito?

Algoritmo per MCD

1. input (x, y)
2. inizializza risultato = min $\{x, y\}$
3. se risultato divide x e risultato divide y,
   allora vai all'istruzione 5
4. decrementa risultato di 1 e
   torna all'istruzione 3
5. output (risultato)

Q: E' corretto? E' finito?



$\Rightarrow 1 \leq mcd(x, y) \leq min\{x, y\}$, poiché $x, y > 0$

$\Rightarrow$ nel range $[1, \min\{x, y\}]$ un numero più grande viene sempre verificato prima di uno più piccolo

Esecuzione

Flusso di esecuzione

Sequenza di istruzioni che vengono eseguite quando un algoritmo viene applicato per risolvere una istanza del problema.

Il flusso di esecuzione consiste di una sequenza di passi (un passo è l'esecuzione di una singola istruzione)

- Una stessa istruzione può essere eseguita più volte.

Traccia del processo di esecuzione

Tabella che mostra i valori assunti dalle variabili durante l'esecuzione.

Esempio di esecuzione

Dati di ingresso: $x = 6; y = 4$ Pre-condizione: $x > 0;\ y > 0$ OK!

x	y	risultato
6	4	-
6	4	4
6	4	4
6	4	3
6	4	3
6	4	2
6	4	2

P1: inizializza risultato = min $\{x, y\}$

P2: risultato non divide $x \Rightarrow$ vai all'istruzione 4

P3: decrementa risultato di 1

P4: risultato non divide $y \Rightarrow$ vai all'istruzione 4

P5: decrementa risultato di 1

P6: risultato divide x e divide $y \Rightarrow$ vai all'istruzione 5

P7: fornisci risultato come soluzione

Dati di uscita: l'intero 2. Post-condizione: 2 è il MCD di 6 e 4 OK!

Non doveva essere una sequenza di istruzioni?

1. input (x, y)
2. inizializza risultato = min $\{x, y\}$
3. se risultato divide x e risultato divide y,
   allora vai all'istruzione 5
4. decrementa risultato di 1 e
   torna all'istruzione 3
5. output (risultato)

È una sequenza di istruzioni, ma l'ordine in cui queste istruzioni sono eseguite può essere controllato, così che la stessa istruzione venga eseguita più volte, o venga eseguita solo se una condizione è soddisfatta.

Istruzioni di controllo

Istruzioni che permettono di controllare (nel senso di regolare, governare, dirigere) il flusso di esecuzione, ovvero quali istruzioni devono essere eseguite e in quale ordine.

- Normalmente le istruzioni che compongono un algoritmo vengono eseguite una dopo l'altra, nell'ordine in cui compaiono.

- Le istruzioni di controllo permettono di modificare l'ordine in cui le istruzioni di un algoritmo vengono eseguite.

Come si realizza il controllo?

1. input (x, y)
2. inizializza risultato = min $\{x, y\}$
3. se risultato divide x e risultato divide y,
   allora vai all'istruzione 5
4. decrementa risultato di 1 e
   torna all'istruzione 3
5. output (risultato)

In questo algoritmo tramite dei salti condizionati (se è vera una certa condizione, allora vai all'istruzione TOT) e tramite dei salti incondizionati (vai all'istruzione TOT).

E’ meglio non saltare

I programmi che utilizzano istruzioni di salto sono contorti, difficili da correggere, manutenere ed estendere — principalmente se il programma deve essere modificato da un programmatore diverso da quello che ha scritto il programma stesso.

Programmazione strutturata

Un programma si dice strutturato se utilizza solo le seguenti istruzioni di controllo:

blocco - permette di eseguire una sequenza di istruzioni.

istruzione condizionale - permette di determinare quale operazione deve essere eseguita in base al soddisfacimento di una condizione (istruzione di controllo decisionale o di selezione).

istruzione ripetitiva - permette di eseguire ripetutamente un'istruzione fintanto che è soddisfatta una condizione (istruzione di controllo iterativo).

Sono sufficienti?

- Teorema [Böhm-Jacopini 1966] qualunque algoritmo può essere realizzato (implementato) da un programma strutturato.

Struttura l'algoritmo!

1. input (x, y)
2. inizializza risultato = min $\{x, y\}$
3. se risultato divide x e risultato divide y,
   allora vai all'istruzione 5
4. decrementa risultato di 1 e
   torna all'istruzione 3
5. output (risultato)

Q: come si scrive una "versione strutturata" di questo algoritmo?

Struttura l'algoritmo!

1. input (x, y)
2. inizializza risultato = min $\{x, y\}$
3. fintanto che (risultato non divide x o risultato non divide y)
   decrementa risultato di 1
4. output (risultato)

Sono utili?

1. input (x, y)
2. inizializza risultato = min $\{x, y\}$
3. fintanto che (risultato non divide x o risultato non divide y)
   decrementa risultato di 1
4. output (risultato)

Q: perché non scrivere direttamente una sequenza di istruzioni, nell'ordine in cui andranno eseguite?

Sono utili?

1. input (x, y)
2. inizializza risultato = min $\{x, y\}$
3. fintanto che (risultato non divide x o risultato non divide y)
   decrementa risultato di 1
4. output (risultato)

Q: perché non scrivere direttamente una sequenza di istruzioni, nell'ordine in cui andranno eseguite?

⇒ perché non sappiamo quante volte (→istruzione ripetitiva) o se (→istruzione condizionale) una condizione si verificherà a tempo di esecuzione.

- Le istruzioni di controllo permettono di regolare il flusso di esecuzione anche se tali informazioni non sono note.

Come si scrive un algoritmo?

- pseudo codice
- diagramma a blocchi

To be continued …