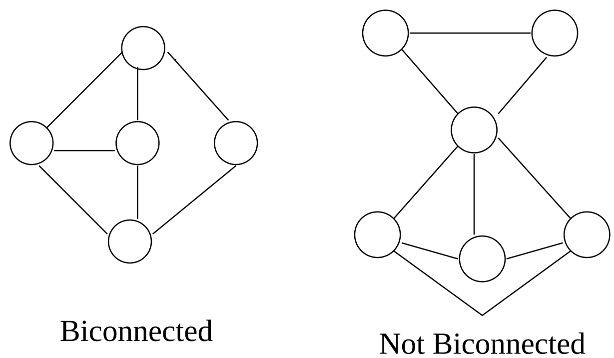
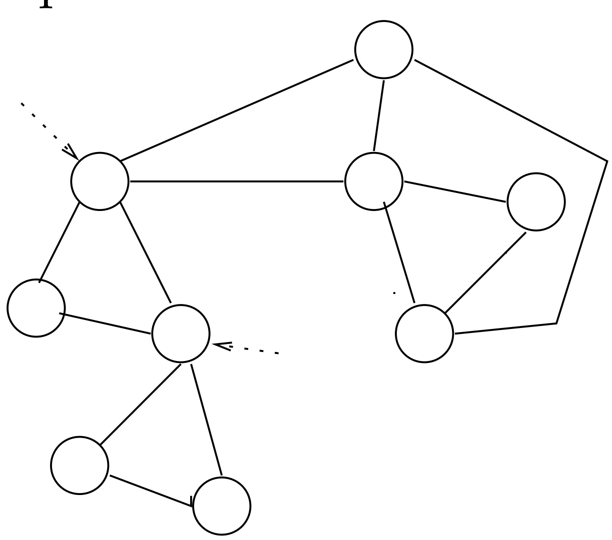
Biconnectivity

- Graph G is biconnected if there are no vertices whose removal disconnects the rest of the graph



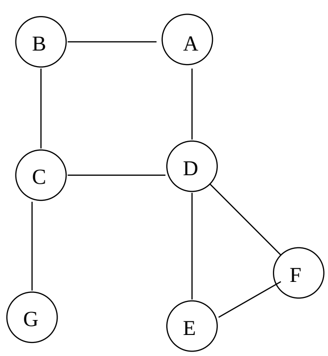
- Articulation Points: Vertex in a graph whose removal disconnects the graph into two or more components

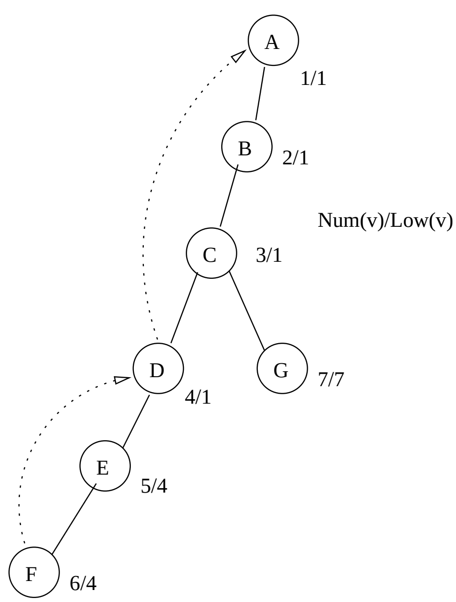


Identifying Articulation Points

(Finding Articulation Points)
Define

- Num(v): DFS NUMBER
- Low(v): Lowest-numbered vertex that is reachable from v by taking zero or more tree edges and then possibly one back edge (in that order)

