Corso di laurea in Informatica
Progettazione d'algoritmi

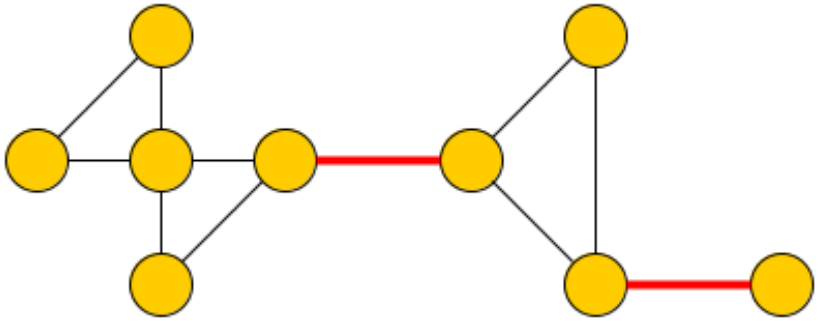
I grafi 6

Angelo Monti

SAPIENZA
UNIVERSITÀ DI ROMA

In un grafo la connessione è una proprietà che può andar persa con la perdita di un arco.

- Un arco la cui eliminazione disconnette il grafo è detto ponte.

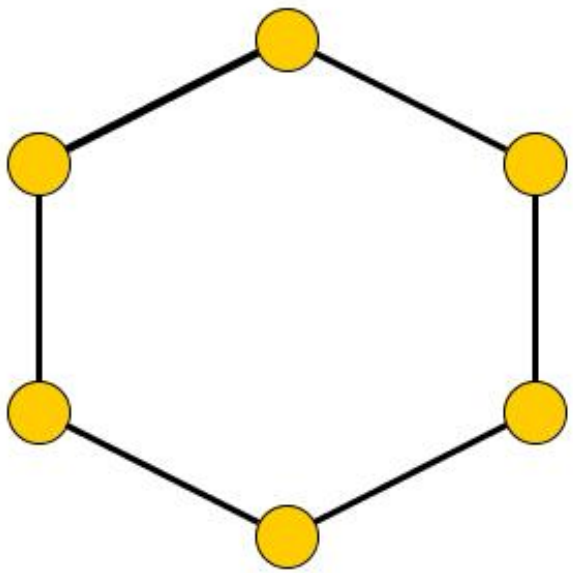


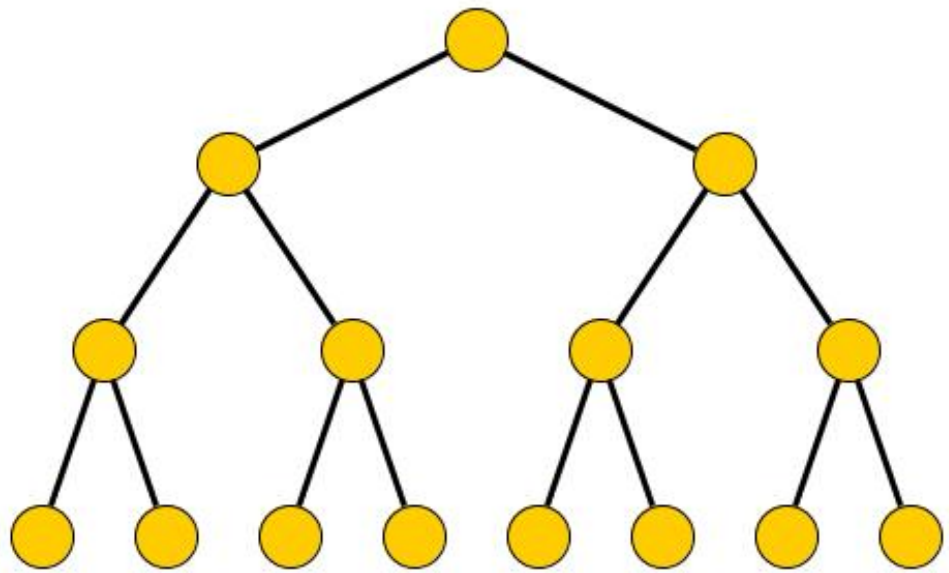
I ponti rappresentano criticità del grafo ed è quindi utile identificarli.

