

Appello del 12/06/2017

ESERCIZIO 1

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

$$a=2 \quad b=2$$

$$n^{\log_b a} = n$$

$$f(n) = \Theta(n^2) = c'n^2$$

$$f(n) = \Omega(n^{\log_b a + \epsilon}) \quad 0 < \epsilon \leq 1$$

Condizione di regolarità:

$$a f\left(\frac{n}{b}\right) \leq c f(n) \quad \text{con } c < 1$$

$$2 f\left(\frac{n}{2}\right) = 2 \cdot \frac{c'n^2}{4} = \underbrace{c'n^2}_{f(n)} \cdot \frac{1}{2} = \frac{1}{2} f(n)$$

verificata per $c = \frac{1}{2} < 1$

Teorema dell'esperto, 3° caso

$$T(n) = \Theta(n^2)$$

ESERCIZIO 2

1) la funzione

$$h(k, i) = (14k + i) \bmod 7$$

~~ma~~ è un'hashabile, ~~in quanto~~ ^{ma} produce la stessa sequenza di scansioni

$$\langle 0, 1, 2, 3, 4, 5, 6 \rangle$$

per ogni chiave, con conseguente degrado di prestazioni.

Inoltre:

$$\begin{aligned} h(k, i) &= ((14k) \bmod 7 + i \bmod 7) \bmod 7 \\ &= i \bmod 7 \end{aligned}$$

$$0 \leq i \leq 6$$

2)

$$h_1(k) = k \bmod 7$$

$$h_2(k) = 1 + k \bmod 6$$

$$h(k, i) = (k \bmod 7 + i \cdot (1 + k \bmod 6)) \bmod 7$$

T =	4	1	2		11		9
	0	1	2	3	4	5	6

chiave	h_1	h_2	sequenza
2	2		2,
9	2	4	2, 6
1	1		1,
11	4		11,
4	4	5	4, 2, 0

ESERCIZIO 3

Centholi (u, cammino)

// prima chiamata:
// Centholi(T.root, 0)

if (u == NIL) return 0;

else ↓

Cammino = cammino + u.key

Somma Sx = Centholi(u.sx, cammino)

Somma Dx = Centholi(u.dx, cammino)

Somma = Somma Sx + Somma Dx + u.key

if (Somma == Cammino) print u;

return Somma;

↓

$T(n) = \Theta(n)$

L'algoritmo esegue una visita dell'albero,
e spende tempo $\Theta(1)$ su ogni nodo.

ESERCIZIO 4

Dizionario	Ricerca chiave massima	Inserimento
ABR	$O(h)$, $h = O(n)$	$O(h)$, $h = O(n)$
max-Heap	$\Theta(1)$	$O(\log n)$
Tabelle hash, con liste	$\Theta(\max\{n, m\})$	$\Theta(1)$
Tabelle hash, indirizzamento aperto	$\Theta(m)$	$O(n)$

ESERCIZIO 5

1) Certificato: ogni binomio che descrive il sottoinsieme S che copre tutti gli archi di G .

la dimensione del certificato è polinomiale

Algoritmo di verifica:

Esamina (G, S, k) // $n = |V|$, $m = |E|$

num = 0

for $i = 1$ to n

if ($S[i] == 1$) num++;

if (num \neq k) return false;

for all $u \in V$ ↓

for all $v \in \text{Adj}[u]$ ↓

if ($S[u] == 0$ && $S[v] == 0$)

return false;

return true;

$$T(n, m) = O(n + m)$$

L'algoritmo di verifica è polinomiale

\Rightarrow il problema appartiene alla classe NP

2)

Copertura Archi (G, k)

S = nuovo array di dim n

GeneraBinomie ($G, S, 0$);

* failure * // non esiste un sottoinsieme di copertura degli archi, di dim k .

Procedura Elabora di GeneraBinomie

Elabora (G, S, k)

~~se~~

num = 0

for $i = 1$ to n

if ($S[i] == 1$) num++;

if (num \neq k) ~~return~~;

for all $u \in V$ {

for all $v \in \text{Adj}[u]$ {

if ($S[u] == 0$ && $S[v] == 0$)
return;

}

* success *

// omesso il calcolo: è stata trovata una soluzione.

3)

$$T(n, m) = O(2^n * \text{CostoElabora})$$

CostoElabora $T_E(n, m) = O(n + m)$

$$\Rightarrow T(n, m) = O(2^n(n + m))$$