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
        $BasicDFS(v)$
**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 $u = v_0 \rightarrow \cdots \rightarrow v_k$
- 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 $v_i \rightarrow v_{i+1}$ — 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

When we explore an edge $u \rightarrow v$ and discover a new, unvisited node $v$, we record the edge $u \rightarrow v$
   • these recorded edges comprise the “DFS forest” (which is acyclic)
   • every node $v$ will have (at most) one predecessor $\pi[v] = u$ 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: $Color[v] \leftarrow white$, $\pi[v] \leftarrow Nil$

time $\leftarrow 0$

for each $v \in V$ do

if $Color[v] = white$ then $RecDFS(v)$

Algorithm $RecDFS(u)$:

$Color[u] \leftarrow gray$

d[u] $\leftarrow ++time$  // discovery time

for each $v \in \text{Successor}(u)$ do:

if $Color[v] = white$ then

$\pi[v] \leftarrow u$, $RecDFS(v)$

Color[u] $\leftarrow black$

d[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 $= \mathcal{O}(|V| + |E|)$
$u$ discovered
- *gray nodes are on run-time stack*

$u$ finished

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

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

**Parenthesis Theorem**

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

1. \([d[u], f[u]] \cap [d[v], f[v]] = \emptyset\), \( u \nsubseteq v \), and \( v \nsubseteq u \)

2. \([d[u], f[u]] \subseteq [d[v], f[v]]\), and \( u \subseteq v \)

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

- **Tree edge:** in the DFS forest ($u \supseteq v$)
  - $v$ 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 \( u, v \in V \).

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

\[ \Rightarrow \text{ Assume } u \supseteq v \]
White Path Theorem

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

\[
\begin{align*}
u \supseteq v \iff & \text{ at the time } u \text{ is discovered, there is a path from } u \text{ to } v \text{ consisting only of white nodes} \\
(\Leftarrow) \text{ Let } u = v_0 \to v_1 \to \cdots \to v_k = v \text{ be the white path} \\
\text{Claim: } u \supseteq v_i \text{ for all } i. \text{ Assume not, and let } i \text{ be minimal such that } u \not\supseteq v_i \ (i > 0) \Rightarrow \Leftarrow
\end{align*}
\]
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
**Lemma**

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

Proof:

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

![Diagram](image)

By the White Path Theorem, \(u\) lies below \(v\) in the DFS forest

\(\therefore\) the edge \(u \rightarrow v\) 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: \(u \sqsubseteq v\) 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: \(u \sqsubseteq v\) and 
    \(d[u] < d[v] < f[v] < f[u]\)

  ◦ \((u, v)\) is a cross edge: \(f[v] < d[u] < f[u]\)

• QED