



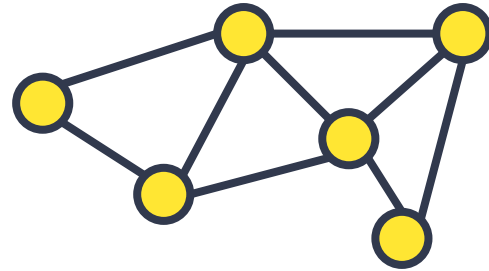
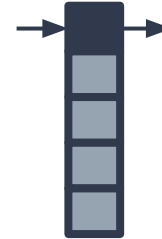
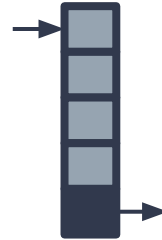
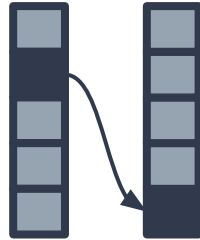
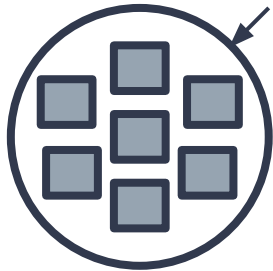
PROGRAMMA **ARNALDO**

Strutture avanzate

A.A 2022/23



Struttura dati = Elementi + Operazioni





Queue

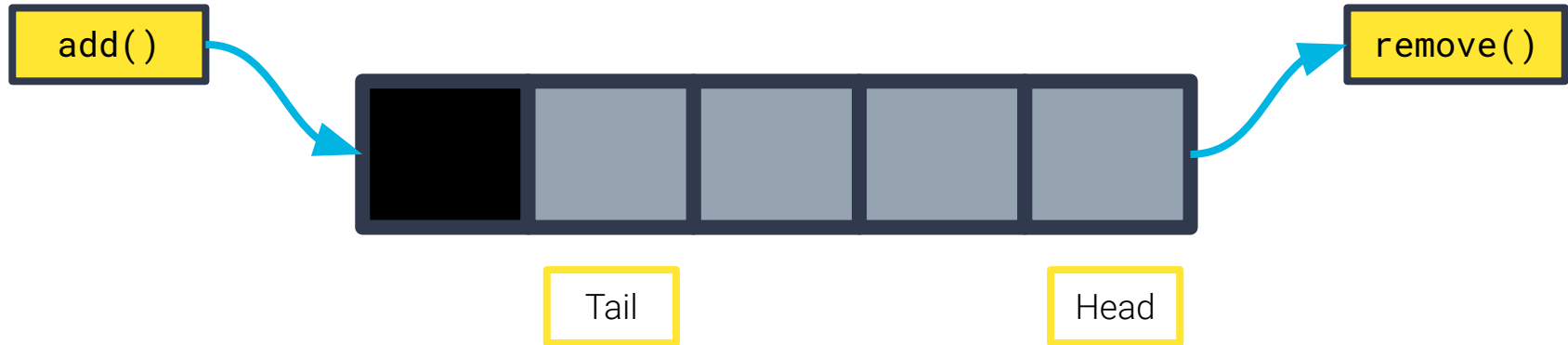
“A collection designed for holding elements prior to processing”



Cos'è una queue



Una queue (coda) è una struttura dati ad accesso **sequenziale** dove gli elementi sono inseriti e rimossi secondo il principio **FIFO**.



Metodi di Queue



L'interfaccia **queue** definisce due tipologie di metodi, la prima lancia eccezioni se l'operazione fallisce, l'altra ritorna un "valore speciale" (null or false)

	Throws exception	Returns special value
Insert	<code>add(e)</code>	<code>offer(e)</code>
Remove	<code>remove()</code>	<code>poll()</code>
Examine	<code>element()</code>	<code>peek()</code>

Queues in Java



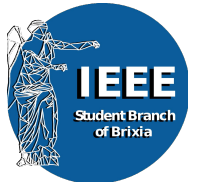
```
Deque<String> clienti = new ArrayDeque<>();
```

```
clienti.add("Bob");  
clienti.add("Paul");  
clienti.add("Tom");  
clienti.add("Jack");  
  
while(!clienti.isEmpty()) {  
    System.out.println(clienti.remove());  
}
```

Quando usare una coda?



