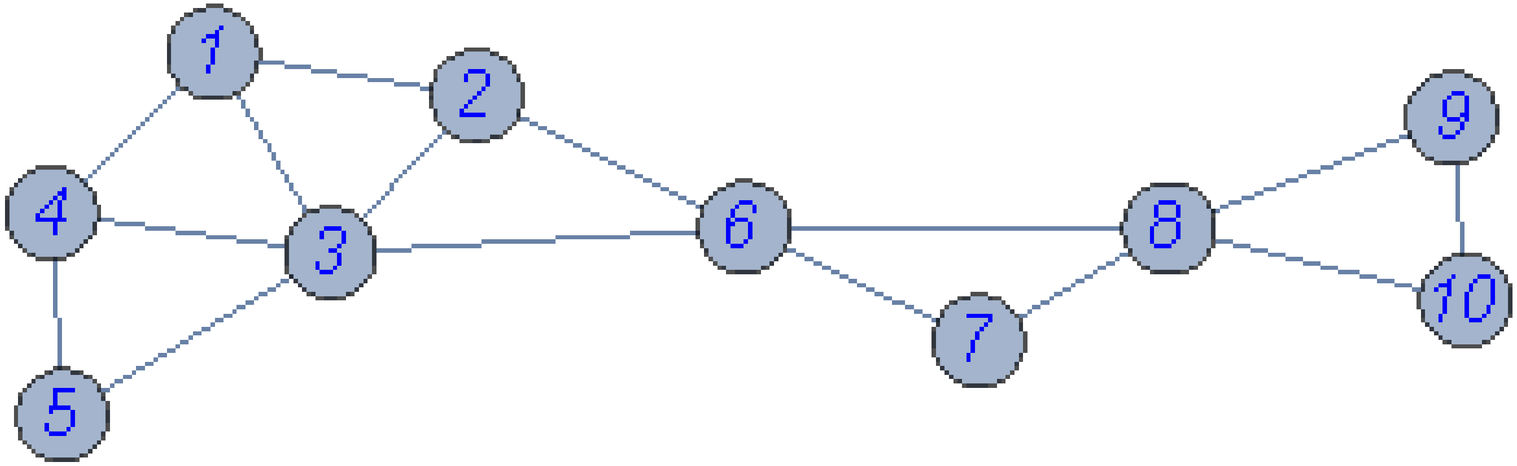
Applications of DFS

Articulation Points

An articulation point/cut vertex in a connected, undirected graph is a vertex that, if removed, will disconnect the graph.



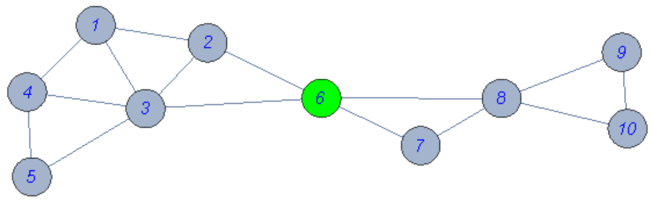
What are the articulation points in this graph? Answer: 6, 8

Applications: network design (indicates network vulnerabilities)

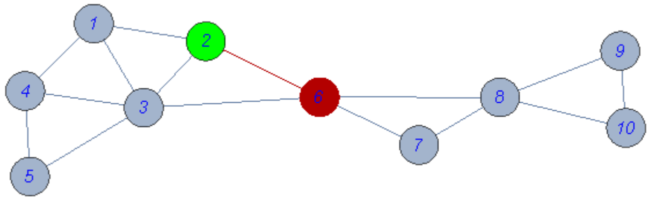
Algorithm (Hopcroft, Tarjan 1973): Perform a single DFS while maintaining the following information: (a) the discovery time of each vertex v and (b) the maximum value over all descendants w of v of the minimum value of the discovery time of any vertex adjacent to w.

Articulation Points

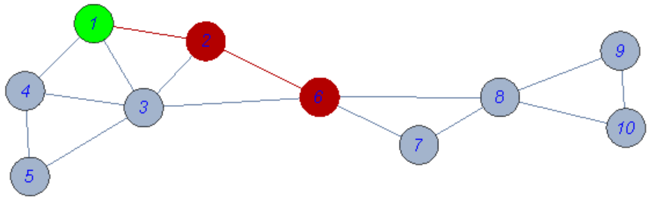
Make two lists: discovery time (known at the time of discovery) and earliest-discovered vertex directly reachable by a child (known at the finishing time).



