Algoritmi & Laboratorio, Grafi
Docente: András Horváth
Esame del 3 luglio 2006

1. (5 punti)

- Sia dato il seguente grafo orientato:
  A: B,F
  B: D
  C: A,F
  D: C
  E: B,D
  F: B,E
- Esiste un ordinamento topologico per questo grafo? Se si, si riporti l'ordinamento topologico. Se no, si spieghi perchè no.
- Si riporti una condizione necessaria e sufficente che garantisce l'esistenza di un ordinamento topologico in un qualsiasi grafo orientato.
- Si disegni un grafo con 5 nodi e 9 archi che ha due ordinamenti topologici diversi. Si riporti i due ordinamenti topologici diversi.

2. (4 punti)

- Si riporti uno dei due schemi di algoritmi greedy visti in aula.
- L'algoritmo di Kruskal procede secondo il schema riportato? (Si motivi la risposta.)
- L'algoritmo di Prim procede secondo il schema riportato? (Si motivi la risposta.)

3. (5 punti)

- Si scriva un algoritmo che
  - prende come argomento un grafo orientato (G) e un vertice di questo grafo (s) e
  - restituisce `true` se nel grafo esiste un ciclo che contiene s, e `false` altrimenti.
- Si discuta la complessità nel caso peggiore.

Soluzioni

1.

- Il grafo in questione contiene cicli (per esempio: A-B-D-C-A) e quindi non esiste un ordinamento topologico.
- Una condizione necessaria e sufficente per l'esistenza di un ordinamento topologico è l'aciclicità.
- Un grafo aciclico con 5 nodi può avere al massimo 10 archi:
  A: B,C,D,E
  B: C,D,E
  C: D,E
  D: E

E:
L'unico ordinamento topologico di questo grafo è A,B,C,D,E. Togliendo l'arco $D \to E$, l'ordine dei nodi D e E non è più determinato dal grafo. Quindi il grafo
A: B,C,D,E
B: C,D,E
C: D,E
D:
E:
che ha 5 nodi e 9 archi ha due ordinamenti topologici diversi: A,B,C,D,E e A,B,C,E,D.

2.

- I due schemi sono riportati sui lucidi della lezione su algoritmi greedy (lucido 4 e 12).
- L'algoritmo di Kruskal procede secondo lo schema I perchè **l'appetibilità** dei archi **non cambia** (e sempre il loro peso).
- L'algoritmo di Prim procede secondo lo schema II perchè **l'appetibilità** dei nodi **cambia** (distanza dei nodi già aggiunti alla soluzione).

3.

- Una soluzione consiste in effettuare una visita partendo dal nodo s e ogni volta quando un nodo u viene scoperto controllare se esiste un arco dal nodo u al nodo s. Se un nodo u si può scoprire partendo dal nodo s ed esiste un arco dal nodo u al nodo s allora esiste un ciclo di cui s fa parte: $s \rightsquigarrow u \to s$. Altrimenti non esiste ciclo di cui s fa parte.
- L'algoritmo può essere implementata basandosi su una visita qualsiasi con due righe aggiunte:
  - una per controllare i vicini del nodo appena scoperto e restituire `true` se c'è un arco dal nodo scoperto al nodo s,
  - e un'altra per restituire `false` se la visita finisce.
- In pseudo codice:
  - **INIZIALIZZA**(G)
    for $\forall u \in V$ **do**
    color[u] $\leftarrow$ bianco
  - **ESISTE_CICLO**(G, s)
    color[s] $\leftarrow$ grigio
    while $\exists$ nodo grigio **do**
    $u \leftarrow$ nodo grigio
    if $s \in$ adj[u] **return** `true`
    if $\exists v$ bianco $\in$ adj[u]
    then color[v] $\leftarrow$ grigio
    else color[u] $\leftarrow$ black
    return `false`

- La complessità nel caso peggiore è uguale alla complessità della visita: $O(|V| + |E|)$.