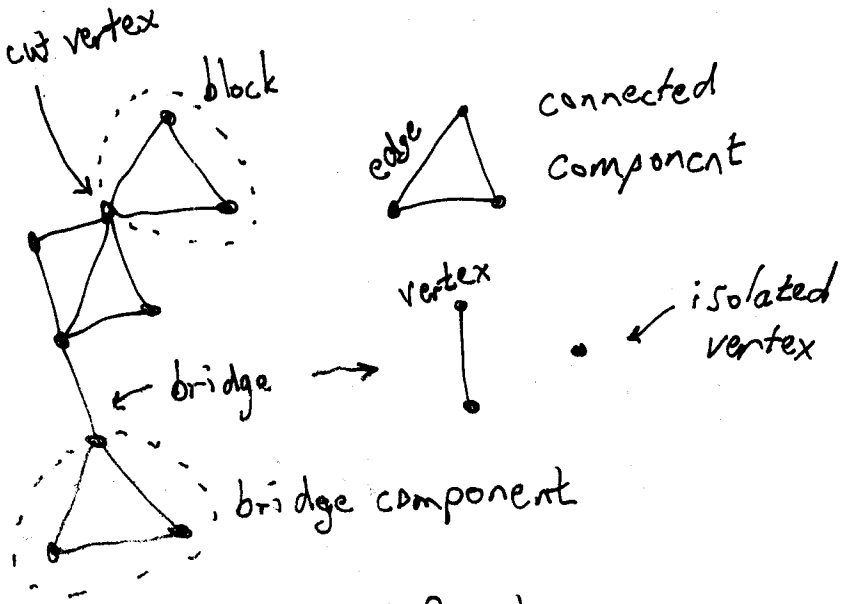
Graphs, Search, Components



Undirected Graph

Representations:

Adjacency Matrix

	a	b	c	d
a	0	1	0	1
b	1	0	1	0
c	0	1	0	0
d	1	0	0	0

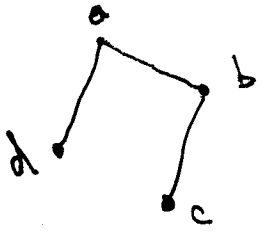
Adjacency Lists

a: b, d

b: a, c

c: b

d: a



Search

Explore vertices systematically by traversing edges

Mark vertices when visited

Traverse an untraversed edge from a visited vertex

or else

Start a new search from an invisited vertex

Breadth-first search: queue of visited vertices

Depth-first search: stack of visited vertices

dfs(v): { previsit(v); scan(v); postvisit(v) }

scan(v): for e ∈ arcs out(v) → traverse(e)

traverse(e): {advance(e);

if not previsited(t(e)) → dfs(t(e));

retreat(e)}

explore(G): for v ∈ G → if not previsited(v) → dfs(v)

Properties of Depth-First Search

First edge into each vertex = tree edge

Tree edges define spanning forest: trees rooted at search start vertices

Previsit order = preorder on tree (discovery order)

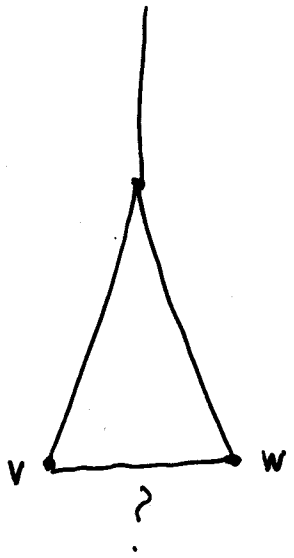
Postvisit order = postorder on tree (finishing order)

v an ancestor of w iff $pre(v) \leq pre(w)$ & $post(v) \geq post(w)$

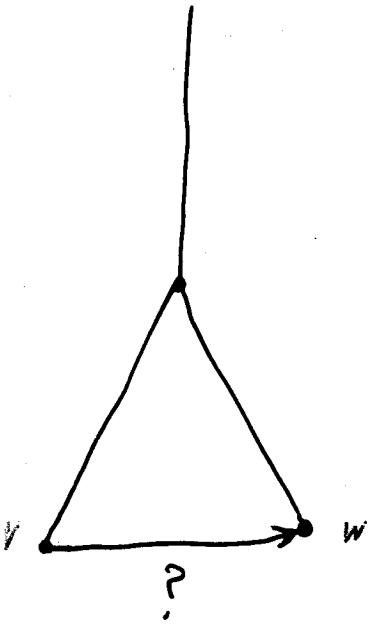
Undirected graph: direct edges along search direction:

