Corso di laurea in Informatica
Progettazione d'algoritmi
Didattica blended

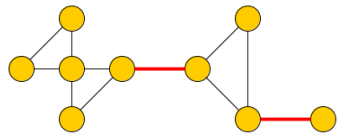
I grafi 5

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.

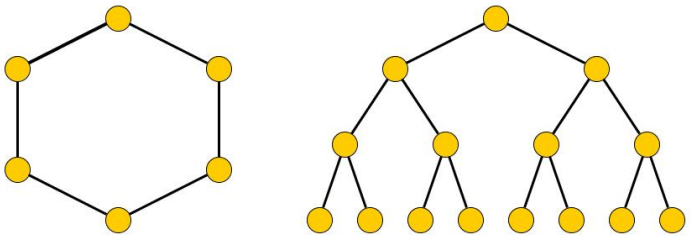


I ponti rappresentano criticità del grafo ed è quindi utile identificarli.

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.

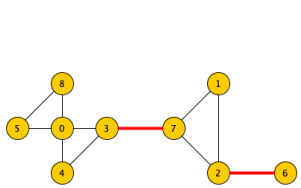
La complessità di questo algoritmo è $m \cdot O(m) = O(m^2) = O(n^4)$

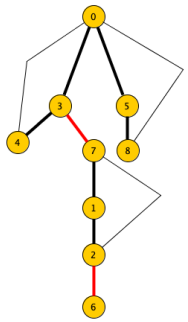
Vedremo ora che il problema è risolvibile in tempo lineare nella dimensione del grafo $O(m)$

IDEA: usare un'unica visita DFS modificata

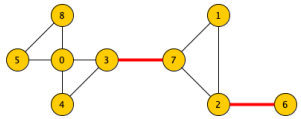
- **I ponti vanno ricercati unicamente tra gli $n-1$ archi dell'albero DFS.**

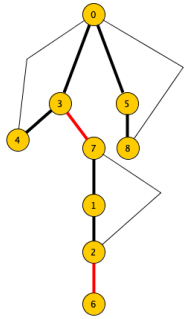
  - **Infatti un arco non presente nell'albero DFS non può essere ponte (se lo elimino gli archi dell'albero DFS continuano a garantire la connessione)**





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





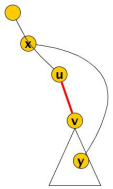
Nota: **gli archi dell'albero che non sono ponti risultano coperti dagli archi all'indietro che non sono stati attraversati**

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:

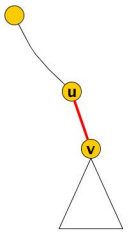
$\Longrightarrow$

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 $\{x,y\}$.

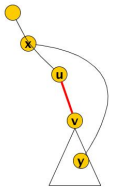


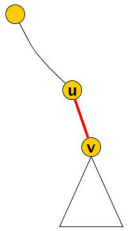
$\Longleftarrow$

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) .

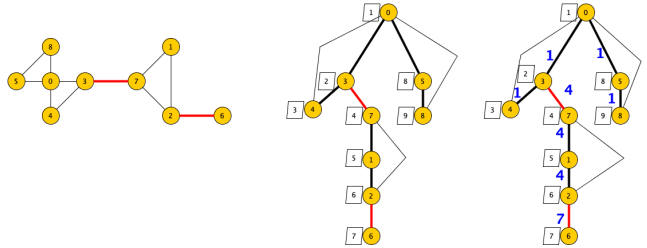


Nel momento in cui l'arco $\{u,v\}$ viene "attraversato" per raggiungere il nodo v da visitare il nodo u non ha informazioni per decidere se dal sottoalbero radicato in v partono archi verso suoi antenati (quindi scoprire se $\{u,v\}$ è ponte o meno). Ma gli sarà possibile scoprirlo al termine della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





Idea: Durante la visita ogni nodo calcola il suo tempo di visita e a fine visita restituisce il tempo di visita minimo per gli estremi degli archi all'indietro che partono dai nodi del suo sottoalbero (il tempo di visita del nodo stesso se non ci sono archi all'indietro che partono dal sottoalbero)



Il nodo u confrontando il suo tempo di visita con il tempo di visita minimo che gli viene segnalato dal figlio v è in grado di decidere se l'arco $\{u, v\}$ è ponte o meno.

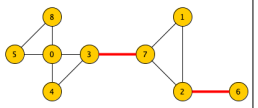
$\{u, v\}$ è ponte se e solo se il tempo di visita di u è strettamente minore del tempo di visita minimo restituito da v.

$6 < 7$ e $\{2,6\}$ **è ponte**

$2 < 4$ e $\{3,7\}$ **è ponte**


```
def ponti(G):
    '''restituisce i ponti di un grafo connesso G'''
    def DFSr(x,G,Tempo,padre,archi_ponte):
        nonlocal t
        t+=1
        ret=Tempo[x]=t
        for y in G[x]:
            if Tempo[y]==0:
                a=DFSr(y,G,Tempo,x,archi_ponte)
                if Tempo[x]<a: archi_ponte.add((x,y))
                ret=min(ret, a)
            elif y!=padre:
                ret = min( ret, Tempo[y])
        return ret
    #
    Tempo=[0 for _  in  G]
    archi_ponte=set()
    t=0
    DFSr(0,G,Tempo,0,archi_ponte)
    return archi_ponte

>>> G={0:[3,4,5,8],1:[2,7],2:[1,6,7],3:[0,4,7],4:[0,3],
       5:[0,8],6:[2],7:[1,2,3],8:[0,5]}
{(3, 7), (2, 6)}
```



ponti=$\{(3,7), (2,6)\}$

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