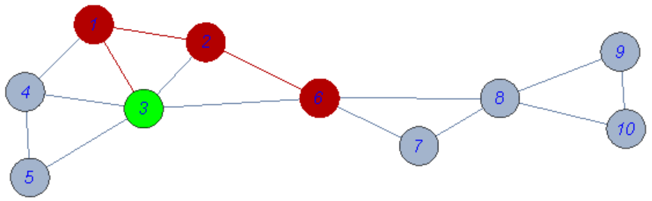
Discovered 6
6



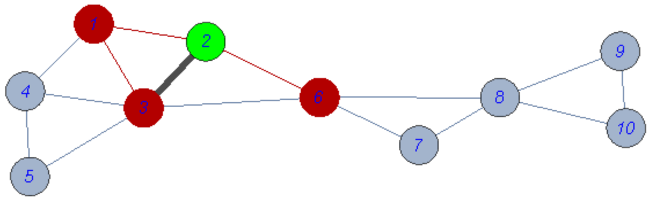
Discovered 2
6, 2

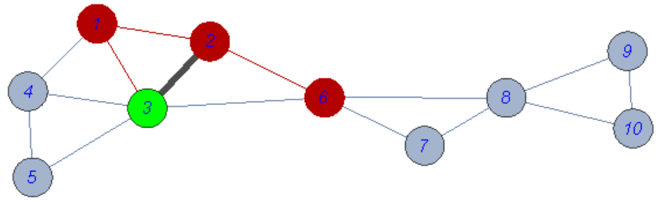


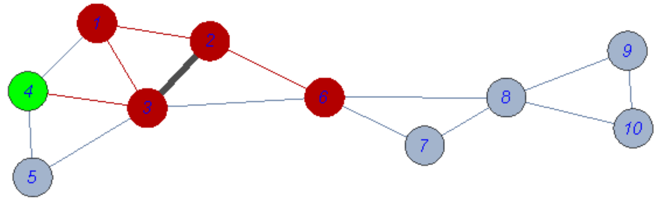
Discovered 1
6, 2, 1



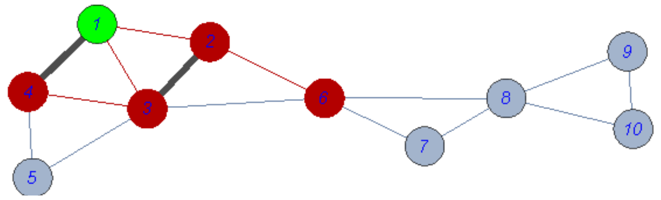
Discovered 3
6, 2, 1, 3

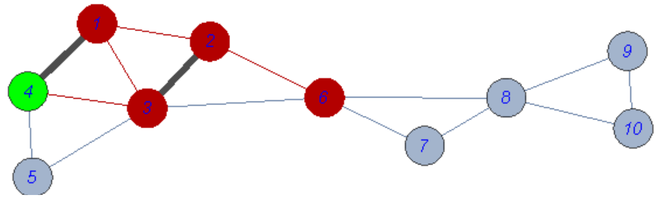


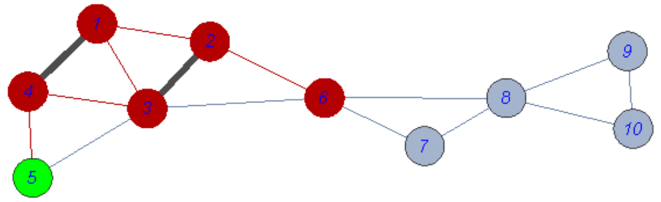




Discovered 4
6, 2, 1, 3, 4

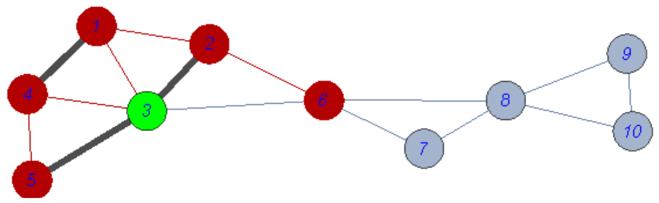


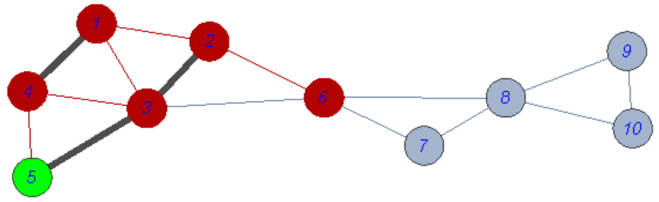




Discovered 5
6, 2, 1, 3, 4, 5

Articulation Points



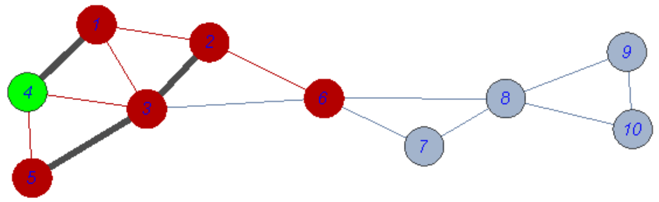


Finished 5

$6, 2, 1, 3, 4, 5$

$5(\emptyset)$

When a node has no children, it is not possible to calculate the red value (the latest-discovered of each child's earliest-reachable neighbors). 5 itself can reach as high as 3.

