Applications of DFS
DFS(G)
1. For each vertex \( u \in G.V \)
2. \( u.color = \text{WHITE} \)
3. \( u.\pi = \text{NIL} \)
4. \( \text{time} = 0 \)
5. For each vertex \( u \in G.V \)
6. if \( u.color == \text{WHITE} \)
7. \( \text{DFS-Visit}(G,u) \)

DFS-Visit(G,u)
1. \( \text{time} = \text{time} + 1 \)
2. \( u.d = \text{time} \)
3. \( u.color = \text{GRAY} \)
4. For each vertex \( v \in G.Adj[u] \)
5. if \( v.color == \text{WHITE} \)
6. \( v.\pi = u \)
7. \( \text{DFS-Visit}(G,v) \)
8. \( u.color = \text{BLACK} \)
9. \( \text{time} = \text{time} + 1 \)
10. \( u.f = \text{time} \)

Each vertex has two timestamps:
\( u.d \): First discovered and GRAYed
\( u.f \): Finished and Blackened
\( \forall u: 1 \leq u.d < u.f \leq 2n \)
DFS(G)
1. For each vertex $u \in G.V$
2. $u.color = WHITE$
3. $u.\pi = NIL$
4. time = 0
5. For each vertex $u \in G.V$
6. if $u.color == WHITE$
7. DFS-Visit(G,u)

DFS-Visit(G,u)
1. time = time + 1
2. $u.d = time$
3. $u.color = GRAY$
4. For each vertex $v \in G.Adj[u]$
5. if $v.color == WHITE$
6. $v.\pi = u$
7. DFS-Visit(G,v)
8. $u.color = BLACK$
9. time = time + 1
10. $u.f = time$

Thm 2: $[v.d, v.f] \subseteq [u.d, u.f]$ iff $v$ is a descendant of $u$ in the DFS-forest, $[v.d, v.f] \cap [u.d, u.f] = \emptyset$ iff $u$ and $v$ are unrelated in the DFS-forest \(\square\)
DFS(G)
1. For each vertex \( u \in G.V \)
2. \( u.\text{color} = \text{WHITE} \)
3. \( u.\pi = \text{NIL} \)
4. time = 0
5. For each vertex \( u \in G.V \)
6. if \( u.\text{color} == \text{WHITE} \)
7. DFS-Visit(G,u)

DFS-Visit(G,u)
1. time = time + 1
2. u.d = time
3. u.\text{color} = \text{GRAY}
4. For each vertex \( v \in G.\text{Adj}[u] \)
5. if \( v.\text{color} == \text{WHITE} \)
6. \( v.\pi = u \)
7. DFS-Visit(G,v)
8. u.\text{color} = \text{BLACK}
9. time = time + 1
10. u.f = time

**Thm 3:** \( v \) is a descendant of \( u \) in the DFS-forest iff when we invoke DFS-Visit(u) there is a white path from \( u \) to \( v \)
The green forward edge should point to \( x \)
Edge classification

• **Tree edges**: \((u,v)\), DFS-Visit\(v\) was called from DFS-visit\(u\)

• **Back edges**: \((u,v)\) such that \(v\) is an ancestor of \(u\) in the DFS-forest

• **Forward edges**: nontree edges \((u,v)\) such that \(v\) is a descendant of \(u\)

• **Cross edges**: \((u,v)\) such that \(v.f < u.d\)

**Observe, among non-tree edges:**
1. (Only) backward edges go to a vertex with a later finish time
2. (Only) forward edges go to a vertex with later discovery time
Is my directed graph acyclic?

Thm 6: $G$ is acyclic iff DFS on $G$ does not produce back edges.

This is actually Thm 5 if we want to be consistent with previous videos.

**Proof**

back-edge $e_i 
\rightarrow e_j$
Topological sort

• Input: $G = (V, E)$ a directed, acyclic graph (DAG)
Topological sort

- **Input:** $G=(V,E)$ a directed, acyclic graph (DAG)
- **Output:** Ordering of the vertices such that if $(u, v) \in E$ then $u$ precedes $v$
Recall:

Observe, among non-tree edges:
1. (Only) backward edges go to a vertex with a later finish time
2. (Only) forward edges go to a vertex with later discovery time
Topological sort

- Let's run DFS
Topological sort

• Lets run DFS
Topological sort

• Let's run DFS
Topological sort

- Let's run DFS
Topological sort

• Let's run DFS
Topological sort

• Lets run DFS
Topological sort

• Lets run DFS
Topological sort

• Lets run DFS
Topological sort

• Let's run DFS
Topological sort

• Lets run DFS
Topological sort

• Order the vertices by reverse finishing times
Topological sort

- Order the vertices by reverse finishing times