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 $A[u, v] = true$ 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 \textit{BFS}(G, s):

for each \( v \in V \)
\begin{align*}
Color[v] &\leftarrow \text{white} \quad // \text{undiscovered} \\
    d[v] &\leftarrow \infty, \pi[v] \leftarrow \text{Nil}
\end{align*}
\begin{align*}
Color[s] &\leftarrow \text{gray} \quad // \text{discovered} \\
d[s] &\leftarrow 0, \pi[s] \leftarrow \text{Nil}
\end{align*}

\begin{align*}
Q &\leftarrow \text{NewQueue()} \quad // \text{a FIFO queue} \\
Q.\text{enqueue}(s)
\end{align*}

while not Q.\text{empty}() do
\begin{align*}
u &\leftarrow Q.\text{dequeue}()
\end{align*}
\begin{align*}
\text{for each } v \in \text{Successor}(u) \text{ do}
    \text{if } Color[v] = \text{white} \text{ then}
    \begin{align*}
    Color[v] &\leftarrow \text{gray} \quad // \text{discovered} \\
d[v] &\leftarrow d[u] + 1, \pi[v] \leftarrow u \\
    Q.\text{enqueue}(v)
    \end{align*}
\end{align*}
\begin{align*}
Color[u] &\leftarrow \text{black} \quad // \text{finished}
\end{align*}
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 \textit{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 \( \nu \):
  \[
  s \rightarrow u \rightarrow \nu
  \]
  
- 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 $m$:

for all $v \in V$, if $\delta(s, v) = m$, 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 $d[v] = \delta(s, v)$ for all $v \in V$

Proof:

- