Graphs

$G = (V, E)$, $V =$ set of nodes (a.k.a., vertices) $E =$ set of edges

$G$ is usually assumed to be directed, so that an edge is a pair of nodes $(u, v)$ (graphically, $u \to v$)

If $(u, v) \in E$, let’s call $v$ a successor of $u$, and $u$ a predecessor of $v$

$\text{Successor}(u) :=$ set of all successors of $u$

An undirected graph is just a special case of a directed graph, where $(u, v) \in E \Rightarrow (v, u) \in E$

One usually assumes an undirected graph contains no self loops, i.e., edges $(u, u)$
Representations

- **Sparse**: an array of adjacency lists
  
  an array $A$ indexed by $V$, where $A[u]$ is a linked list containing all successors of $u$
  
  size: $O(|V| + |E|)$
  
  this will be the “default”

- **Dense**: an boolean array $A$ indexed by $V \times V$, where $A[u, v] = \text{true}$ iff $(u, v) \in E$
  
  size: $O(|V|^2)$
Breadth first search (BFS)

Input: a graph $G = (V, E)$, and a node $s \in V$

Outputs:

- the “shortest distance” array $d$, indexed by $V$, so that $d[\nu] =$ length of shortest path from $s$ to $\nu$
- a “breadth first search” tree $T$, represented as an array $\pi$ indexed by $V$
  
  $\pi[\nu] = u$ means $u$ is $\nu$’s parent in $T$

the root $T$ is $s$, and paths in $T$ are shortest paths in $G$
Algorithm $BFS(G, s)$:

for each $v \in V$
    $Color[v] \leftarrow \text{white}$  // undiscovered
    $d[v] \leftarrow \infty$, $\pi[v] \leftarrow \text{Nil}$

$Color[s] \leftarrow \text{gray}$  // discovered
$d[s] \leftarrow 0$, $\pi[s] \leftarrow \text{Nil}$

$Q \leftarrow \text{NewQueue()}$  // a FIFO queue
$Q.enqueue(s)$

while not $Q.empty()$ do
    $u \leftarrow Q.dequeue()$
    for each $v \in \text{Successor}(u)$ do
        if $Color[v] = \text{white}$ then
            $Color[v] \leftarrow \text{gray}$  // discovered
            $d[v] \leftarrow d[u] + 1$, $\pi[v] \leftarrow u$
            $Q.enqueue(v)$
    $Color[u] \leftarrow \text{black}$  // finished
Example:

BFS Tree:
Running time:

- Each node enqueued at most once (by coloring)
- Each node dequeued at most
- Each adjacency list scanned at most once
- \( \therefore \) Running time = \( O(|V| + |E|) \)

Invariant:

- At the beginning of each loop iteration, \( Q \) contains all nodes that are colored *gray*. 
Correctness

Notation: \( d[\nu] = \text{computed distance} \)
\( \delta(s, \nu) = \text{length of shortest path from } s \text{ to } \nu \)

**Shortest Path Lemma**

If \( \delta(s, \nu) = m > 0 \), then \( \nu \) is the successor of some node \( u \) with \( \delta(s, u) = m - 1 \)

Proof:

• Consider a shortest path from \( s \) to \( \nu \):

\[
\begin{align*}
S & \rightarrow u \rightarrow \nu \\
\text{ } & \quad \text{ } \underline{m - 1} \\
\text{ } & \quad \text{ } \underline{m}
\end{align*}
\]

• The path \( S \rightarrow u \) must be a shortest path from \( s \) to \( u \) (otherwise, we could find an even shorter path to \( \nu \)). QED
Theorem
Algorithm BFS eventually discovers every node reachable from $s$

Prove by induction on $m$:

\[ \text{for all } v \in V, \text{ if } \delta(s, v) = m, \text{ then BFS discovers } v \]

$m = 0$: clear; $m > 0$:

- Suppose $v \in V$ with $\delta(s, v) = m$
- By Shortest Path Lemma, $v$ has a predecessor $u$ with $\delta(s, u) = m - 1$
- By induction, BFS discovered $u$, and placed $u$ in $Q$
- When BFS removes $u$ from $Q$, it discovers $v$ (or finds that it was already discovered)
Theorem

BFS correctly computes $d[\nu] = \delta(s, \nu)$ for all $\nu \in V$

Proof:

• Let $\nu_0, \nu_1, \ldots$ be the nodes listed in the order they are removed from $Q$

• We can partition the execution of BFS into epochs 0, 1, 2, ...