Problema: dato un grafo connesso G determinare l'insieme di ponti del grafo.

Problema: dato un grafo connesso G determinare l'insieme di ponti del grafo.

Il grafo può non avere neanche un ponte così come avere tutti i suoi archi come ponti





A sinistra un esempio di grafo senza ponti: il ciclo.
A destra un esempio di grafo in cui tutti gli archi sono ponti: l'albero.

Una prima soluzione basata sulla ricerca esaustiva:

- **Prova per ogni arco del grafo se questo è ponte o meno.**

Verificare se un arco $\{a, b\}$ è un ponte per G richiede tempo $O(m)$

Basta ad esempio eliminare l'arco $\{a, b\}$ da G e, con una visita DFS, controllare se b è raggiungibile a partire da a.

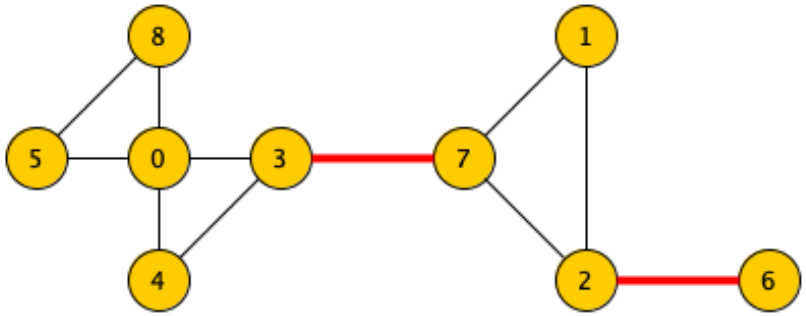
La complessità di questo algoritmo è $m \cdot O(m) = O(m^2) = O(n^4)$

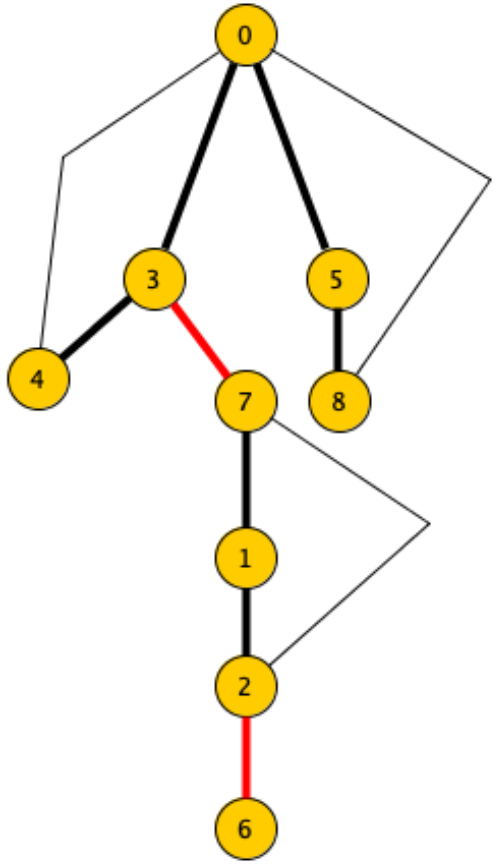
Vedremo ora che il problema è risolvibile in tempo lineare nella dimensione del grafo $O(m)$

