Depth First Search (DFS)

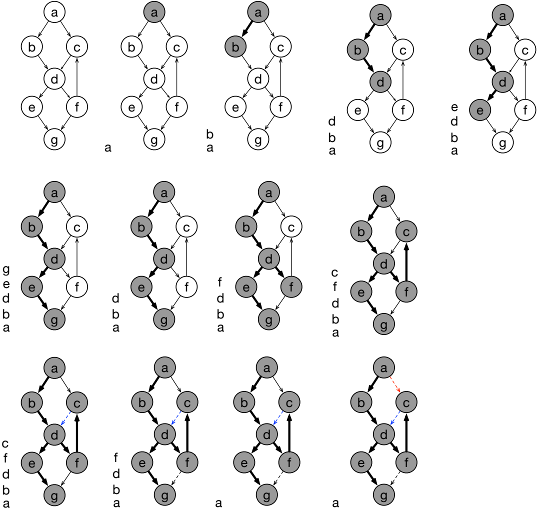
An extremely simple, fast, recursive agorithm to visit all nodes reachable from a given node

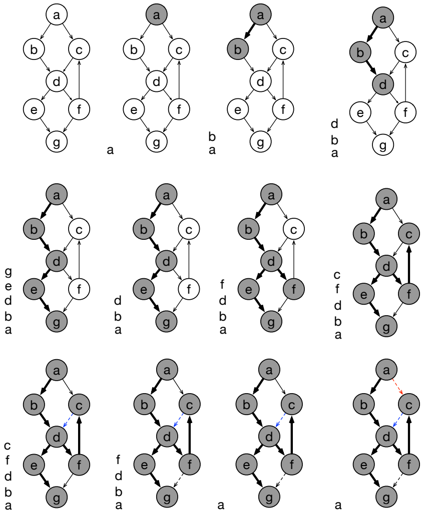
Let $G = (V, E)$ be a graph

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

Algorithm $BasicDFS(u)$:

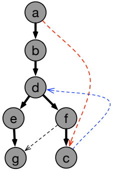
```
// Visit u
mark u as "visited"
for each v ∈ Successor(u) do
    // Explore the edge u → v
    if v is not marked "visited" then
        BasicDFS(v)
```


a
b a
d b a
e d b a
g e d b a
d b a
f d b a
c f d b a
c f d b a
f d b a
a
a



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: $i = 0$ ✓
  - Assume for $i - 1$ and prove for i: when we visit v_{i-1}, since $v_i \in \mathit{Successor}(v_{i-1})$, we explore the edge $v_{i-1} \to v_i$... either v_i 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

- We record information about the structure of the DFS forest using an array π indexed bt $v \in V$
- When we explore an edge $u \to v$ and discover a new, unvisited node v, we record the edge $u \to v$ by setting $\pi[v]$ to 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

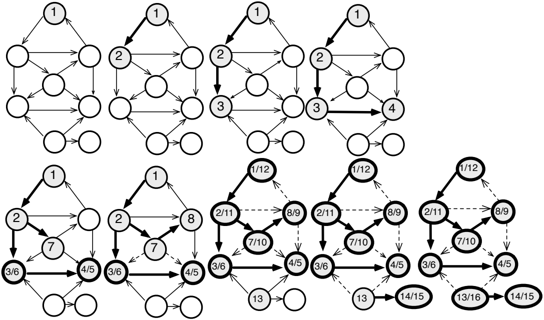
"Full" DFS

Algorithm $DFS(G)$:

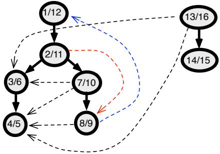
for each $v \in V$ do: $Color[v] \leftarrow white$, $\pi[v] \leftarrow Nil$
$time \leftarrow 0$
for each $v \in V$ do
    if $Color[v] = white$ then $RecDFS(v)$

Algorithm $RecDFS(u)$:

$Color[u] \leftarrow gray$
$d[u] \leftarrow ++time$ // discovery time
for each $v \in Successor(u)$ do:
    if $Color[v] = white$ then
        $\pi[v] \leftarrow u$, $RecDFS(v)$
$Color[u] \leftarrow black$
$f[u] \leftarrow ++time$ // 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 $= O(|V| + |E|)$

For $u, v \in V$, "$u \sqsubseteq v$" 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 $u \sqsupseteq v$ to mean $v \sqsubseteq u$, 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. $[d[u], f[u]] \cap [d[v], f[v]] = \emptyset$
   and neither $u \sqsubseteq v$ nor $v \sqsubseteq u$
2. $[d[u], f[u]] \subseteq [d[v], f[v]]$
   and $u \sqsubseteq v$
3. $[d[u], f[u]] \supseteq [d[v], f[v]]$
   and $u \sqsupseteq v$

