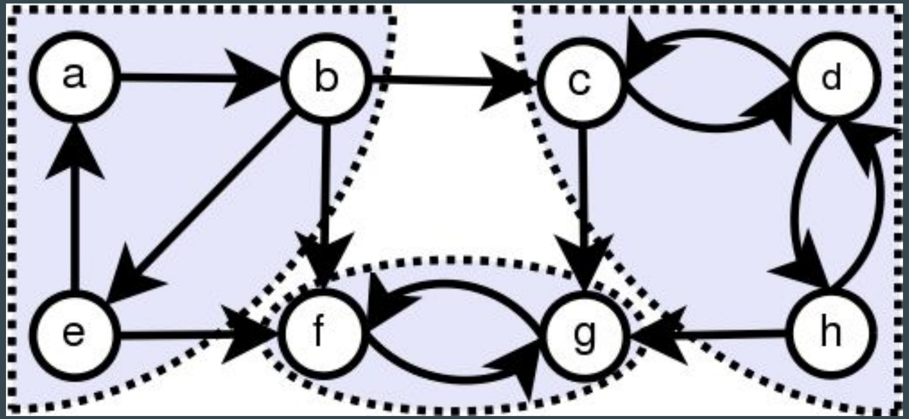
CSE525 Lec14
Graph: DFS algorithms

•••

Debajyoti Bera (M2~~0~~1)

~~https://sites.google.com/a/iiitd.ac.in/cse525-m20~~

Kosaraju ('78) Sharir ('81) SCC



Source SCC = component with no incoming edge

Sink SCC = component with no outgoing edge

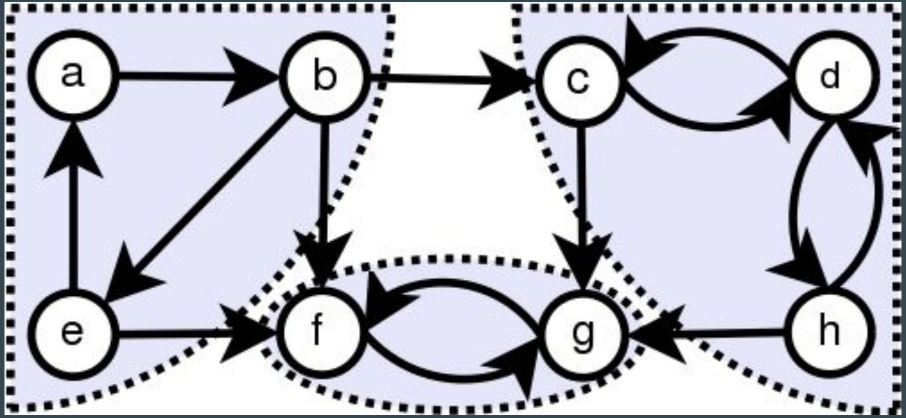
Component graph is acyclic.

Proof:
Let there be cycle, say among some of the components.
Without loss of generality, let the cycle be among components C1, C2, C3, ... Ck.

Let u be some vertex in C1. There is an edge from some vertex in C1, say u1, to some vertex in C2. Since every vertex (including u1) in C1 is reachable from u, and u2 is reachable from u1, therefore, u2 is reachable from u. Since every vertex in C2 is reachable from u2, therefore, every vertex in C2 is reachable from u. There is an edge from some vertex in C2 to some vertex in C3.
Applying the same argument as above we get that every vertex in C3 is reachable from u. Continuing this for all the components in C4, C5, ..., we get that all the vertices in Ck is reachable from u.

Let uk from Ck have an edge to some w in C1. So, u has a path to uk. Furthermore, uk has path to w and w has to path to u => uk has a path to u. Thus, u and uk have a path to each other.
So uk must belong to SCC(u) which contradicts the fact that SCC(u) is different from SCC(uk).