IDEA: usare un'unica visita DFS modificata

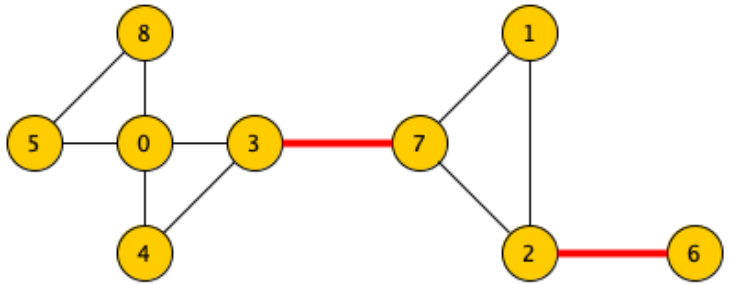
Nota che:

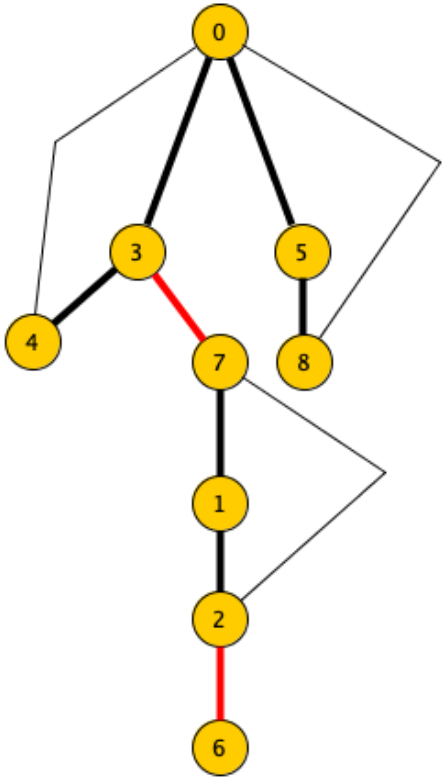
- **I ponti vanno ricercati unicamente tra gli $n-1$ archi dell'albero DFS.**
  - Infatti un arco non presente nell'albero DFS non può essere ponte (se lo elimino gli archi dell'albero DFS continuano a garantire la connessione)





Serve una caratterizzazione facile da calcolare che permetta, durante la visita, di determinare se un arco dell'albero è un ponte o meno.





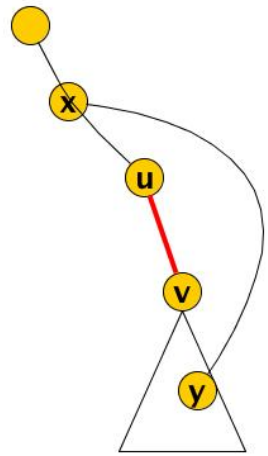
Nota: gli archi dell'albero che non sono ponti risultano **coperti** dagli archi che non sono stati attraversati

Proprietà: Sia $\{u, v\}$ un arco dell'albero DFS con u padre di v.
L'arco $\{u, v\}$ è un ponte se e solo se non ci sono archi tra i nodi del sottoalbero radicato in v e il nodo u o nodi antenati di u.

Prova:

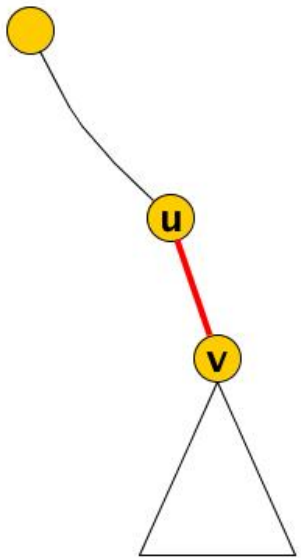
$\Longrightarrow$

Per assurdo: sia $\{x, y\}$ un arco tra un antenato di u e un discendente di v. Dopo l'eliminazione dell'arco $\{u, v\}$ tutti i nodi dell'albero restano comunque connessi grazie all'arco $\{x, y\}$.