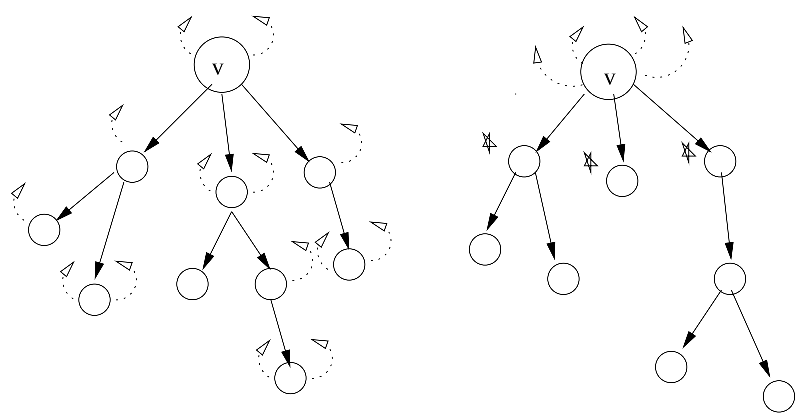


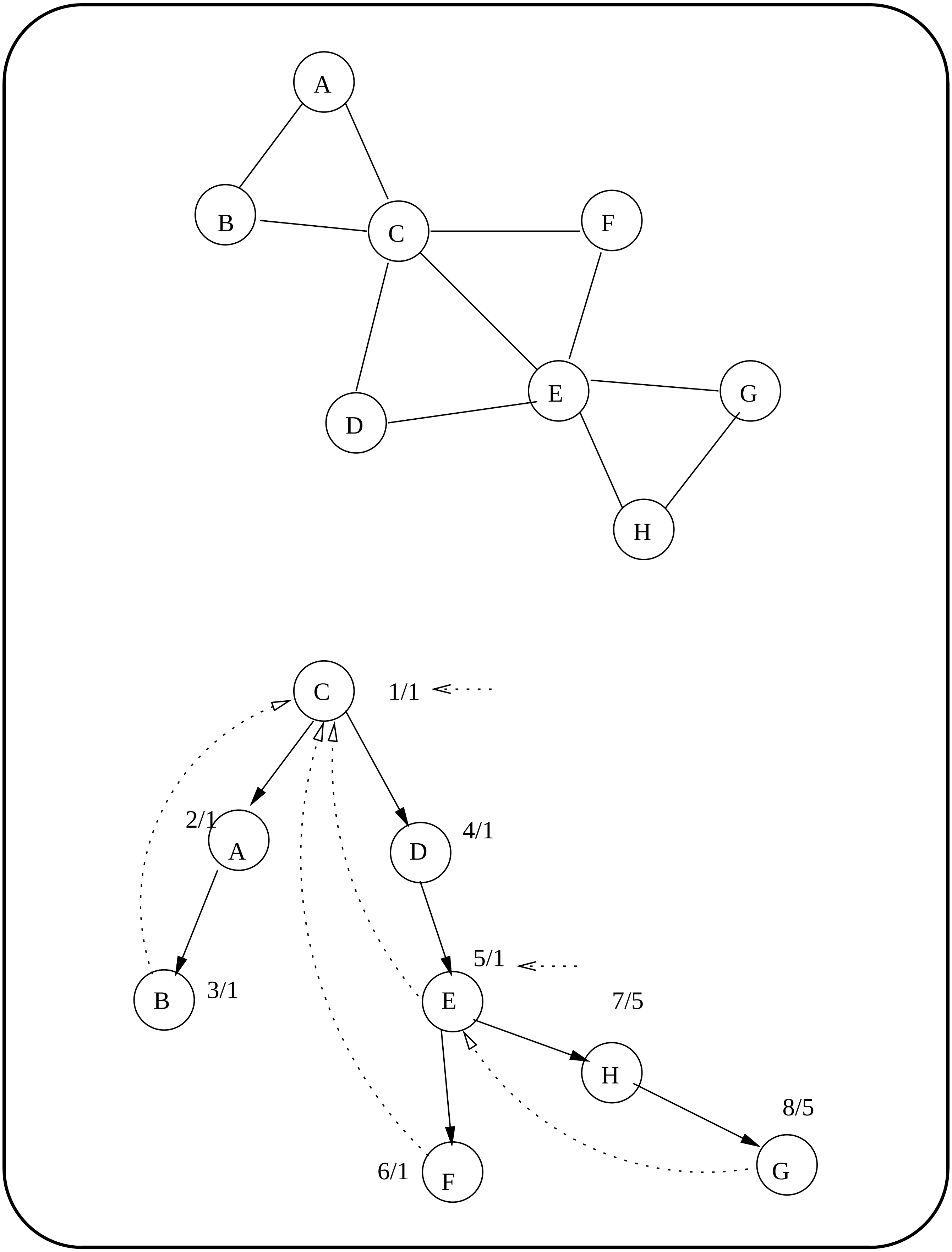


Computation of Low(v)

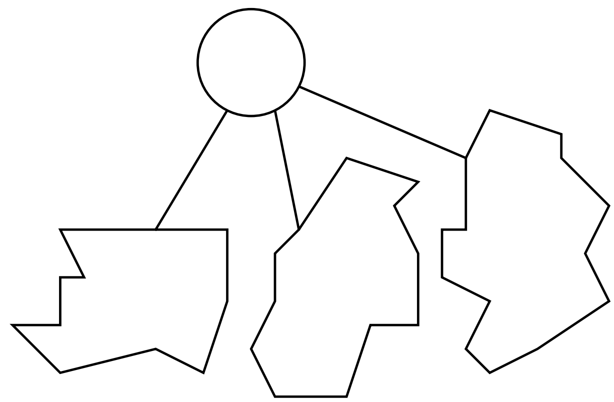
minimum of

- Num(v)
- lowest Num(w) among all back edges (v,w)
- the lowest Low(w) among all tree edges (v,w).

