Soluzioni della terza esercitazione di Algoritmi 1

Beniamino Accattoli

24 ottobre 2007

1 Esercizi

1. Il problema dell'indice speciale: data una sequenza ordinata in modo crescente di n interi distinti, sia positivi che negativi, determinare se esiste un indice i tale che $a_i = i$. La prima posizione di un vettore ha indice 1, la sua lunghezza è indicata con n, ed è assunto essere non vuoto.

   (a) Dire se esiste, e in caso qual'è, l'indice speciale delle seguenti sequenze: $[-3, 0, 1, 5, 6]$, $[0, 1, 2, 3]$, $[-1, 0, 2, 4, 6]$

   risposta: i primi due vettori non hanno indice speciale, il terzo sì, ed è 4

   (b) Se v è un vettore ordinato in modo crescente e $v_i > i$ cosa si può dedurre su v_{i+1}? Cosa si può dedurre su tutti gli elementi compresi tra v_i e v_n? E se $v_i < i$ cosa si può dedurre su v_{i-1}? E su tutti gli elementi tra v_i e v_1? Ci può essere più di un indice speciale?

   risposta: se $v_i > i$, essendo il vettore ordinato e formato da interi, si ha $v_{i+1} \geq v_i + 1 > i + 1$, e quindi neanche $i + 1$ può essere un indice speciale. Iterando il ragionamento si ottiene che se $v_i > i$ allora nessun indice tra i e n (compresi) può essere un indice speciale. Similmente se $v_i < i$ allora $i - 1$ non può essere un indice speciale, cosìccome ogni altro indice compreso tra i e 1. Questi due fatti mostrano che volendo cercare un'indice speciale in un vettore, se si analizza l'elemento di mezzo del vettore, se questi non è un indice speciale, si può sempre 'buttare' una metà del vettore. Possono esserci più indici speciali in uno stesso vettore ma questi devono essere tutti consecutivi.

   (c) progettare un algoritmo iterativo che stabilisce *se esiste* un indice speciale in un dato vettore in tempo $O(\log(n))$ nel caso peggiore (l'algoritmo deve dire se c'è o meno un indice speciale, ma non deve restituirlo in output)

   risposta:
   algoritmo IndiceSpeciale(*vettore di interi v*) $\longrightarrow$ *boolean*
   1. $i_{sin} \leftarrow 1$
   2. $i_{des} \leftarrow$ lunghezza di v
   3. **if** $(i_{des} = 1)$ **then return** $(v[1] = 1)$

4. **else**
5. $\quad i_{med} \leftarrow \left\lceil \frac{i_{sin}+i_{des}}{2} \right\rceil$
6. $\quad$ **while** $(v[i_{med}] \neq i_{med})$ **do**
7. $\quad\quad$ **if** $(v[i_{med}] > i_{med})$ **then** $i_{des} = i_{med} - 1$
8. $\quad\quad$ **else** $i_{sin} = i_{med} + 1$
9. $\quad\quad$ **if** $(i_{sin} > i_{des})$ **then return false**
10. $\quad\quad$ **else** $i_{med} \leftarrow \left\lceil \frac{i_{sin}+i_{des}}{2} \right\rceil$
11. **return true**

(d) dare un algoritmo ricorsivo con le stesse caratteristiche è un po' più complicato. Il seguente algoritmo:

algoritmo IndSpRic(*vettore di interi* v) $\longrightarrow$ *boolean*
1. $n \leftarrow$ lunghezza di v
2. **if** $(n = 1)$ **then return** $(v[1] = 1)$
3. **else**
4. $\quad i \leftarrow \lceil n/2 \rceil$
5. $\quad$ **if** $(v[i] = i)$ **then return true**
6. $\quad$ **else if** $(v[i] > i)$ **then return** IndSpRic($v[1; max\{1, i-1\}]$)
7. $\quad\quad$ **else return** IndSpRic($v[i+1, n]$)

ottenuto modificando il corrispondente algoritmo per la ricerca binaria non funziona. Nella chiamata ricorsiva alla riga 6 si usa $max\{1, i-1\}]$ invece che semplicemente $i-1$, perché? E perché tale algoritmo non funziona?

risposta: si usa $max\{1, i-1\}]$ perché se $n = 1$ o $n = 2$ allora si ha $i = 1$ e in tal caso $i-1$ vale 0 e la chiamata ricorsiva userebbe il vettore mal definito $v[1; 0]$. L'algoritmo non funziona perché nel caso che la chiamata ricorsiva effettuata sia quella della riga 7 nella nuova chiamata l'indice $i+1$ diventa l'indice 1 falsando la ricerca.

