GRAFI

Angelo Di Iorio
Università di Bologna

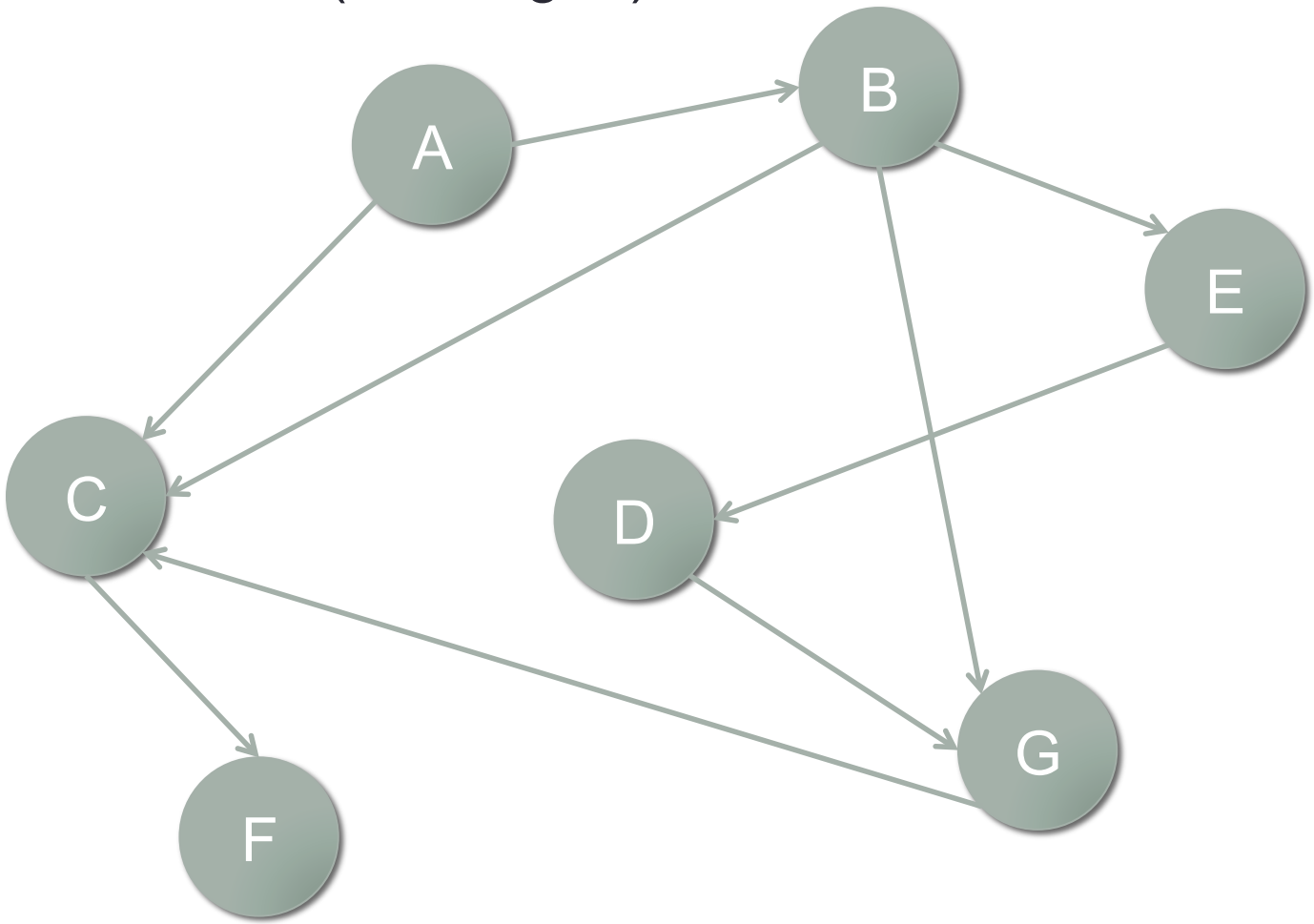
Esercizio 1

- Implementare una classe Java per memorizzare e manipolare un grafo orientato
  - Applicazioni: rete di fermate di autobus, città, reti sociali, etc.
- Usare i Generics di Java per creare una struttura dati flessibile
- Usare le strutture dati disponibili in Java Collection Framework (`java.util.*`)
- Realizzazione basata su liste di adiacenza
- Interfaccia mostrata nella prossima slide

```java
import java.util.Set;

public interface IGraph<T> {

	public void insertNode(Node<T> u);

	public void deleteNode(Node<T> u);

	public void insertEdge(Node<T> u, Node<T> v);

	public void deleteEdge(Node<T> u, Node<T> v);

	public Set<Node<T>> adj(Node<T> u);

	public Set<Node<T>> V();

	// utility: stampa nodi e liste di adiacenza
	public void print();
}
```