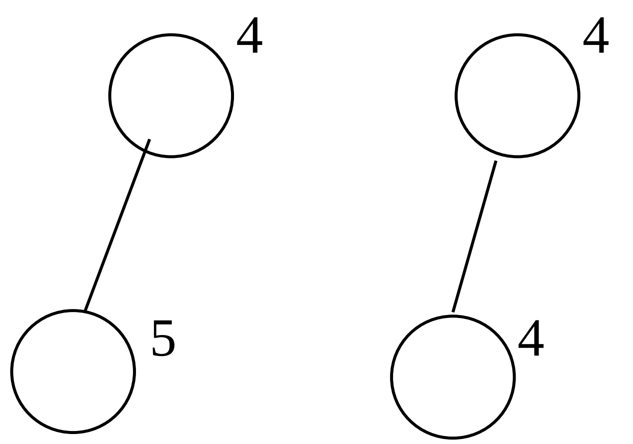



A
B
C
F
D
E
G
H
C
1/1
2/1
A
D
4/1
B
3/1
5/1
E
7/5
H
8/5
6/1
F
G

- Root is an articulation point if it has two or more tree edges

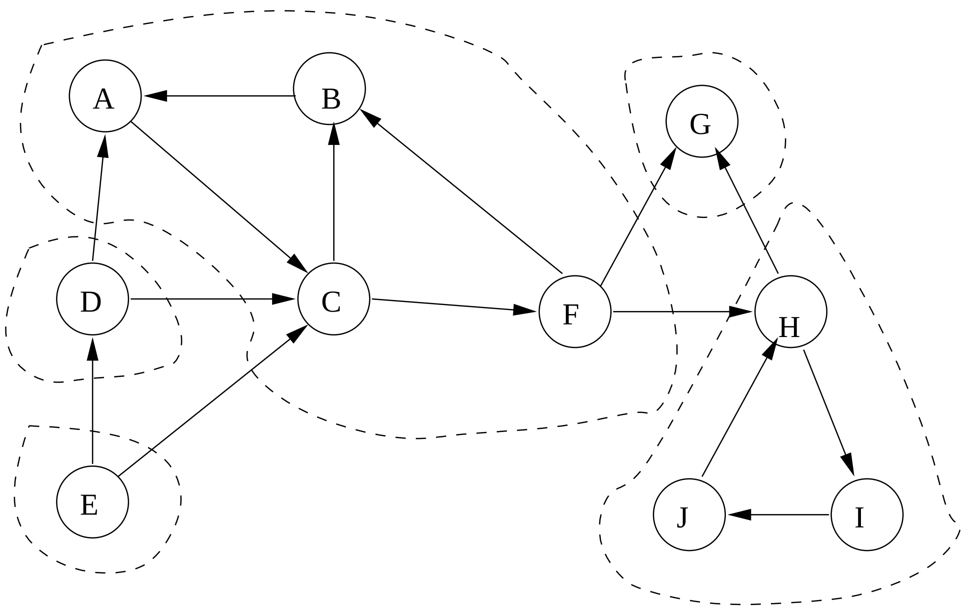


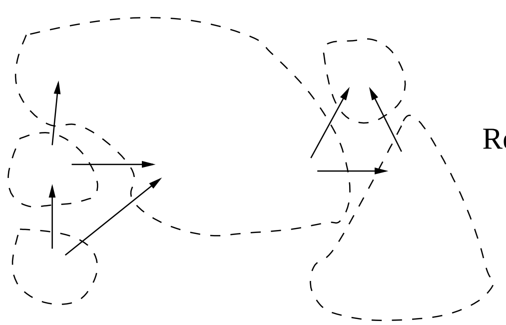
- Vertex v (other than a root) is an articulation point iff v has some child w such that Low(w) $\geq$ Num(v), i.e., w cannot be higher than than v.



