Grafi e Grafi Diretti Aciclici

Laboratorio di Programmazione II

Corso di Laurea in Bioinformatica

Dipartimento di Informatica - Università di Verona

Sommario

- Grafi ed implementazione in Java
- Visita di un grafo
- Grafi Diretti Aciclici

Grafi

Grafi

Concetti di base

- Struttura dati per rappresentare relazioni binarie
  - reti di trasporto
  - reti stradali
  - grafo web
  - relazioni tra classi nei linguaggi OO

Terminologia

Definizioni

- $G(V, E)$ V insieme di vertici, E insiemi di archi
- Arco: $< v_i, v_j > | v_i, v_j \in V$
- Cammino (di lunghezza k):
  $< v_1, v_2, \cdots, v_k > | \forall i = 1, \cdots, k-1 < v_i, v_{i+1} > \in E$
- Circuito: cammino con $v_1 = v_k$
- Ciclo: Circuito con $v_i \neq v_j$, $1 < i, j < k, i \neq j$
- Grafo pesato: valore reale $w_{i,j}$ associato ad ogni arco $< v_i, v_j >$

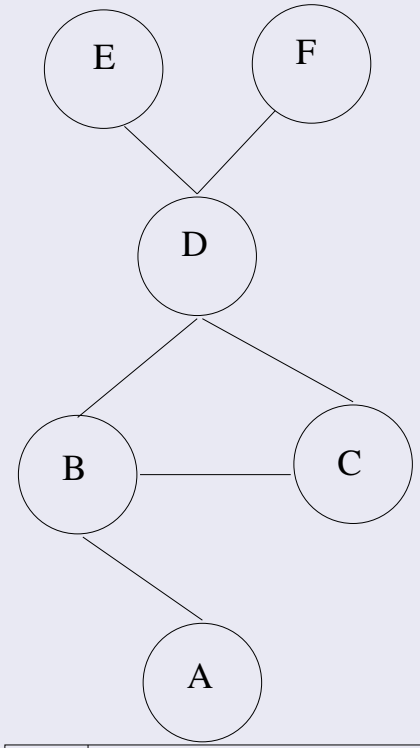
Rappresentazioni

Rappresentare grafi

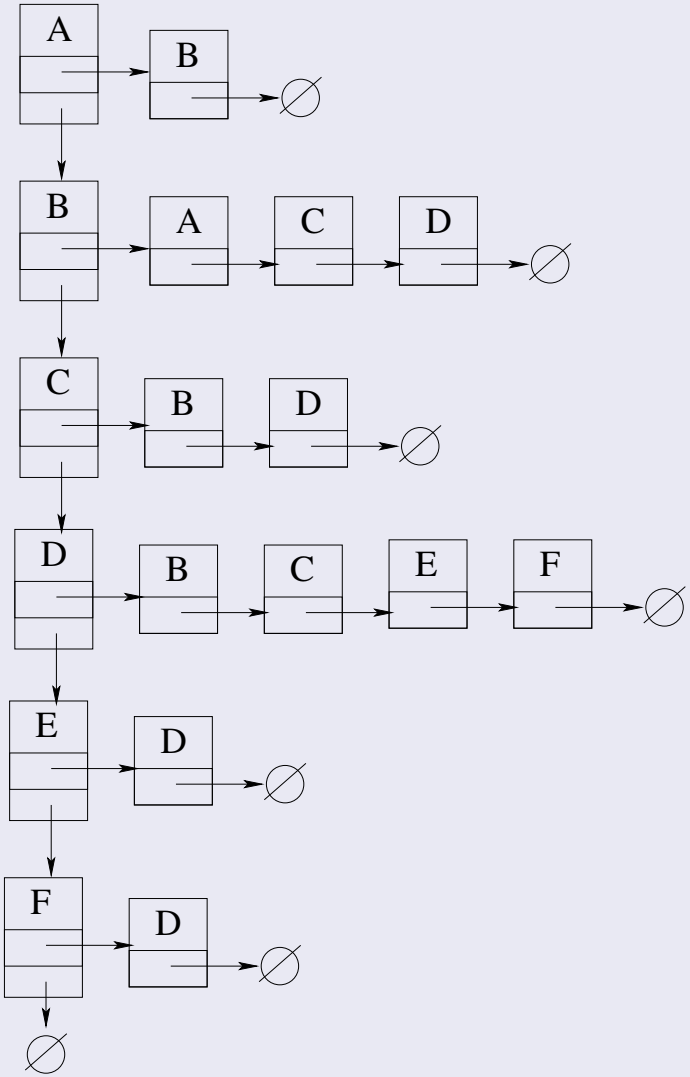
- Liste di adiacenza:
  - ad ogni vertice e' associata la lista dei vertici adiacenti
- Matrice di adiacenza:
  - matrice $|V| \times |V|$
  - $a_{ij} = 1$ se $< v_i, v_j > \in E$, $a_{ij} = 0$ altrimenti