Classification of edge $u \to v$

- Tree edge: in the DFS forest ($u \sqsupset v$)
  - v was *white* when $u \to v$ was explored;
    ($d[u] < d[v] < f[v] < f[u]$)
- Back edge: $u \sqsubseteq v$ (includes self loops)
  - v was *gray* when $u \to v$ was explored
    ($d[v] \leq d[u] < f[u] \leq f[v]$)
- Forward edge: a non-tree edge, $u \sqsupset v$
  - v was *black* when $u \to v$ was explored, but *white* when u was discovered ($d[u] < d[v] < f[v] < f[u]$)
- Cross edge: neither $u \sqsubseteq v$ nor $u \sqsupseteq v$
  - v was *black* when $u \to v$ was explored, and *black* when u was discovered; ($d[v] < f[v] < d[u] < f[u]$)
  - points "into the past" (right to left)

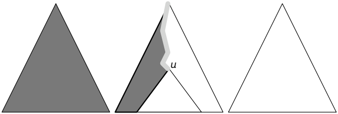
White Path Theorem

Let $u, v \in V$.

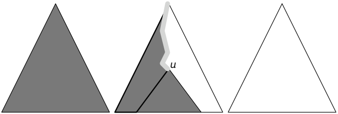
$$u \sqsupseteq v \Leftrightarrow \left\{ \begin{array}{l} \text{at the time } u \text{ is discovered, there is} \\ \text{a path from } u \text{ to } v \text{ consisting only of} \\ \textit{white} \text{ nodes} \end{array} \right.$$

Intuition:

u discovered:



u finished:



White Path Theorem

Let $u, v \in V$.

$$u \sqsupseteq v \Leftrightarrow \left\{ \begin{array}{l} \text{at the time } u \text{ is discovered, there is} \\ \text{a path from } u \text{ to } v \text{ consisting only of} \\ \textit{white} \text{ nodes} \end{array} \right.$$

($\Rightarrow$) Assume $u \sqsupseteq v$

The DFS tree path from u to v is the white path

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

Claim: $u \sqsupseteq v_i$ for all i

Proof by induction on i: $u \sqsupseteq v_0$ ✓

Assume $u \sqsupseteq v_{i-1}$ and show $u \sqsupseteq v_i$

By white path assumption: $d[u] \leq d[v_i]$ [v_i is white when u is discovered]

Since we explore the edge $v_{i-1} \to v_i$: $d[v_i] \leq f[v_{i-1}]$

By induction hypothesis and Parenthesis Theorem: $f[v_{i-1}] \leq f[u]$

[$v_{i-1} \sqsubseteq u \implies [d[v_{i-1}], f[v_{i-1}]] \subseteq [d[u], f[u]]$]

$\therefore d[u] \leq d[v_i] \leq f[u]$

By Parenthesis Theorem: $u \sqsupseteq v_i$

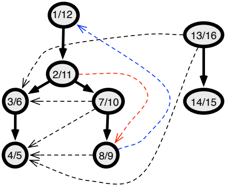
Topological Sorting — Tarjan's Algorithm

Algorithm DFSTopSort

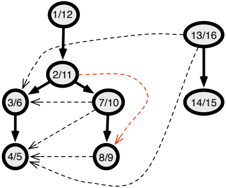
- initialize an empty list
- Run DFS: When a node is painted *black*, insert it at the front of the list
- If we ever discover a back edge, report that the graph is cyclic

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

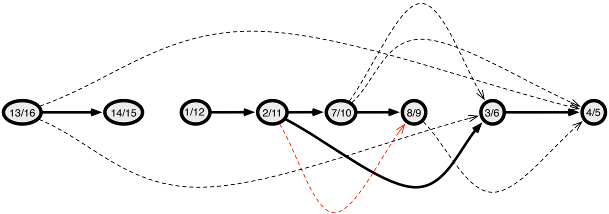
As a bonus, if there is a cycle, we can actually print it out



Let’s get rid of the back edge



Arrange from highest to lowest finishing time

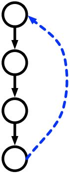


Lemma

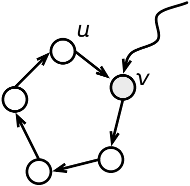
G has a cycle $\Leftrightarrow$ DFS produces a back edge

Proof:

- ($\Leftarrow$) A back edge trivially yields a cycle



- (⇒) Suppose G has a cycle C of vertices, and let v be the first first vertex discovered in C:



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

∴ the edge $u \to v$ 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: $u \sqsupset v$ 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: $u \sqsupset v$ and $d[u] < d[v] < f[v] < f[u]$
  - (u, v) is a cross edge: $f[v] < d[u] < f[u]$
- QED