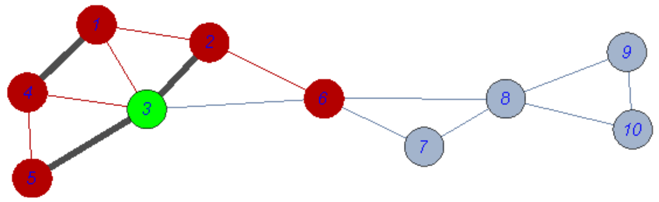


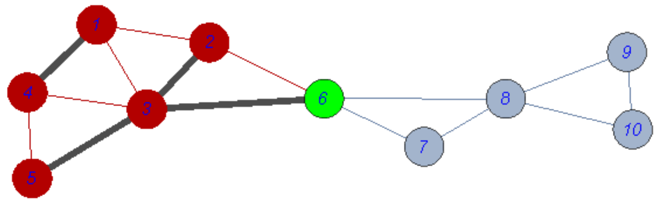
Finished 4

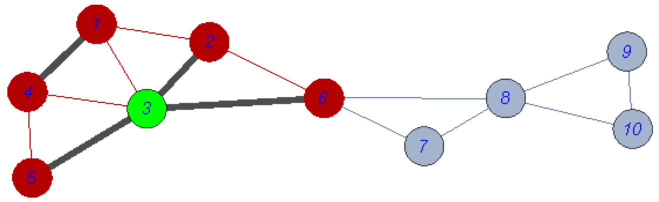
$6, 2, 1, 3, 4, 5$

$5(\emptyset), 4(3)$

4 has a single child 5. (You know this because it is the only vertex that immediately follows 4 in the path list that was not already discovered by the time 4 was is 5.) We know that 5 can reach as high as 3. We also record that 4 can reach as high as 1 so 4's subtree can theoretically reach as high as 1.







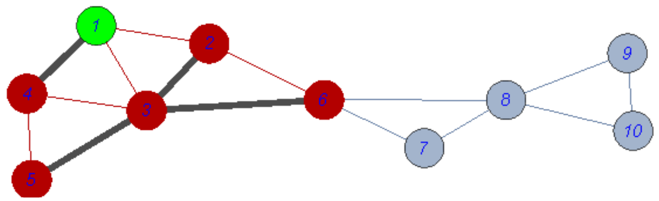
Finished 3

$6, 2, 1, 3, 4, 5$

$5(\emptyset), 4(3), 3(1)$

3's only child is 4, and we already know that 4's subtree can reach as high as 1.

Articulation Points

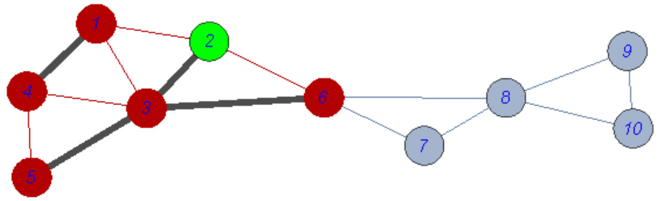


Finished 1

6, 2, 1, 3, 4, 5

$5(\emptyset), 4(3), 3(1), 1(6)$

1's only child is 3, and we know that the highest that 3's subtree can reach is 6.

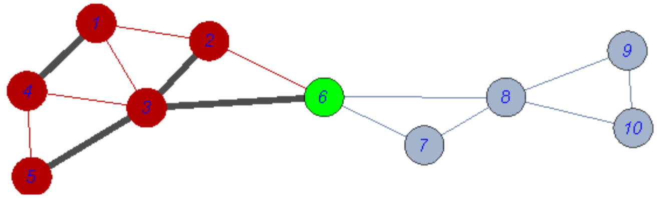


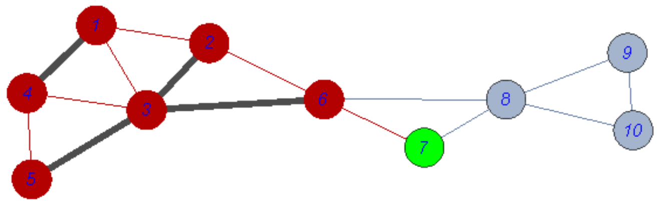
Finished 2

6, 2, 1, 3, 4, 5

$5(\emptyset), 4(3), 3(1), 1(6), 2(6)$

2's only child is 1, and we know that 1's subtree can reach as high as 6.
From this point onward, we'll only point out major events and leave the path computation out.