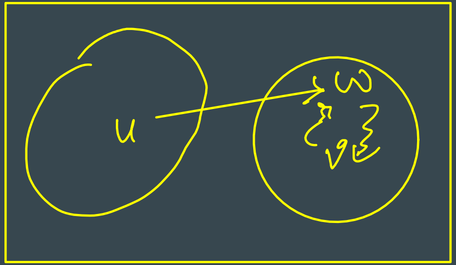
Kosaraju ('78) Sharir ('81) SCC

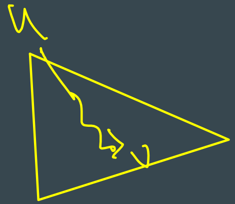


post(v) is largest among all vertices

Lemma: Let v be the vertex to finish last in some DFS. Then, v belongs to a source SCC.

Proof: Suppose not, so, let u -> w and w is in the same component as v. There are two cases (a) pre(u) < pre(v), (b) pre(u) > pre(v).





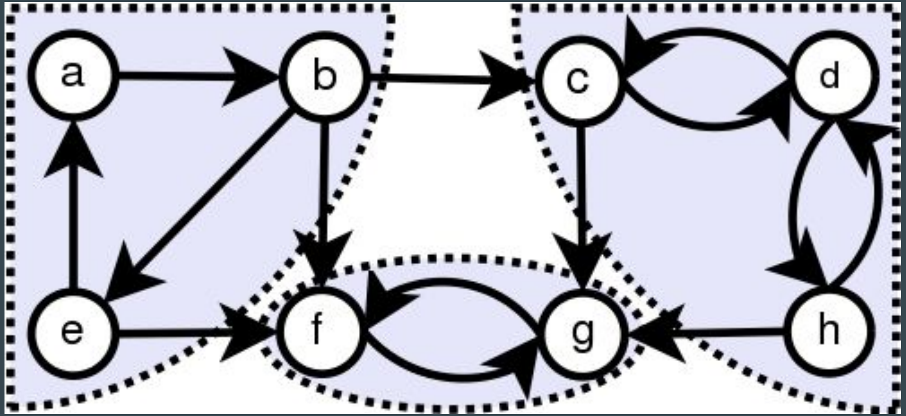
pre(u) < pre(v)

Since u → w ⇝ v is a path & v was not visited when u was being visited, DFS would visit v while u is still not finished. ⇒ v would be finished earlier than u #

pre(u) > pre(v)

v was visited before u was visited. Note that there is no path from v to u since o/w u, v would be in the same component. When v would be finished, u will remain unvisited ∴ u will finish later

Kosaraju ('78) Sharir ('81) SCC



Lemma: If v belongs to a sink SCC, then SCC(v) = all vertices reachable from v.

$\{u : v \rightsquigarrow u\}$

Proof of 1st part: If u is in SCC(v), then by definition of SCC, u has a path to and from v.

If $w \in SCC(v)$, then $v \rightsquigarrow w$.
$\therefore SCC(v) \subseteq reachable(v)$.

Proof of 2nd part: (Proof by contradiction) Suppose v has a path to u and u is not in the SCC(v), so in a different SCC.

Consider that edges on the path from v to u and let e denote the edge that _first_ goes out of SCC(v), probably to SCC(u) or some other SCC. This edge indicates that there is an outgoing edge from SCC(v) and contradicts that fact that SCC(v) is a sink SCC.