- Invio richieste ad un server.
- Come buffer di memoria nei dispositivi di networking
- Nel sistema operativo per quanto riguarda il job scheduling
- BFS
- ...





Stack

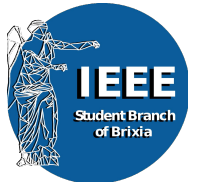
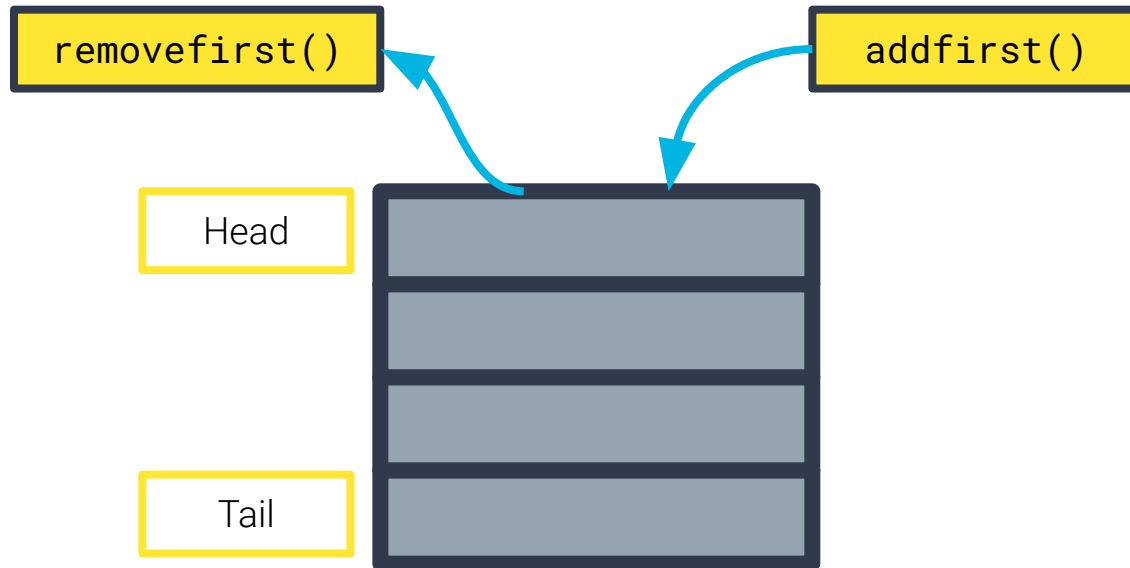
*“A pile of things
arranged one on top of another”*



Cos'è uno stack



Uno **stack** (pila) è una struttura dati ad accesso **sequenziale** dove gli elementi sono inseriti e rimossi secondo il principio LIFO.



Metodi di Deque



L'interfaccia **Deque** definisce due tipologie di metodi, la prima lancia eccezioni se l'operazione fallisce, l'altra ritorna un "valore speciale" (null or false)

	Throws exception	Returns special value
Insert	<code>addFirst(e)</code>	<code>offerFirst(e)</code>
Remove	<code>removeFirst()</code>	<code>pollFirst()</code>
Examine	<code>getFirst()</code>	<code>peekFirst()</code>

Stack in Java



```
Deque<String> clienti = new ArrayDeque<>();
```

```
clienti.addFirst("Bob");  
clienti.addFirst("Paul");  
clienti.addFirst("Tom");  
clienti.addFirst("Jack");  
  
while(!clienti.isEmpty()) {  
    System.out.println(clienti.removeFirst());  
}
```

Quando usare uno stack?

