Prova intermedia di Algoritmi (19 Aprile 2017)

Cognome ..... Nome ..... Matricola .....

Note

1. La leggibilità è un prerequisito: parti difficili da leggere potranno essere ignorate.
2. Quando si presenta un algoritmo è fondamentale spiegare l'idea soggiacente il suo funzionamento e motivarne la correttezza.

Domanda 1 Si determini una soluzione asintotica della ricorrenza

$$T(n) = 4T\left(\frac{n}{2}\right) + n^2(n+1)$$

DOMANDA 1

$T(n) = 4T(\frac{n}{2}) + n^2(n+1)$

$\log_b a = \log_2 4 = 2$

$\lim_{n\to\infty} \frac{2}{n^2(n+1)} = \frac{1}{\infty} = 0$ $\Theta(n^2)$

$T(n) \leq cn^2$

$T(n) = 4T(\frac{n}{2}) + n^2(n+1)$

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

$= An^2 + n^3 + n^2 = n^3 + 2n^2 \sim$ ~~n^3~~ $n^3 + 2n^2 \leq cn^2$

~~$n^2(n+2) \leq cn^2$~~ $n+2 \leq c$

$T(n) \geq dn^2$

$T(n) = 4T(\frac{n}{2}) + n^2(n+1)$

$\geq 4(\frac{n}{2})^2 + n^2(n+1)$

$\sim n^3 + 2n^2 \geq dn^2$

~~$dn^2(n+2) \geq dn^2$~~ $d \geq n+2$

$n \geq d-2$

$c \leq n+2$

$\rightarrow n \leq c-2$

e.g. $c=3$ $n=1$

Domanda 2 Si consideri una tabella hash di dimensione $m = 8$, e indirizzamento aperto con doppio hash basato sulle funzioni $h_1(k) = k \bmod m$ e $h_2(k) \doteq 1 + 2 * (k \bmod (m-1))$. Si descriva in dettaglio come avviene l'inserimento della sequenza di chiavi: 34, 12, 18, 23, 15.

DOMANDA 2

$m = 8$

$h_1(k) = k \bmod m$

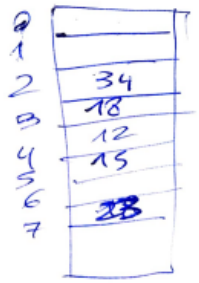
$h_2(k) = 1 + 2(k \bmod (m-1))$

DOPPIO HASHING: $(h_1(k) + i\,h_2(k)) \bmod m$

dove i sta per "tentativo i-esimo"

$1 + 2(3) = 7 + 6 = 13 \bmod 8 = 5$

k: 34, 12, 18, 23, 15



$18 \bmod 8 = 2 \rightarrow h_1 k$

$2 + 1(1 + 2(18 \bmod 7))$

$2 + 1(1 + 2(4)) = 11 \bmod 8 = 3$

$23 \bmod 8 = 7 \rightarrow h_1 k$

$7 + 1(1 + 2(23 \bmod 7))$

$15 \bmod 8 \rightarrow 7$

$= 7 + 1(1 + 2(15 \bmod 7))$

$= 7 + 1(1 + 2(1))$

$= 7 + 3 = 10 \bmod 8 = 2$ COLLISIONE

Domanda 3 Realizzare una funzione `FirstMax` che, dato un array `A[1,n]` ordinato, determina l'indice più piccolo del valore massimo ovvero il minimo i tale che `A[i]` è il valore massimo dell'array. Ad esempio, per l'array che segue si vuole ottenere 5.

1	2	3	4	5	6	7
2	4	4	7	8	8	8

Scrivere lo pseudocodice e valutare la complessità.

$FirstMax(A, n)$
    $i = 0;$
    $while(A[i] < A[n-1])$
        $i++;$
$return\ i;$

Considerando che l'array risulta essere ordinato, basterà trovare un elemento maggiore della prima metà di array (scorsa in avanti), tale che rispetto alla fine l'elemento massimo trovato sia effettivamente quello cercato. L'invariante di questo ciclo è dato dal fatto di avere l'elemento $i < n - 1$; questo garantisce di poter ciclare trovando iterativamente almeno un elemento che soddisfa la proprietà richiesta dall'esercizio. Quando ciò accade, esso viene restituito come indice i. Se questo indice viene trovato, da invariante, risulta essere il primo elemento massimo (in quanto, essendo ordinato, tutti gli indici a lui superiori sono più grandi e altri massimi "non minimi"; ciò significa che l'indice attuale è l'estremo superiore, il più piccolo dei massimi.
Da tali osservazioni, si può anche concludere che la complessità dell'algoritmo sia $T(n) = O(n)$.