To show } $reachable(v) \subseteq SCC(v)$

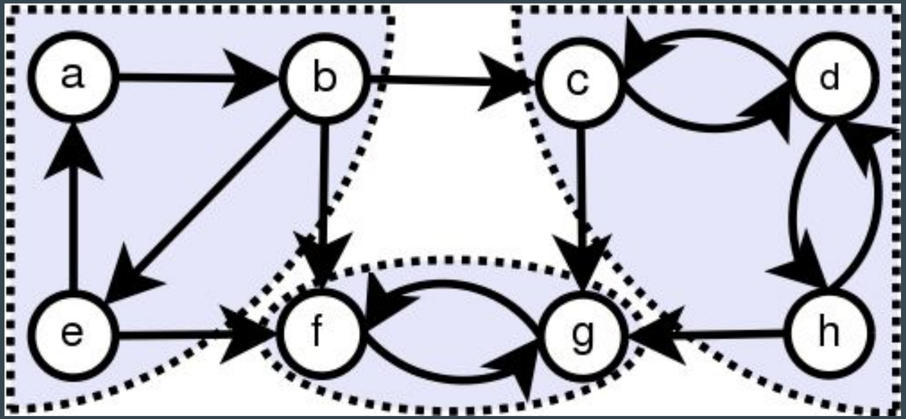
Assume $reachable(v) \not\subseteq SCC(v)$.

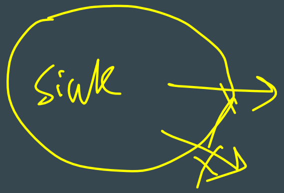
$\Rightarrow v \rightsquigarrow u$ & $u \notin SCC(v)$

↳ path: $v = w_0$, $\boxed{w_1 w_2 \cdots w_k u}$

first vertex that is not in $SCC(v) \rightarrow w_i$ $\therefore w_{i-1} \rightarrow w_i$ is an outgoing edge #

Kosaraju ('78) Sharir ('81) SCC





Lemma: A sink SCC in G is a source SCC in rev(G).

reverse the edge directions

Proof: Let C be a sink SCC in G. So, it has no edges going out from any vertex in C to a vertex in any other component. In rev(G), there would be no edges coming in from a vertex in any other component to any vertex in C. This is same as the condition for C to be a source SCC in rev(G).

Algorithm for finding all SCC

Lemma: Let v be the vertex to finish last in DFS. Then, v belongs to a source SCC.

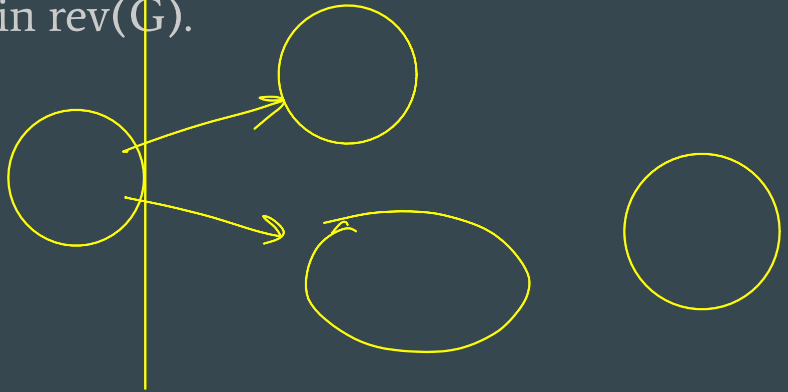
Lemma: If v belongs to a sink SCC, then SCC(v) = vertices reachable from v.

Lemma: A sink SCC in G is a source SCC in rev(G).

Q: How can we find one source SCC?

Q: How can we find one sink SCC?

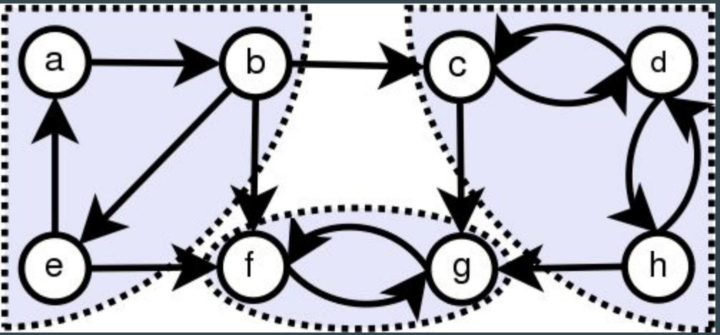
Q: How can we find all SCCs?



Hint: Removing a source SCC or sink SCC does not change other SCCs.

Hint: Reverse graph has same SCCs.

