Grafi

Usati per rappresentare dati con relazioni più complesse tra di loro

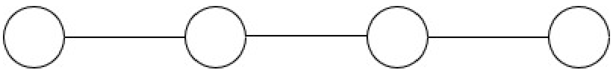
G = (V, E) → DIRETTO/ ORIENTATO → NON ORIENTATO

V nodi/vertici

$E \subseteq V \times V$ archi

Ognuno degli archi è formato da una coppia di Nodi.

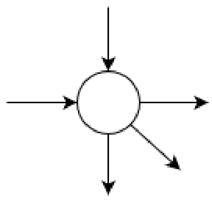
Non Orientati



Catena

- **Grado**

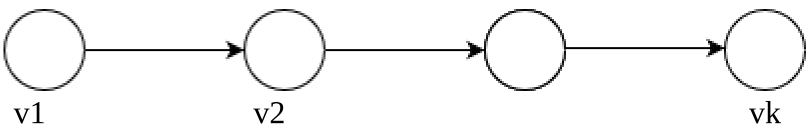
Grado: 5



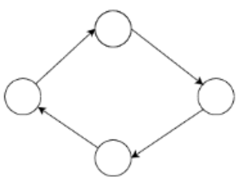
In-Degree : 2

Out-degree: 3

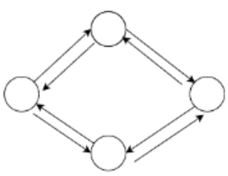
Orientati



Cammino <v1, v2, v3, …, vk>

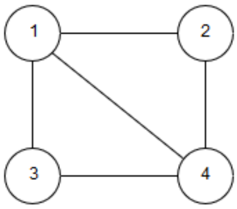


$$n° \; archi : \frac{(n-1)n}{2}$$



$$n° \; archi: \; (n-1)n$$

Rappresentazione

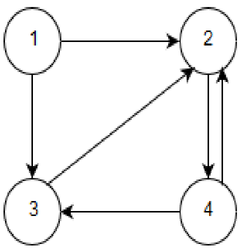


- **Matrice di adiacenza**

	1	**2**	**3**	**4**
1	0	1	1	0
2	1	0	1	1
3	1	1	0	1
4	0	1	1	0

$$a[i;j] = \begin{cases} 1, & se\ (i;j) \in E \\ & 0\ altrimenti \end{cases}$$

Caso grafo diretto



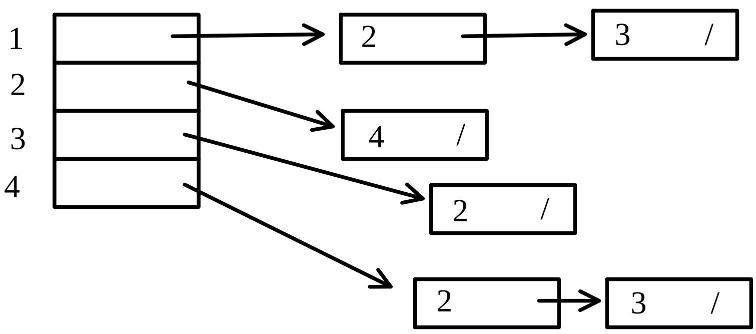
	1	**2**	**3**	**4**
1	0	1	1	0
2	0	0	0	1
3	0	1	0	0
4	0	1	1	0

Check_arco -> O(1) *controllo l'esistenza di un arco*
Lista_vicini $\rightarrow O(n)\ stila\ una\ lista\ dei\ vicini\ al\ nodo$

- **Liste di Adiacenza**

Lista in ogni nodo che contiene i vicini di quel nodo

Caso grafo diretto:



Spazio: $O(n\ +\ m)$

$Check\_Arco\ \rightarrow O(\max\_out\_degree)$

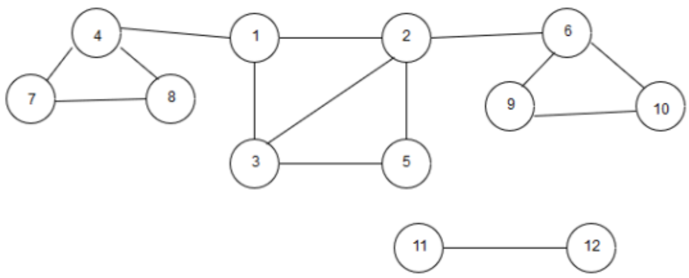
$Lista\_Vicini\ \rightarrow O(\max\_out\_degree)$

Visite

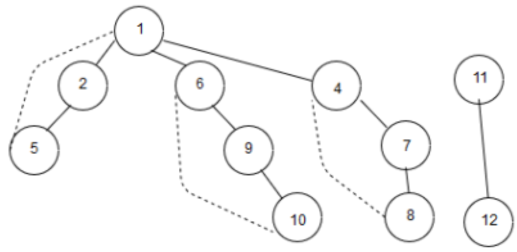
Di due tipi:

- DFS
- BFS

Grafi Indiretti



Esplorazione



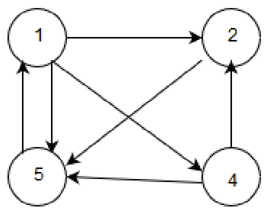
Per ogni nodo nell'insieme dei nodi

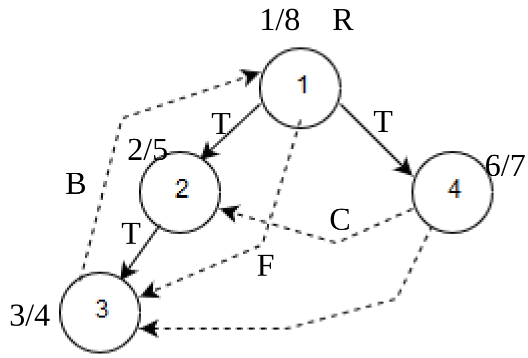
```
DFS(G)
    for all v ∈ V
        Visited[v] := False
    for  all v ∈ V
        if Visited[v] = False
            then DFS_VISIT(G, V)
```

```
DFS_VISIT(G, v)
    Visisted[v] := True
    Caso previsita analisi di V
    for all (v, u) ∈ E
        if Visited[u] = False
            then DFS_VISIT(G, u)
    Caso postvisita analisi V
```

Il numero totale dei gradi è il doppio del numero di archi.

DFS-Grafi Diretti



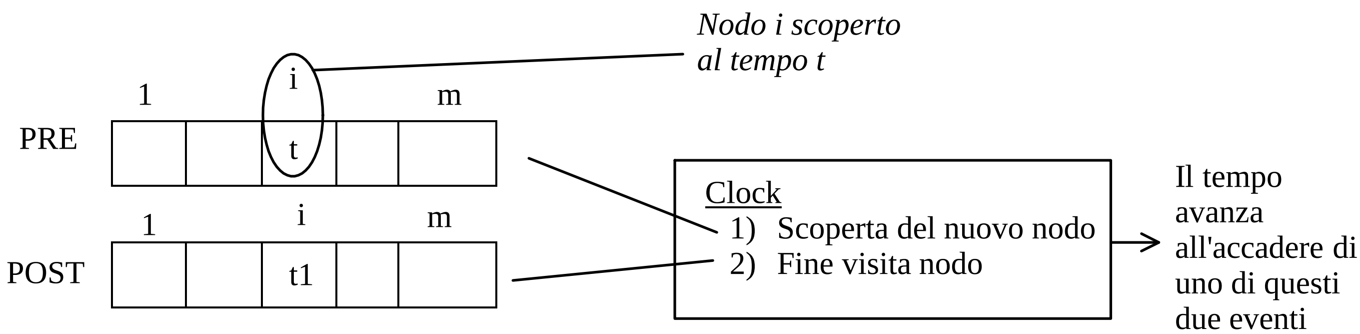


- **Tree(linee solide)**
  Mi permettono di scoprire un nuovo nodo durante la visita e di ottenere un albero di copertura.

- **Back** - verso antenato

- **Forward** - Discendente

- **Cross**
  gli altri: collegano nodi che non hanno tra loro un rapporto antenato/discendente o viceversa



Accanto ad ogni nodo troviamo una coppia di numeri
- I: quando il nodo è stato scoperto
- II: quando si finisce di visitarlo

Con l'ausilio di questi tempi posso classificare gli archi nel corso della mia visita

Durante la visita:

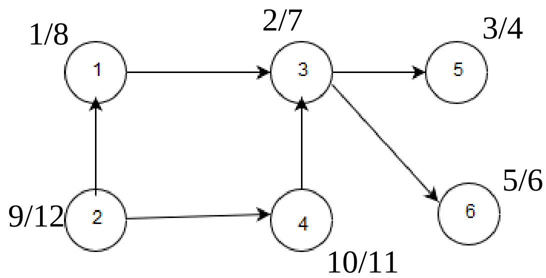
```
TREE - se pre[v] = 0

BACK - se pre[u] > pre[v]
               e post[v] = 0

FORWARD - se pre[u] < pre[v]
              e post[v] > 0

CROSS - se post[v] < pre[u]
```

DAG(Direct Acyclic Graph)



Sto cercando un <u>ordinamento tra i nodi</u>.

Se $(u, v) \in E$
$\Rightarrow u$ viene prima nell'ordinamento di v

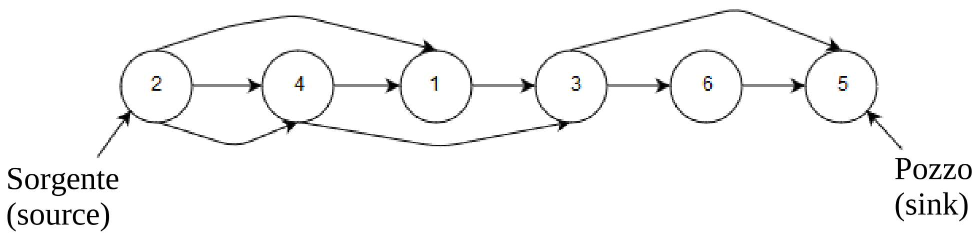
ORDINAMENTO TOPOLOGICO
linearizzazione

<u>A fine visita</u>

```
TREE - post[u] > post[v]

FORWARD - post[v] > post[u]

CROSS - post[u] > post[u]
```



Ordine di post visita dal più grande al più piccolo

<u>Componenti fortemente connesse</u> $\longrightarrow$ Un sottografo
G' = (V', E') di G = (V, E)

$V' \subseteq V$
$E' \subseteq E$

1) $\forall$ coppia u, v $\in V$; $\exists$ un cammino da u a v in G'
2) Massimale