Depth First Search (DFS)

An extremely simple, fast, recursive algorithm to visit all nodes reachable from a given node.

Let $G = (V, E)$ be a graph.

We assume adjacency list (i.e., sparse) representation.

Algorithm $BasicDFS(u)$:

```
// Visit u
mark u as “visited”
for each $v \in Successor(u)$ do
  // Explore the edge $u \rightarrow v$
  if $v$ is not marked “visited” then
      $BasicDFS(v)$
```
DFS Tree
Solid edge from u to v
means recursive call on u
made recursive call on v
DFS Tree

Solid edge from \( u \) to \( v \) means recursive call on \( u \) made recursive call on \( v \)
BasicDFS: essential properties

Fact: BasicDFS runs in linear time — $O(|V| + |E|)$

Each node gets visited at most once
Each edge gets explored at most once
**BasicDFS: essential properties**

**Fact:** a node $v$ in $V$ gets marked “visited” $\iff$ there is a path from (initial) $u$ to $v$ (i.e., $v$ is “reachable” from $u$)

($\implies$): obvious (only actual paths are explored)

($\impliedby$): kind of obvious...

- consider a path $u = v_0 \to \cdots \to v_k$
- prove by induction on $i$ that $v_i$ gets marked visited...
  - Base case: $i = 0 \checkmark$
  - Assume for $i$ and prove for $i + 1$: when we visit $v_i$, since $v_{i+1} \in \text{Successor}(v_i)$, we explore the edge $v_i \to v_{i+1}$ — either $v_{i+1}$ has already been visited or we will visit it immediately
“Full” DFS: bells and whistles

We visit all the nodes in the graph
while some nodes are unvisited do:
    pick one and start “Basic DFS” from there

Instead of a single DFS tree, the defines a “DFS forest” with one or more DFS trees

• When we explore an edge $u \rightarrow v$ and discover a new, unvisited node $v$, we record the edge $u \rightarrow v$ we will set $\pi[v] = u$
• $\pi[v] = u$ means $u$ is the parent of $v$ in the DFS forest

We “timestamp” each node with a “discovery time” and a “finish time”

We “color” each node:
• white: undiscovered
• gray: visited but not finished (still on the call stack)
• black: finished
“Full” DFS

Algorithm $DFS(G)$:

for each $v \in V$ do: $Color[v] \leftarrow \text{white}$, $\pi[v] \leftarrow \text{Nil}$ 
$time \leftarrow 0$

for each $v \in V$ do
    if $Color[v] = \text{white}$ then $RecDFS(v)$

Algorithm $RecDFS(u)$:

$Color[u] \leftarrow \text{gray}$
$d[u] \leftarrow ++time$  // discovery time

for each $v \in \text{Successor}(u)$ do:
    if $Color[v] = \text{white}$ then
        $\pi[v] \leftarrow u$, $RecDFS(v)$

$Color[u] \leftarrow \text{black}$
$f[u] \leftarrow ++time$  // finish time
DFS Forest:

- Tree edge
- Forward edge
- Back edge
- Cross edge
Running Time Analysis:

- Each node is discovered once
- Each edge is explored once
- Running time $= O(|V| + |E|)$
$u$ discovered
- gray nodes are on run-time stack

$u$ finished

Some Back, Forward, and Cross edges
For $u, \nu \in V$, “$u \subseteq \nu$” means that $u$ lies below $\nu$ in the DFS forest (possibly $u = \nu$), and “$u \sqsubset \nu$” means $u$ lies strictly below $\nu$ (so $u \neq \nu$)

We can also write $u \supseteq \nu$ to mean $\nu \subseteq u$, i.e., $u$ lies above $\nu$ in the DFS forest

**Parenthesis Theorem**

For all $u, \nu \in V$, exactly one of the following holds:

1. $[d[u], f[u]] \cap [d[\nu], f[\nu]] = \emptyset$, $u \not\sqsubset \nu$, and $\nu \not\sqsubset u$

2. $[d[u], f[u]] \subseteq [d[\nu], f[\nu]]$, and $u \sqsubseteq \nu$

3. $[d[u], f[u]] \supseteq [d[\nu], f[\nu]]$, and $u \supseteq \nu$
Classification of edge $u \rightarrow v$

- **Tree edge**: in the DFS forest ($u \subseteq v$)
  - $v$ was *white* when $u \rightarrow v$ was explored;
    $(d[u] < d[v] < f[v] < f[u])$

- **Back edge**: $u \subseteq v$ (includes self loops)
  - $v$ was *gray* when $u \rightarrow v$ was explored
    $(d[v] \leq d[u] < f[u] \leq f[v])$

- **Forward edge**: a non-tree edge, $u \supseteq v$
  - $v$ was *black* when $u \rightarrow v$ was explored, but *white* when $u$ was discovered
    $(d[u] < d[v] < f[v] < f[u])$

- **Cross edge**: $u \not\subseteq v$ and $u \not\supseteq v$
  - $v$ was *black* when $u \rightarrow v$ was explored, and *black* when $u$ was discovered
    $(d[v] < f[v] < d[u] < f[u])$
  - points “into the past” (right to left)
White Path Theorem

Let \( u, \nu \in V \).

\[ u \succeq \nu \iff \begin{cases} \text{at the time } u \text{ is discovered, there is} \\ \text{a path from } u \text{ to } \nu \text{ consisting only of} \\ \text{white nodes} \end{cases} \]

(\Rightarrow) Assume \( u \succeq \nu \)
**White Path Theorem**

Let $u, v \in V$.

$u \supseteq v \iff$

- at the time $u$ is discovered, there is a path from $u$ to $v$ consisting only of white nodes

$(\Leftarrow)$ Let $u = v_0 \rightarrow v_1 \rightarrow \cdots \rightarrow v_k = v$ be the white path

Claim: $u \supseteq v_i$ for all $i$. Assume not, and let $i$ be minimal such that $u \nsubseteq v_i$ for $i > 0$ $\Rightarrow \Leftarrow$
Topological Sorting — Tarjan’s Algorithm

**Algorithm DFSTopSort**

- initialize an empty list
- Run DFS: When a node is painted *black*, insert it at the front of the list
- If we ever discover a back edge, report that the graph is cyclic

So we output vertices on order of *decreasing* finishing time

As a bonus, if there is a cycle, we can actually print it out
Let's get rid of the back edge

Arrange from highest to lowest finishing time
Lemma

$G$ has a cycle $\iff$ DFS produces a back edge

Proof:

- ($\Leftarrow$) A back edge trivially yields a cycle
• (⇒) Suppose $G$ has a cycle $C$ of vertices, and let $\nu$ be the first vertex discovered in $C$:

By the White Path Theorem, $u$ lies below $\nu$ in the DFS forest

:. the edge $u \rightarrow \nu$ is a back edge
Theorem

Algorithm DFSTopSort is correct

Proof:

• Let \((u, v) \in E\)
• We want to show \(f[u] > f[v]\)
• Cases:
  ◦ \((u, v)\) is a tree edge: \(u \preceq v\) and \(d[u] < d[v] < f[v] < f[u]\)
  ◦ \((u, v)\) is a back edge: impossible, since \(G\) is acyclic
  ◦ \((u, v)\) is a forward edge: \(u \preceq v\) and \(d[u] < d[v] < f[v] < f[u]\)
  ◦ \((u, v)\) is a cross edge: \(f[v] < d[u] < f[u]\)
• QED