Esercizio 1b

- Utilizzare la precedente API per memorizzare il seguente grafo orientato (di stringhe):

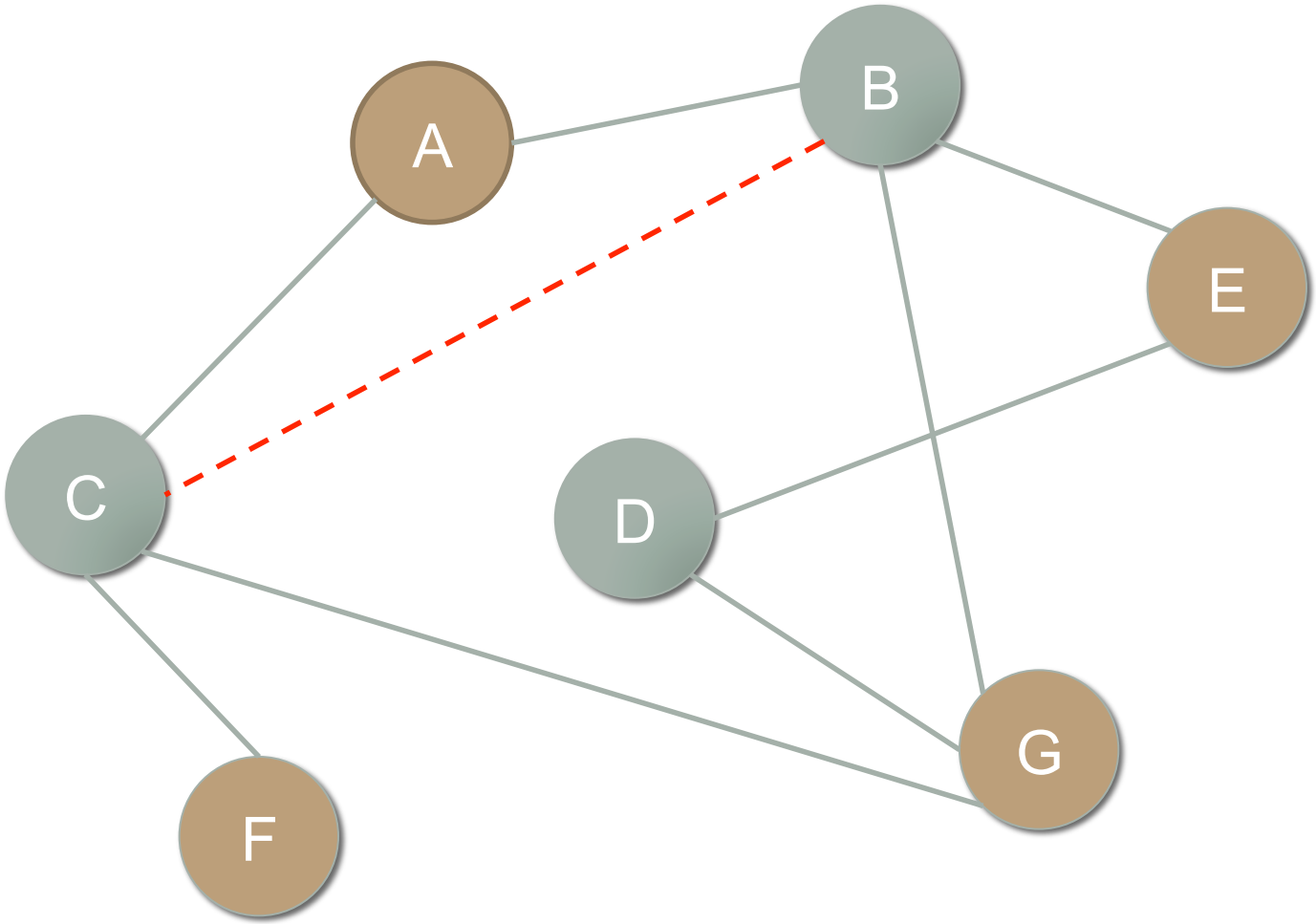


Esercizio 2 (esercizio 9.4 del libro)