(e) suggerire una modifica dell'algoritmo precedente in modo che sia corretto.

risposta: si deve passare un altro parametro all'algoritmo, uno shift sulla posizione iniziale di cui si deve tenere conto mentre si fanno i confronti. Il testo:

algoritmo IndSpRic(*vettore di interi* v, *intero* shift) $\longrightarrow$ *boolean*
1. $n \leftarrow$ lunghezza di v
2. **if** $(n = 1)$ **then return** $(v[1] = 1+$shift
3. **else**
4. $\quad i \leftarrow \lceil n/2 \rceil$
5. $\quad$ **if** $(v[i] = i+$shift$)$ **then return true**
6. $\quad$ **else if** $(v[i] > i+$shift **then return** IndSpRic($v[1; max\{1, i-1\}]$, shift)
7. $\quad\quad$ **else return** IndSpRic($v[i+1, n]$, shift$+i$)

La prima invocazione dell'algoritmo deve avere $shift$ uguale a 0.

2. Una sequenza $a_1, a_2, .., a_n$ di elementi distinti è ciclicamente ordinata se

il più piccolo elemento è a_i per qualche indice i non noto, e la sequenza $a_i, a_{i+1}, .., a_n, a_1, .., a_{i-1}$ è ordinata in modo crescente. Problema della ricerca in una sequenza ciclicamente ordinata:

> *data una sequenza ciclicamente ordinata di n elementi, trovare l'elemento minimo della sequenza*

(a) sia v un vettore ciclicamente ordinato. Se $v_{n/2} > v_n$ cosa si può dedurre? Qual'è il caso speculare?

risposta: Se $v_{n/2} > v_n$, allora esiste un indice i, con $n/2 < i \leq n$, tale che $v_i < v_{i-1}$, ovvero tale che v_i è il minimo del vettore. Quindi nella ricerca del minimo si può 'buttare' tutta la metà sinistra del vettore, poiché il minimo è sicuramente in quella destra. Viceversa se $v_{n/2} < v_1$ vuol dire che il minimo è compreso tra v_2 e $v_{n/2}$.

(b) progettare un algoritmo ricorsivo che risolve il problema della ricerca in una sequenza ciclicamente ordinata in tempo $O(\log(n))$

risposta:

algoritmo RicercaCicl(*vettore di interi* v) $\longrightarrow$ *intero*
1. **if** $(dim(v) = 1)$ **then return** $v[1]$
2. **else if** $(dim(v) = 2)$ **then return** $min\{v[1], v[2]\}$
3. **else**
4. $i \leftarrow \lceil dim(v)/2 \rceil$
5. **if** $(v[i] > v[n])$ **then return** RicercaCicl($v[i+1; n]$)
6. **else return** RicercaCicl($v[1; i]$)

3. Definizioni (dove f e g sono funzioni sempre positive):

- $f(n) = O(g(n))$ se esistono c e n_0 costanti positive tali che per ogni $n > n_0$ vale $f(n) \leq c \cdot g(n)$
- $f(n) = \Omega(g(n))$ se esistono c e n_0 costanti positive tali che per ogni $n > n_0$ vale $f(n) \geq c \cdot g(n)$
- $f(n) = \Theta(g(n))$ se $f(n) = \Omega(g(n))$ e $f(n) = O(g(n))$
- $\lim_{n\to+\infty} f(n) = 0 \quad \Leftrightarrow \quad \forall\delta > 0 \ \ \exists n_0$ tale che $\forall n > n_0$ vale $f(n) < \delta$
- $\lim_{n\to+\infty} f(n) = +\infty \quad \Leftrightarrow \quad \forall A > 0 \ \ \exists n_0$ tale che $\forall n > n_0$ vale $f(n) > A$

Usando le definizioni fornite, dimostrare (suggerimento: non spaventatevi, questo è l'esercizio più facile):

(a) $\lim_{n\to+\infty} \frac{f(n)}{g(n)} = 0 \quad \Rightarrow \quad f(n) = O(g(n))$

risposta: se $\lim_{n\to+\infty} \frac{f(n)}{g(n)} = 0$ allora $\forall\delta > 0 \ \ \exists n_0$ tale che $\forall n > n_0$ vale $\frac{f(n)}{g(n)} < \delta$ (che è ben definito perché g è sempre positiva). Moltiplicando tutto per $g(n)$ si ottiene $f(n) < \delta \cdot g(n)$. Scegliendo $c = \delta$ nella definizione di $O(g(n))$ si ha che $f(n) = O(g(n))$.

(b) $\lim_{n\to+\infty} \frac{f(n)}{g(n)} = +\infty \quad \Rightarrow \quad f(n) = \Omega(g(n))$

risposta: speculare al punto precedente: se $\lim_{n\to+\infty} \frac{f(n)}{g(n)} = +\infty$ allora $\forall A > 0 \;\; \exists n_0$ tale che $\forall n > n_0$ vale $\frac{f(n)}{g(n)} > A$. Moltiplicando tutto per $g(n)$ si ottiene $f(n) > A \cdot g(n)$. Scegliendo $c = A$ nella definizione di $\Omega(g(n))$ si ha che $f(n) = \Omega(g(n))$.