Finding Strong Components

- A directed graph is strongly connected iff for every $i \neq j$ there is a directed path from i to j and one from j to i.
- Partition the set of vertices in $G = (V, E)$ into sets $V_1, V_2, \ldots, V_k$. The graph $G_i = (V_i, E(V_i))$ is said to be a strongly connected component iff for every $l \neq j$ in V_i there is a path from l to j and one from j to l; and for no vertex $j \in V_i$ and $q \in V - V_i$, there is a path from q to j and from j to q in G.

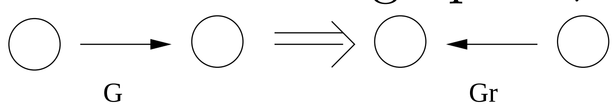




Residual graph is acyclic graph

Identifying Strongly Connected Componen

- Perform a dfs on G (number vertices in the order in which you end their recursive calls)
- Construct the reversed graph G_r from G

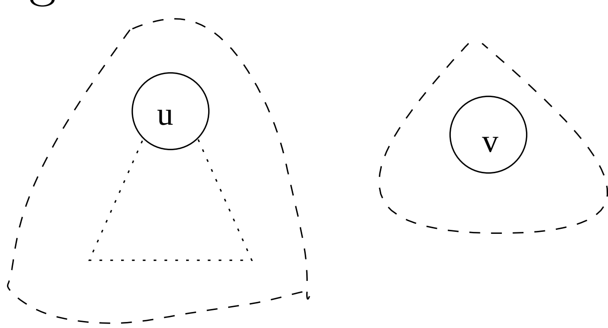


- Perform a dfs on G_r always starting a new dfs search at the vertex with highest number (last one to end recursive call in past (first item))

*Every tree in the dfs forest is a strongly connected component.

Theorem

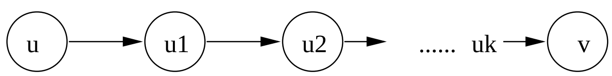
Theorem:
There is a path from u to v in G and a path from v to u in G, if and only if u and v end up in the same spanning tree in the $2^{\underline{nd}}$ DFS traversal.
Proof:
($\rightarrow$) If there is a path from u to v in G and a path from v to u in G, then u and v end up in the same spanning tree in the $2^{\underline{nd}}$ DFS traversal.



In the $2^{\underline{nd}}$ DFS assume the dfs(u) is called before dfs(v).

Proof: Cont'

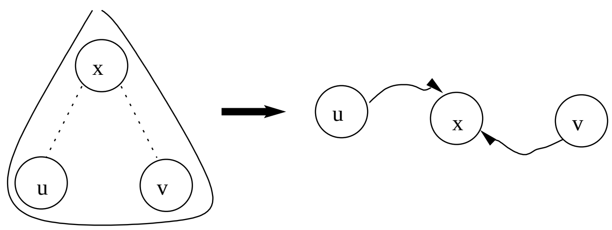
We know there is a path from u to v.



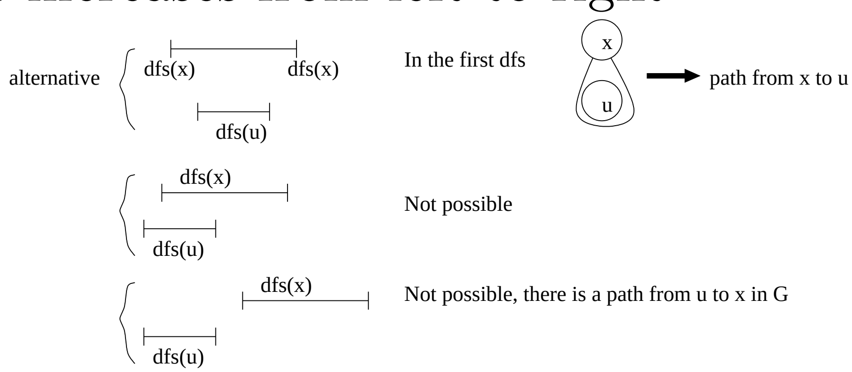
u_i must appear in the same spanning tree as u or in a previous one. The same holds for v. Since u is visited before v then u and v are in the same spanning tree.

Theorem

Proof for (←)
If u and v end up in the same spanning tree in the $2^{\underline{nd}}$ DFS traversal, then there is a path from u to v in G and a path from v to u in G.
Assume without of generality that the spanning tree for u and v is



Therefore, $\#x > \#u$, $\#x > \#v$. This implies that dfs(x) terminated after dfs(u) in the first dfs.
→ time increases from left to right



Using similar argument we know that there is a path from x to v. This concludes the proof.