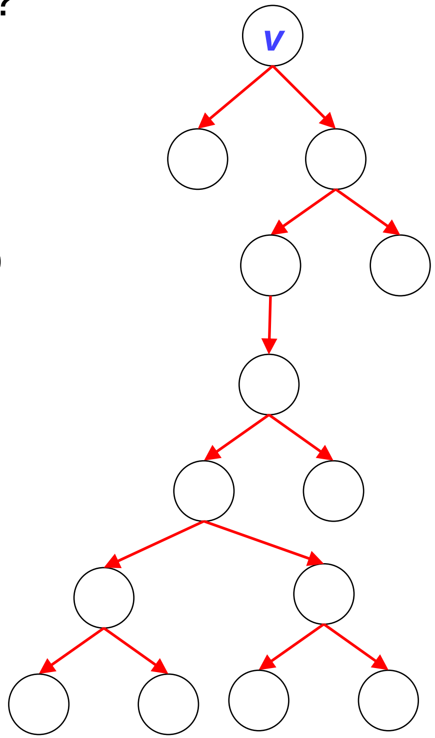
Applications of DFS in Directed Graphs

Dr. Shrutilipi Bhattacharjee, Assistant Professor, NIT Karnataka, India

Application: Graph is Strongly Connected using DFS

- How to check if there exists a path from every vertex in ***G*** to ***v***?
  - Lets get $\boldsymbol{G^R}$ by reversing the edges in ***G***
  - Then do DFS(***v***)
  - If all vertices are visited in $\boldsymbol{G^R}$, then there is a path from ***v*** to every vertex in $\boldsymbol{G^R}$
  - That implies, in the original graph ***G***, there is a path from every vertex to ***v***
  - If in second DFS, if any vertex ***x*** in not visited by ***v***, that implies in the original graph ***G***, there is no path from ***x*** to ***v***
  - Time complexity: ***2*DFS***



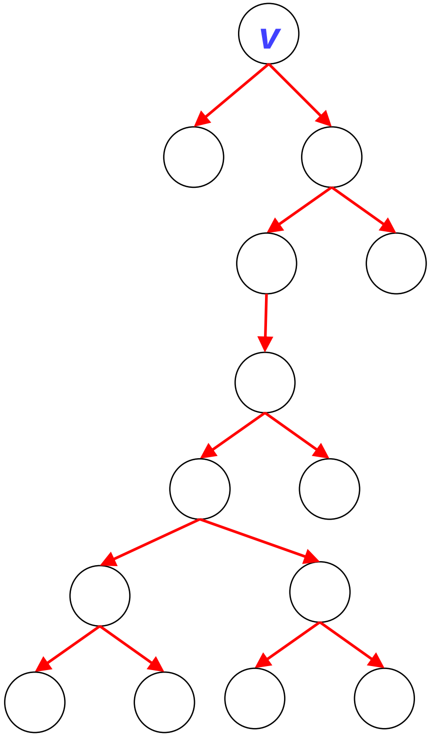
Application: Graph is Strongly Connected using DFS

Pick an arbitrary vertex ***v***

do DFS(***v***)

Reverse ***G***, do DFS(***v***) on $\boldsymbol{G^R}$

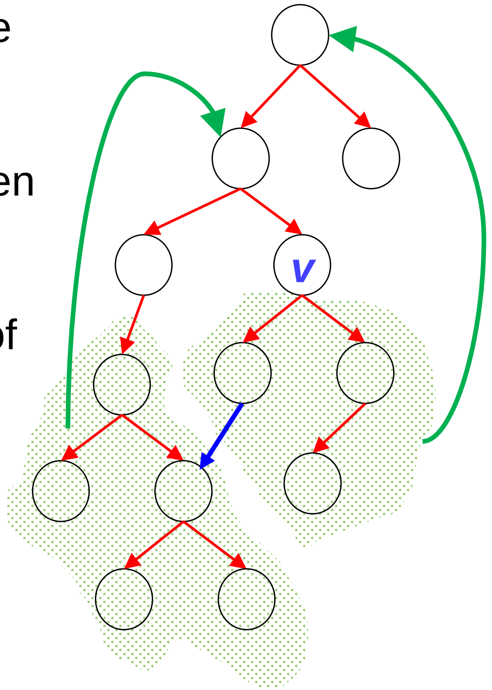
if all vertices are visited in both DFS's **then**
    G is strongly connected
else
    G is not strongly connected



- Can we do better? Can we do it with one DFS?