Esempio: lista adiacenze

Esempio: lista adiacenze

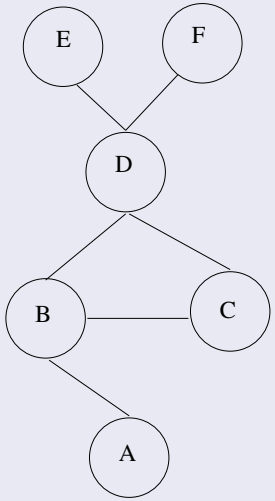


A	B
B	A C D
C	B D
D	B C E F
E	D
F	D



Esempio: matrice adiacenza

Esempio: matrice adiacenza



	A	B	C	D	E	F
A	0	1	0	0	0	0
B	1	0	1	1	0	0
C	0	1	0	1	0	0
D	0	1	1	0	1	1
E	0	0	0	1	0	0
F	0	0	0	1	0	0

Caratteristiche

Vantaggi e svantaggi delle reppresentazioni

- Liste di adiacenza (Memoria O(|E|)):
  - Vantaggi: posso scorrere i nodi adiacenti a v in O(grado(v))
  - Svantaggi: inserimento e cancellazione O(grado(v))
- Matrice di adiacenza: (Memoria O($|V|^2$))
  - Vantaggi: inserimento e cancellazione O(1)
  - Svantaggi: posso scorrere i nodi adiacenti a v in O(|V|)

Implementazione in Java

Implementazione in java

- Utilizziamo una rappresentazione basata su liste di adiacenza
- La lista di nodi e' realizzata con una HashMap
  - non voglio avere nodi replicati
  - inserimento e cancellazione dei nodi sono efficienti
- La lista di nodi successori viene rappresentata come un insieme di archi
  - Insieme: un solo arco tra ciascuna coppia di nodi
  - Arco: piu' efficiente per algoritmi che operano sull'insieme degli archi
- Vedere classi Grafo.java e Arco.java

Visite

Vistare il grafo

Visitare il grafo

- Obiettivo: visitare una sola volta tutti i nodi del grafo
- Difficolta':
  1. Cicli: marcare i nodi visitati
  2. nodi isolati: visita termina quando tutte le componenti del grafo sono state visitate

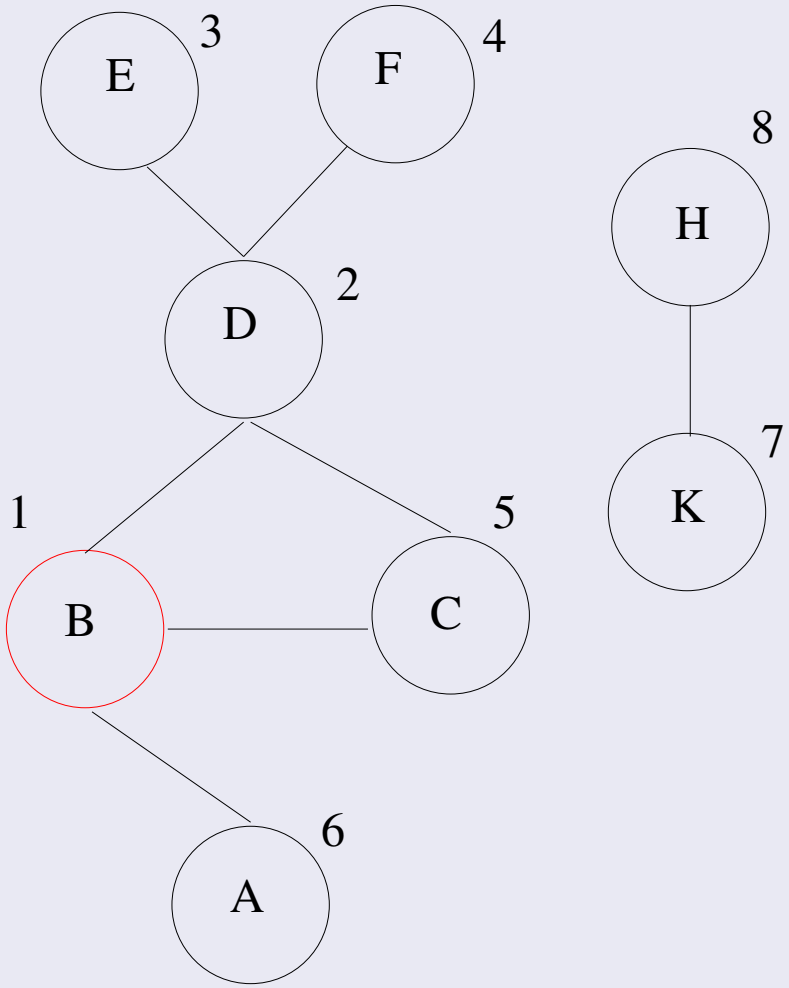
Visita in profondita'

