Principi di progettazione degli algoritmi

Lezione 2 di Programmazione 2

Docente: Giuseppe Scollo

Università di Catania, sede di Comiso (RG)
Facoltà di Scienze Matematiche, Fisiche e Naturali
Corso di Studi in Informatica applicata, AA 2006-7

Indice

prospettiva metodologica

che "**lezione di metodo**" trarre dallo studio condotto sul problema della connettività?
punti salienti dello schema di studio degli algoritmi **union-find:**

- specifica **chiara** ed **esatta** dell'**enunciato del problema**
- **analisi dell'enunciato**
  - per l'identificazione delle **operazioni astratte** intrinseche del problema
- sviluppo di un **primo algoritmo**
  - **semplice** e di immediata correttezza, in termini delle **operazioni astratte**
- implementazione **semplice** in un **primo programma**
  - con una codifica delle operazioni astratte su **strutture dati** atte allo scopo
- miglioramento dell'efficienza per **raffinamenti successivi**
  - valutandone l'efficacia con **analisi formali e/o valutazioni empiriche**
- **uso di rappresentazioni astratte** di strutture dati e algoritmi impiegati
  - per la **progettazione ad alto livello** delle successive versioni
- **analisi prestazionale** (caso peggiore e medio) per trarne **garanzie di efficienza**

esercizi: livelli di difficoltà

convenzioni: esercizi di

- ▷ **verifica** della comprensione
- ○ **approfondimento**

  aggiungono materiale o stimolano nuove riflessioni

- • **sfida**, livello **normale** di difficoltà
- ? **sfida**, livello **soggettivo** di difficoltà
- •• **sfida**, livello **eccezionale** di difficoltà

  (relativamente ai precedenti)

esercizi di verifica della comprensione

- **1.3** descrivere un metodo semplice per contare il numero di componenti connesse che si hanno al termine dell'esecuzione di un algoritmo per il problema della connettività che usi le operazioni **union** e **find**
- **1.4** mostrare il contenuto dell'array **id** dopo ogni operazione **union** nell'algoritmo **quick-find**, con la sequenza di input **0-2, 1-4, 2-5, 3-6, 0-4, 6-0, 1-3**; per ogni coppia in input, contare anche il numero di accessi del programma all'array **id**
- **1.5 idem** nell'algoritmo **quick-union**
- **1.7 idem** nell'algoritmo **quick-union pesata**

esercizi di approfondimento

aggiungono materiale o stimolano nuove riflessioni

- **riflessione critica sull'enunciato** dell'Esercizio **1.14** :

  trovare una sequenza di coppie che, data in ingresso all'algoritmo di **quick-union pesata**, produca in uscita un cammino di lunghezza 4

- **1.19** fornire un esempio che dimostri che aggiungere la compressione dei cammini completa (v. Esercizio 1.16) all'algoritmo **quick-union** non è sufficiente ad assicurare che gli alberi definiti nell'algoritmo non abbiano cammini lunghi

esercizi di sfida, livello normale o soggettivo

? **1.12** calcolare la distanza media fra un nodo e la radice nel caso peggiore di un albero con 2^n nodi costruito dall'algoritmo di **quick-union pesata**

? **1.16** modificare il programma dell'algoritmo di **quick-union pesata** per implementare la compressione dei cammini completa, dove si termini l'esecuzione dell'operazione **union** facendo in modo che ogni nodo visitato punti alla radice dell'albero

• **1.20** modificare il programma dell'algoritmo di **quick-union pesata** in modo da usare l'**altezza** degli alberi (massima lunghezza dei cammini dai nodi alla radice) invece del peso (numero dei nodi) quale criterio di scelta della radice del nuovo albero; confrontare empiricamente questa variante con il programma suddetto

esercizi di sfida, livello eccezionale

•• **1.18** (riformulato, v. Esercizio 1.14) trovare una sequenza di coppie che, data in ingresso all'algoritmo di **quick-union pesata** con compressione dei cammini completa, produca un cammino di lunghezza 4 nella struttura ad albero definita nell'algoritmo

•• **1.21** mostrare che l'algoritmo descritto nell'Esercizio 1.20 gode della seguente proprietà, valida per l'algoritmo di **quick-union pesata** presentato nel testo:

> l'algoritmo segue al più **2 lg N** puntatori per decidere se due nodi, fra **N**, siano connessi