Problemi e soluzioni

- Ogni problema di *elaborazione di informazione* è caratterizzato da :
  - un insieme di dati di partenza,
  - da un risultato cercato.
- Ogni soluzione al problema dato è una procedura che genera un risultato sulla base dei dati indicati.
- A-priori

  il *soggetto* che trova la soluzione ad un problema e

  il *soggetto* che effettivamente esegue la soluzione

  possono essere diversi

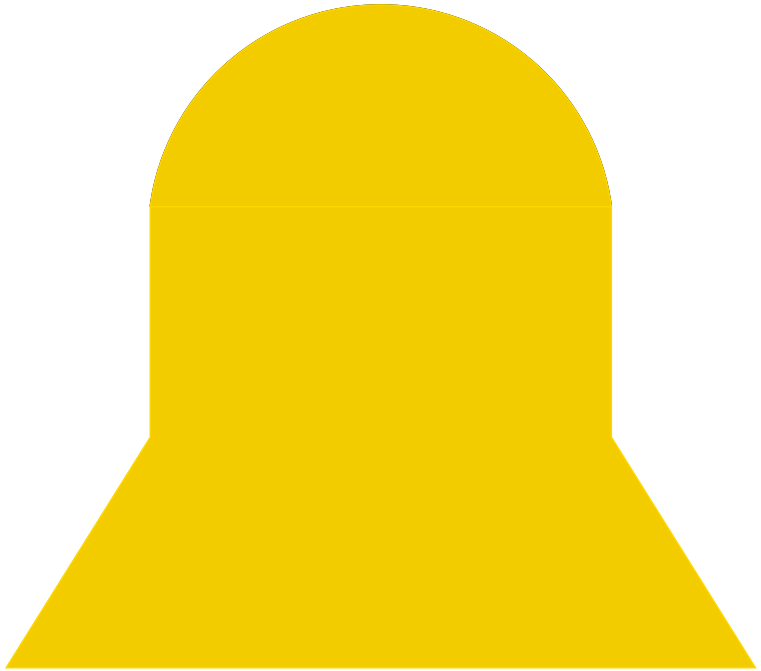
- **Fasi del procedimento di soluzione del problema**:
  1. **analisi** ed **identificazione di una soluzione**, da parte del primo soggetto;
  2. **descrizione della soluzione**, da parte del primo soggetto, mediante la *specifica di azioni effettivamente eseguibili* da parte del secondo soggetto;
  3. **interpretazione corretta della soluzione**, fornita dal primo soggetto, da parte del secondo soggetto;
  4. **attuazione della soluzione**, da parte del secondo soggetto mediante *azioni* che questi è *effettivamente* in grado di eseguire
- Un **calcolatore** è essenzialmente un esecutore di soluzioni identificate e descritte preventivamente da esseri umani, caratterizzato:
  - dal **linguaggio** in grado di interpretare, e
  - dalle **istruzioni** in grado di eseguire.

Procedura di soluzione di un problema

- **Scomposizione di un problema** in:
  - sotto-problemi,
  - sotto-sotto-problemi,
  - ... e così via,
  - fino a giungere a **problemi elementari** (o primitivi), la soluzione di ognuno dei quali corrisponde ad una **azione elementare**, eseguibile immediatamente dall'esecutore
- Un insieme di problemi è una **procedura effettiva** quando:
  - tutti i problemi dell'insieme sono elementari;
  - è fissato l'*ordine* di esecuzione dei problemi;
  - è specificato il *modo* in cui un problema utilizza i risultati dei problemi che lo precedono