- Implementare un metodo per verificare se un grafo non orientato è bipartito oppure no

Esercizio 2b

- Dati di esempio

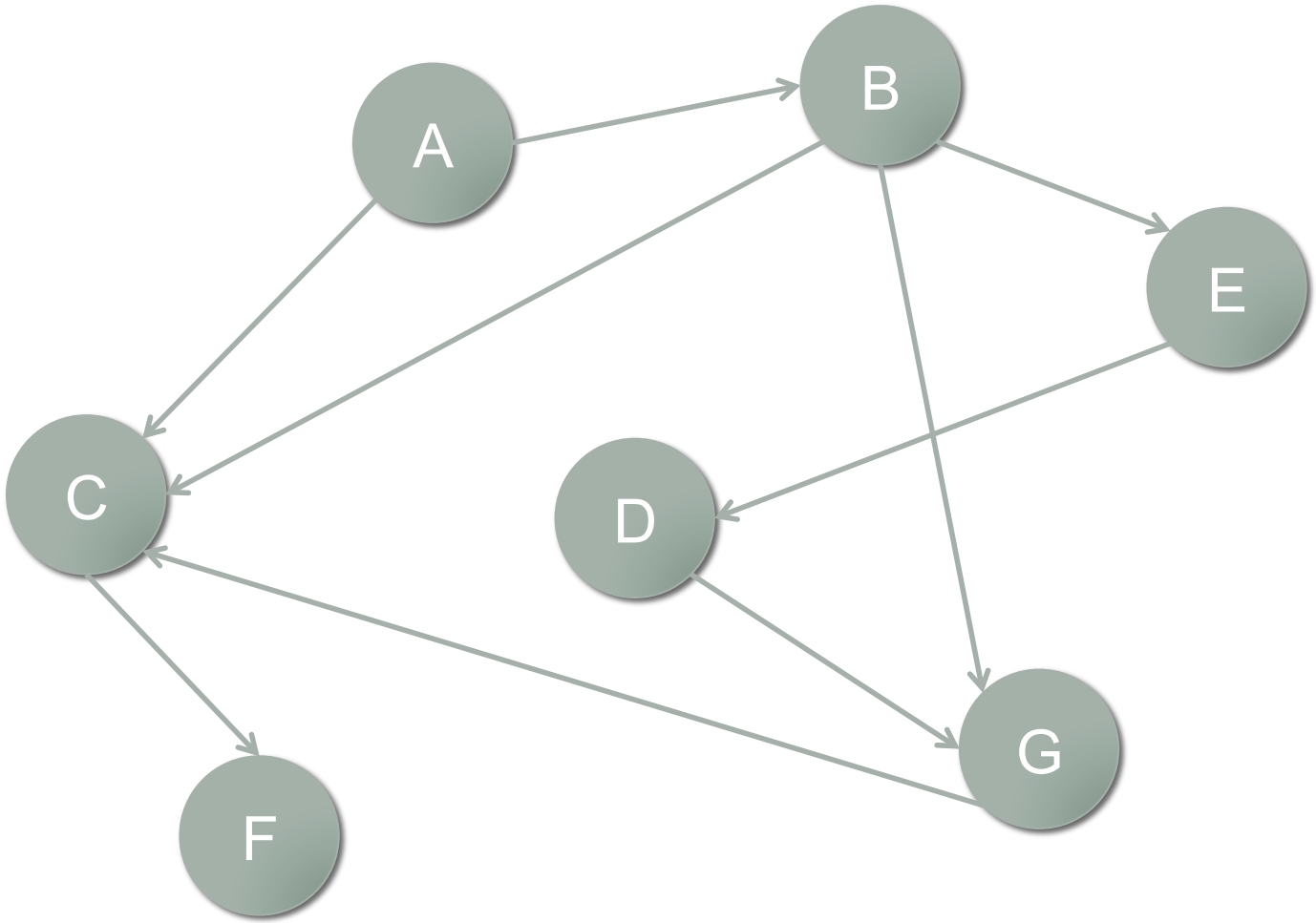


Esercizio 3

- Implementare un metodo per stampare il cammino BFS da un nodo `u` ad un nodo `v` in un grafo orientato (o un messaggio di errore se il cammino non esiste)