Visita in Profondita': DFS

- Tipicamente ricorsivo
- Dato un nodo, visita uno qualsiasi dei nodi adiacenti non ancora visitati
- Se tutti i nodi adiacenti sono stati visitati, ritornare al predecessore
- I nodi possono essere enumerati secondo l'ordine di visita

Esempio: DFS

Esempio DFS



DFS

Visita in Profondita': DFS

- Inizializzare tutti i nodi come non visitati
- Fino a che esiste un nodo v non visitato
  - eseguire DFS(v)
- DFS(v): vista il nodo e lo marca come visitato
- per ciascun nodo v adiacente non visitato
  - eseguire DFS(v)

DFS: discussione

Visita in Profondita': discussione

- Marcatura: varie opzioni:
- Mantenere una Map di nodi e interi
  - inizializzo la map con tutti i nodi e valore pari a zero
  - un nodo con valore maggiore di zero e' visitato
- Un insieme di nodi non visitati
  - inizializzo il vettore con tutti i nodi
  - quando visito un nodo lo rimuovo dal vettore
- Creare un oggetto nodo che contiene un campo visitato ed aggiornarlo di conseguenza

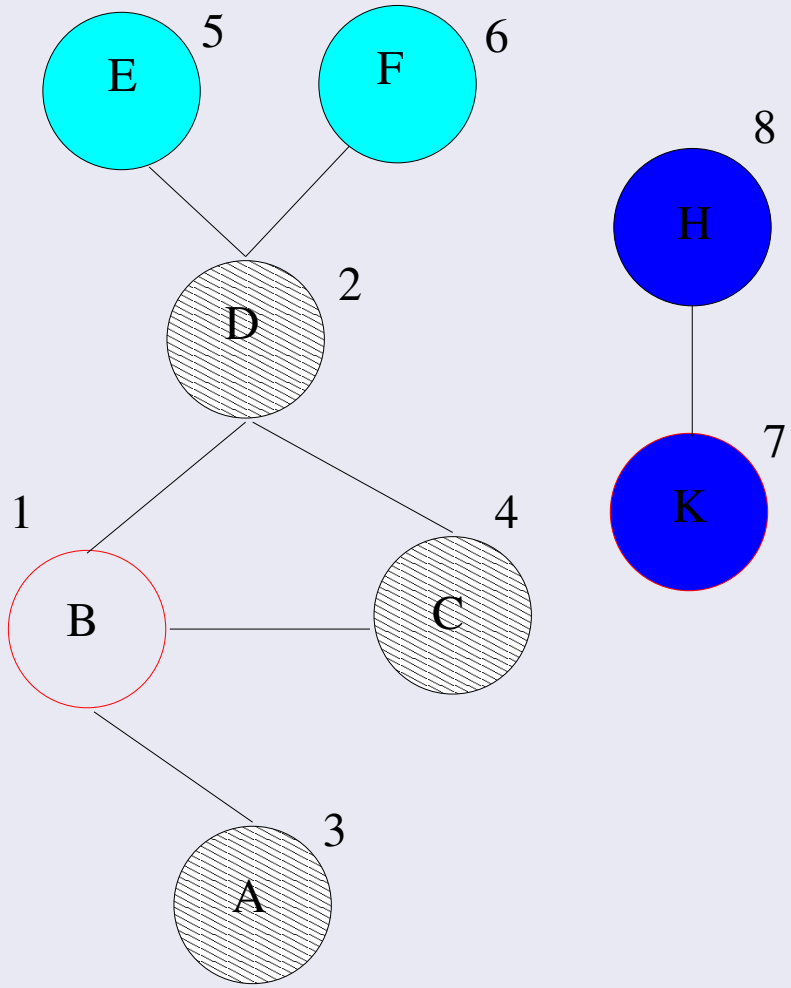
Visitare in ampiezza

BFS

- Tipicamente iterativo: usa una coda di appoggio
- metti in coda il nodo iniziale
- Fino a che la coda non e' vuota:
  - estrai il primo nodo e visitalo
  - metti in coda tutti i nodi adiacenti non visitati
- quando la coda e' vuota controllare se ci sono altri nodi non visitati se si metterli in coda e continuare.

Esempio: BFS

Esempio BFS



Esercizi

Esercizi

- Implementare il metodo dfs(...) della classe Visite.java
- Implementare il metodo visitaBFS() della classe Visite.java

Grafi Diretti Aciclici (DAG)

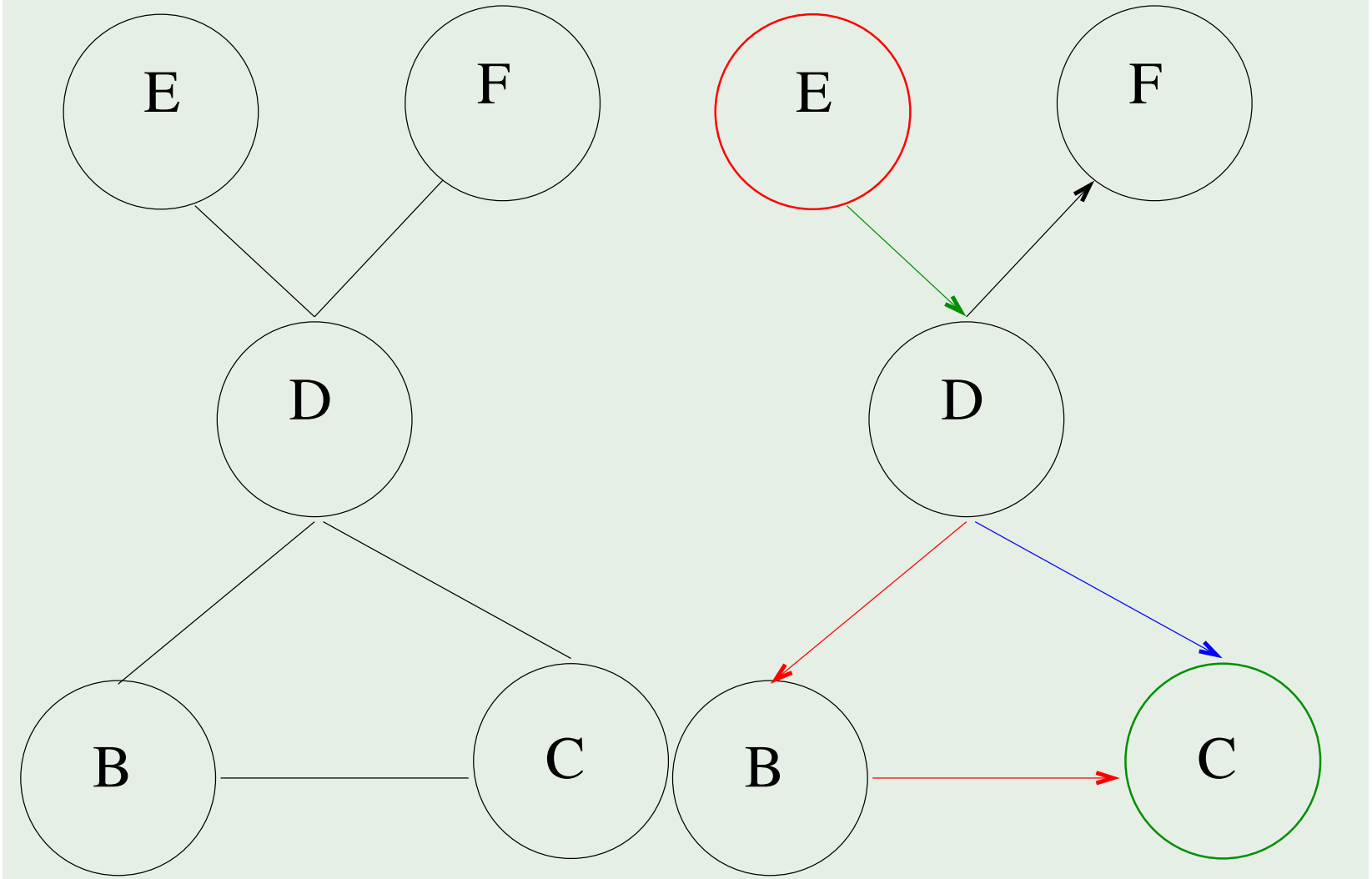
Grafi Diretti Aciclici (DAG)