(c) Riguardo le implicazioni opposte? Che si può dire?

risposta: le implicazioni opposte non valgono. Infatti per essere un $O(g(n))$ basta che *esiste* una costante c tale che *bla bla bla*. Mentre per la definizione di limite si richiede che la limitazione sia possibile *per ogni* costante δ, che è una cosa più forte. Similmente l'altro caso. *A titolo informativo*: in effetti il quadro si completa mostrando che $\lim_{n\to+\infty} \frac{f(n)}{g(n)} = k$ con k reale positivo se e solo se $f(n) = \Omega(g(n))$ e $f(n) = O(g(n))$, ovvero $f(n) = \Theta(g(n))$. L'inversione vale quindi in questa forma $f(n) = O(g(n)) \quad \Rightarrow \quad \lim_{n\to+\infty} \frac{f(n)}{g(n)} = 0$ o $\lim_{n\to+\infty} \frac{f(n)}{g(n)} = k$, con k positivo, ovvero quando tale limite esiste finito (essendo f e g positive il limite non può essere negativo). Invece si ha $f(n) = \Omega(g(n)) \quad \Rightarrow \quad \lim_{n\to+\infty} \frac{f(n)}{g(n)} = +\infty$ o $\lim_{n\to+\infty} \frac{f(n)}{g(n)} = k$, ovvero quando tale limite esiste ed è non nullo.

4. Per le seguenti coppie di funzioni dire se vale $f(n) = O(g(n))$, $f(n) = \Omega(g(n))$ o $f(n) = \Theta(g(n))$ (aiutarsi con l'esercizio precedente):

(a) $f(n) = 9^{\log_3 n}$ e $g(n) = n\sqrt{n}$
risposta: si ha $f(n) = 9^{\log_3 n} = \left(3^2\right)^{\log_3 n} = \left(3^{\log_3 n}\right)^2 = n^2$, e $\lim_{n\to+\infty} \frac{n^2}{n\sqrt{n}} = \lim_{n\to+\infty} \sqrt{n} = +\infty$, ovvero $f(n) = \Omega(g(n))$.

(b) $f(n) = n \log\log n$ e $g(n) = n^{1+\epsilon}$ con $\epsilon > 0$
risposta: $\lim_{n\to+\infty} \frac{n \log\log n}{n^{1+\epsilon}} = \lim_{n\to+\infty} \frac{\log\log n}{n^{\epsilon}} = 0$, ovvero $f(n) = O(g(n))$

(c) $f(n) = 2^{n/2}$ e $g(n) = n \log\log n$
risposta: $f(n) = 2^{n/2} = \left(\sqrt{2}\right)^n$. Vale $g(n) = n \log\log n < n^2$ e quindi $\lim_{n\to+\infty} \frac{(\sqrt{2})^n}{n \log\log n} > \lim_{n\to+\infty} \frac{(\sqrt{2})^n}{n^2} = +\infty$

(d) $f(n) = 1$ e $g(n) = 2^9$
risposta: vale che $1 < 1 \cdot 2^9$ per ogni n, e quindi $f(n) = O(g(n))$, e $1 \geq 1/2^9 \cdot 2^9$ per ogni n, e quindi $f(n) = \Omega(g(n))$. Dunque $f(n) = \Theta(g(n))$.

(e) $f(n) = n^2 + 8n$ e $g(n) = (n \log n)^2$
risposta: dividendo ogni termine per quello di grado massimo (ovvero n^2) si ottiene $\lim_{n\to+\infty} \frac{n^2+8n}{(n\log n)^2} = \lim_{n\to+\infty} \frac{1+8/n}{(\log n)^2} = \lim_{n\to+\infty} \frac{1}{(\log n)^2} = 0$, cioè $f(n) = O(g(n))$.

(f) $f(n) = n^n$ e $g(n) = n!$
risposta: $\lim_{n\to+\infty} \frac{n^n}{n!} = +\infty$, cioè $f(n) = \Omega(g(n))$

(g) $f(n) = 2^n$ e $g(n) = 2^{n/4}$

risposta: $g(n) = 2^{n/4} = \left(\sqrt[4]{2}\right)^n$. E $\lim_{n\to+\infty} \frac{2^n}{\left(\sqrt[4]{2}\right)^n} = \lim_{n\to+\infty} \left(\frac{2}{\sqrt[4]{2}}\right)^n = +\infty$, cioè $f(n) = \Omega\left(g\left(n\right)\right)$.