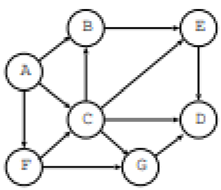
Ordinamento Topologico

E' una funzione σ definita nel seguente modo:
$\sigma : V \rightarrow \{1, \ldots, |V|\}$ tale che $\sigma(u) < \sigma(v)$ se esiste un cammino da u a v in G

Ovvero è una funzione che, dato un vertice V in *input* restituisce un $intero > 0$ che rappresenta l'ordine di tale vertice.

L'idea di fondo è che , dati due vertici A e B, se esiste un cammino da A a B, allora A viene sicuramente prima (parti da A per effettuare il cammino)

Esempio:



$\sigma(A) = 1$, $\sigma(F) = 2$,
$\sigma(C) = 3$, $\sigma(B) = 4$,
$\sigma(E) = 5$, $\sigma(G) = 6$,
$\sigma(D) = 7$

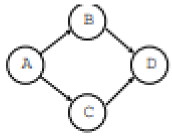
A è il primo dell'ordine topologico, infatti $\forall V \in G$ esiste un cammino da A in V, dunque applicando la definizione della funzione, abbiamo che $\sigma(A) < \sigma($ogni altro vertice$)$.

Ovviamente può esistere un **ordinamento topologico** solo se il grafo è **DAG**(grafo diretto aciclico).
Infatti se esistesse un ordinament topologico in un grafo aciclico avremmo che:

- $\sigma(u) < \sigma(v)$
- $\sigma(v) < \sigma(u)$

Ma ciò sarebbe una contraddizione!

Inoltre su un grafo possono esistere più ordinamenti topologici.
Esempio:



$\sigma_1(A) = 1$
$\sigma_1(C) = 2$
$\sigma_1(B) = 3$
$\sigma_1(D) = 4$

$\sigma_2(A) = 1$
$\sigma_2(B) = 2$
$\sigma_2(C) = 3$
$\sigma_2(D) = 4$

Algoritmo per ottenere l'ordinamento topologico

Un primo approccio sarebbe quello di controllare il **numero di archi entranti** in ogni nodo:

- il primo nodo o_1 avrà 0 archi entranti
- il secondo nodo o_2 potrà avere soltanto 1 arco entrante da o_1
- il terzo nodo o_3 potrà avere soltanto archi entranti da o_1 e o_2
- etc..

tuttavia non è molto efficiente.

Possiamo tuttavia usare un algoritmo basato su **DFS**

Topological Sort DFS

L'idea è quella di controllare gli intervalli di attivazione ($inizio\_visita$,$fine\_visita$):

- Il nodo con $fine\_visita$ più grande sicuramente non avrà archi entranti
  - Questo nodo sarà il primo nell'ordinamento topologico, lo denotiamo con o_1
- Il nodo con $fine\_visita$ secondo più grande potrà avere soltanto archi entranti da o_1

    - Questo nodo dunque sarà il secondo nell'ordinamento e lo denotiamo con o_2
- etc..

Possiamo facilmente adattare una visita **DFS** dunque al problema dell'**ordinamento topologico.**
Basta creare una lista dei vertici in ordine decrescente dei tempi di $fine\_visita$.

```
TOPOLOGICAL-SORT(G)
  L ← lista vuota di vertici
  INIZIALIZZA(G)
  for ∀u ∈ V do
      if u.color = bianco then
          DFS-TOPOLOGICAL(G, u, L)
  restituisci L
```

Per ogni vertice ancora bianco, effettuo una visita in profondità.

```
DFS-TOPOLOGICAL(G, s, L)
  s.color ← grigio
  s.d ← time
  time ← time + 1
  for ∀v : v è bianco ed è ∈ adj[s] do
      v.π = s
      DFS-TOPOLOGICAL(G, v, L)
  s.color ← nero
  s.f ← time
  time ← time + 1
  in testa di L inserisci s
complessità è uguale alla complessità della visita in profondità
```

Effettuo una visità in profondità sul vertice S e appena S diventa nero, lo metto in tesa sulla lista L.
Per ogni vertice adiacente ad S, per definizione di DFS, effettuo una visita in profondità e anche qua, appena tale adiacente diventa nero, lo metto in lista.
Dunque, prima che S venga messo in lista, verranno messi in lista tutti i suoi adiacenti.