Depth First Search
Depth First Search

- Strategy – to search “deeper” into the graph
- Edges are explored from the most recently discovered vertex \( v \)
- When all of \( v \)’s edges have been discovered, DFS **backtracks** to explore edges from which \( v \) was discovered
- Unlike predecessor graph of DFS may have several trees because a search may be repeated – DFS Forest
Book-keeping

• Colors – white, gray, black
• DFS also timestamps vertices:
  – \(d[v]\) records the time when \(v\) was discovered
  – \(f[v]\) records the time when \(v\)’s list is completely examined (\(v\) is blackened)
• Useful in many other algorithms that use DFS as a building block
• \(d[u]<d[v]\)
DFS(G)

1. for each vertex $u \in V[G]$
2. do $color[u] \leftarrow \text{WHITE}$
3. $\pi[u] \leftarrow \text{NIL}$
4. $time \leftarrow 0$
5. for each vertex $u \in V[G]$
6. do if $color[u] = \text{WHITE}$
7. then DFS-VISIT(u)
DFS-VISIT(u)
1. \(color[u] \leftarrow \text{GRAY}\) ▷ White vertex \(u\) has just been discovered.
2. \(time \leftarrow time + 1\)
3. \(d[u] \leftarrow time\)
4. \(\text{for each } v \in \text{Adj}[u]\) ▷ Explore edge \((u, v)\).
5. \(\text{do if } color[v] = \text{WHITE}\)
6. \(\text{then } \pi[v] \leftarrow u\)
7. \(\text{DFS-VISIT}(v)\)
8. \(color[u] \leftarrow \text{BLACK}\) ▷ Blacken \(u\); it is finished.
9. \(f[u] \leftarrow time \leftarrow time + 1\)
Figure 22.4 The progress of the depth-first-search algorithm DFS on a directed graph. As edges are explored by the algorithm, they are shown as either shaded (if they are tree edges) or dashed (otherwise). Nontree edges are labeled B, C, or F according to whether they are back, cross, or forward edges. Vertices are timestamped by discovery time/finishing time.
Running time

- DFS – each loop $\Theta(V)$ exclusive to DFS-VISIT
- DFS-VISIT is called once for each vertex $v$
- Each vertex has $|\text{adj}[v]|$ calls to DFS-VISIT
- In total $\Theta(E)$ such edges
- Hence, $\Theta(V+E)$
Properties of DFS

• Represent discovery of a vertex $u$ as “(u”
• Represent its finishing by “u)”
• History of discovering and finishing gives a well-formed regular expression where parenthesis are properly nested
Figure 22.5 Properties of depth-first search. (a) The result of a depth-first search of a directed graph. Vertices are timestamped and edge types are indicated as in Figure 22.4. (b) Intervals for the discovery time and finishing time of each vertex correspond to the parenthesization shown. Each rectangle spans the interval given by the discovery and finishing times of the corresponding vertex. Tree edges are shown. If two intervals overlap, then one is nested within the other, and the vertex corresponding to the smaller interval is a descendant of the vertex corresponding to the larger. (c) The graph of part (a) redrawn with all tree and forward edges going down within a depth-first tree and all back edges going up from a descendant to an ancestor.
**Theorem  (Parenthesis theorem)**

For all \( u, v \), exactly one of the following holds:

1. \( d[u] < f[u] < d[v] < f[v] \) or \( d[v] < f[v] < d[u] < f[u] \) and neither of \( u \) and \( v \) is a descendant of the other.
2. \( d[u] < d[v] < f[v] < f[u] \) and \( v \) is a descendant of \( u \).
3. \( d[v] < d[u] < f[u] < f[v] \) and \( u \) is a descendant of \( v \).


Like parentheses:

- **OK**: ( [ ] ) [ () ]
- **Not OK**: ( [ ] ) [ ( ) ]

**Corollary**

\( v \) is a proper descendant of \( u \) if and only if \( d[u] < d[v] < f[v] < f[u] \).
White-Path theorem

**Theorem (White-path theorem)**

\( \nu \) is a descendant of \( u \) if and only if at time \( d[u] \), there is a path \( u \sim \nu \) consisting of only white vertices. (Except for \( u \), which was *just* colored gray.)
Edge types

- **Tree edges** – those edges in the DFS forest
- **Back edges** – edges \((u,v)\) connecting \(u\) to an ancestor \(v\) in the depth first tree
- **Forward edges** – \((u,v)\) connects \(u\) to a descendant \(v\)
- **Cross edges** – All other edges.
Edge types
Topological sorting

- On a DAG $G=(V,E)$
- Gives a linear ordering of its vertices such that if an edge $(u,v)$ appears in $G$, the $u$ appears before $v$ in the ordering
DAG