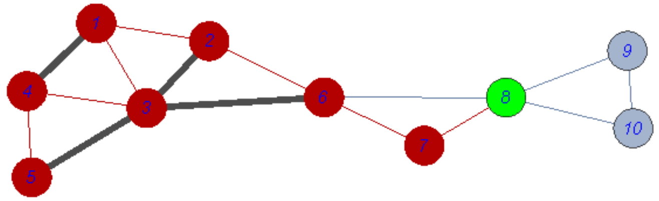




Discovered 7

6, 2, 1, 3, 4, 5, 7

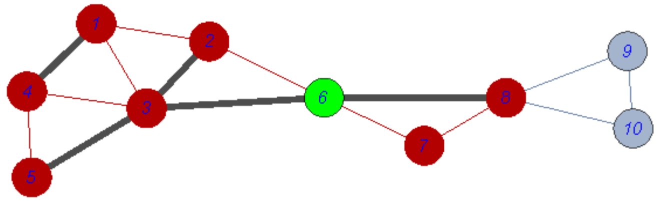
$5(\emptyset), 4(3), 3(1), 1(6), 2(6)$



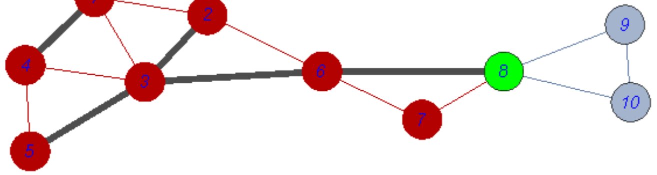
Discovered 8

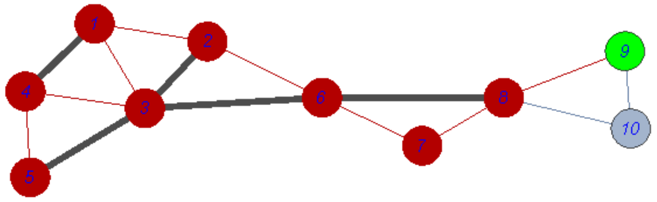
6, 2, 1, 3, 4, 5, 7, 8

$5(\emptyset), 4(3), 3(1), 1(6), 2(6)$



Articulation Points

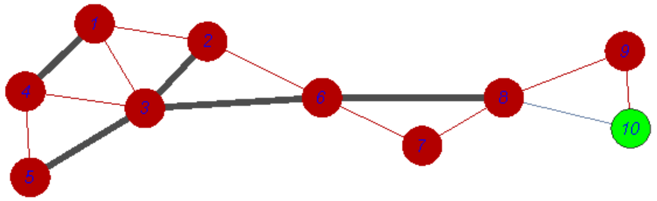




Discovered 9

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

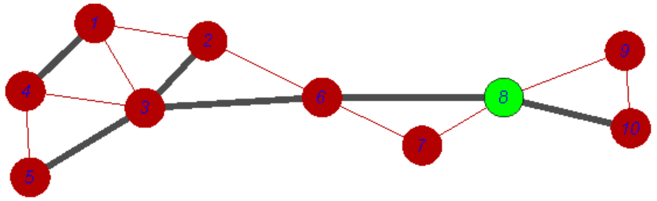
$5(\emptyset), 4(3), 3(1), 1(6), 2(6)$

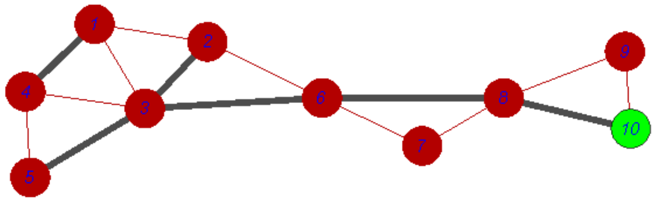


Discovered 10

6, 2, 1, 3, 4, 5, 7, 8, 9, 10

$5(\emptyset), 4(3), 3(1), 1(6), 2(6)$

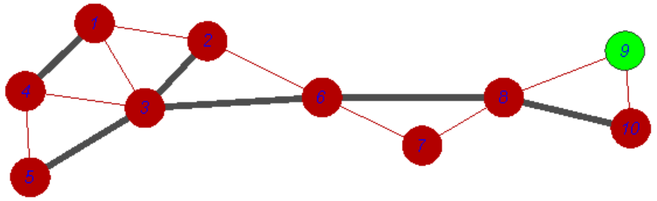




Finished 10

6, 2, 1, 3, 4, 5, 7, 8, 9, 10

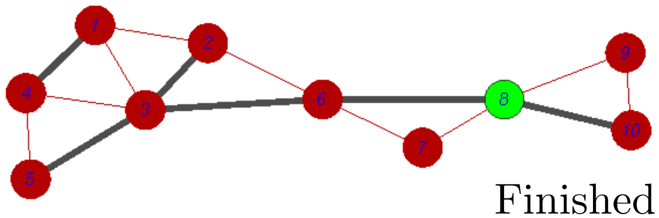
$5(\emptyset), 4(3), 3(1), 1(6), 2(6),$
$10(\emptyset)$



