Grafi

I grafi sono strutture che rappresentano coppie di oggetti in relazione tra di loro.

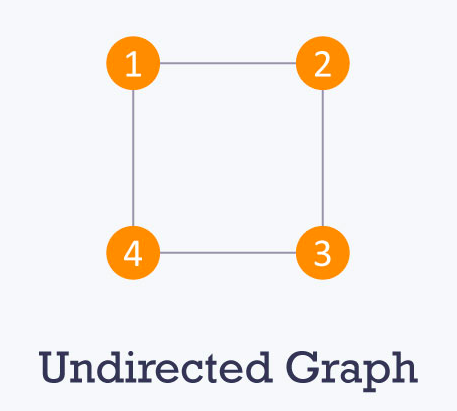
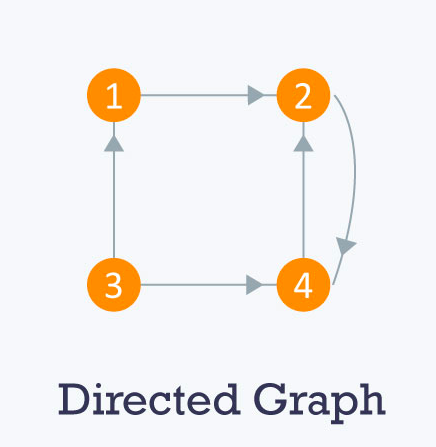
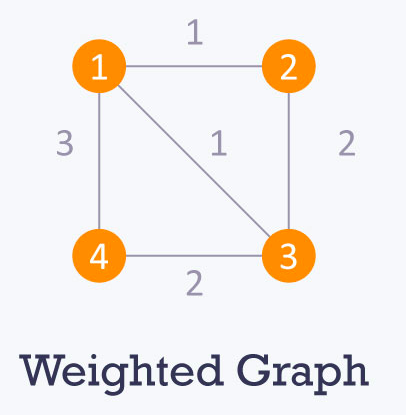
Possono essere visualizzati usando due componenti basi:

* nodi: sono la componente *chiave* in un grafo. Sono entità che vengono messe in relazioni tramite edges.   
  Se un grafo comprende 2 nodi, A e B, e un arco non direzionato tra di loro, allora tra i due nodi vale una **relazione biunivoca**.
* edges (archi): sono le componenti usate per rappresentare le relazioni tra i vari nodi in un grafo.   
  Un arco esprime una relazione univoca o biunivoca tra i nodi.

Tipi di nodi

* Root Node (nodo radice): è il nodo antenato di tutti gli altri nodi nel grafo. Non ha un antenato. Generalmente è il primo nodo del grafo.
* Leaf nodes (nodi foglia): in un grafo, i nodi foglia rappresentano i nodi che non hanno successori, ma hanno soltanto nodi antenati.   
  Possono avere un numero qualsiasi di archi entranti ma nessun arco uscente.

Tipi di grafo

* Undirected (non orientato): un grafo non orientato è un grafo in cui tutti gli archi sono bi-direzionali.
* Directed (orientato): un grafo orientato è un grafo in cui tutti gli archi sono uni-direzionale, quindi vanno soltanto in una direzione
* Weighted (ponderato): in un grafo ponderato/pesato, ad ogni arco è assegnato un peso o costo.   
    
  Esempio: consideriamo un grafo di 4 nodi come nella foto:  
    
    
    
    
    
    
    
    
    
    
    
    
    
    
  Ad ogni arco è assegnato un peso. Se dal nodo 1 voglio arrivare al nodo 3, ho diverse strade possibili:   
    
  1 —> 2 —> 3 (costo 1+2)  
  1 —> 3 (costo 1)  
  1 —> 4 —> 3 (costo 3+2)  
    
  Queste 3 strade portano tutte da 1 a 3, ma con un “costo” diverso tra loro.  
  Notiamo che il secondo percorso è quello con costo minore.
* Cyclic (ciclico): un grafo è ciclico se il grafo comprende un percorso che parte da un vertice (nodo) e finisce allo stesso vertice (nodo).  
  Il percorso è chiamato “ciclo”. Un grafo aciclico è un grafo che non ha cicli.  
    
  Un albero è un grafo aciclico dove due vertici sono connessi da un solo percorso.   
  Un albero è un grafo aciclico che ha N-1 archi dove N è il numero dei nodi.   
  Tuttavia in un grafo ogni nodo può avere uno o più nodi genitori, mentre in un albero si ha esattamente un solo nodo genitore.

Rappresentazione del grafo

Un grafo può essere rappresentato in diversi modi, le più comuni sono:

**Adjacency Matrix (Matrice delle adiacenze):**

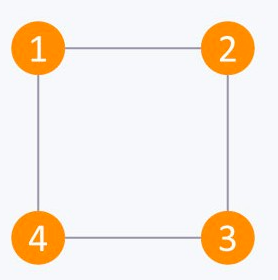
Una matrice delle adiacenze è una matrice A binaria VxV.

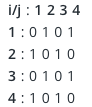
L’elemento è 1 se è presente un arco che collega il nodo i col nodo j mentre è 0 se non è presente.

***Note:*** *una matrice binaria è una matrice dove le celle possono avere solo valore 0 o 1.*

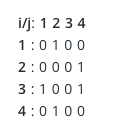
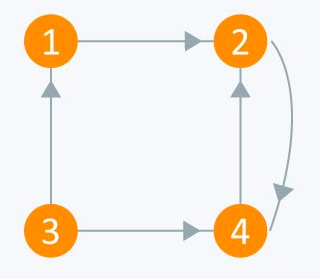
La matrice delle adiacenze può esser modifica dal grafo pesato dove invece di inserire o 0 o 1, viene inserito il “peso” dell’arco.

In un grafo orientato, se = 1, allora anche = 1, mentre in un grafo orientato questo non è sempre valido.

La **matrice delle adiacenze** impiega tempo (O(1)) per determinare se un arco è presente tra due nodi.

Lo spazio di complessità è .

Un esempio di matrice delle adiacenze di un grafo:

Mentre considerando il caso di Grafo orientato avremo:

**Adjacency List (Lista delle adiacenze):**

Un altro modo per rappresentare i grafi è mediante l’uso delle liste di adiacenza.

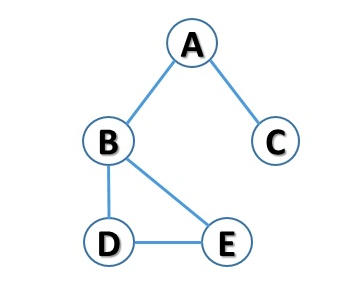
Una lista di adiacenza è un array A di liste separate.

Ogni elemento dell’array è una lista, che contiene tutti i nodi che sono adiacenti al nodo “i”.

Lo spazio di complessità è poiché in una lista delle adiacenze l’informazione è immagazzinata solo per gli archi esistenti del grafo.

***Note:*** *una matrice sparsa è una matrice dove la maggior parte degli elementi è zero mentre una matrice densa è una matrice dove la maggior parte degli elementi non è zero.*

Creazione di un Grafo su PyCharm (GraphsProject)

Link: https://www.meccanismocomplesso.org/programming-graphs-in-python-part-1-grafi/

Riferimenti:

<https://www.hackerearth.com/practice/algorithms/graphs/graph-representation/tutorial/>