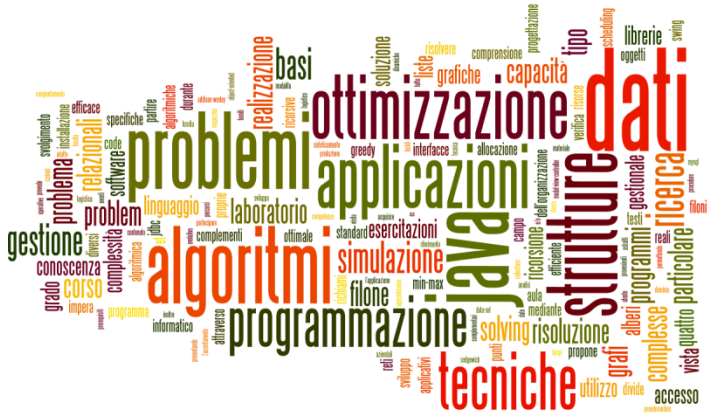
POLITECNICO DI TORINO

e-Lite



Recursion – Some design tips

Tecniche di Programmazione – A.A. 2015/2016

Goal

1. Analysis of a problem to be solved with recursive techniques
2. Identification of the main design choices
3. Identification of the main implementation strategies

Analizzare il problema

- Come imposto in generale la ricorsione?
- Che cosa mi rappresenta il "livello"?
- Com'è fatta una soluzione parziale?

Generale le possibili soluzioni

- Qual è la regola per generare tutte le soluzioni del livello+1 a partire da una soluzione parziale del livello corrente?
- Come faccio a riconoscere se una soluzione parziale è anche completa? (terminazione con successo)
- Come viene avviata la ricorsione (livello 0)?

Identificare le soluzioni valide

- Data una soluzione **parziale**, come faccio a
  - sapere se è valida (e quindi continuare)?
  - sapere se non è valida (e quindi terminare la ricorsione)?
  - nb. magari non posso…
- Data una soluzione **completa**, come faccio a
  - sapere se è valida?
  - sapere se non è valida?
- Cosa devo fare con le soluzioni complete valide?
  - Fermarmi alla prima?
  - Generarle e memorizzarle tutte?
  - Contarle?

Progettare le strutture dati

- Qual è la struttura dati per memorizzare una soluzione (parziale o completa)?
- Qual è la struttura dati per memorizzare lo stato della ricerca (della ricorsione)?

Scheletro del codice

```
// Struttura di un algoritmo ricorsivo generico

void recursive (... , level) {

  // E -- sequenza di istruzioni che vengono eseguite sempre
  // Da usare solo in casi rari (es. Ruzzle)
  doAlways();

  // A
  if (condizione di terminazione) {
    doSomething;
    return;
  }

  // Potrebbe essere anche un while ()
  for () {

    // B
    generaNuovaSoluzioneParziale;


    if (filtro) { // C
      recursive (..., level + 1);
    }

    // D
    backtracking;
  }

}
```

Riempire lo scheletro (del codice)

Blocco	Frammento di codice
A	
B	
C	
D	
E	

```
// Struttura di un algoritmo ricorsivo

void recursive (... , level) {

  // E -- sequenza di istruzioni che ve
  // Da usare solo in casi rari (es. Ru
  doAlways();

  // A
  if (condizione di terminazione) {
    doSomething;
    return;
  }

  // Potrebbe essere anche un while ()
  for () {

    // B
    generaNuovaSoluzioneParziale;


    if (filtro) { // C
      recursive (..., level + 1);
    }

    // D
    backtracking;
  }

}
```

Licenza d'uso

- Queste diapositive sono distribuite con licenza Creative Commons "Attribuzione - Non commerciale - Condividi allo stesso modo (CC BY-NC-SA)"
- Sei libero:
  - di riprodurre, distribuire, comunicare al pubblico, esporre in pubblico, rappresentare, eseguire e recitare quest'opera
  - di modificare quest'opera

- Alle seguenti condizioni:
  - **Attribuzione** — Devi attribuire la paternità dell'opera agli autori originali e in modo tale da non suggerire che essi avallino te o il modo i cui tu usi l'opera.

  - **Non commerciale** — Non puoi usare quest'opera per fini commerciali.

  - **Condividi allo stesso modo** — Se alteri o trasformi quest'opera, o se la usi per crearne un'altra, puoi distribuire l'opera risultante solo con un licenza identica o equivalente a questa.

- http://creativecommons.org/licenses/by-nc-sa/3.0/