Finished 9

6, 2, 1, 3, 4, 5, 7, 8, 9, 10

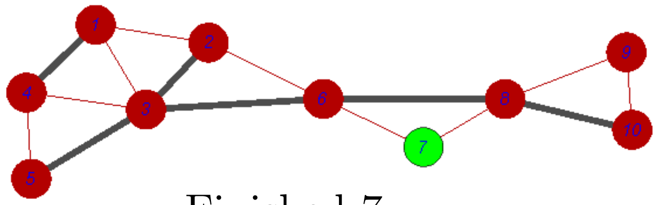
$5(\emptyset), 4(3), 3(1), 1(6), 2(6),$
$10(\emptyset), 9(8)$



Finished 8

6, 2, 1, 3, 4, 5, 7, 8, 9, 10

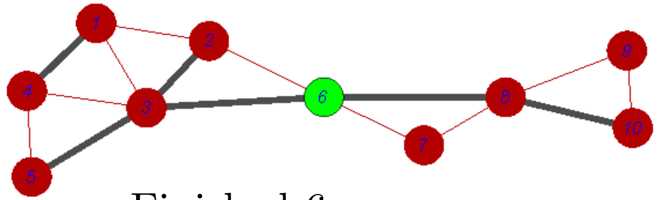
$5(\emptyset), 4(3), 3(1), 1(6), 2(6),$
$10(\emptyset), 9(8), 8(8)$



Finished 7

6, 2, 1, 3, 4, 5, 7, 8, 9, 10

$5(\emptyset), 4(3), 3(1), 1(6), 2(6),$
$10(\emptyset), 9(8), 8(8), 7(6)$

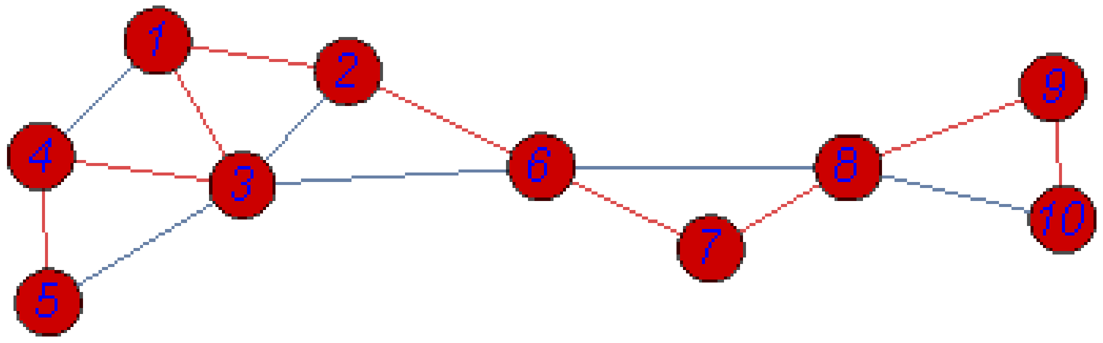


Finished 6

6, 2, 1, 3, 4, 5, 7, 8, 9, 10

$5(\emptyset), 4(3), 3(1), 1(6), 2(6),$
$10(\emptyset), 9(8), 8(8), 7(6), 6(6)$

Articulation Points

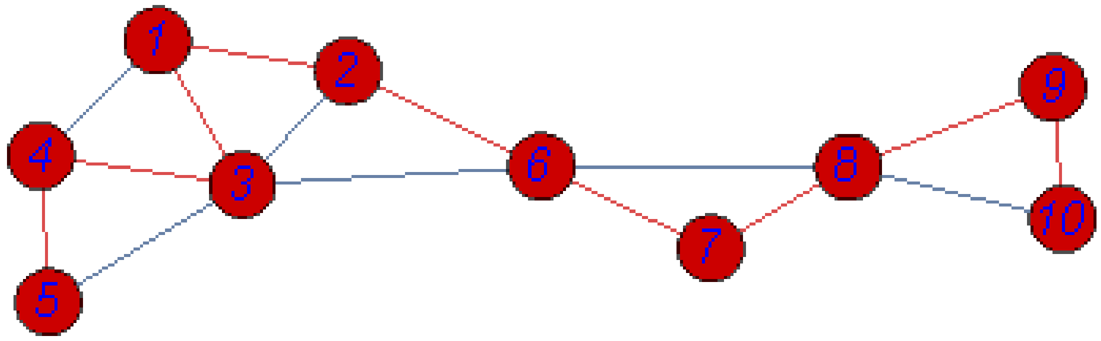


Is the root an articulation point?