Concetti di base

- Grafo Diretto: archi sono coppie ordinate di nodi
  - $< u, v > \neq < v, u >$
- Grafo diretto aciclico: grafo diretto senza cicli
  - non esiste nessun cammino $< v_1, v_2, \cdots, v_k >$ tale che $v_1 = v_k$
  - possono esistere diversi cammini da v_1 a v_k

DAG: Esempio

Example (DAG)



Ordinamento parziale e DAG

Ordinamento parziale

- Dato un insieme A, ordinamento parziale: relazione di ordine transitiva parziale sugli elementi di A
- parziale: possono esistere coppie di elementi per cui non e' definito alcun ordine
- Un DAG rappresenta un ordinamento parziale:
  - Vertici: elementi di A
  - Archo $< u, v >$ se $u < v$ secondo l'ordine parziale
- Esempio: sviluppo di moduli per un progetto software
- Ciclo nel grafo $\rightarrow$ il grafo non puo' rappresentare un ordine parziale

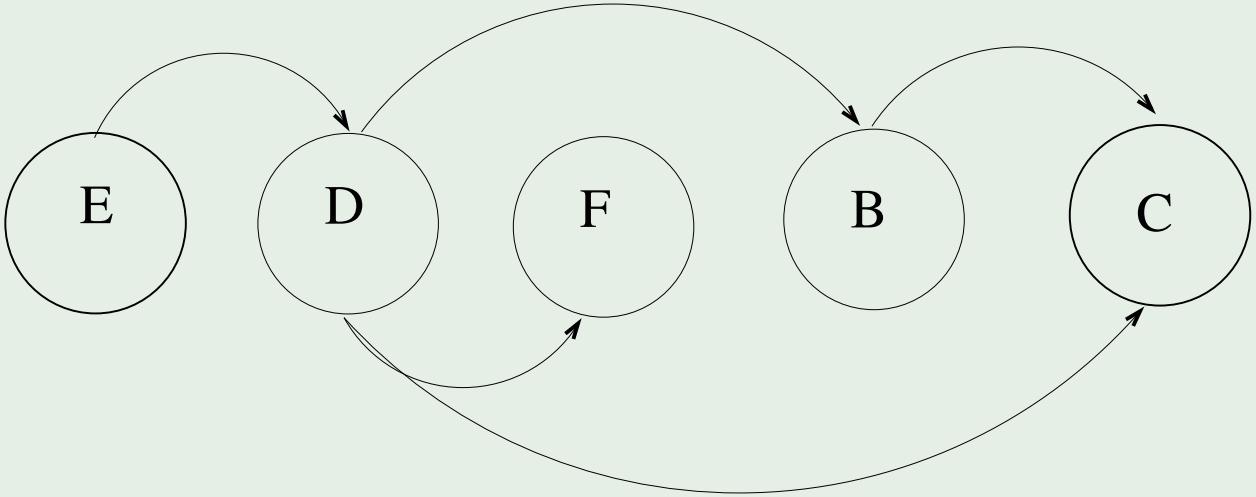
Ordinamento Topologico di un DAG

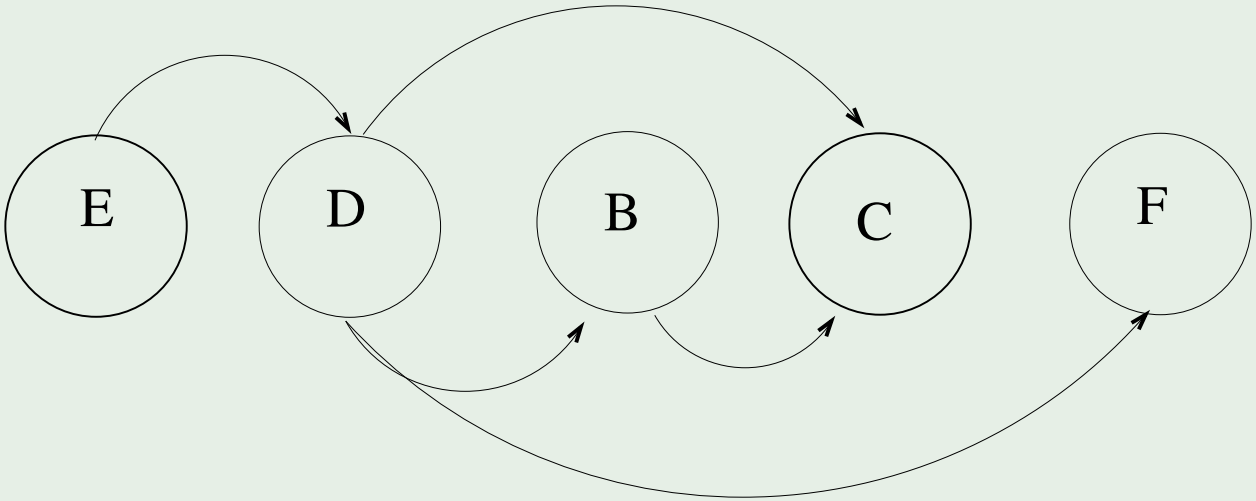
Ordinamento Topologico

- Ordinamento lineare dei veritci tale che:
  - per ogni arco $< u, v >$ u precede v nell'ordinamento
- ad ogni vertice u associamo un intero p(u) tale che se esiste l'arco $< u, v >$ allor $p(u) < p(v)$
- di conseguenza se esiste un cammino da u a w allora p(u) $<$ p(w): ogni nodo e' seguito nell'ordine dai suoi successori
- e' sempre possibile trovare un ordinamento topologico per un DAG
- l'ordinamento topologico non e' univoco

Ordinamenti Topologici: Esempio

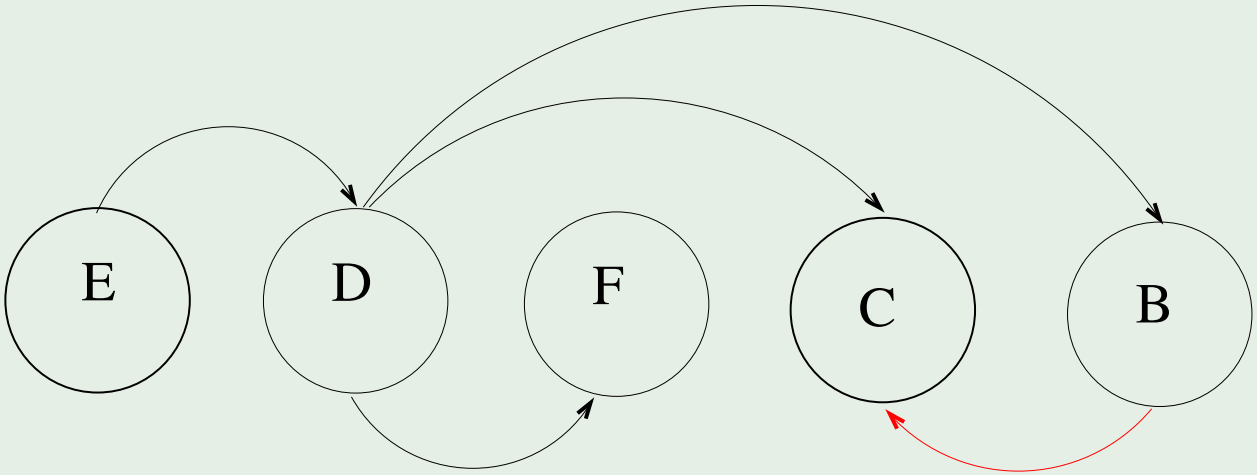
Example (ordinamenti topologici validi)





Ordinamenti Topologici: Esempio II

Example (ordinamento topologico non valido)



Ordinamento Topologico e DFS

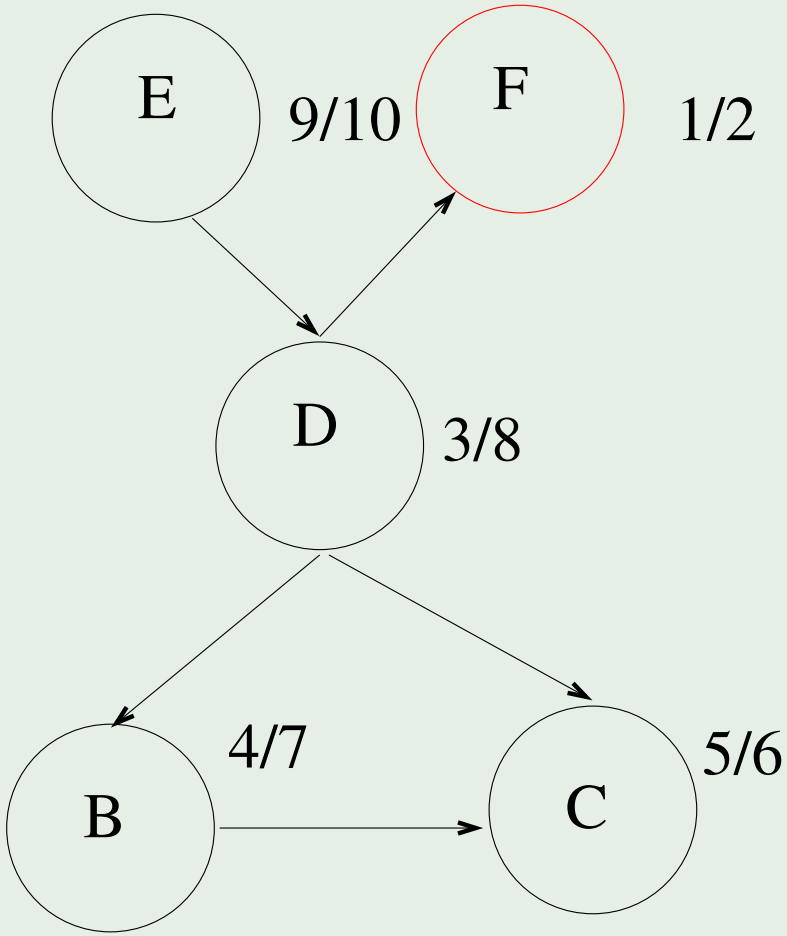
DFS di un DAG

Una DFS di un DAG induce un ordinamento topologico

- Eseguo una DFS registrando il tempo di inizio visista (startValues) e fine visita (finalValues) per ogni nodo
- ad ogni esecuzione della DFS aggiorno il tempo
- Ordinamento topologico: ordine inverso di visita dei vertici
- Utilizzo il tempo di fine visita come $p(u)$
- Intuizione: ciascun nodo avra' un ordine di fine visita maggiore dei suoi successori.

Esempio: DFS per Ordinamenti Topologici

Example (esempio esecuzione di DFS per ordinamento topologico)



DFS per ordine topologico

Pseudocodice ordine topologico

Algorithm 1 OrdineTopologico

```
Require: G grafo
    for ogni vertice v di G do
        inizializza a zero startValues e endValues di v
    end for
    for ogni vertice v di G do
        if startValues di v == 0 then
            dfs(G,v);
        end if
    end for
    costruisci una lista res con i vertici in ordine inverso di fine visita
    return res
```

DFS per ordine topologico

Pseudocodice per DFS

Algorithm 2 dfs

```
Require: G grafo, v node
    time++;
    startValues di v = time;
    for ogni vertice u adiacente a v do
        if startValues di u == 0 then
            dfs(G,u);
        else
            if finalValues di u == 0 then
                errore G non e' un DAG
            end if
        end if
    end for
    time++
    finalValues di v = time
```

Realizzazione DAG in java

DAG in java

- Classe ArcoOrdinato.java
  - estende la classe Arco definendo un ordine tra i nodi dell'arco
- classe DAG.java
  - estende la classe Grafo
  - ridefinisce il metodo add(Object from, Object to, Object Value)
  - ridefinisce il metodo toString()
  - implementa nuove funzionalita: ordineTopologico(...)

Esercizi: DAG

Esercizi sui DAG

- (valido per il progetto) realizzare il metodo getTopologicalOrder(...) della classe DAG.java
  - utilizzare il metodo pullMax(...) (implementato) per estrarre il nodo con endValues piu' alta alla fine della visita
  - utilizzare il metodo ricorsivo dfs(...) (da implementare) per richiamare la visita in profondita' sui nodi
  - le strutture startValues, endValues e la variabile time sono gia dichiarate come variabili di classe