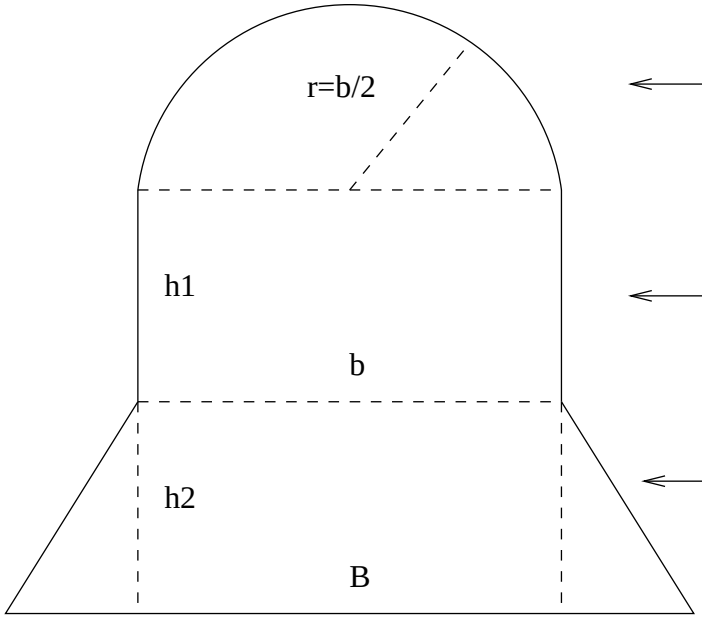
il **problema elementare** corrisponde all'*aspetto descrittivo*
l'**azione elementare** corrisponde all'*aspetto esecutivo*

Problema : calcolo della superficie di una figura complessa



PROBLEMA COMPLESSO

Soluzione : calcolo della superficie di una figura complessa



sottoproblema 1

$S1 = 1/2\ \pi\ r^2$

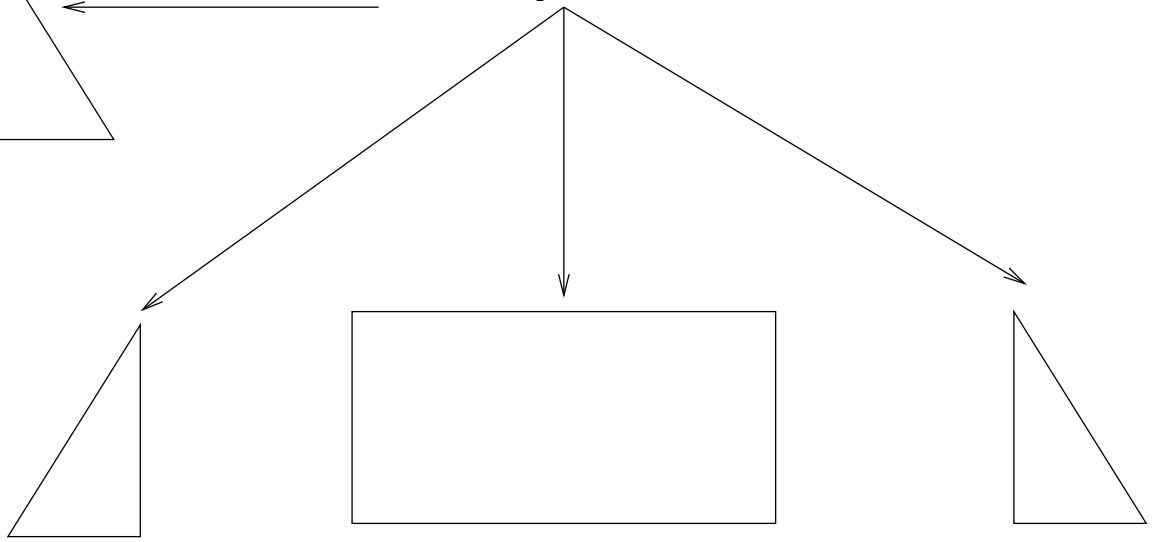
sottoproblema 2

S2= b h1

sottoproblema 3

Problema Complesso: S ?

S= S1+S2+S31+S32+S33



sotto−problema 3.1

$S31 = \frac{(1/2(B-b))\ h2}{2}$

sotto−problema 3.2

S32 = B h2

sotto−problema 3.3

$S33 = \frac{(1/2(B-b))\ h2}{2}$

Problemi elementari

Problemi, Algoritmi, Programmi

- **Caratterizzazione dell'esecutore**:
  1. **sintassi**: il linguaggio che l'esecutore è in grado di interpretare deve essere definito in termini formali;
  2. **determinismo**: l'insieme delle azioni che l'esecutore è in grado di compiere deve essere univocamente definito, e tali azioni devono essere deterministiche (l'esecuzione di una stessa azione deve produrre sempre lo stesso risultato);
  3. **semantica**: l'insieme delle *regole* di associazione tra *costrutti del linguaggio* e *azioni* deve essere univocamente definito.
- Le soluzioni effettive per esecutori, caratterizzati secondo i punti 1-3, sono chiamate

Algoritmi

- Se l'esecutore è un calcolatore, allora l'algoritmo viene detto

Programma

- Il linguaggio formale usato usato per la specifica dei costrutti sintattici di un programma, viene detto:

Linguaggio di programmazione

Algoritmi

- Un algoritmo è una descrizione della soluzione di un problema espressa in modo da poter essere interpretata ed eseguita da un esecutore.
- Un algoritmo consente di ottenere dei risultati (la soluzione del problema) mediante un un insieme finito di regole che operano sui dati iniziali:
  - le regole sono determinate tramite la scomposizione iterativa del problema di partenza in sottoproblemi elementari,
  - la soluzione di ogni (sotto)problema elementare è detta passo dell'algoritmo.

Algoritmi

- Un algoritmo descrive, non la soluzione per un singolo problema, ma per una

  classe di problemi equivalenti

- Ad esempio l'algoritmo per la *moltiplicazione di due numeri naturali* spiega all'esecutore come effettuare il prodotto di tutte le coppie di numeri naturali

$$\forall n\, \forall m\, (n \in \mathbb{N} \wedge m \in \mathbb{N}) \rightarrow (n \times m) \in \mathbb{N})$$

- Le operazioni espresse nell'algoritmo fanno riferimento non direttamente a valori utilizzabili dall'esecutore ma a variabili il cui valore non è fissato a-priori ma può cambiare di volta in volta
- Le variabili possono essere intese come contenitori per i dati;
- Ogni variabile ha :
  - un **nome**, che la identifica, ed
  - un **valore**, che in ogni istante corrisponde al dato contenuto nella variabile.
- I nomi di variabli:
  - possono comparire all'intero di espressioni, ad esempio

$$op_1 + op_1 \times op_3$$

  - possono essere usati in operazioni di assegnamento, ad esempio

$$x ::= 7 \qquad z \leftarrow true \qquad \texttt{let pi = 3.14}$$

Problema 1 : determinare il maggiore di due numeri interi, x, y

Assumiamo i seguenti problemi elementari (la loro soluzione è immediata per l'esecutore)

- **differenza** tra due numeri;
- valutazione del **segno** di un numero, positivo o negativo: $x > 0, x < 0$

Soluzione : x è maggiore di y se $(x - y) > 0$

ALGORITMO 1

```
passo 1 : LEGGI un valore dall'esterno e inseriscilo nella variabile x ;
passo 2 : LEGGI un secondo valore dall'esterno e inseriscilo nella variabile y ;
passo 3 : CALCOLA la differenza (x - y) e inseriscila nella variabile d;
passo 4 : SE d e' maggiore di 0 ALLORA  esegui il passo 5,
          ALTRIMENTI (d<0) esegui il passo 6;
passo 5 : STAMPA la frase `` il massimo e' '' seguita dal valore contenuto in x ;
passo 6 : STAMPA la frase `` il massimo e' '' seguita dal valore contenuto in y ;
passo 7 : TERMINA l'esecuzione.
```

- Al `passo 4` è presente una condizione logica (a valori booleani, `0,1`).
- A seconda del valore di verità della condizione si opera una scelta tra due azioni esclusive:

`IF` condizione `THEN` azione-1 `ELSE` azione-2

si tratta di **strutture condizionali**, molto usate negli algoritmi.

Problemi terminali

- Allo scopo di semplificare un algoritmo e ridurne il numero di passi si può arrestare la scomposizione a **sottoproblemi non elementari**, purchè di questi sia comunque nota una scomposizione in problemi elementari.
- Chiameremo tali sotto-problemi risolubili, elementari o meno, **problemi terminali**.
- L'esecuzione di un algoritmo prevede la soluzione di un insieme di problemi terminali:
  - *ogni* **problema terminale elementare** *viene risolto direttamente, eseguendo la corrispondente azione elementare*,
  - *ogni* **problema terminale non elementare** *richiede che sia dapprima resa esplicita una sua scomposizione in sotto-problemi elementari.*
- Nei **linguaggi di programmazione**:
  - ai problemi terminali elementari corrispondono le **istruzioni del linguaggio**;
  - ai problemi terminali non elementari corrispondono i **sotto-programmi**.

Problema 2

determinare il maggiore di tre numeri x, y, z

Soluzione

riduciamo il problema a quello terminale (non elementare)
di trovare il maggiore di due numeri (problema 1)

Algoritmo 2

```
passo 1: LEGGI x, y, z;
passo 2: IF x e' maggiore di y THEN esegui il passo 3     (* problem 1 *)
                              ELSE esegui il passo 4;
passo 3: la soluzione corrisponde al maggiore tra x e z ; (* problem 1 *)
passo 4: la soluzione corrisponde al maggiore tra y e z ; (* problem 1 *)
```

Problema 3

determinare il maggiore di n numeri interi

Soluzione

si arriva alla soluzione risolvendo successivamente i seguenti problemi

Algoritmo 3

```
passo 1   : trova il maggiore tra i primi due numeri        (* problem 1 *);
passo 2   : trova il maggiore tra il terzo numero e
            il risultato del passo precedente               (* problem 1 *);
passo 3   : trova il maggiore tra il quarto numero e
            il risultato del passo precedente               (* problem 1 *);
 ...
passo n-1: trova il maggiore tra l'ultimo numero e
            il risultato del passo precedente               (* problem 1 *);
passo n   : restituisci la soluzione che corrisponde
            al risultato del passo precedente.
```

...è possibile scrivere l'algoritmo in una forma "più compatta" ?

Soluzione "compatta" del Problema 3
(determinare il maggiore di tre numeri n numeri interi)

Algoritmo 4

```
passo 1: trova il maggiore fra i primi due numeri;
passo 2: FINCHE' ci sono numeri da esaminare, RIPETI il passo 3,
         altrimenti vai al passo 4;
passo 3: trova il maggiore tra il nuovo numero da esamniare
         e il piu' grande trovato in precedenza;
passo 4: presenta all'utente la soluzione che corrisponde
         al risultato dell'ultima esecuzione del passo 3.
```

- Il `passo 3` mostra una struttura usata nella descrizione di problemi ripetitivi e detta struttura iterativa o ciclo:

  `while` *condizione* `do` *azione* (finchè ... ripeti ...)

- In generale, un insieme di problemi che soddisfa tale struttura si dice soluzione iterativa.
- Ogni problema di tale insieme è detto iterazione elementare o passo dell'iterazione
- Esistono diverse strutture iterative/cicliche per controllare il flusso dell'elaborazione:

  `FOR...TO...DO...; REPEAT...UNTIL...`, ecc.

Algoritmi e tipi di dati

Problema – Trovare una algoritmo che implementi la **congiunzione logica,** $\wedge$:

date due *proposizioni logiche* (valori booleani: `TRUE, FALSE`) restituisce il valore booleano `TRUE` se entrambe le proposizoni hanno valore `TRUE`, altrimenti `FALSE`

Algoritmo 5

```
passo 1: LEGGI i valori booleani di A e B;
passo 2: SE il valore di A e' TRUE ALLORA restituisci il valore di B
         altrimenti restituisci il valore FALSE
```

Esercizio – trovare un algoritmo che implementi la **disgiunzione classica $\vee$ (vel)**

Esercizio – trovare un algoritmo che implementi **la negazione classica $\neg$**.

OSSERVAZIONE

L'istruzione `IF...THEN...ELSE` basta, da sola, ad implementare tutti i connettivi base.

Algoritmi e strutture dati

- Per uno stesso problema si possono avere soluzioni (algoritmi) differenti:
  alcune soluzioni possono essere più efficienti/efficace di altre.
- La soluzione dipende dalla "struttura" nella quale si presentano i dati.
- **Problema del bibliotecario:** inserire la scheda relativa ad una nuova accessione in uno schedario ordinato *lessicograficamente* (A-Z) per autore.

  Algoritmo A (lineare):

  consiste nello sfogliare le schede una per una, iniziando dalla prima nell'ordine.

  Lo sfoglio viene interrotto non appena viene trovata la prima scheda che segue la scheda da inserire nell'ordinamento lessicografico.

  Algoritmo B (binario):

  se lo schedario è vuoto **allora** inserisci la nuova scheda, **altrimenti**

  1. *seleziona* la scheda che si trova circa a metà dello schedario,
  2. *confronta* la scheda selezionata con quella da inserire:
     - **se** la scheda selezionata precede la scheda da inserire, nell'ordinamento lessicografico, **allora** inserisci la nuova scheda nella seconda metà dello schedario (ripeti l'algoritmo B sulla seconda metà dello schedario e la scheda da inserire)
     - **altrimenti** inserisci la nuova scheda nella prima metà dello schedario (ripeti l'algoritmo B sulla prima metà dello schedario e la scheda da inserire)

Efficienza e costi degli algoritmi (*complessità*)

- Consideriamo i due algoritmi precedenti, A e B.
- **Domanda :** quanto tempo impiegheremo per inserire la nuova scheda nel posto giusto, considerando che lo schedario contiene 100.000 schede ed ogni confronto costa 3 secondi?
- Ipotizziamo il **caso peggiore**, quando cioè dobbiamo scorrere l'intero catalogo:
  - nel caso lineare A impiegheremo 100.000×3 secondi, **più di 3 giorni**!
  - nel caso binario B impiegheremo **meno di 1 minuto**!
    ogni volta divideremo lo schedario per 2, quindi il numero dei confronti è

$$log_2 100000 \; < \; 17$$

infatti, ricordando che il logaritmo è l'inverso della potenza:

$$100000 < (2^{10} \times 2^7) \; = \; 2^{17}$$

quindi, impiegheremo meno di : 17 (controlli) $\times$ 3 (secondi) = 51 (secondi) !

Sviluppo di un programma

- Il processo di sviluppo di un programma è organizzato in:

  1. **analisi** del problema e identificazione di una soluzione;
  2. **formalizzazione** della soluzione e identificazione dell'algoritmo risolutivo;
  3. **programmazione**, cioè scrittura di un programma in un linguaggio di programmazione di "alto livello"
  4. **traduzione o compilazione** del programma nel linguaggio di "basso livello" riconoscibile dalla macchina, detto **linguaggio macchina**.

- Un programma P_1 scritto in un linguaggio $\mathcal{L}_1$ può essere trasformato, in due modi, in un programma P_2 equivalente scritto in un linguaggio $\mathcal{L}_2$:

  traduzione: ogni istruzione del codice di P_1 viene tradotta nell'istruzione equivalente di P_2 ed eseguita;

  compilazione: l'intero programma P_1 viene trasformato, in una sola volta, in un programma eseguibile P_2.