APA Modulo 1 Esercizi 1

Elena Zucca

4 aprile 2020

Notazioni asintotiche

- usando la definizione dimostrare che:

  $$\textit{se } f(n) = \Theta(g(n)) \textit{ allora anche } g(n) = \Theta(f(n))$$

- soluzione: $f(n) = \Theta(g(n))$ significa che $\exists c_1, c_2 > 0, n_0 \geq 0$ tali che

  $$c_1 g(n) \leq f(n) \leq c_2 g(n) \textit{ per ogni } n \geq n_0$$

- dobbiamo trovare delle costanti tali che

  $$?f(n) \leq g(n) \leq ?f(n) \textit{ per ogni } n \geq n_0$$

  come possiamo sceglierle?

- $$\frac{1}{c_2} f(n) \leq g(n) \leq \frac{1}{c_1} f(n) \textit{ per ogni } n \geq n_0$$

Notazioni asintotiche

- assumendo f, g (asintoticamente) non negative, dimostrare che
$$\max(f(n), g(n)) = \Theta(f(n) + g(n))$$
- dobbiamo provare che $\exists c_1, c_2 > 0, n_0 \geq 0$ tali che
$$c_1(f(n) + g(n)) \leq \max(f(n), g(n)) \leq c_2(f(n) + g(n))$$
per ogni $n \geq n_0$
- come possiamo sceglierle?
- $c_2 = 1$ il massimo è sicuramente minore o uguale della somma
- $c_1 = \frac{1}{2}$ il massimo è sicuramente maggiore o uguale della media

Notazioni asintotiche

- è vero che $\min(f(n), g(n)) = \Theta(f(n) + g(n))$?
- no, controesempio?
- $n + 1$ non è $\Theta(\min(n, 1)) = \Theta(1)$

Relazioni di ricorrenza

- si risolva la seguente relazione di ricorrenza:

$$T(n) = 4T(n/2) + n^2 \log n$$

- conviene assumere $n = 2^k$, quindi $\log n = k$ e $n^2 = (2^k)^2 = 4^k$ sostituendo successivamente si ha:

$$T(2^k) = 4T(2^{k-1}) + 4^k k =$$
$$4(4T(2^{k-2}) + 4^{k-1}(k-1)) + 4^k k =$$
$$4^2 T(2^{k-2}) + 4^k(k-1) + 4^k k =$$
$$\ldots =$$
$$4^k T(2^{k-k}) + 4^k + \ldots + 4^k(k-1) + 4^k k =$$
$$4^k(1 + \ldots + k) = 4^k \frac{k(k+1)}{2}$$

Relazioni di ricorrenza

- ora proviamo $T(2^k) = 4^k \frac{k(k+1)}{2}$ per induzione aritmetica su k:

  Base $T(2^0) = 0$

  Passo induttivo $T(2^k) = 4\,T(2^{k-1}) + 4^k k =$ (per ipotesi induttiva)

  $$4(4^{k-1}\frac{k(k-1)}{2}) + 4^k k =$$
  $$4^k \frac{k(k-1)}{2} + 4^k k =$$
  $$4^k \frac{k(k-1)+2k}{2} = 4^k \frac{k(k+1)}{2}$$

- quindi $T(n) = n^2 \frac{\log n(\log n+1)}{2} = \Theta(n^2 \log^2 n)$.

Design e analisi di algoritmi imperativi

- grafo orientato con n nodi (no cappi) rappresentato con matrice di adiacenza `M`
- (solo in questo esercizio) *pozzo* = nodo che ha un arco entrante da ciascun altro nodo, quindi $n-1$ archi entranti, e nessun arco uscente
- descrivere un algoritmo che determina se esiste un nodo "pozzo"
- giustificarne la correttezza con opportuna invariante
- l'algoritmo deve essere $O(n)$, giustificarlo
  in particolare esaminare solamente $O(n)$ elementi della matrice

Osservazione di partenza

- ogni volta che si esamina una casella `M[i,j]` della matrice cosa possiamo concludere?
- se l'arco esiste `i` non è un pozzo
- se l'arco non esiste `j` non è un pozzo
- quindi esaminando $n-1$ caselle ($O(n)$) riusciamo a scartare come possibili pozzi tutti i nodi meno uno
- a questo punto basta controllare questo ultimo nodo, ossia controllare la sua riga e la sua colonna ($O(n)$)
- NB: può esserci più di un pozzo?

Algoritmo in versione molto astratta e semplice

```
candidati = 1..n
while (|candidati| > 1)
  scegli a caso i,j ∈ candidati (i ≠ j)
  if M(i,j) elimina i da candidati
  else elimina j da candidati
controlla che l'unico nodo k ∈ candidati sia un pozzo
```

Correttezza

- invariante?
- tutti i nodi non in `candidati` non sono pozzi

```
candidati = 1..n //candidati = 1..n
while (|candidati| > 1)
  //INV: ∀ n ∉ candidati. n non pozzo ∧ |candidati| ≥ 1
  scegli a caso i,j ∈ candidati
  if M(i,j) elimina i da candidati
  else elimina j da candidati
//Post: ∀ n ∉ candidati. n non pozzo ∧ |candidati| = 1
controlla che l'unico nodo k ∈ candidati sia un pozzo
```

È immediato vedere che:

- l'invariante vale all'inizio
- si preserva a ogni iterazione
- garantisce la postcondizione
- il ciclo termina perché a ogni passo la cardinalità di `candidati` decresce di uno
- il numero di elementi esaminati e la complessità temporale dell'algoritmo sono $O(n)$
- il ciclo while viene eseguito $n - 1$ volte esaminando ogni volta un elemento
- il controllo finale esamina una riga e una colonna
- assumiamo a costo costante le operazioni sull'insieme

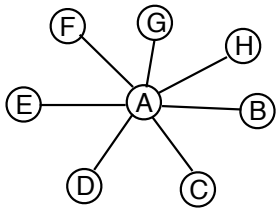
Esercizio da fare

Dare una versione più "concreta" dell'algoritmo in cui si parte prendendo come "candidato" il primo nodo e di volta in volta si cerca se c'e' un arco dal candidato a un nodo successivo. Se l'arco c'è il successivo diventa il nuovo candidato. In questa versione, l'invariante è che tutti i nodi prima del candidato corrente e tra il candidato corrente e il successivo non sono pozzi.

Algoritmi su grafi

Si consideri la visita DFS imperativa: si disegnino due grafi (connessi) con nodi A,B,C,D,E,F,G,H tali che:

- nel primo, ogni nodo entri nella pila una volta sola e l'albero DFS abbia altezza minima



- nel secondo, il nodo H entri nella pila il maggior numero di volte possibile e l'albero DFS abbia altezza massima