Esercizio

Sia $G = (V, E)$ un grafo diretto con n vertici e m archi. Dato un vertice $s \in V$ si definisca E_s l'insieme di archi uscenti da reachable(s). È noto che esiste un ciclo diretto in E_s se e solo se DFS(G, s) etichetta un arco come BACK EDGE. Usando questa proprietà, progettare e analizzare un algoritmo che dato $s \in V$ restituisce una lista L che contiene gli archi di un ciclo in E_s, se ne esiste uno, altrimenti è vuota.

Supponiamo che se (v, w) è un BACK EDGE allora w è un antenato di v nell'albero definito da DISCOVERY EDGE, e quindi (v, w) forma un ciclo con tutti i DISCOVERY EDGE che tornano risalendo da v a w di "parent" in "parent"

⇒ L'algoritmo per trovare un ciclo diretto in E_s (se ne esiste uno) si articola così:

* Esegui la DFS(G, s) modificata in modo che
  - i BACK EDGE siano etichettati come tali (vedi precedente esercizio)
  - se u scopre v, si imposta un campo $v.parent = u$

x faccio una scansione di tutti gli archi e se non trovo BACK EDGE restituisco una lista vuota, altrimenti, se trovo un BACK EDGE (v, w) restituisco una lista L che contiene i seguenti archi

- (v, w)
- tutti i DISCOVERY EDGE che incontro risalendo da v a w di parent in parent

ESERCIZIO: Scrivere lo pseudo codice in dettaglio

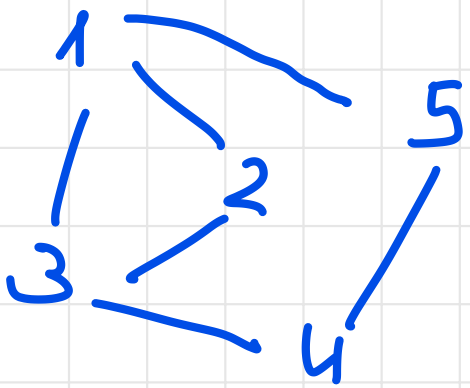
Complessità: $O(n+m)$

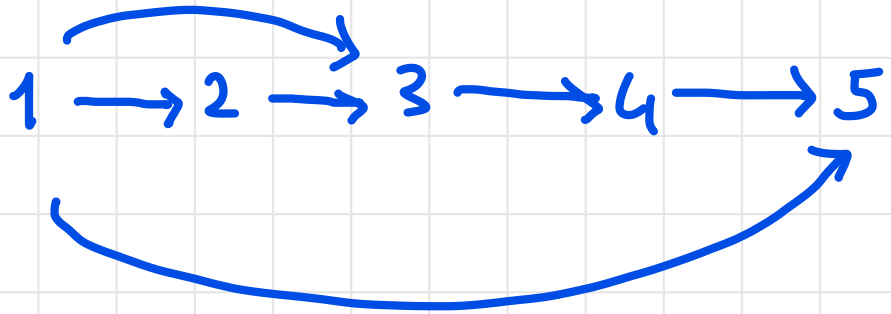
OSS. Facendo restituire alla DFS(G, s) il primo BACK EDGE trovato, o null se non ne trova, e assumendo che i campi dei vertici e degli archi usati dall'algoritmo siano già inizializzati correttamente prima che l'algoritmo sia invocato, la complessità si riduce a $O(|E_s|)$

Esercizio

Dimostrare che dato un grafo semplice non diretto $G = (V, E)$ è possibile dare un orientamento ai suoi archi in modo che esso diventi un DAG.

Es.





Metodo generale:

* Fisso un ordinamento arbitrario tra i vertici: $V = \{v_1, v_2, \ldots, v_n\}$
* Orento ogni arco (v_i, v_j) in modo che parta da $v_{\min\{i,j\}}$ e arrivi a $v_{\max\{i,j\}}$

È facile vedere che quello risultante è un DAG perché un ciclo diretto richiederebbe sicuramente un arco (v_i, v_j) con $i > j$

che non può esistere.

Esercizio

Sia G un DAG i cui vertici rappresentano task e un arco (u, v) indica che v deve essere eseguito dopo u. Per ogni task v, un campo $v.\texttt{length}$ contiene la durata del task. Progettare un algoritmo che in tempo $O(n+m)$ assegna a ciascun task v un tempo di inizio compatibile con le informazioni contenute in G, memorizzandolo in un campo $v.\texttt{start}$.

Suggerimento: usare l'algoritmo $\texttt{TopologicalSort}(G)$ come primitiva.

IDEA

* Calcolo un ordinamento topologico dei vertici di G: $V = v_1\ v_2\ \ldots\ v_n$
* Imposto: $v_1.start \leftarrow 0$

  $v_i.start \leftarrow v_{i-1}.start + v_{i-1}.length$

per ogni $i = 2, 3, \ldots, n$

Complessità: $\Theta(n+m)$

* $\Theta(n+m)$ per calcolare l'ordinamento topologico
* $\Theta(n)$ per calcolare i valori V. start