• Directed Acyclic Graphs
• Directed graph with no directed cycles
• Commonly used to indicate precedence amongst events
• Good for modeling *partial order*
• Topological sort creates a *total order* from a partial order
DAG for putting on goalie equipment
Figure 22.7  (a) Professor Bumstead topologically sorts his clothing when getting dressed. Each directed edge \((u, v)\) means that garment \(u\) must be put on before garment \(v\). The discovery and finishing times from a depth-first search are shown next to each vertex. (b) The same graph shown topologically sorted. Its vertices are arranged from left to right in order of decreasing finishing time. Note that all directed edges go from left to right.
**TOPOLOGICAL-SORT***(G)**

1. call **DFS***(G)* to compute finishing times **f**[**v**] for each vertex **v**
2. as each vertex is finished, insert it onto the front of a linked list
3. **return** the linked list of vertices
Complexity

- $\Theta(V+E)$ to do DFS
- $O(1)$ to insert each of the $|V|$ vertices into the linked list
Lemma: A DAG G is acyclic if and only if a depth-first search of G yields no back edges

Proof $\Rightarrow$: Show that back edge $\Rightarrow$ cycle.
Suppose there is a back edge $(u, v)$. Then $v$ is ancestor of $u$ in depth-first forest.

Therefore, there is a path $v \leadsto u$, so $v \leadsto u \rightarrow v$ is a cycle.

$\Leftarrow$: Show that cycle $\Rightarrow$ back edge.
Suppose $G$ contains cycle $c$. Let $v$ be the first vertex discovered in $c$, and let $(u, v)$ be the preceding edge in $c$. At time $d[v]$, vertices of $c$ form a white path $v \leadsto u$ (since $v$ is the first vertex discovered in $c$). By white-path theorem, $u$ is descendant of $v$ in depth-first forest. Therefore, $(u, v)$ is a back edge. ■ (lemma)
Correctness

Correctness: Just need to show if $(u, v) \in E$, then $f[v] < f[u]$. When we explore $(u, v)$, what are the colors of $u$ and $v$?

- $u$ is gray.
- Is $v$ gray, too?
  - No, because then $v$ would be ancestor of $u$.
    $\Rightarrow (u, v)$ is a back edge.
    $\Rightarrow$ contradiction of previous lemma (dag has no back edges).
- Is $v$ white?
  - Then becomes descendant of $u$.
- Is $v$ black?
  - Then $v$ is already finished.
    Since we’re exploring $(u, v)$, we have not yet finished $u$.
    Therefore, $f[v] < f[u]$.
Strongly connected components

Given directed graph $G = (V, E)$.

A **strongly connected component (SCC)** of $G$ is a maximal set of vertices $C \subseteq V$ such that for all $u, v \in C$, both $u \rightarrow v$ and $v \rightarrow u$.

**Example:**

![Diagram of a directed graph with strongly connected components](image)
Algorithm uses $G^T = \text{transpose}$ of $G$.

- $G^T = (V, E^T), E^T = \{(u, v) : (v, u) \in E\}$.
- $G^T$ is $G$ with all edges reversed.

Can create $G^T$ in $\Theta(V + E)$ time if using adjacency lists.

**Observation:** $G$ and $G^T$ have the same SCC’s. ($u$ and $v$ are reachable from each other in $G$ if and only if reachable from each other in $G^T$.)
Figure 22.9  (a) A directed graph $G$. The strongly connected components of $G$ are shown as shaded regions. Each vertex is labeled with its discovery and finishing times. Tree edges are shaded. (b) The graph $G^T$, the transpose of $G$. The depth-first forest computed in line 3 of STRONGLY-CONNECTED-COMPONENTS is shown, with tree edges shaded. Each strongly connected component corresponds to one depth-first tree. Vertices $b$, $c$, $g$, and $h$, which are heavily shaded, are the roots of the depth-first trees produced by the depth-first search of $G^T$. (c) The acyclic component graph $G^{SCC}$ obtained by contracting all edges within each strongly connected component of $G$ so that only a single vertex remains in each component.
**STRONGLY-CONNECTED-COMPONENTS** (*G*)

1. call **DFS**(*G*) to compute finishing times *f*[u] for each vertex *u*
2. compute *G*\(^T\)
3. call **DFS**(*G*\(^T\)), but in the main loop of **DFS**, consider the vertices in order of decreasing *f*[u] (as computed in line 1)
4. output the vertices of each tree in the depth-first forest formed in line 3 as a separate strongly connected component
Figure 22.10  The articulation points, bridges, and biconnected components of a connected, undirected graph for use in Problem 22-2. The articulation points are the heavily shaded vertices, the bridges are the heavily shaded edges, and the biconnected components are the edges in the shaded regions, with a bcc numbering shown.