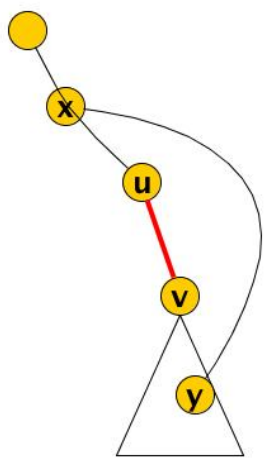


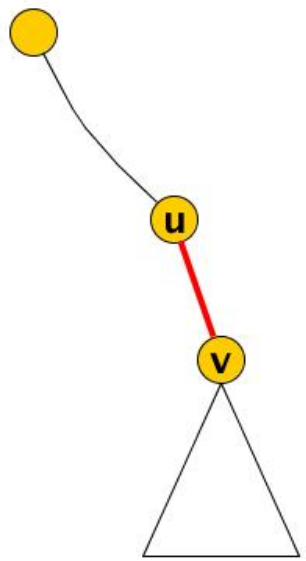
$\Longleftarrow$

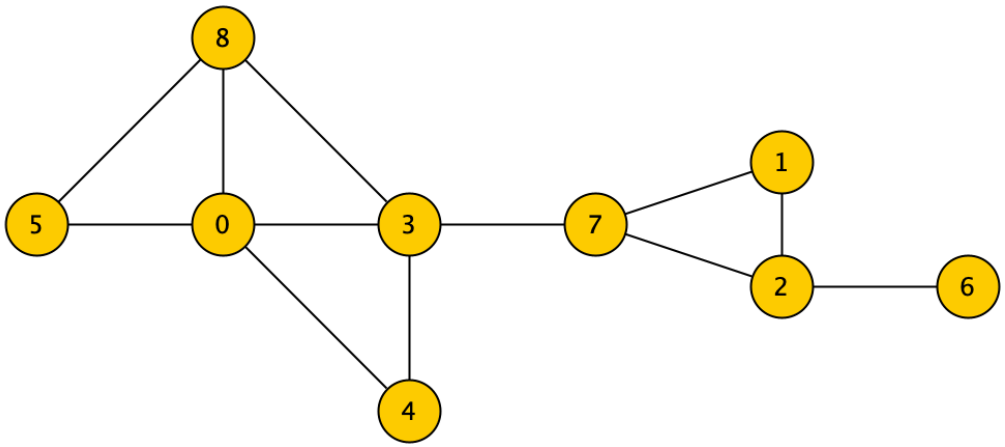
l'eliminazione dell'arco $\{u, v\}$ disconnette i nodi dell'albero radicato in v dal resto del grafo (ricorda che tutti gli archi non nell'albero che partono da nodi nel sottoalbero radicato in v andrebbero verso u o antenati di u) .

