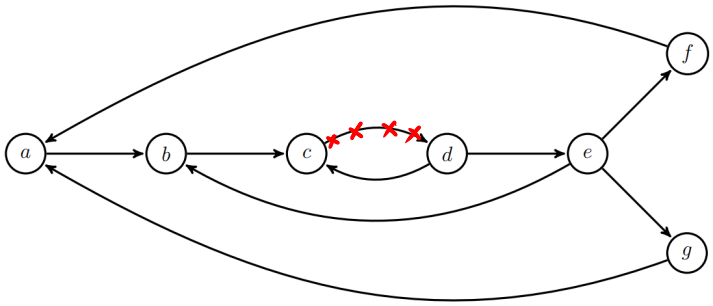
Esercizio 1. In un grafo non diretto e connesso $G = (V, E)$ un vertice v si dice di *articolazione* (*cutvertex* in inglese) se $G - v$, il grafo ottenuto da G rimuovendo v e tutti gli archi ad esso incidenti, non è connesso. Modificare l'algoritmo della ricerca in profondità in maniera da poter ottenere tutti e soli i vertici di articolazione di un grafo connesso; è possibile fare questa modifica in modo che il controllo avvenga in $\Theta(|V| + |E|)$?

Un nodo u e' un articolazione se $\exists v, k$ t.c. $u \sim v \wedge u \sim k \wedge \neg\exists$ un cammino $v \rightsquigarrow k$ che non faccia uso di u.

```
Search_Art ( G:grafo){
    art : Set
    for each u∈V(G){
        For each x∈u.adj{
            for each y∈u.adj{
                if (x≠y){
                    if(x∉ DFS(G\{u},y)){
                        art.add(u)
                        break
                    }
                }
            }
        }
    }
return art
}
```

Questa versione e' inefficente, possiamo considerare l'albero di ricerca di una DFS ed i sui archi all'indietro.

Esercizio 4 (I. Salvo)**.** Sia G il grafo raffigurato in figura. Determinare il minimo numero di archi che devono essere eliminati da G affinchè G ammetta ordinamenti topologici. Una volta rimosso questo insime minimo di archi, determinare tutti gli ordinamenti topologici di G.



Possibili ordinamenti topologici:

$O_1 = \{c, b, a, f, g, e, d\}$

$O_2 = \{c, b, a, g, f, e, d\}$

Esercizio 6 (I. Salvo)**.** Descrivere in pseudo-codice un algoritmo che, dato un grafo non diretto G, descrivere un algoritmo che ne orienta gli archi in modo da creare un grafo G' diretto e aciclico. Questo algoritmo deve avere tempo di esecuzione $\Theta(n+m)$.

L'idea e' quella di fare un DFS impostando il verso degli archi come il verso di visita, quando si considera un nodo gia' visitato, si cambia di verso.

```
VARIABILI GLOBALI
ED={ }
Vis[n]={0,0,0...0}
DFS_archi(G:grafo, x:vertice){
    Vis[x]=1
    For each(y~x){
        if(Vis[y]==1){
            ED.add((y,x))
        }else{
            ED.add((x,y))
            DFS_archi(G,y)
        }
    }
}
```

ED conterra' gli archi del nuovo grafo.