Recitation 12   1 April 2008

Note Title 4/1/2009

- Graph representation in python
- Topological sort
- Articulation points

Topological sort

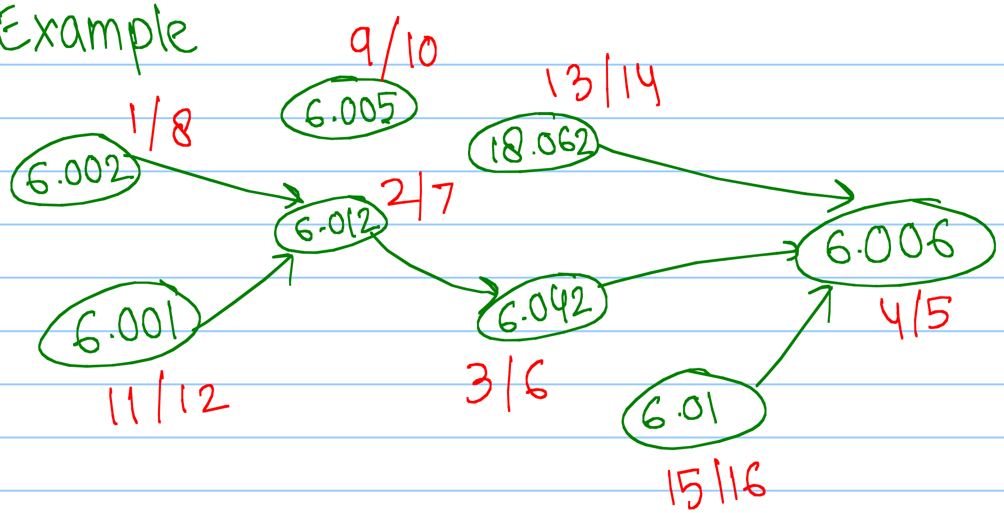
→ a linear ordering of all vertices of $G=(V,E)$

if $(u,v) \in E(G)$, then u appears before v in the ordering.

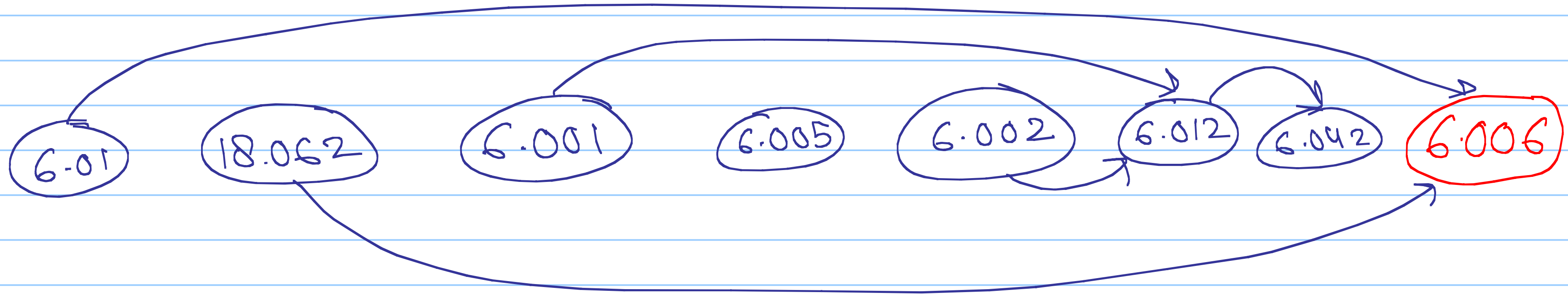
```
TOPOLOGICAL-SORT(G)
1. Call DFS(G) to compute f[v] $\forall v \in V(G)$        $\Theta(E+V)$
                                                          (L : linked list)
2. As each vertex finishes, insert in front of L           $O(1)$
3. return L
```

$\Theta(V+E)$ time

Example



(Random!)



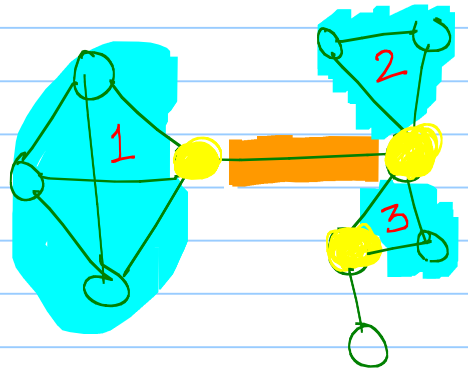
Articulation Problem (CLRS Problem 22.2)

$G=(V,E)$ connected, undirected graph

articulation point → vertex whose removal disconnects G

bridge → edge whose removal disconnects G

biconnected component → maximal set of edges, any two edges in the set lie on a common simple cycle.



a) Prove that root of G_π is an articulation point of G iff it has at least two children.

$\Rightarrow$ if root is articulation point, then it has at least two children

Consider root r with only one children.

Case 1: there is a back edge from a node in subtree pointing back to r
Case 2: r has no back edge pointing back to itself

Case 1, because of backedge, a simple cycle is formed involving r. r belongs to only one biconnected component. Remaring r doesn't disconnect G.

Case 2, removing r only removes one edge, can't disconnect G.

$\Leftarrow$ If root has at least two children, then it is an articulation point.

If root has two or more children, and there can not be any cross edges accross subtrees of those children (DFS), therefore the edges in the subtrees can't lie together in same simple cycle.
remaving r would disconnect these two (or more) components.

b) Let v be a non-root vertex of G_π. Prove that v is an articulation point of G iff v has a child s s.t. there is no back edge from s or any descendant of s to a proper ancestor of v.

$\Rightarrow$ if v is an articulation point, then there is no back edge from s or any descendant of s to a proper ancestor of v.

We can prove this by contradiction. Assume all subtrees rooted at child of v have back edges to a proper ancestor of v. If we remove v, every subtree of v is still reachable through back edges, G is still connected. Contradiction!

$\Leftarrow$ if $\exists$ a subtree rooted at child of v that has no back edge to a proper ancestor of v, then v is an articulation point.

→ No back edges from subtree to any ancestor of v, there can only be tree edges. Removing v will remove the tree edge and disconnect that subtree from root. Hence, v is an articulation point.

c) $$low[v] = \min \begin{cases} d[v] \\ d[w] : (v,w) \text{ is a back edge for some descendant of } v \end{cases}$$

compute low[v] for all $v \in V(G)$ in $O(E)$ time.

```
visit (v,u)        /* visit v from u */
time = time + 1
 d[v] = time
 low[v] = d[v]

 for all vertex w ≠ u and (w,v) ∈ E(G)

        if d[w] = 0 then
            visit (w,v)
            low[v] = min( low[v], low[w] )
       else
            low[v] = min( low[v], d[w] )


initially call visit(r,0)
```

d) Show how to compute all articulation points in O(E) time.

v is an articulation point
iff low[w] ≤ v for some child w of v

```
visit(v,u)
 time = time+1
 d[v] = time
 low[v] = d[v]

 for all vertex w ≠ u , (w,v) ∈ E(G)
    if d[w] = 0
       visit(w,v)
       low[v] = min(low[v], low[w])
       if (d[v] = 1 and d[w] ≠ 2)        /* root */
            print v is articulation point
       if (d[v] ≠ 1 and low[w] ≥ d[v])
            print v is an articulation point
   else
      low[v] = min( low[v], d[w])
```

e) Prove that e is a bridge iff it does not lie on any simple cycle.

$\Rightarrow$ if e is a bridge, it does not lie on any simple cycle.
otherwise there is an alternate path available after removing e, contradictio

$\Leftarrow$ if e does not lie on any simple cycle, then it is a bridge.

Removing e disconnects the two components as it is the only connecting link between them.

f) Show how to compute all bridges of G in O(E) time.

bridge either connects two articulation points or a leaf vertex in G
(vertex with one edge)

For all $e \in E(G)$, $e=(u,v)$
    if u & v are articulation points, or degree(u) = 1 or degree(v) = 1
        print e is a bridge.

g) Biconnected components partition non-bridge edges of G.
(no non-bridge edge can connect two biconnected components, otherwise same component, contradiction).

h)

```
visit(v,u)
  low[v]=d[v]=time=time+1
  for all $w \neq u$, $(w,v) \in E(G)$
     if d[w]<d[v] add (w,v) to stack
     if d[w]=0
         visit(w,v)
         low[v] = min(low[v],low[w])
         if low[w] > d[v] then
            pop off edges from stack until edge (v,w)  /* edges form a
                                                          biconnected
                                                          component */
     else
         low[w]= min (low[v], d[w])
```