ALGORITMI II (a.a. 2005-2006) Prova esame del 17 LUGLIO 2006

Esercizio 1 Sia $G = (\{1, 2, \ldots, n\}, E)$ un grafo diretto con n nodi. Chiamiamo il grafo *ordinato* se ha le seguenti proprietà.

- Ciascun arco va da un nodo di indice inferiore ad un nodo di indice superiore. Vale a dire che ogni arco diretto ha la forma $< i, j >$ con $i < j$.
- Ciascun nodo, eccetto il nodo n, ha almeno un arco uscente. Vale a dire che per ogni nodo i, $1 \leq i < n$ c'è almeno un arco della forma $< i, j >$.

La lunghezza di un cammino è data dal numero di archi di cui è composto. Bisogna risolvere il seguente problema:
Dato un grafo ordinato G di n nodi, trovare il cammino di lunghezza massima che va dal nodo 1 al nodo n.

Per risolvere il problema si propone il seguente algoritmo greedy

```
INPUT: il grafo di n nodi G
    SOL ← 1
    i ← 1
    WHILE i < n DO
        j ← i + 1
        WHILE < i, j > ∉ E DO j ← j + 1
        SOL ← (SOL, j)
        i ← j
    ENDWHILE
OUTPUT SOL
```

1. *(max 10 punti)* provare che l'algoritmo proposto non è corretto fornendo un'istanza per cui il cammino SOL prodotto non è quello di lunghezza massima.
2. *(max 12 punti)* proporre un algoritmo che in tempo $O(n^2)$ calcola la lunghezza del cammino di lunghezza massima (*Suggerimento: utilizzare la programmazione dinamica calcolando la lunghezza dei cammini di lunghezza massima per tutte le destinazioni j con $1 \leq j \leq n$.*
3. *(max 8 punti)* modificare l'algoritmo proposto al punto 2 in modo da avere in output un cammino di lunghezza massima anzicché la sua lunghezza. La modifica deve avere un costo addittivo $O(n)$.

Soluzione Esercizio 1

1. Si consideri l'istanza in cui il grafo ordinato G ha 5 nodi e matrice di adiacenza come segue:

0	1	1	0	0
0	0	0	0	1
0	0	0	1	0
0	0	0	0	1
0	0	0	0	0

L'algoritmo produce il cammino$(1, 2, 5)$ di lunghezza 2 mentre il cammino più lungo ha lunghezza 3 ed è $(1, 3, 4, 5)$.

2. Definiamo $M[i]$ come la lunghezza del cammino più lungo dal nodo 1 al nodo i, $1 \leq i \leq n$, se non c' è alcun cammino che dal nodo 1 porta al nodo i pongo $M[i] = -1$.

   Per $i = 1$ ovviamente si ha $M[1] = 0$. Per $i > 1$ gli eventuali cammini che portano ad i toccano come ultimo nodo (prima della destinazione) un nodo j con $j < i$ a cui si poteva arrivare a partire dal nodo 1, quindi se per tutti i nodi con $j < i$ si ha $M[j] = -1$ oppure $< j, i > \notin E$ allora non c' è cammino da 1 a i e quindi $M[i] = -1$, in caso contrario vale

$$M[i] = \max_{(j<i)AND(M[j]\neq -1)AND(<j,i>\in E)} \{M[j]\} + 1$$

   in quanto tutti i cammini che portano a i provengono da nodi j con $j < i$ e $< j, i > \in E$ e fra questi devo prendere quello di lunghezza massima.

   Poiché la lunghezza massima è calcolata in $M[n]$ l'algoritmo per ottenerla è il seguente

```
INPUT: il grafo di n nodi G
M[1] ← 0
FOR i = 2 TO n DO
    M[i] ← −1
    FOR j ← 1 TO j − 1 DO
        IF M[j] ≠ −1 AND < j, i >∈ E AND M[i] ≤ M[j] THEN M[i] ← M[j] + 1
    ENDFOR
ENDFOR
OUTPUT M[n]
```

3. Per ottenere il cammino di lunghezza massima basta ripercorrere all'indietro le scelte che hanno portato a calcolarne la lunghezza, basta quindi sostituire l'ultima riga di codice dell'algoritmo precedente con quelle che seguono

```
SOL ← n
dest ← n
i ← n − 1
WHILE i > 0 DO
    IF M[i] + 1 = M[dest] THEN
            SOL ← (i, SOL)
            dest ← i
        ENDIF
    i ← i − 1
ENDWHILE
OUTPUT SOL
```