Kosaraju ('78) Sharir ('81) SCC



2-DFS O(n+m) algorithm

finish time
unmarked

Run DFS: L is ordered according to finish time

Reverse the edges of the graph: Grev

On Grev DFS(last finished yet-unmarked vertex in L)

Output everything discovered as an SCC

and mark all those that are output

// This is source SCC of the current graph

// Ignore/implicitly remove this SCC

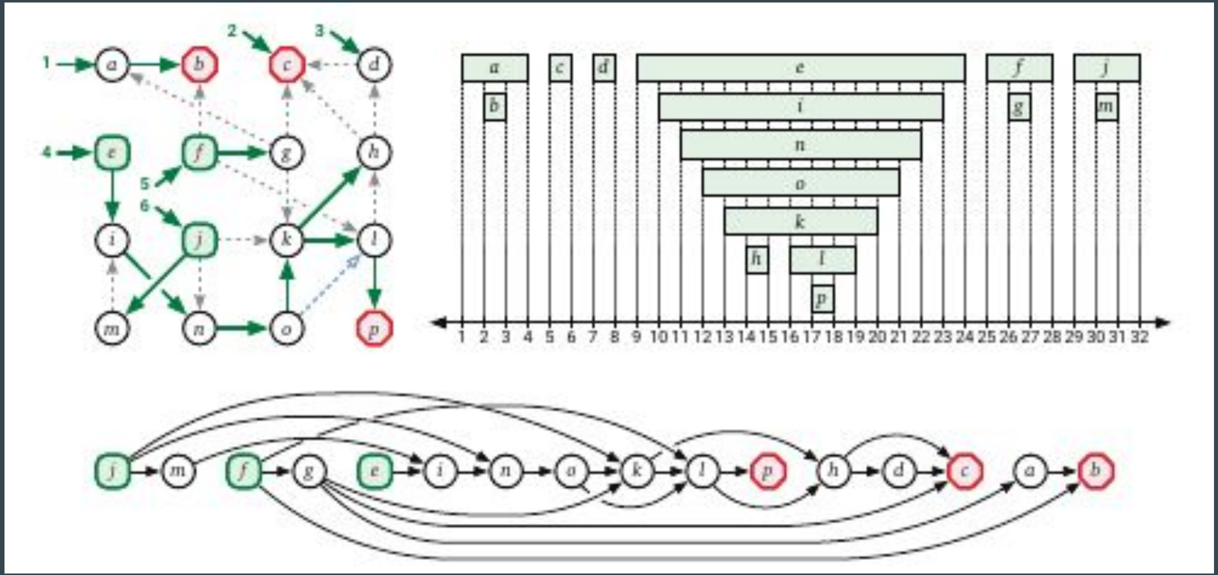
Goto: DFS(last finished... in L)

not defined for $u \rightleftarrows v$

u

v

Topological Sort



Order u before v if u → v

Consider "post(v)" !

[pre,post] intervals are either nested or disjoint.

Claim:

If u → v and post(u) < post(v), then ...? G is cyclic.

- pre(u) < pre(v)

If u is visited before v, & u ⇝ v, then v will be visited & finished before u.
post(v) < post(u) #

- pre(v) < pre(u) < post(u) < post(v)

--- There must be v ⇝ u
∴ u & v belong to a cycle

Topological Sort

T.order(G.rev) = rev(T.order(G))

rev (T.ordering(Grev))

= T.order(G)

Goal: Order u before v if u → v

Consider “post(v)” !

Lemma: If u → v and post(u) < post(v), v ⇝ u and u, v belong to some cycle.

Lemma: If G is acyclic then for any u → v edge, post (u) > post (v) .

∴ print vertices in the decreasing order of post values.

How to construct a topological ordering (ordering of vertices such that if u → v, then u is ordered before v) ? How to “print u before v” ?

Claim :- Topo.Ordering(rev(G)) = rev(Topo.Ordering(G))

reverse graph Grev

DFS on Grev, print vertices in increasing order of post. values