each nontree edge leads from a descendant to an ancestor

Digraph: (v,w) an edge implies v and w are related in depth-first spanning forest or $pre(v) > pre(w)$ (& $post(v) > post(w)$)



would be traversed from v
before w is reached,
hence a tree edge

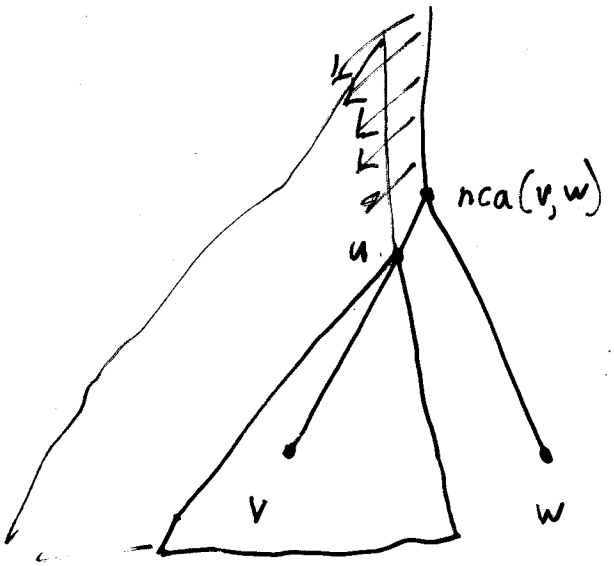


(same argument)

Path lemma:

undigraph: Any path from v to w contains a common ancestor of v and w.

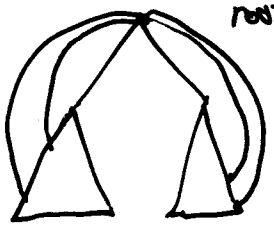
digraph: Any path from v to w with pre(v) < pre(w) (or post(v) < post(w)) contains a common ancestor of v and w



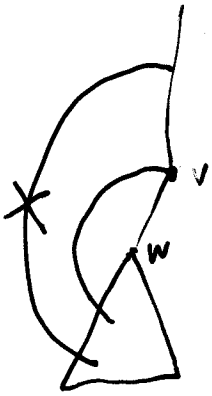
common ancestor = first vertex on the path that is not a descendant of u

(digraph) = first vertex on the path that is ≥ u in postorder

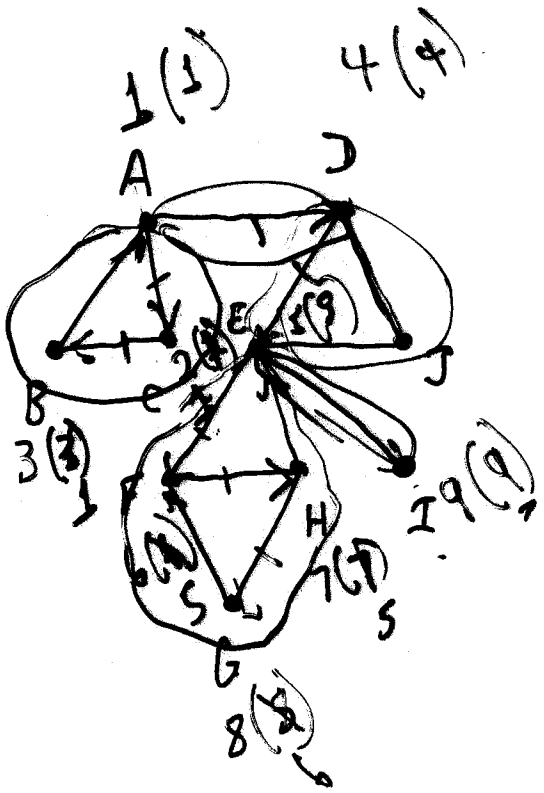
Cut vertex:

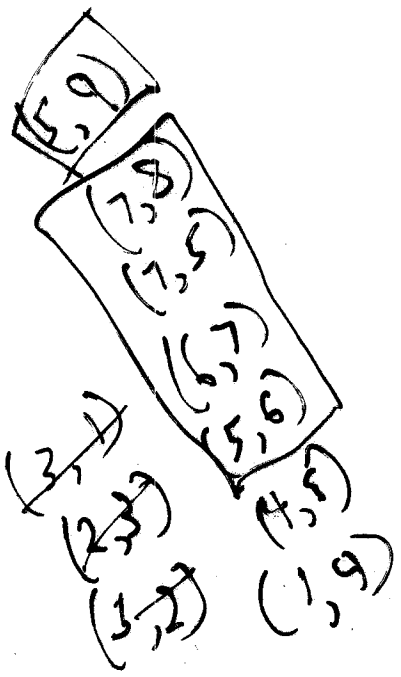


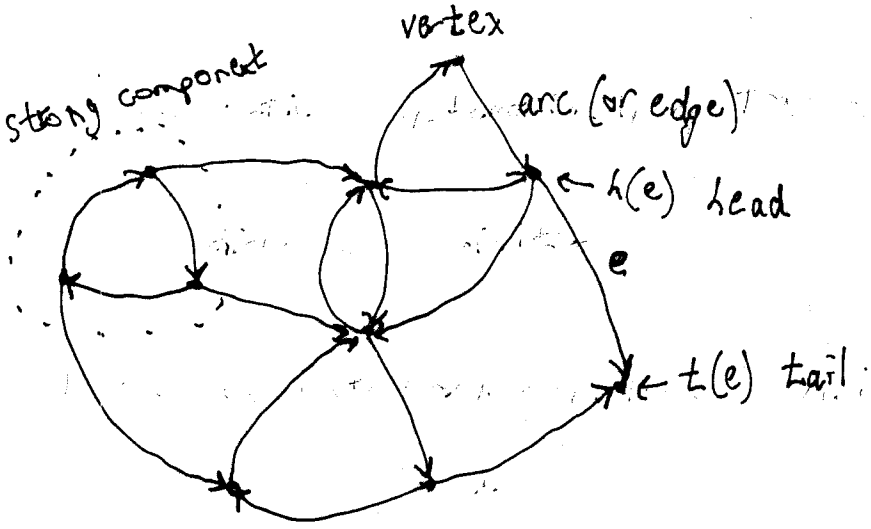
root with ≥2 children
(no cross edges)



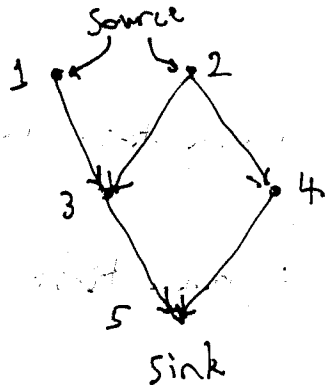
non root with child w with no edges from a descendant of w to a proper ancestor of v (iff low(w) ≥ v)


1(1)
4(4)
A
D
E
B
3(3)
J
I 9(9)
H
6(6)
5
7(7)
5
G
8(8)


(5,9)
(7,8)
(7,5)
(6,7)
(5,6)
(3,1)
(2,3)
(1,2)
(4,5)
(1,9)



Di(rected) Graph



acyclic digraph

topological order: $i \to j \Rightarrow n(i) < n(j)$

Strong Components by DFS (Gabow)

Maintain stack of tentative components

Add each new vertex to stack as a singleton component

When advancing along an edge, if it leads from a component to a lower component on stack, combine all components down to the lower component

When postvisiting the last vertex in a component, output the component

Needs disjoint set union to maintain components:

$$O((m+n)\alpha(n))$$

Components output in reverse topological order

Linear-Time Version

Maintain stack of vertices not in permanent components in preorder

Observe: tentative components are intervals on this stack

Maintain separate stack of (indices of) bottom vertices in tentative components

vertex number: 0 if not previsited
stack position if positive
− component number if negative