If the root has a single subtree, obviously not.

If the root has at least two distinct subtrees, then choose any two vertices a and b in different subtrees. WLOG, assume that a was discovered first.
Can there exist a path from a to b that does not include the root? No. If so, then how could a have been finished before b was discovered?
But if every path from a to b includes the root, then removing the root will disconnect a from b $\Rightarrow$ The root is an articulation point.
In this case, 6 is an articulation point because it has 2 children.

Articulation Points



$6,\ 2,\ 1,\ 3,\ 4,\ 5,\ 7,\ 8,\ 9,\ 10$

$5(\emptyset), 4(3), 3(1), 1(6), 2(6),$
$10(\emptyset), 9(8), 8(8), 7(6), 6(6)$

Assume that v is a non-root vertex.

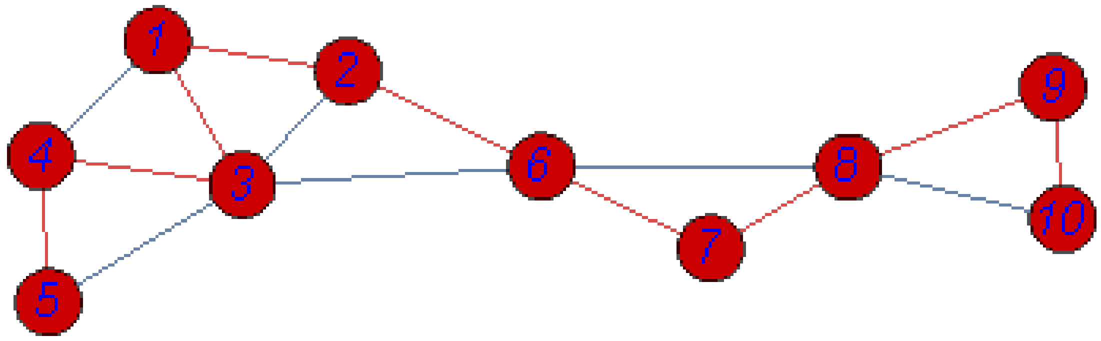
Fact: The discovery/forward edges of the graph form a tree (connected, undirected, acyclic).

If v is removed from the graph, v's children can only reach the root by taking a back edge. To determine whether v's children (and therefore v's children's subtrees) can reach the root, we only need to determine whether there exists a back edge within v's subtree that can reach the root.

Let c be a child of v.

Is it possible for there to be a back edge to c's subtree from a node discovered after c's finishing time? Once c is marked finished, it implies that all nodes that are discoverable by c have been discovered by the DFS so there cannot be a node directly connected to a subtree of c that is discovered after c's finishing time; it would have been discovered by the DFS before c was marked finished. The only back edges from c's subtree must be to nodes with discovery times before c's finishing time.

Articulation Points



6, 2, 1, 3, 4, 5, 7, 8, 9, 10

$5(\emptyset), 4(3), 3(1), 1(6), 2(6),$
$10(\emptyset), 9(8), 8(8), 7(6), 6(6)$

Assume that v is a non-root vertex.

Let c be a child of v. The only back edges from c's subtree must be to nodes with discovery times before c's finishing time.

Assume that there exists a back edge from c to a node n with a discovery time earlier than v's discovery time. Then there exists a path through the forward edges from n to the root so c's subtree can reach the root.

Assume that the only back edges from c's subtree are to nodes with discovery times after v's discovery time but before v's finishing time. Then the only paths that exist from c's subtree are strictly within c's subtree, and it is not possible for c to reach the root.

Conclusion: A non-root vertex v is an articulation point iff there exists a child c of v whose subtree cannot directly reach a vertex with an earlier discovery time than v.

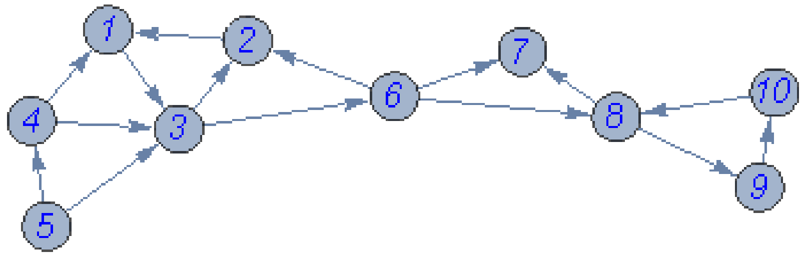
The only non-root vertex that has a child that cannot reach "before" it is 8, and it is therefore an articulation point also.