Esercizio 3b

- Dati di esempio



Esercizio 3 (esercizio 9.2 sul libro)

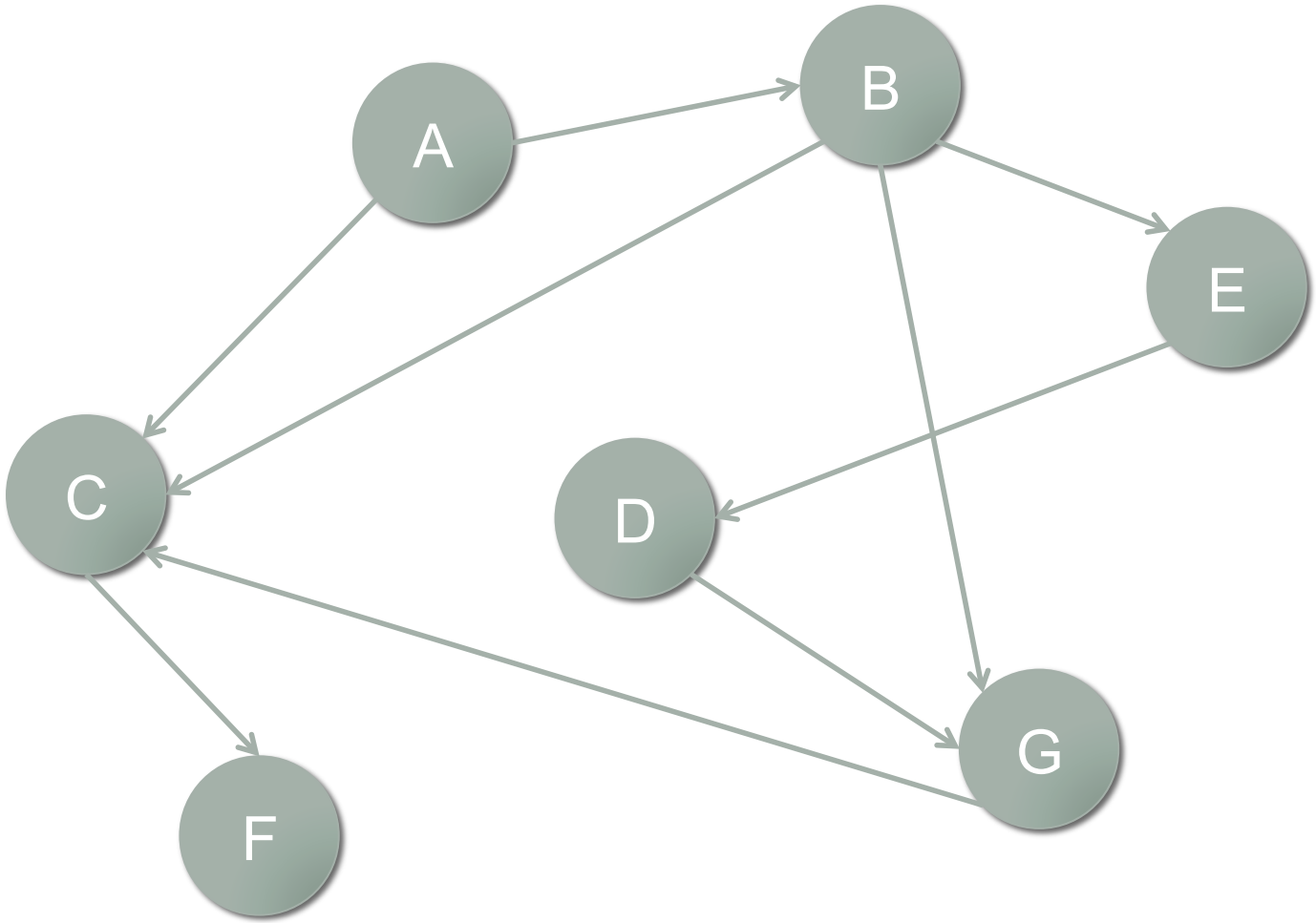
- Scrivere una versione iterativa della DFS dove l'ordine di visita dei nodi e degli archi sia lo stesso di quello della versione ricorsiva.

