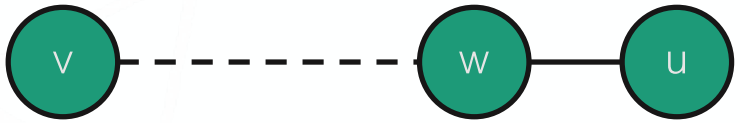
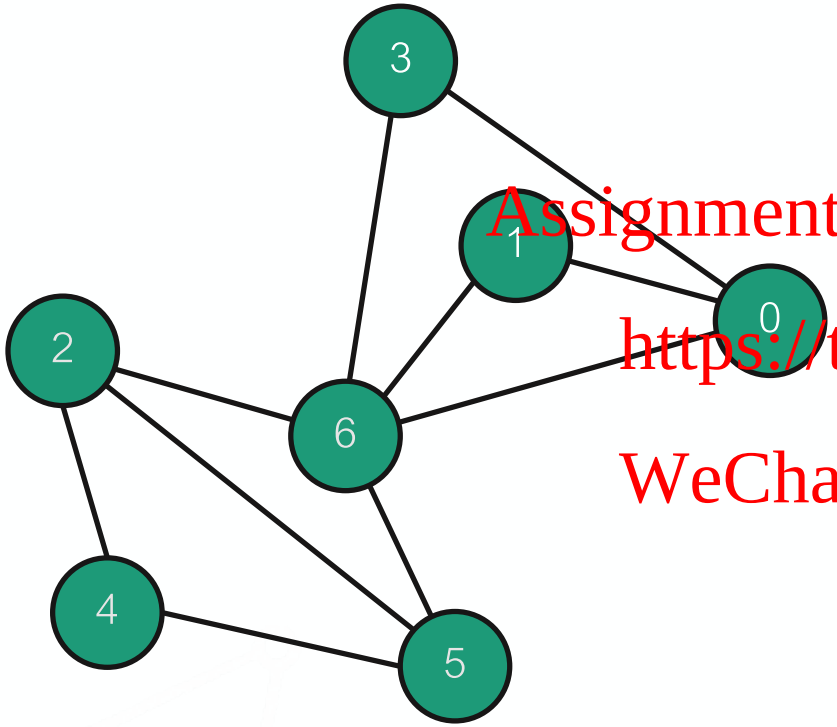
DFS DISCOVERS COMPONENTS

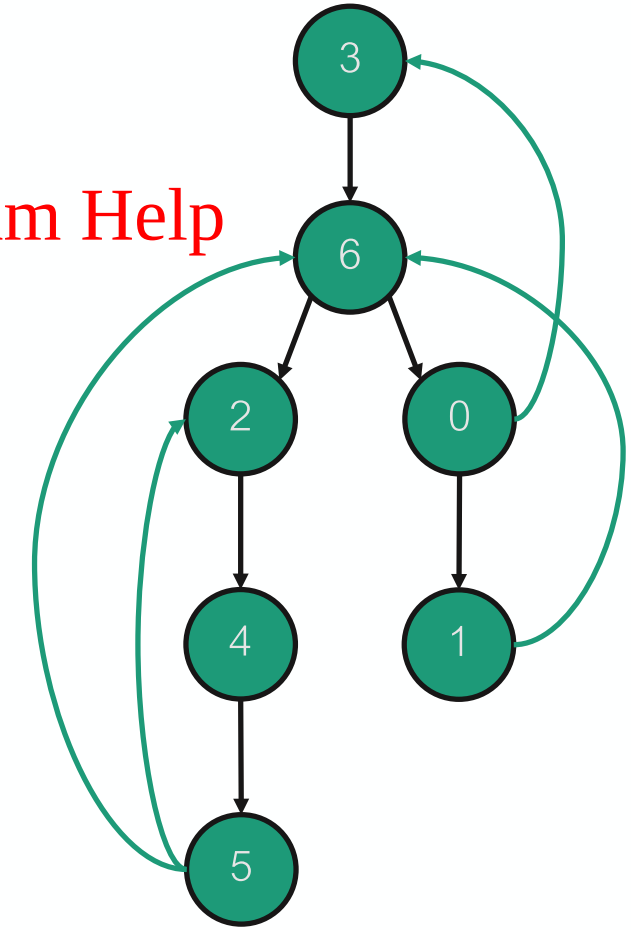
- Note: DEPTH-FIRST-SEARCH calls DFS repeatedly.
- DFS (inner function) calls itself on unseen neighbors.
- Theorem: If DFS is started on vertex v then it will finish all the vertices in the connected component of v before terminating.
- Proof:
  - Suppose for contradiction that a vertex in the v's component is not visited.
  - Let u be the unvisited vertex that is the fewest hops away from v



  - w must have been visited, and from there u. Contradiction!

TYPES OF EDGES IN DFS EXPLORATION





Tree edges: edges on which new vertices are discovered

Back edges: all other edges

THE DFS TREE

- G is a connected graph and we start depth-first search at node v.
- Tree edges actually form a tree rooted at v.
- There is a unique path from v to every vertex in tree.
- Structure of tree = recursive structure of DFS.
  - If x is parent of y then $\mathrm{DFS}(x)$ calls $\mathrm{DFS}(y)$
  - $\mathrm{DFS}(y)$ finishes before $\mathrm{DFS}(x)$
  - z is a descendant of x if it lies in subtree rooted at x (x is an ancestor of z)
  - Define $s(x), f(x)$: start and finish time of x
  - z descendant of $x \Leftrightarrow s(x) < s(z) < f(z) < f(x)$

ONE MORE OBSERVATION:

“NO STONE UNTURNED”

- Suppose we are starting $\mathrm{DFS}(v)$.
- Suppose:
  - There is a path P from v to u.
  - Every vertex on P is unseen at the moment.
- Then u will become a descendant of v in the DFS tree.

- Proof: Similar to the proof that DFS started at a vertex will discover every vertex in the component.

ABOUT BACK EDGES

- Back edges always go between descendants and ancestors in DFS Tree.
- Why?
  - Suppose (u, v) is a back edge.
  - Say u was discovered before v (no loss of generality here).
  - (u, v) must be tried before $\mathrm{DFS}(u)$ finishes.
  - v must be discovered before $\mathrm{DFS}(u)$ finishes.
  - So v is a descendant of u.
  - (u, v) goes between ancestor and descendant.