\[ \underbrace{\nu_0, \ldots, \nu_{j_0}}_{\text{epoch 0}}, \underbrace{\nu_{j_0+1}, \ldots, \nu_{j_1}}_{\text{epoch 1}}, \ldots \]

• A new epoch starts at $\nu_j$ if $\delta(s, \nu_j) \neq \delta(s, \nu_{j-1})$
Prove by induction on $i$:

At the beginning of epoch $i$, $Q$ contains precisely all nodes $v$ such that $\delta(s, v) = i$, and $d[v] = i$ for all these nodes

$i = 0$: clear

Assume for $0, \ldots, i$ and prove for $i + 1$:

- During epoch $i$, by the lemma, and the induction hypotheis, all nodes $v$ with $\delta(s, v) = i + 1$ will be discovered and placed at the end of $Q$ during epoch $i$

- Epoch $i$ ends when all nodes $v$ with $\delta(s, v) = i$ have been removed from $Q$

QED. One can also easily show that $T$ is correct
Depth First Search (DFS)

Algorithm $DFS(G)$:

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

$\textit{time} \leftarrow 0$

for each $v \in V$ do

if $Color[v] = \textit{white}$ then $\text{RecDFS}(v)$

Algorithm $\text{RecDFS}(u)$:

$Color[u] \leftarrow \textit{gray}$

$d[u] \leftarrow \textit{++time}$  // discovery time

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

if $Color[v] = \textit{white}$ then

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

$Color[u] \leftarrow \textit{black}$

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

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

- Each node is discovered once
- Each edge is traversed once
- Running time $= O(|V| + |E|)$
\(u\) discovered

- gray nodes are on run-time stack

\(u\) finished

Some Back, Forward, and Cross edges
For $u, v \in V$, “$u \subseteq v$” means that $u$ is a descendent of $v$ in the DFS forest (possibly $u = v$), and “$u \sqsubseteq v$” means $u$ is a proper descendent of $v$ (so $u \neq v$)

**Parenthesis Theorem**

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

1. $[d[u], f[u]] \cap [d[v], f[v]] = \emptyset$, $u \not\sqsubseteq v$, and $v \not\sqsubseteq u$
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 ($u \sqsubseteq v$)
  - $v$ 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, $u \sqsubseteq v$
  - $v$ was *black* when $u \rightarrow v$ was explored, but *white* when $u$ was discovered
    \[d[u] < d[v] < f[v] < f[u]\]

- **Cross edge:** $u \not\sqsubseteq v$ and $u \not\sqsubseteq v$
  - $v$ 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 $u, v \in V$.

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

$(\Rightarrow)$ Assume $u \geq v$
White Path Theorem

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

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

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

Claim: \( u \sqsupseteq v_i \) for all \( i \). Assume not, and let \( i \) be minimal such that \( u \not\sqsubseteq v_i \) (\( i > 0 \)) \( \Rightarrow \Leftarrow \)
Topological Sorting

Suppose $G = (V, E)$ is a DAG (Directed Acyclic Graph)

A topological sort of $G$ is an ordering of the vertices $\nu_1, \nu_2, \ldots, \nu_n$ such that $(\nu_i, \nu_j) \in E \Rightarrow i < j$

“all arrows go from left to right”

Algorithm TopSort

- initialize an empty list
- Run DFS: When a node is painted black, insert it at the front of the list

So we output vertices on order of decreasing finishing time
Lemma

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

Proof:

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

\[
\begin{tikzpicture}
  \node (A) at (0,0) {\circ};
  \node (B) at (1,1) {\circ};
  \node (C) at (1,-1) {\circ};
  \draw[->] (A) -- (B);
  \draw[->] (B) -- (C);
  \draw[->, dashed] (C) -- (A);
\end{tikzpicture}
\]
• (⇒) Suppose $G$ has a cycle $C$ of vertices, and let $v$ be the first vertex discovered in $C$:

By the White Path Theorem, $u$ is a descendent of $v$ in the DFS forest

∴ the edge $u \rightarrow v$ is a back edge
**Theorem**

Algorithm TopSort is correct

**Proof:**

- Let \((u, v) \in E\)
- We want to show \(f[v] < f[u]\)
- **Cases:**
  - \((u, v)\) is a tree edge: \(u \prec v\) 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: \(u \prec v\) and \(d[u] < d[v] < f[v] < f[u]\)
  - \((u, v)\) is a cross edge: \(f[v] < d[u] < f[u]\)
- QED