008AA – ALGORITMICA E LABORATORIO

Secondo Compitino, 30 Maggio 2013

Esercizio 1. *(8 punti)*

Date due stringhe $P = p_1 p_2 \dots p_m$ e $T = t_1 t_2 \dots t_n$, un *allineamento globale* consiste nell'inserzione di spazi in P o in T in modo da ottenere due stringhe della stessa lunghezza. Ad un allineamento globale si associa un punteggio ottenuto assegnando punteggio +1 ad ogni *match* tra le coppie di caratteri corrispondenti nell'allineamento, -1 ad ogni *mismatch* tra coppie di caratteri, e punteggio -2 ad ogni allineamento con uno spazio. Ad esempio l'allineamento

$-$	R	I	S	O	T	T	O
P	R	E	S	$-$	$-$	T	O
-2	$+1$	-1	$+1$	-2	-2	$+1$	$+1$

ha punteggio -3.

Si progetti un algoritmo di programmazione dinamica per trovare il punteggio del miglior allineamento globale tra due stringhe e si simuli l'algoritmo sulla coppia di stringhe PRESTO e RISOTTO.

SOLUZIONE

Assumendo che P e T inizino con un carattere vuoto p_0 e t_0, si definisce una matrice D tale che $D[i][j]$ è il punteggio del miglior allineamento globale tra il prefisso $p_0 p_1 \dots p_i$ e $t_0 t_1 \dots t_j$. Risulta:

$$D[i][j] = \begin{cases} -2i & \text{se } j = 0 \\ -2j & \text{se } i = 0 \\ \max\{D[i][j-1] - 2, D[i-1][j] - 2, D[i-1][j-1] + \delta(i,j)\} & \text{altrimenti} \end{cases}$$

dove $\delta(i,j) = 1$ se $p_i = t_j$ e $\delta(i,j) = -1$ se $p_i \neq t_j$. Il valore $D[m][n]$ individua il punteggio dell'allineamento globale ottimo tra P e T.

	Ø	R	I	S	O	T	T	O
Ø	0	-2	-4	-6	-8	-10	-12	-14
P	-2	-1	-3	-5	-7	-9	-11	-13
R	-4	-1	-2	-4	-6	-8	-10	-12
E	-6	-3	-2	-3	-5	-7	-9	-11
S	-8	-5	-4	-1	-3	-5	-7	-9
T	-10	-7	-6	-3	-2	-2	-4	-6
O	-12	-9	-8	-5	-2	-3	-3	-3

Esercizio 2. *(8 punti)*

Dato un albero binario, progettare un algoritmo efficiente per determinare il minimo valore di Δ per cui l'albero risulti Δ-bilanciato. Analizzare la complessità dell'algoritmo proposto.

SOLUZIONE

```
CercaDelta(T) //T albero binario

<delta,h> = DeltaBil(T.radice);
return t;

DeltaBil(u)

if (u == null) return <0,-1>;
<delta_sx,h_sx> = DeltaBil(u.sx);
<delta_dx,h_dx> = DeltaBil(u.dx);
delta_u = |h_sx – h_dx|;
delta = Math.max{delta_u, Math.max{delta_sx, delta_dx}};
h = 1 + Math.max{h_sx, h_dx};
return <delta,h>;
```

L'algoritmo esegue una postvisita e la sua complessità è $\Theta(n)$, dove n è il numero dei nodi.

Esercizio 3. *(6 punti)*
Descrivere a parole (o in pseudo-codice) un algoritmo efficiente per ordinare n interi positivi di valore $O(n^k)$ con $k > 1$ costante, e analizzarne la complessità.

SOLUZIONE

- Si rappresentano gli interi da ordinare in base n.
- Ogni numero in base n è composto da un numero costante di cifre, ciascuna di valore $O(n)$.
- Si ordinano i numeri cosi' rappresentati con il Radix Sort (usando il Counting Sort come ordinamento stabile).

L'algoritmo così ottenuto ha complessità $O(n)$.

Esercizio 4. *(8 punti)*
Dato un grafo non orientato, progettare un algoritmo che restituisca il minimo numero di archi da aggiungere al grafo per renderlo connesso.

SOLUZIONE

ContaArchi`(G)`

```
numCC = 0;
for (s = 0; s < n; s++) raggiunto[s]=0;
for (s = 0; s < n; s++)  {
   if (!raggiunto[s])  {
      numCC++;
      DFSric(s);
   }
}
return numCC-1;
```

`DFSric(s):` vedi libro di testo

Complessità: $T(|V|, |E|) = O(|V| + |E|)$.