5 - Tipi di problemi e scomposizione

Ce ne sono due tipi:

- **Semplici**: l'individuazione degli algoritmi è semplice, sono risolvibili con un azione
- **Complessi**: Difficilmente la soluzione si trova pensando a tutto il problema nella sua interezza
  Un algoritmo non è immediatamente progettabile, deve avvenire tramite fasi di analisi e scelte realizzative, da un analisi generale del problema si passa ad una soluzione a grandi linea e poi specifica per la soluzione finale con tutti i dettagli

Problemi complessi

Un problema complesso si risolve in questo modo:

- Scomporlo in sotto problemi per trovare una soluzione primitiva che equivalga al problema di partenza
- Individuazione del procedimento che porta alla soluzione

Approccio/Principio del Divide et Impera

Il principio del Divide et Impera si basa sul ridurre la complessità del problema fino ad arrivare a problemi primitivi risolubili tramite azioni note, scomponendo in sotto problemi più semplici, individuarne una struttura e relazioni fra di essi. Se il problema è troppo complesso, riapplicare la scomposizione

Scomposizione di problemi

Step-Wise refinement o Top Down design

È una tecnica fondamentale per la programmazione strutturata, trasforma il problema in una gerarchia di problemi di difficoltà decrescente, affrontando prima il suo aspetto generale e poi sempre più dettagli per arrivare agli elementi fondamentali.
Nel processo di raffinamento inizialmente l'attenzione è rivolta a cosa poi, man mano, raffinando si passa a come ovvero ad un algoritmo che indichi come fare per ottenere cosa.
Per ogni passo che della scomposizione ci si allontana dal linguaggio naturale (ad alto livello) e ci si avvicina al linguaggio di programmazione
Può capitare che il sotto problema debba venire risolto più volte, applicandolo a dati diversi, per poter risolvere il problema complessivo

Requisiti

Ogni passo della suddivisione o specificazione deve garantire che la soluzione dei sotto problemi riconduca alla soluzione generale, che i passi e la loro successione abbiano senso e siano possibili e che i sotto problemi siano il più vicino possibili agli strumenti disponibili

Ma quando ci si ferma?

Quando tutti i problemi sono diventati primitivi, è previsto il modo in cui un sotto problema usi i risultati prodotti dalla soluzione dei sotto problemi precedenti ed è fissato l'ordine di soluzione dei sotto problemi

I problemi che si ottengono dalla scomposizione sono:

- **Indipendenti:** per ognuno si può generare un algoritmo e la soluzione possa essere parallela a diversi solutori
- **Cooperanti:** ciascuno può usare il risultato della soluzione di altri

Sviluppo BOTTOM-UP

È il metodo opposto al Top-Down, si parte dal basso verso l'alto, cioè si parte dalle azioni primitive per poter costruire algoritmi semplici al fine di ottenere algoritmi più complessi e infine all'algoritmo finale che diventa la soluzione del problema

In generale, utilizziamo i diversi tipi di scomposizione per diversi casi d'uso:

- Top-down nella costruzione di un nuovo algoritmo
- Bottom-up nell'adattamento per scopi diversi di programmi già scritti

Strumenti

I costrutti offerti dai linguaggi di programmazione strutturata supportano le scomposizioni come:

- Sequenziale (suddividere un problema in parti disgiunte)
- Selettiva (trattamento differenziato in casi differenti)
- Iterativa
- Ricorsiva

Scomposizione Sequenziale

Per la scomposizione sequenziale la soluzione si ottiene tramite una sequenza di passi, eseguiti uno alla volta e uno solo alla volta, non sono mai ripetuti od omessi.
L'ordine in cui i passi sono scritti è lo stesso dell'esecuzione e l'ultimo passo equivale alla terminazione del procedimento

✓Problema: Preparare una tazza di tè

✓Soluzione: l'algoritmo per la tazza di tè

- Bollire l'acqua
- Mettere il tè nella tazza
- Versare l'acqua nella tazza
- Lasciare in infusione per 3 minuti

✓Non sono problemi primitivi

- Ciascuno va ulteriormente scomposto

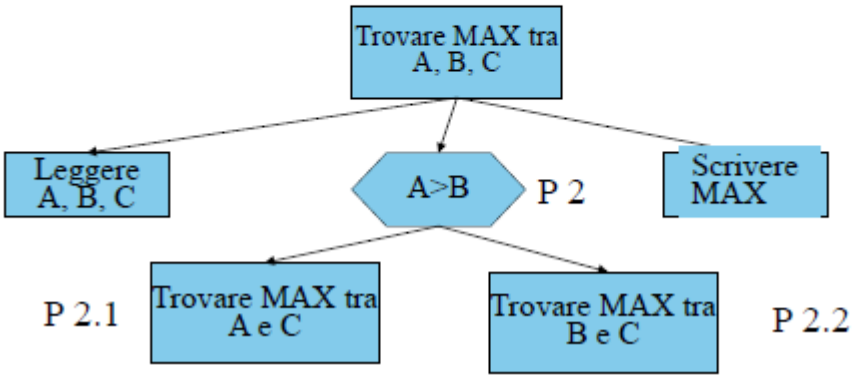
1.
   1.1 Riempire d'acqua il bollitore
   1.2 Accendere il gas
   1.3 Aspettare che l'acqua bolla
   1.4 Spegnere il gas
2.
   2.1 Aprire la scatola del tè
   2.2 Prendere un sacchetto-filtro
   2.3 Chiudere la scatola del tè
   2.4 Mettere il sacchetto nella tazza
3.
   3.1 Versare l'acqua bollente nella tazza finché non è piena
4.
   4.1 Aspettare 3 minuti
   4.2 Estrarre il sacchetto-filtro

Scomposizione selettiva

La soluzione nella scomposizione selettiva viene ottenuta tramite scelte multiple, cioè strutture di selezione all'interno di altre strutture di selezione, con il concetto della nidificazione (o annidamento) di strutture

✓Problema: Trovare il più grande dei numeri *a, b, c* diversi tra loro

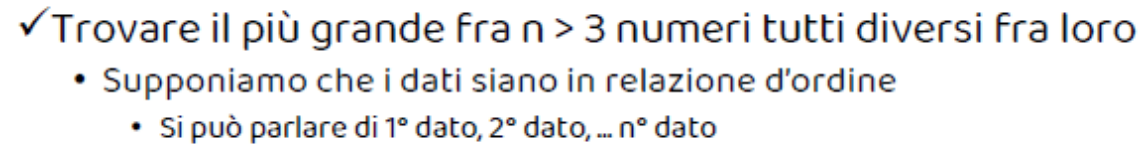
- Azione primitiva (nota):
  - Verificare se un numero è maggiore di un altro

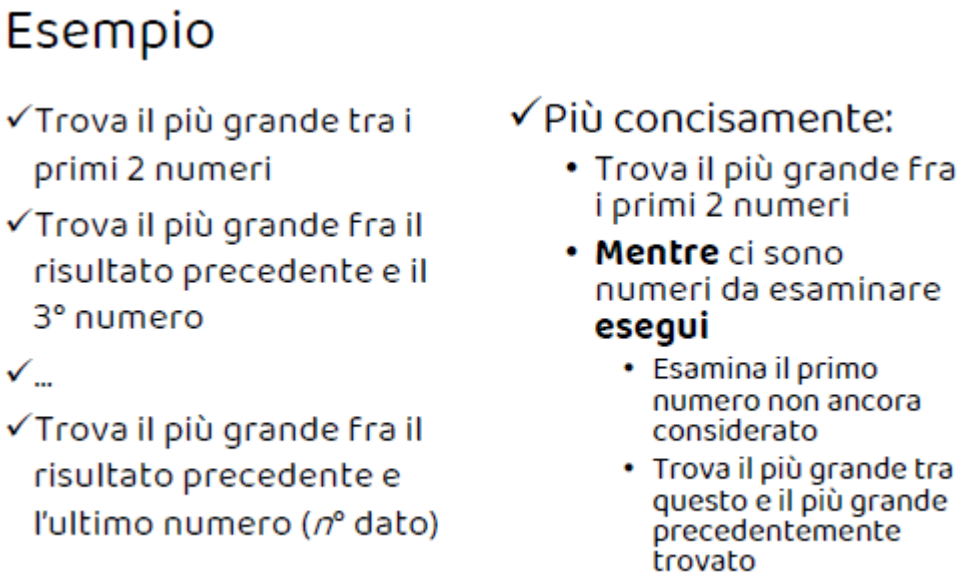


Scomposizione iterativa

I problemi in successione sono tutti dello stesso tipo, si ripete un numero finito di volte un passo di soluzione in modo da avere alla fine, la soluzione completa
La scomposizione di un problema iterativa porta all'individuazione di problemi che soddisfano le seguenti condizioni

- I sotto problemi sono uguali o differiscono unicamente per i dati su cui agiscono
- I dati su cui agiscono i sotto problemi sono in relazione d'ordine e un sotto problema differisce dal precedente solo perché usa il dato successivo





Scomposizione Ricorsiva

Un **oggetto ricorsivo** è un'entità definita in termini di se stessa, questo significa che la definizione dell'oggetto include una versione più semplice dello stesso oggetto. La ricorsione è una tecnica potente perché permette di risolvere problemi complessi scomponendoli in problemi più semplici dello stesso tipo.
Esempio
Numeri Naturali:

- **Definizione**: 1 è un numero naturale. Se (n) è un numero naturale, allora (n+1) è un numero naturale.
- **Ricorsione**: Ogni numero naturale è definito in termini del numero naturale precedente

Una **definizione ricorsiva** è un metodo per definire un problema o una struttura in termini di versioni più semplici di se stessa. Questo tipo di definizione si basa su due componenti

principali:

1. **Passo ricorsivo**: un'operazione che riconduce la definizione ad un livello inferiore.
2. **Caso base**: il punto di partenza della definizione, necessario per ricostruire i livelli successivi.

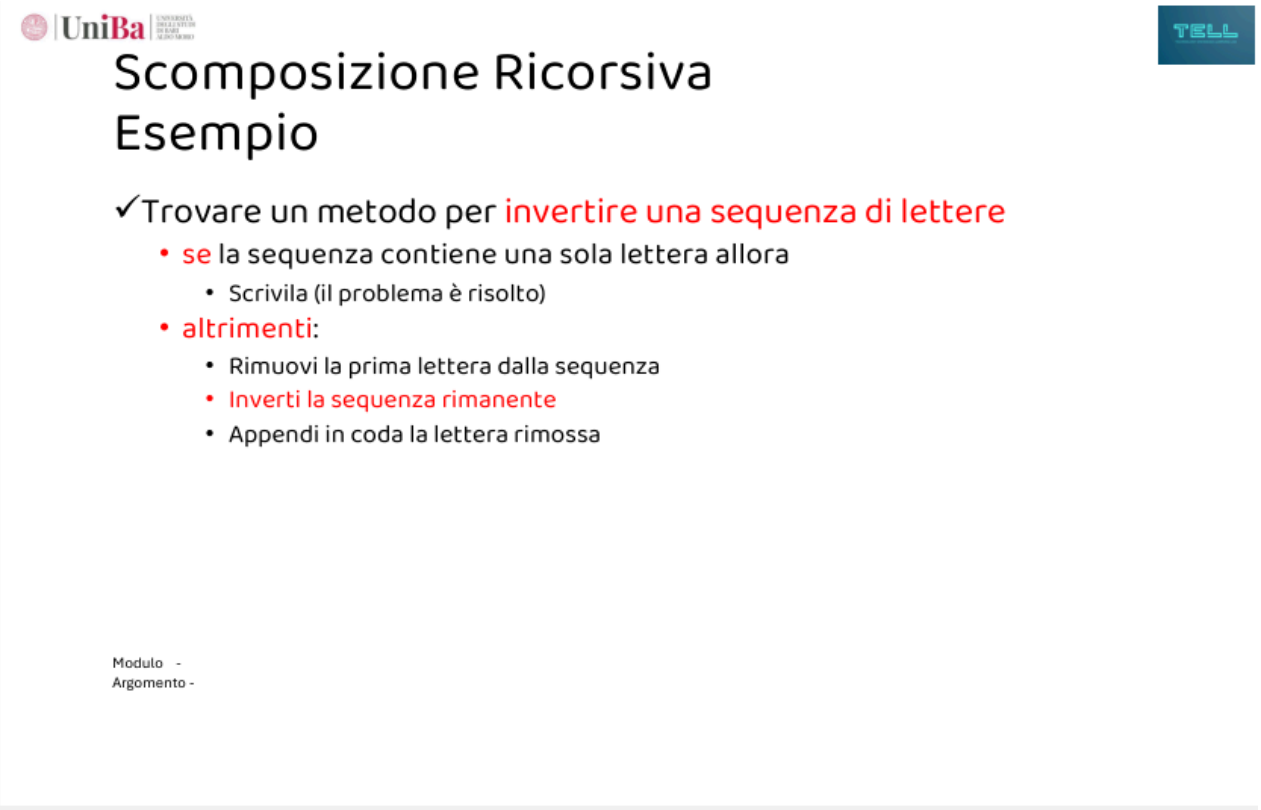
Si può definire quindi la **scomposizione ricorsiva** come un problema di ordine n definito in termini del medesimo problema di ordine inferiore $n-1$, dove l'entità è definita in termini più semplici della stessa entità
La scomposizione ricorsiva ha alcuni requisiti per poter essere effettivmamente tale:

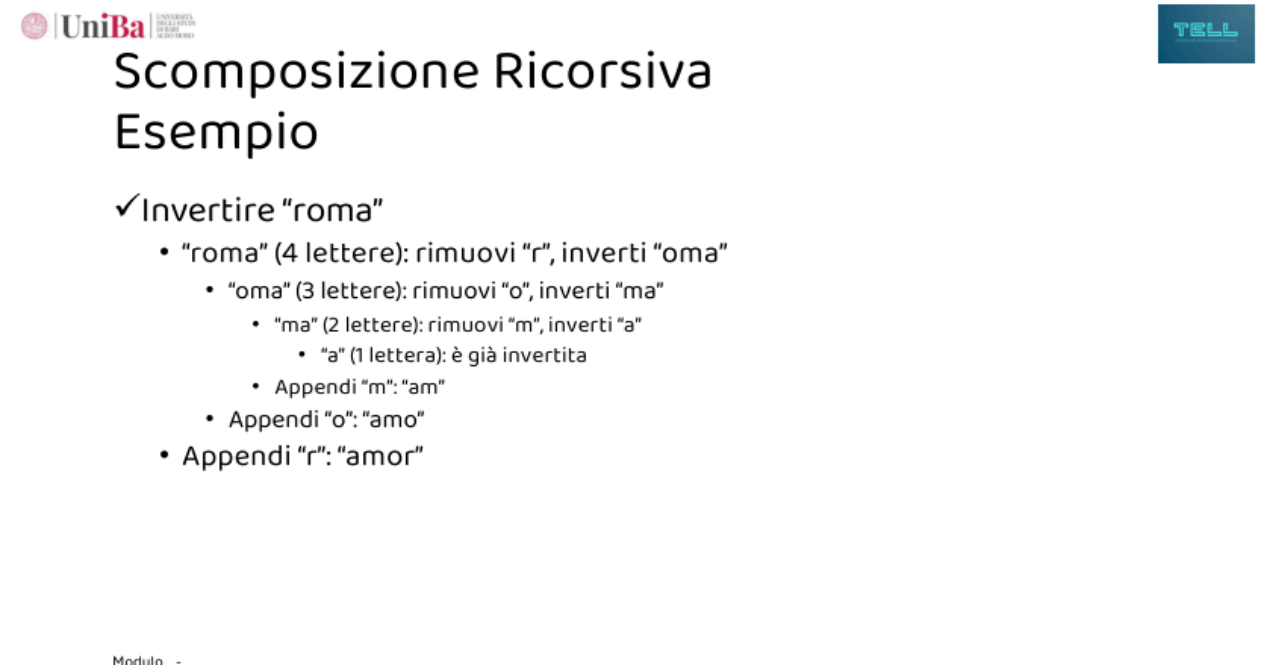
- Almeno uno dei sottoproblemi è formalmente uguale al problema di partenza, ma di ordine inferiore
- Al momento della scomposizione è noto l'ordine del problema

Una scomposizione ricorsiva presenta queste caratteristiche:

- Un problema espresso ricorsivamente termina sempre.
- Un problema esprimibile ricorsivamente può essere risolto anche iterativamente.
- Una funzione esprimibile ricorsivamente è sempre computabile.

- Ogni funzione computabile per mezzo di un programma è ricorsiva





Conclusione

Per poter risolvere un problema può esserci più di una scomposizione in sotto problemi. Alcune cause che rendono la scomposizione difficoltosa possono essere:

- Comprensione intuitiva della complessità di un problema
- Scelta fra le possibile scomposizione

- Necessità di una formulazione "adeguata" per ciascun sottoproblema