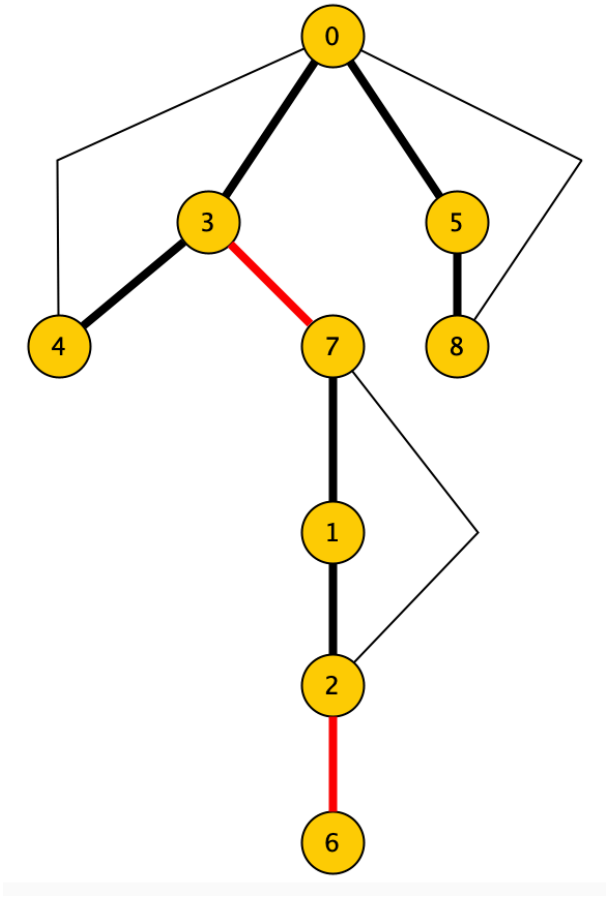


Nel momento in cui l'arco $\{u, v\}$ viene "attraversato" per raggiungere il nodo v da visitare il nodo u non ha informazioni per decidere se dal sottoalbero radicato in v partono archi verso suoi antenati (quindi scoprire se $\{u, v\}$ è ponte o meno). Ma gli sarà possibile scoprirlo al termine della visita di v se riceve da *v* la giusta informazione.









