Algoritmica e Laboratorio
Soluzione Compito del 12/6/2015

Esercizio 1

$T(n) = 5\ T(n/6) + 2n^2$

$a=5, b=6, f(n) = 2n^2$

$n^{\log_b a} = n^{\log_6 5}$ $\log_6 5 < 1$ quindi $f(n) = 2n^2 = \Omega(n^{\log_6 5+e})$ per $0< e\leq 2 - \log_6 5$.

Condizione di regolarità

$af(n/b) \leq cf(n)$, $c<1$

$5f(n/6) = 5x2x(n/6)^2 = 10/36n^2 \leq c2n^2$ soddisfatta per $c = 5/36 <1$

Teorema dell'esperto, terzo caso: $T(n) = \Theta(n^2)$

$T(n) = 7T(n/6) + 3n$

$a=7, b=6, f(n) = 3n$ $\log_6 7 > 1$ quindi $f(n)=3n = O(n^{\log_6 7 - e})$ per $0 < e \leq \log_6 7 - 1$.

Teorema dell'esperto, primo caso: $T(n) = \Theta(n^{\log_6 7})$

Il secondo algoritmo è da preferire per input di dimensione sufficientemente grande.

Esercizio 2

Algoritmo di programmazione dinamica

1) **Sottoproblema i-esimo:** la più lunga sottosequenza crescente nella porzione di array L[1..i] che termina con l'elemento L[i]. Ciò implica che la tabella di PD sia un array M[1..n] ove M[i] è la lunghezza della più lunga sottosequenza in L[1..i] che termina con L[i].
2) **Sottoproblema elementare:** per i=1, la più lunga sottosequenza crescente è composta dall'unico elemento L[1], dunque ha lunghezza 1.
3) **Regola ricorsiva:** Posso allungare di una unità tutte le sequenze che terminano con un elemento L[j] tale che L[j]<L[i]. Tra queste scelgo la sottosequenza di lunghezza massima:
   $M[i] = 1+ \max\{ M[j] \mid j<i \text{ e } L[j]<L[i]\}$
4) **La soluzione** (lunghezza della sottosequenza di lunghezza massima) è data dall'elemento massimo nell'array M.

Pseudocodice

```
Sottocrescente(L)
M=nuovo array di dimensione n;
M[1]=1;
for i=2 to n {
    max=0;
    for j=1 to i-1{
        if (L[i]>L[j]&& M[j]>max)
          max= M[j];
    }
    M[i]=1+max;
}
max=1;
```

```
for i=2 to n {
     if (M[i]>M[max]) max=i;
 }
return M[max];
```

Complessità

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

Ricostruzione della soluzione (sull'esempio)

L = 9 15 **3** 6 **4** 2 **5** **10**

M = 1 2 **1** 2 **2** 1 **3** **4**

Esercizio 3

```
Minpositivo(T)

u=T.root;

while (u!=NIL) {
     if (u.key>0) {
          min=u.key;
          u=u.left;
     }
     else u=u.right;
}
if (min>0)return min;
else return 'no positive key';
```

Dato che il nodo u scende di un livello ad ogni iterazione del while, il numero di iterazioni è proporzionale al più all'altezza dell'albero T, e in ciascuna operazione si esegue un numero costante di operazioni. Al di fuori del ciclo si eseguono al più O(1) operazioni. La complessità è quindi O(h), con h altezza di T.

Esercizio 4

Un heap di altezza h possiede il minimo numero di elementi quando ha la forma di un albero completamente bilanciato di altezza h-1, con una sola foglia sul livello h, dunque:

$$n_{min} = 2^{h-1} - 1 + 1 = 2^h$$

L'heap ha invece il massimo numero si nodi quando è un albero completamente bilanciato di altezza h, cioè:

$$n_{max} = 2^{h+1} - 1 .$$