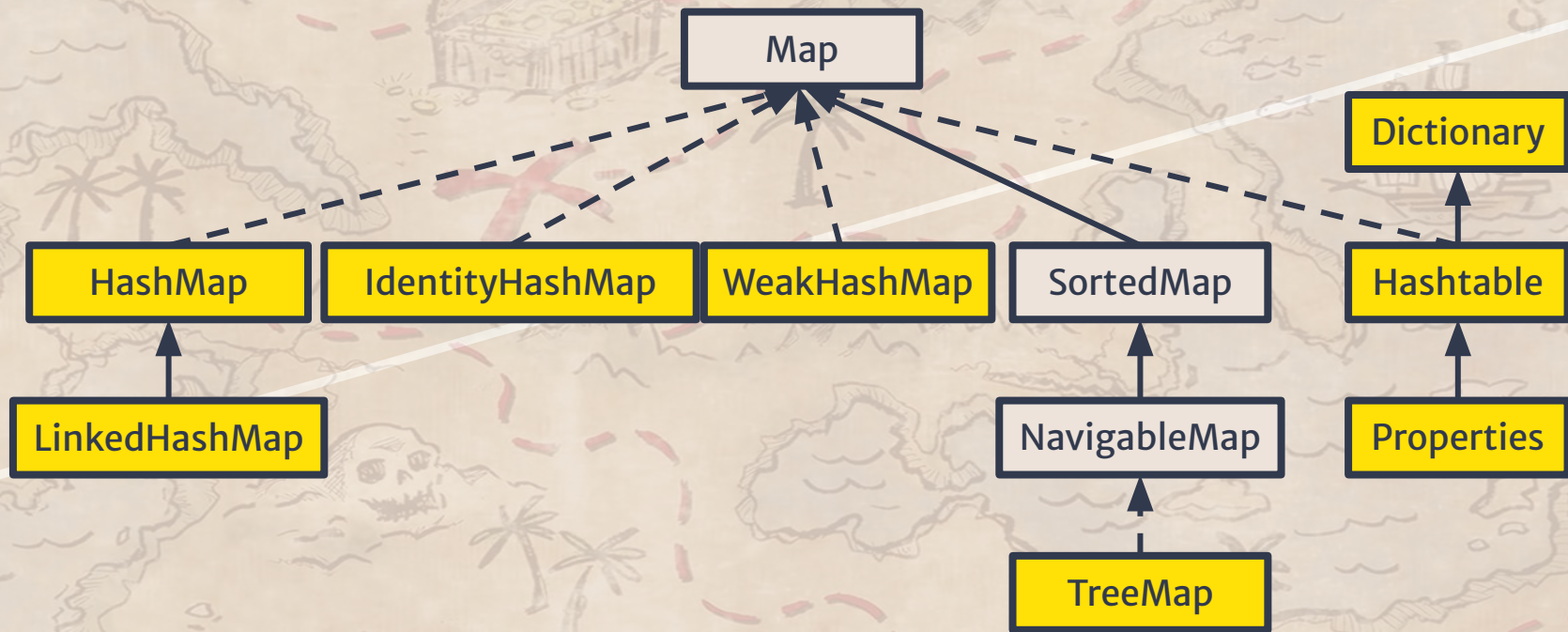


- Controllo parentesi
- Chiamate a funzioni
- Ctrl-z
- Backtracking in un labirinto
- DFS



Map

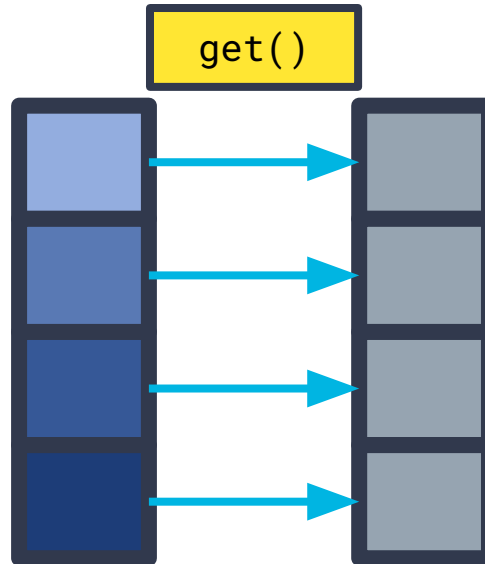
“An object that maps keys to values.”