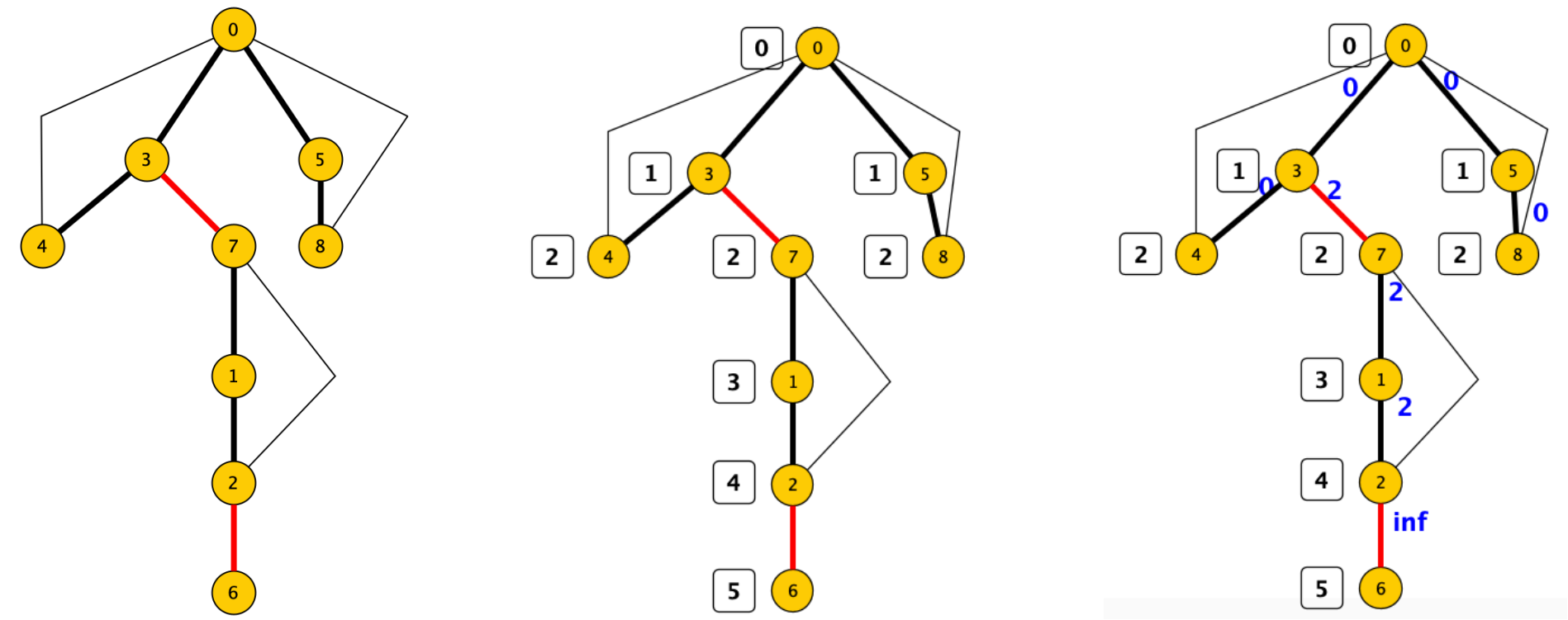
$\{2,6\}$ **è ponte**

$\{3,7\}$ **è ponte**

Idea: Durante la visita ogni nodo x calcola la sua altezza nell'albero DFS e a fine visita restituisce l'altezza minima raggiungibile da nodi del sottoalbero radicato in x tramite gli archi non presenti nell'albero DFS. Se non ci sono archi di questo tipo allora viene restituito $+\infty$.

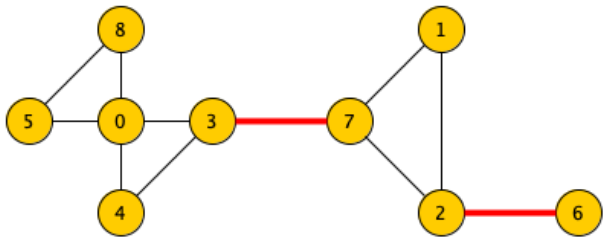
Il nodo u confrontando la sua altezza con il valore che gli viene segnalato dai figlio v è in grado di decidere se l'arco $\{u, v\}$ è ponte o meno.

$\{u, v\}$ è ponte se e solo se l'altezza di u è strettamente minore del valore restituito da v.



4 < $+\infty$ e $\{2,6\}$ è ponte $1 < 2$ **e $\{3,7\}$ è ponte**

```python
def TrovaPonti(G):
    '''restituisce la lista dei ponti di G'''
    Altezza = [-1]*len(G)
    Ponti = []
    DFSModificata(0, 0, G, Altezza, Ponti)
    return Ponti
```




```python
G=[
[3,4,5,8],
[2,7],
[1,6,7],
[0,4,7],
[0,3],
[0,8],
[2],
[1,2,3],
[0,5]
]
```

```python
def DFSModificata(x, t, G, Altezza, Ponti):
    from math import inf
    Altezza[x]  = t
    ret = inf
    for y in G[x]:
        if Altezza[y] == -1:
            a = DFSModificata(y, t+1, G, Altezza, Ponti)
            if t < a:
                Ponti.append((x,y))
            ret = min(ret, a)
        elif Altezza[y] != t-1:
                ret = min(ret, Altezza[y])
    return ret
```

```
>>> TrovaPonti(G)
[(2, 6), (3, 7)]
```

La complessità della procedura è $O(n + m)$

Corso di laurea in Informatica
Introduzione agli Algoritmi

Esercizi per casa

SAPIENZA
UNIVERSITÀ DI ROMA

un vertice la cui rimozione è in grado di sconnettere il grafo è detto **punto di articolazione.**

- Dimostrare o confutare che esiste un grafo G con la proprietà di avere un arco la cui rimozione fa aumentare di 2 il numero dei suoi punti di articolazione e di 3 il numero dei suoi ponti.

- Dimostrare o confutare che non esistono grafi i cui nodi sono tutti punti di articolazione.

- Dimostrare o confutare che in un grafo connesso G esiste sempre un ordinamento dei suoi nodi tale che la rimozione dei nodi in quell'ordine mantiene sempre il grafo connesso. In caso questa sequenza esista progettare un algoritmo che la fornisce in tempo $O(m)$.

Sia G un grafo connesso con m archi.

- dimostrare che il nodo radice dell'albero DFS di G è un punto di articolazione di G se e solo se ha almeno due figli.

- dimostrare che un nodo x dell'albero DFS di G diverso dalla radice è un punto di articolazione di G se e solo se non esiste in G un arco (a, b) tale che nell'albero DFS di G a è un discendente di x e b un antenato di b.

- Progettare un algoritmo che dato il grafo G restituisce la lista dei suoi nodi di articolazione in tempo $O(m)$.