Appello del 3 luglio 2017

Esercizio 1

1) $T_1(n) = 9\,T_1(n/3) + 2n^2$

$$n^{\log_b a} = n^{\log_3 9} = n^2$$
$$f(n) = 2n^2 = \Theta(n^{\log_b a})$$

Teorema fondamentale, II° caso:
$$T_1(n) = \Theta(n^2 \log n)$$

2) $T_2(n) = 3\,T_2(n/2) + n^2 \log^2 n$

$$n^{\log_b a} = n^{\log_2 3} \qquad 1 < \log_2 3 < 2$$
$$f(n) = n^2 \log^2 n = \Omega(n^{\log_2 3 + \varepsilon}) \qquad 0 < \varepsilon < 2 - \log_2 3$$

Condizione di regolarità:

$$a f\left(\frac{n}{b}\right) \leq c f(n) \qquad c < 1$$
$$3\left(\left(\frac{n}{2}\right)^2 \log^2 \frac{n}{2}\right) = \frac{3}{4} n^2 \log^2 \frac{n}{2} \leq \frac{3}{4} n^2 \log^2 n$$
$$= \frac{3}{4} f(n)$$

verificata con $c = 3/4 < 1$

Teorema fondamentale, III° caso
$$T_2(n) = \Theta(n^2 \log^2 n)$$

E' DA PREFERIRE IL PRIMO ALGORITMO

ESERCIZIO 2

① Cerca Coppia (a)

```
HeapSort(a);
i = 1
j = i+1;
while (i < n) {
        k = 2*a(i);
        while ( j ≤ n && a(j) < k) j++;
        if ( j > n) return <-1, -1>;
        else if (a(j) == k) return <i, j>;
        else i++;    //caso a(j) > k
}
return <-1, -1>
```

$$T(n) = \underbrace{\Theta(n \log n)}_{\text{costo Heapsort}} + \underbrace{O(n)}_{\text{costo ciclo while}} = \Theta(n \log n)$$

(j non riparte dall'inizio quando si incrementa i).

② cerca2 (a)

```
T = nuova tabella hash, di dim 2n, con concatenamento
for i = 1 to n
      e = nuovo elemento per T
      e.indice = i
      e.key = 2*a(i);
      Insert(T, e);
for j = 1 to n {
      u = Search(T, a(j))
      if (u ≠ NIL) return < u.indice, j >
}
return <-1, -1>
```

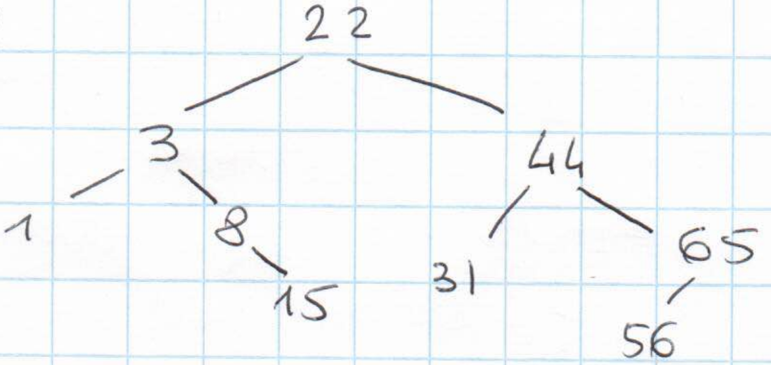
$T_{average}(n) = \Theta(n)$

ESERCIZIO 3

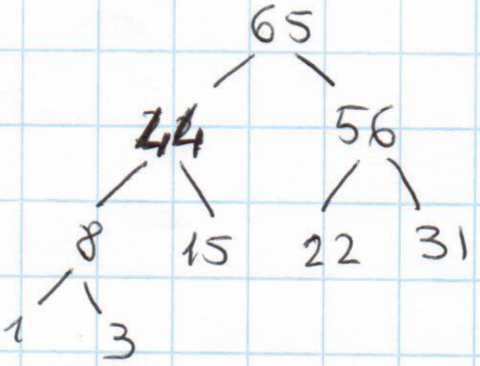
Si veda CLRS, cap 11.

ESERCIZIO 4

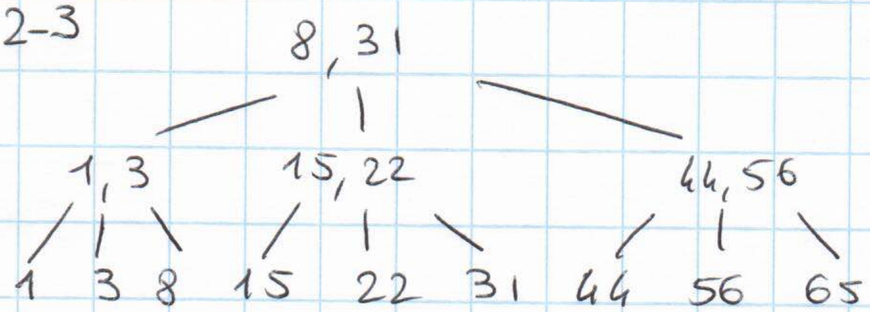
1) ABR



2) Heap di massimo



3) Albero 2-3



ESERCIZIO 5

1) CLIQUE(G, k)

S = nuovo array di dim $n = |V|$
GeneraBinarie(G, S, 0)
* failure * // G non contiene una clique di k nodi

Procedura ELABORA di GeneraBinarie

G sia rappresentato con matrice di adiacenza, per rendere più immediato il test di adiacenza tra due vertici.

ELABORA(G, S, k)

```
num = 0
for i = 1 to n
    if (S[i] == 1) num ++;
if (num ≠ k) return; // il sottoinsieme descritto da S non contiene k nodi
for i = 1 to n {
    for j = i+1 to n {
        if (S(i) == 1 && S(j) == 1) { // i e j ∈ al sottoinsieme descritto da S
            if (A[i, j] == 0) return; // (i, j) ∉ E
        }
    }
}
* success * // si arresta il calcolo: si è trovata una clique.
```

COMPLESSITÀ $T(n, m) = O(2^n * \text{costo Elabora})$ $n = |V|$, $m = |E|$

costo di Elabora: $T_E(n, m) = O(n^2)$

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

2) CLIQUE è un problema NP-completo, infatti:
1) CLIQUE ∈ NP e
2) SAT $\leq_p$ ~~…~~ CLIQUE e SAT ∈ NP-completo.