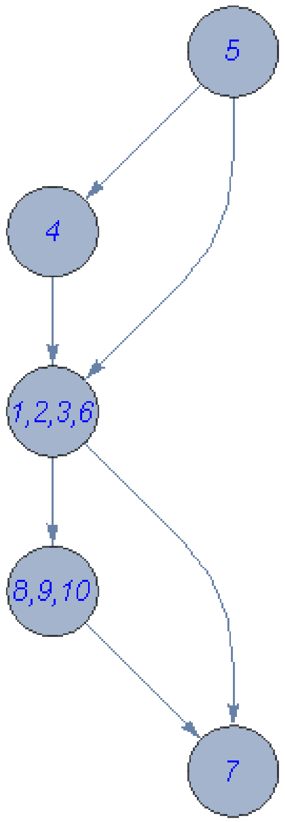
Strongly Connected Components

A strongly connected component of a directed graph is a maximal set of vertices such that for every pair of vertices v_1, v_2 in the component, there exists a path $v_1 \leadsto v_2$.

Problem: Given a directed graph, find all the strongly connected components.

Example:





Answer: {1,2,3,6},{4},{5},{7},{8,9,10}

Why isn't $\{1, 2, 3\}$ a SCC?

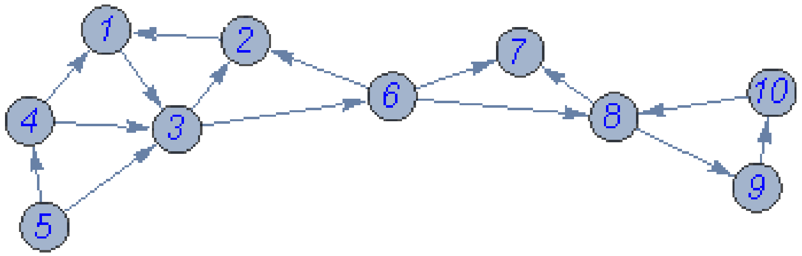
Given two different SCC's $C_1 \neq C_2$, if $v_1 \in C_1$ and $v_2 \in C_2$, is it possible for both $v_1 \leadsto v_2$ and $v_2 \leadsto v_1$?

A directed acyclic graph (DAG) is a graph that is encountered fairly often in computer science. Why is the underlying graph of the SCC's of *any* directed graph a DAG?

Strongly Connected Components

Kosaraju's Algorithm

1. Do a DFS of $G = (V, E)$ to compute finishing times $f[v]$ for each vertex $v \in V$.
2. Do a DFS of G^T, but consider the vertices in order of decreasing $f[v]$ when performing the search.
3. Output the vertices of each tree in the depth-first forest formed in the second DFS as a separate strongly-connected component.

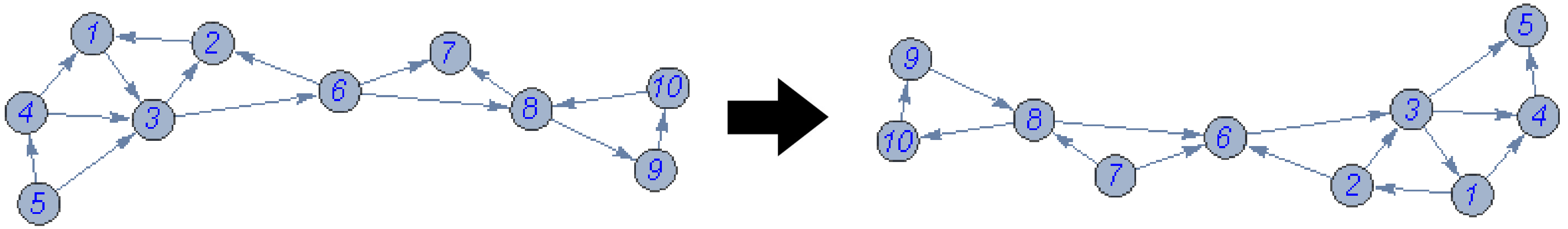


Consider a DFS that takes the following path: 6, 2, 1, 3, 6, 3, 2, 3, 1, 2, 6, 7, 6, 8, 7, 8, 9, 10, 8, 10, 9, 8, 6, 5, 4, 1, 4, 3, 4, 5, 3, 5
The finishing times of the vertices in order from first to last is: 3, 1, 2, 7, 10, 9, 8, 6, 4, 5