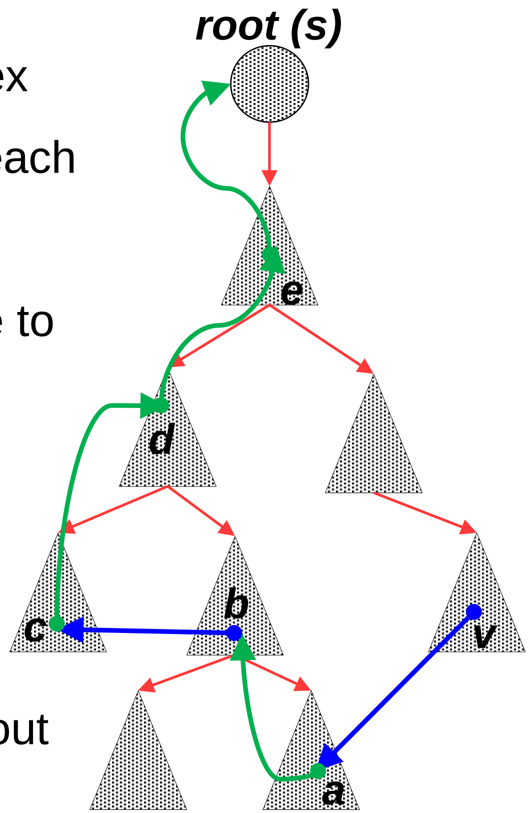
Application: Graph is Strongly Connected using DFS

- Suppose my DFS tree looks like this:
- Now we want to ensure, while processing a vertex ***v***, there are some edges going out from either ***v*** or from the subtree rooted at ***v***
- If from a subtree rooted at ***v*** there is no edge going out from this, then the graph is not strongly connected
  - Because then we can only enter the subtree, we can not go out of this set of vertices
  - Therefore, it is necessary that an edge goes out of every subtree
    - Back edge
    - Cross edge
  - Is it a sufficient condition?



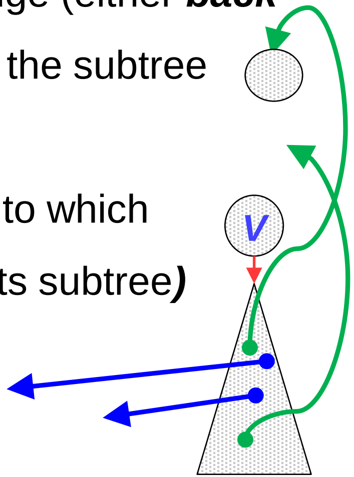
Application: Graph is Strongly Connected using DFS

- To check for sufficiency:
  - Given the DFS tree, from the root we can reach every vertex
  - Now, it is enough to show that from every vertex, we can reach the root as well
  - With both a ***cross edge*** and a ***back edge***, we can traverse to a vertex which is having arrival time strictly less than ***v***
  - Every time with DFS, we will get to the vertex with smaller arrival time: *arr[**v**] > arr[**a**] > arr[**b**] > arr[**c**] > arr[**d**] > arr[**e**] > arr[**s**]*
  - Therefore, it is also ***sufficient*** to check that an edge goes out of every subtree



Application: Graph is Strongly Connected using DFS

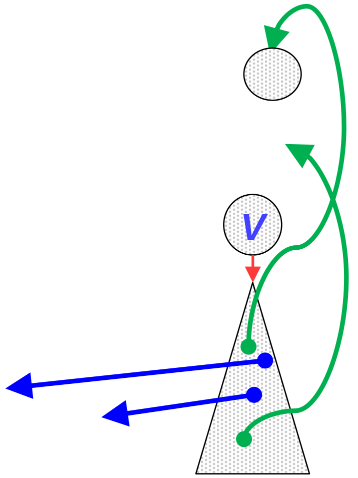
- To modify DFS to check if there is an edge going out of every sub tree?
  - **Subtree**: The part of the tree which is composed of the descendants of any vertex
  - DFS(***v***) should return the smallest arrival time to which there is an edge (either ***back edge*** or ***cross edge***) from the subtree rooted at ***v***
  - i.e., ***min(***all the arrival time to which there is an edge from ***v*** or its subtree***)***
  - Algorithm?
  - Any special case?




```
time = 0;
SC(v)
{
  visited[v] = 1;
  arr[v] = time++;
  SEdge = arr[v];
  for all vertex u out-adjacent from v do
    if !visited[u] then
        SEdge = min(SEdge, SC(u));
    else
        SEdge = min(SEdge, arr[u]);
 if SEdge = arr[v] then
    return SEdge;
}
```

Application: Graph is Strongly Connected using DFS

- To modify DFS to check if there is an edge going out of every sub tree?
  - Any special case?
  - What is the time complexity?




```
time = 0;
SC(v)
{
  visited[v] = 1;
  arr[v] = time++;
  SEdge = arr[v];
  for (for all vertex u out-adjacent from v) do
    if !visited[u] then
        SEdge = min(SEdge, SC(u));
    else
        SEdge = min(SEdge, arr[u]);
 if SEdge = arr[v] && v != starting vertex then
    return SEdge;
}
```

Summary of BFS and DFS Applications

- **Applications of BFS in undirected graphs:**
  - Connected components
  - Bipartite
  - Shortest distance of the vertices to the start vertex
- **Applications of DFS in undirected graphs:**
  - 2-Edge connectivity
  - Planer (only pointer)
  - 2-Vertex connectivity (only pointer)
- **Applications of DFS in directed graphs:**
  - Acyclic
  - Topological sort
  - Strongly connected

What is the time complexity for each of them?

Next Lecture

Dijkstra's Shortest Path Algorithm

Thank you for your attention...

Any question?

Contact:
Department of Information Technology, NITK Surathkal, India
6^{th} Floor, Room: 13
Phone: +91-9477678768
E-mail: shrutilipi@nitk.edu.in, shrutilipi.bhattacharjee@tum.de