**WhateverFirstSearch**($s$):

- put $s$ into the bag
- while the bag is not empty
  - take $v$ from the bag
  - if $v$ is unmarked
    - mark $v$
    - for each edge $vw$
      - put $w$ into the bag

Running time?

Each vertex is marked at most once
Each edge is put into bag at most twice

$$O(V + E) \text{ time} = O(V^2) \text{ time}$$

Connected $\Rightarrow E \geq V-1 \Rightarrow V = O(E) \Rightarrow O(E)$ time

Shortest path tree

Bag = stack
Depth-First search

Bag = queue
Breadth-First search
WFSALL($G$):
for all vertices $v$
  unmark $v$
for all vertices $v$
  if $v$ is unmarked
    WHATEVERFIRSTSEARCH($v$)

COUNTCOMPONENTS($G$):
$\text{count} \leftarrow 0$
for all vertices $v$
  unmark $v$
for all vertices $v$
  if $v$ is unmarked
    $\text{count} \leftarrow \text{count} + 1$
    WHATEVERFIRSTSEARCH($v$)
return $\text{count}$

COUNTANDLABEL($G$):
$\text{count} \leftarrow 0$
for all vertices $v$
  unmark $v$
for all vertices $v$
  if $v$ is unmarked
    $\text{count} \leftarrow \text{count} + 1$
    LABELONE($v$, $\text{count}$)
return $\text{count}$

((Label one component))

LABELONE($v$, $\text{count}$):
while the bag is not empty
  take $v$ from the bag
  if $v$ is unmarked
    mark $v$
    $\text{comp}(v) \leftarrow \text{count}$
  for each edge $vw$
    put $w$ into the bag

$v$.comp
**Depth First Search**

**DFS(v):**
- mark v
- PreVisit(v)
  - for every edge v→w
    - if w is unmarked
      - parent(w) = v
      - DFS(w)
- PostVisit(v)

**DFSAll(G):**
- Preprocess(G)
  - for all vertices u
    - unmark u
- for all vertices v
  - if v is unmarked
    - DFS(v)
**DFS(v):**
- mark \( v \)
- \( v.pre \leftarrow \text{clock}++ \)
- for every edge \( v \rightarrow w \)
  - if \( w \) is unmarked
    - \( \text{parent}(w) \leftarrow v \)
    - \( \text{DFS}(w) \)
- \( v.post \leftarrow \text{clock}++ \)

**DFSAll(G):**
- \( \text{clock} \leftarrow 0 \)
- for all vertices \( v \)
  - unmark \( v \)
  - for all vertices \( v \)
    - if \( v \) is unmarked
      - \( \text{DFS}(w) \)

Sort by \( v.pre \) — preorder
Sort by \( v.post \) — postorder

**Lemma:** \( G \) has a directed cycle iff for some edge \( v \rightarrow w \) we have \( v.post < w.post \)

**Proof:** Let \( v \rightarrow w \) be an arbitrary edge

3 cases:

1. **DFS(v) called before DFS(w)**
   - \( v.pre < w.pre < w.post < v.post \)

2. **DFS(w) called before DFS(v)**
   - and \( w \) can reach \( v \)
   - Directed Cycle!
   - \( w.pre < v.pre < v.post < w.post \)

3. **DFS(w) before DFS(v)**
   - \( w \) cannot reach \( v \)
   - \( w.pre < w.post < v.pre < v.post \)

Is \( v.post > w.post \) for all \( v \rightarrow w \) then \( G \) is a dag
see "DAG" or "dag" => think "topological sort"

Order the vertices s.t. \( \text{num}(v) < \text{num}(w) \) for all \( v \rightarrow w \)

\( G \) is a dag \( \Rightarrow \) top. order exists

\( \text{num}(v) = 2V - v.\text{post} \)

**Top Sort(G)**

Preprocess \( G \): Clock \( \leftarrow \) V (# vertices)
Previsit \( v \): return
Postvisit \( v \): Top[
\[\text{clock}--1\] \( \leftarrow \) v

**Dynamic Programming !!**

\( LP(v) = \text{length of the longest path in } G \) from \( v \) to \( t \)

\[
LP(v) = \begin{cases} 
0 & \text{if } v = t \\
\max_{v \rightarrow w} \{ 1 + LP(w) \} & \text{if } v \neq t
\end{cases}
\]

Memoize? Use the graph? \( v.\text{LP} \)
Eval order? reverse top. order = postorder
Running time? \( \mathcal{O}(V+E) \)