Strongly Connected Components

Kosaraju's Algorithm

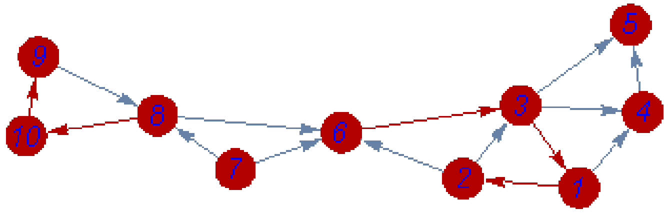
1. Do a DFS of $G = (V, E)$ to compute finishing times $f[v]$ for each vertex $v \in V$.
2. Do a DFS of G^T, but consider the vertices in order of decreasing $f[v]$ when performing the search.
3. Output the vertices of each tree in the depth-first forest formed in the second DFS as a separate strongly-connected component.



The finishing times of the vertices in order from first to last is: 3, 1, 2, 7, 10, 9, 8, 6, 4, 5

Performing a DFS on the right-hand graph in order from last-finished to first: 5, 4, 5, 6, 3, 5, 3, 4, 3, 1, 4, 1, 2, 3, 2, 6, 2, 1, 3, 6, 8, 6, 8, 10, 9, 8, 9, 10, 8, 7, 6, 7, 8, 7

Answer: {1,2,3,6},{4},{5},{7},{8,9,10}



Strongly Connected Components

Claim: The strongly connected components of G and G^T are the same.

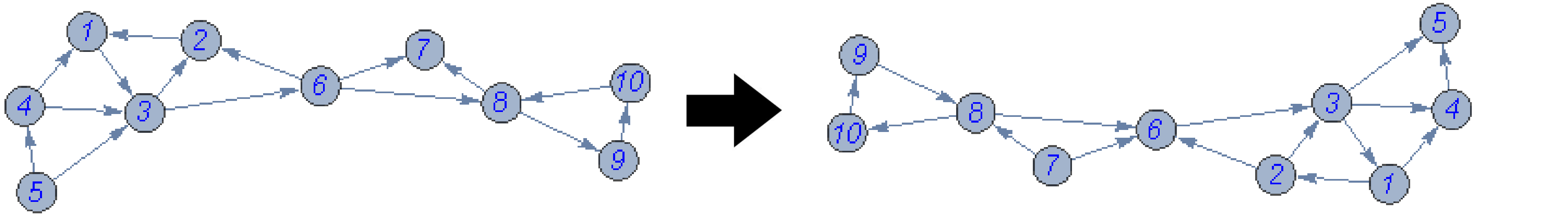
From this point onwards, let $C \neq C'$ be two distinct SCC's.

Claim: If $x \in C$ and $x' \in C'$ and $(x, x') \in E$, then there cannot exist a path from x' to x in G.

Claim: If there is an edge $(x, x') \in E$, with $x \in C$ and $x' \in C'$, x must be finished later than the "latest" vertex finished in C' by the DFS of G. (Two cases: either C or C' is visited first...)

Claim: The strongly connected component that contains the vertex with the latest finishing time from the DFS of G will not have any edges pointed *into* it from any other components. (By the above...)

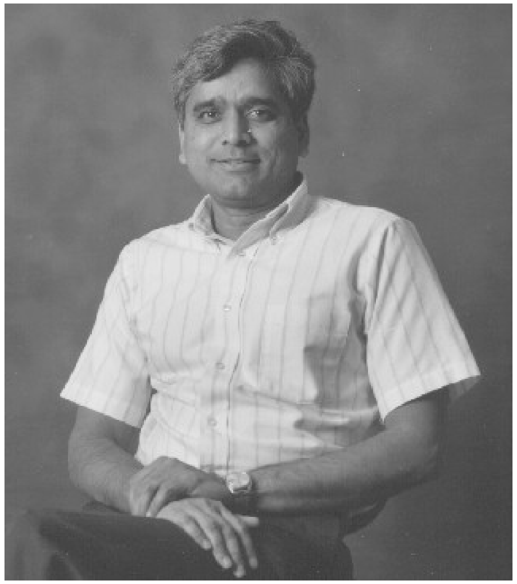
Because G^T has edges pointed in the opposite direction as G, the strongly connected component with the latest finishing time from the DFS from G will not have any edges pointing *out from* it to any other components... so it will pop out as its own tree in the DFS of G^T. Now remove it from the graph and repeat... What happens?



Strongly Connected Components

This algorithm was discovered by S. Rao Kosaraju in 1978 and again by Micha Sharir in 1981. The algorithms require 2 DFS passes.

This problem was first solved optimally by Tarjan in 1972. He used a single DFS.



Kosaraju, Sharir, Tarjan, Hopcroft