Cos'è una map



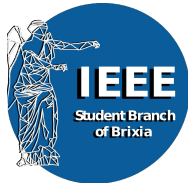
Una map rappresenta una successione di elementi (**chiave, valore**), dove la chiave ha lo scopo di individuare univocamente il valore associato.





- **put(key : K, value : V) : V**
Associa il valore *value* alla chiave *key*, sostituendo il valore precedente se già presente, viene reso il valore precedente se già presente, *null* altrimenti.
- **get(key : K) : V**
Restituisce il valore associato alla chiave *key*, oppure *null* se la chiave non è presente.
- **remove(key : K) : V**
Rimuove la chiave *key*, e il valore associato se già presenti, viene reso il valore precedente se già presente, *null* altrimenti.

Per ulteriori informazioni consultare la documentazione ufficiale Java





Una **funzione hash** è qualsiasi funzione che può essere usata per mappare dati di lunghezza arbitraria in un dato di dimensione fissa.

$$h(s) = \sum_{i=0}^{n-1} s[i] \cdot 31^{n-1-i}$$

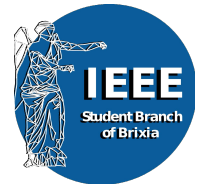
Hashcode String class

Una funzione hash:

- Mappare oggetti uguali in hash uguali;
- Mappa oggetti diversi in hash possibilmente diversi (evitando le collisioni);
- Non è invertibile.

Le migliori funzioni hash sono **MOLTO** complesse e sono usati per la crittografia e anche nel mining delle cryptocurrency.

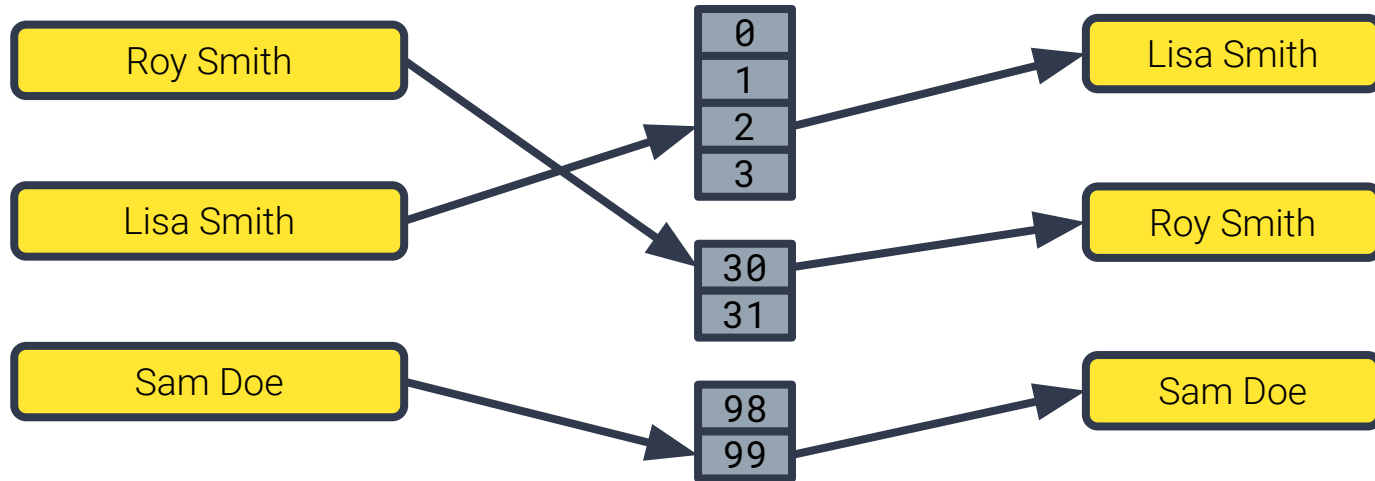
Esempi: SHA-256, MD5, ...



Hash Map



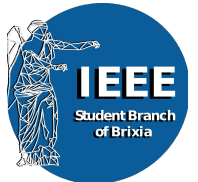
- Molto veloci nella ricerca e nell'inserimento di elementi
- Non garantiscono l'ordinamento degli elementi
- Permette null per chiave e per valore



Tree Map



- Meno veloci rispetto alla HashMap ma garantiscono l'ordinamento degli elementi
- La classe Key deve implementare l'interfaccia Comparable
- $O(\log n)$ per `get()`, `put()`, `remove()` e `containsKey()`



HashMap



```
Map<Integer, String> names = new HashMap<>();
```

```
names.put("Walt".hashCode(), "Walt");  
names.put("Paul".hashCode(), "Dave");  
names.put("Tom".hashCode(), "Adam");  
names.put("Jack".hashCode(), "Jack");  
  
for (Map.Entry<Integer, String> entry : names.entrySet()) {  
    System.out.println(entry.getKey() + " -> " + entry.getValue());  
}
```

```
84274 -> Adam  
2688498 -> Walt  
2300927 -> Jack  
2480232 -> Dave
```

TreeMap



```
Map<Integer, String> names = new TreeMap<>();
```

```
names.put("Walt".hashCode(), "Walt");  
names.put("Paul".hashCode(), "Dave");  
names.put("Tom".hashCode(), "Adam");  
names.put("Jack".hashCode(), "Jack");  
  
for (Map.Entry<Integer, String> entry : names.entrySet()) {  
    System.out.println(entry.getKey() + " -> " + entry.getValue());  
}
```

```
84274 -> Adam  
2300927 -> Jack  
2480232 -> Dave  
2688498 -> Walt
```

Quando usare una map



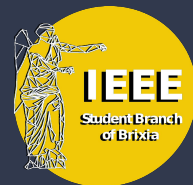
- Dizionario (parola, definizione)
- Lookup table
- Ricerca veloce
- ...





Set

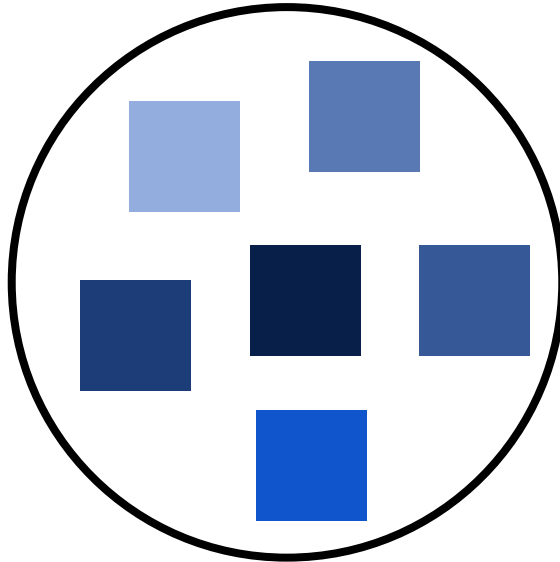
*“A collection that contains
no duplicate elements”*



Cos'è un set



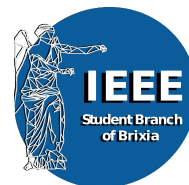
Un set (insieme) è una struttura dati non ordinata che **non** ammette elementi **duplicati**.
È un modello di un insieme matematico.





L'interfaccia Set definisce metodi per effettuare operazioni tra insiemi.

- **`addAll(c : Collection<? Extends E>) : boolean`**
Rappresenta l'operazione di unione di insiemi e ritorna se il set è stato cambiato o meno
- **`retainAll(c : Collection<?>) : boolean`**
Rappresenta l'operazione di intersezione di insiemi e ritorna se il set è stato cambiato o meno
- **`containsAll(c : Collection<?>) : boolean`**
Rappresenta la relazione di sottoinsieme, ritorna true se la collection c è sottoinsieme.



