Appello del 12/06/2017

ESERCIZIO 1

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

$a = 2 \quad b = 2$

$n^{\log_b a} = n$ $\qquad f(n) = \Theta(n^2) = n^2$

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

Condizione di regolarità:

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

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

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

Teorema dell'esperto, 3° caso

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

ESERCIZIO 2

1) la funzione

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

è utilizzabile, ma produce la stessa sequenza di scansione

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

per ogni chiave, con conseguente degrado di prestazioni.

Infatti:

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

$$0 \leq i \leq 6$$

2)

$$h_1(k) = k \bmod 7$$
$$h_2(k) = 1 + k \bmod 6$$

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

$T =$

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

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

```
Cerhali (u, cammino)        // prima chiamata:
                            // Cerhali (T.root, 0)

if (u==NIL) return 0;
else {
        cammino = cammino + u.key
        somma Sx = Cerhali (u.sx, cammino)
        Somma Dx = Cerhali (u.dx, cammino)
        somma = SommaSx + somaDx + 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	$\Theta(1)$	$O(\log n)$
Tabella hash, con liste	$\Theta(\max\{n, m\})$	$\Theta(1)$
Tabella hash, indirizzamento aperto	$\Theta(m)$	$O(n)$

ESERCIZIO 5

1) Certificato: array binario che descrive il sottinsieme 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 ≠ k) return false;
for all u ∈ V {
    for all v ∈ 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

GeneraBinarie (G, S, 0);

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

Procedura Elabora di GeneraBinarie

Elabora (G, S, k)

num = 0

for i = 1 to n

if (S(i) == 1) num ++;

if (num ≠ k) return;

for all u ∈ V {

for all v ∈ Adj[u] {

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

return;

}

}

* success * // questa è la codice: è stata trovata una soluzione

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

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

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