Esercizio 4

- Implementare un metodo per verificare se un grafo orientato è aciclico oppure no

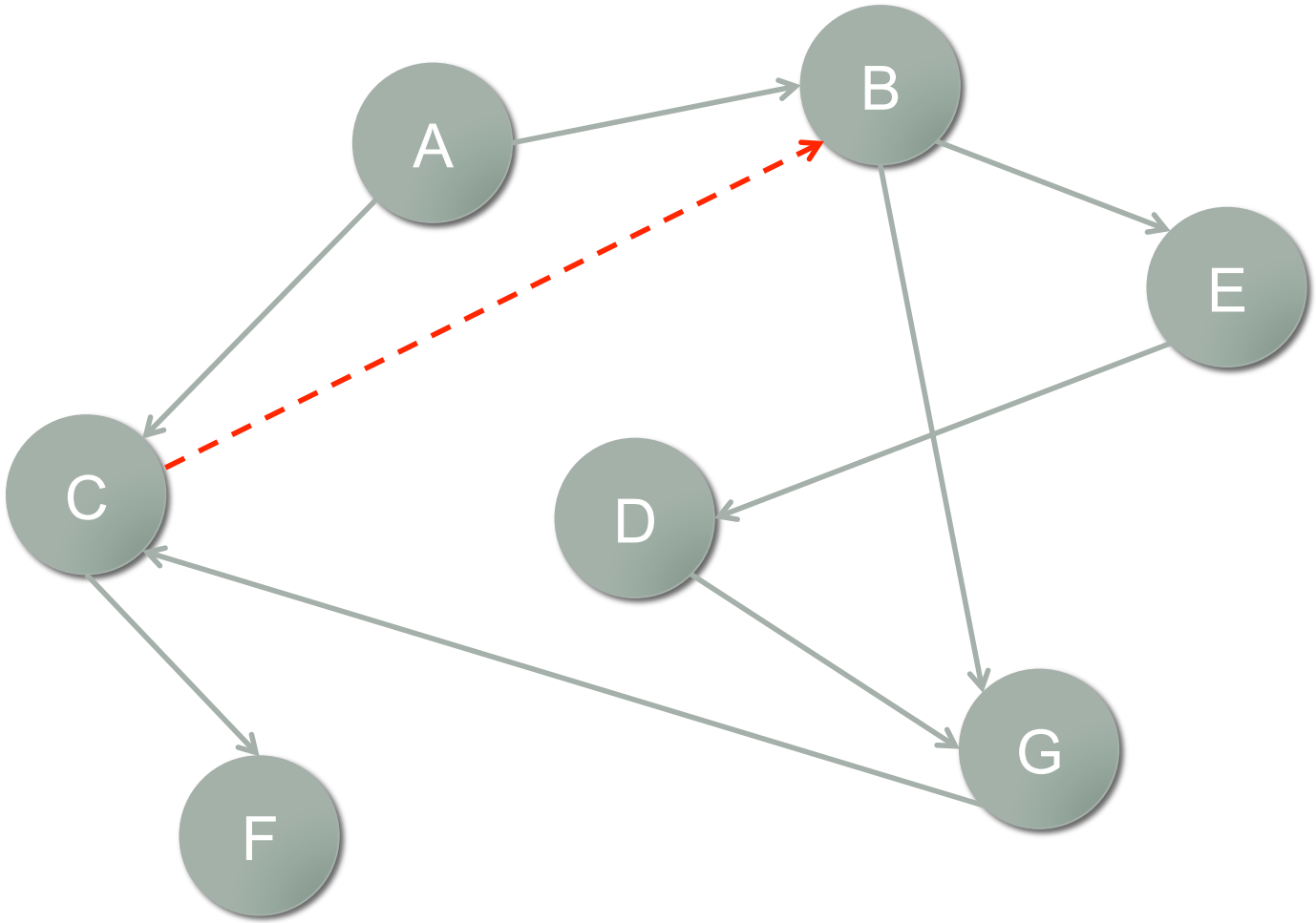
Esercizio 4b

- Dati di esempio



Esercizio 4c

- Dati di esempio



Esercizio 5

- Implementare un metodo per calcolare un ordinamento topologico su un grafo orientato (aciclico)