HashSet



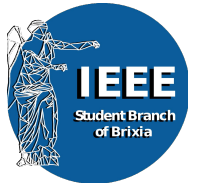
```
Set<Character> letters = new HashSet<>();
```

```
String phrase = "IEEE - Programma Arnaldo";
```

```
for (int i = 0; i < phrase.length(); i++)  
    letters.add(phrase.charAt(i));
```

```
for(char c : letters)  
    System.out.print(c);
```

aAdEgIl-mnoPr



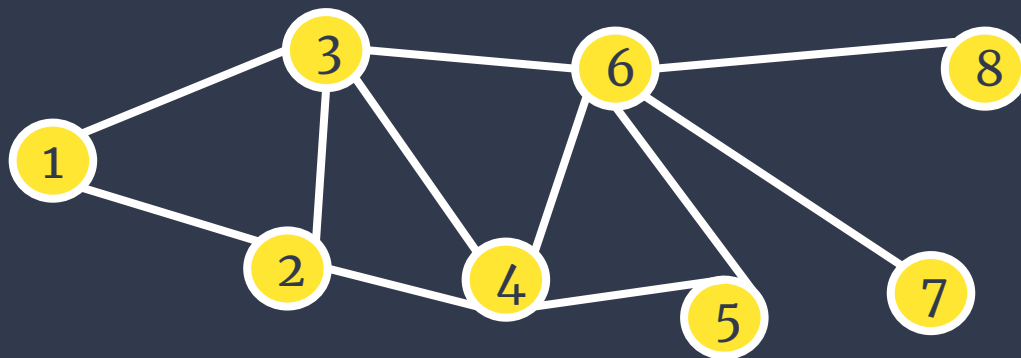
Quando usare un set



- Per estrarre le chiavi da una map (keySet)
- Per rendere una collection con soli elementi distinti
- ...



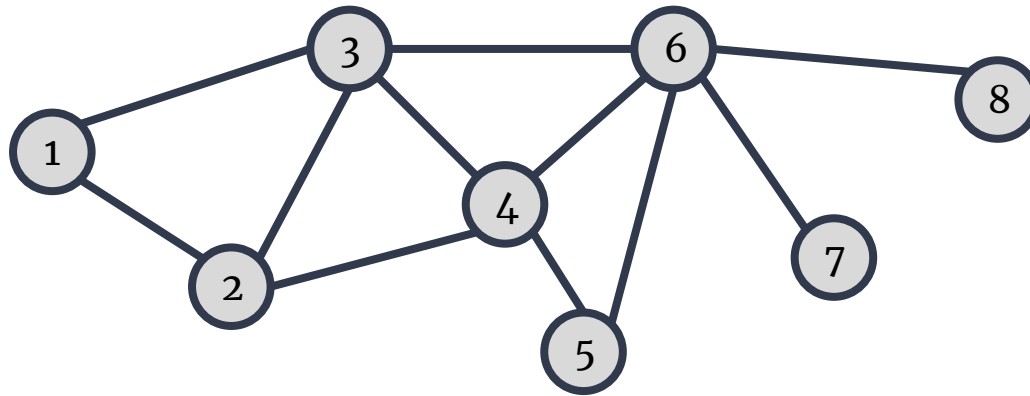
Grafi



Cos'è un grafo



Si dice **Grafo** una coppia $G = (V, E)$ di insiemi, con V insieme dei nodi ed E insieme degli archi, tale che E sia un sottoinsieme del prodotto cartesiano di V con se stesso.

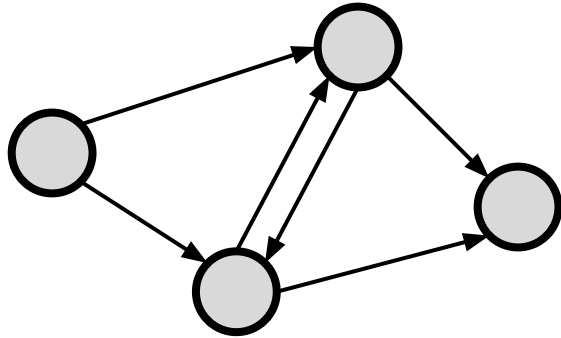


Grafi: classificazione

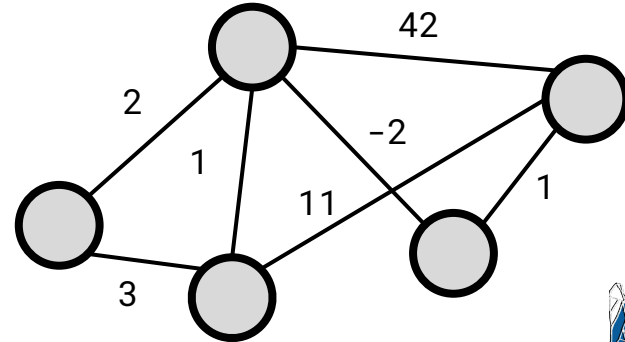


Un grafo si dice **Grafo Pesato** se a ciascun arco è associato un valore numerico, detto peso dell'arco.

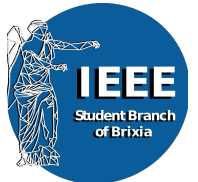
Un grafo si dice **Grafo Orientato** se a ciascun arco è associata una *direzione* (ossia se viene effettuata una distinzione fra la coppia di nodi $\langle A, B \rangle$ e quella $\langle B, A \rangle$).



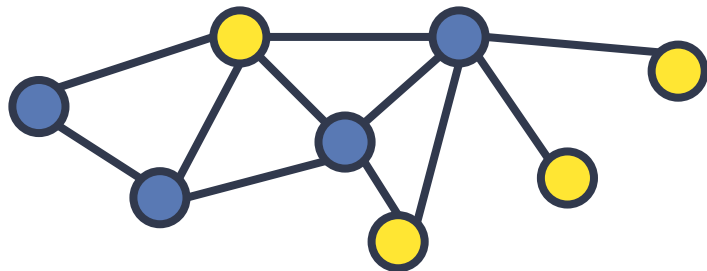
Grafo orientato



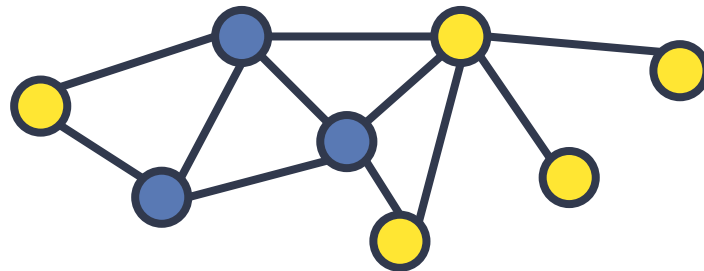
Grafo pesato



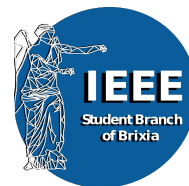
Cicli e cammini



Cammino tra nodi



Ciclo





In Java non esiste una collection che modella un grafo...
Dobbiamo realizzare noi questa struttura.

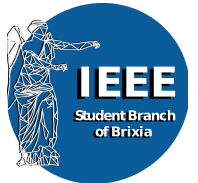
- Tramite una matrice di adiacenza
- Tramite una lista di adiacenza
- Tramite varie classi che rappresentano i vari componenti di un grafo



Quando usare un grafo

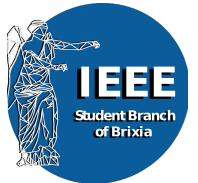


- Cammino più breve tra due nodi
 - Google maps
 - Networking
- PageRank
- In chimica per la rappresentazione di atomi e legami
- Facility location
- Social network
- ...





- [Documentazione ufficiale Java 8](#)
- [Stack and Queues](#)
- [Map](#)
- [Trattazione più completa sui concetti fondamentali dei grafi](#)

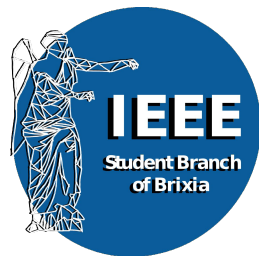


Presentazione realizzata per lo
Student Branch IEEE
dell'Università degli Studi di
Brescia, in occasione del
Programma Arnaldo 2023

*Si prega di non modificare o
distribuire il contenuto di tale
documento senza essere in possesso
dei relativi permessi*

corazzinamarco33@ieee.org
matteo.boniotti@ieee.org
stefano.agnelli@ieee.org
kibo@ieee.org

ieeesb.unibs.it



Grazie per l'attenzione

BUON LAVORO!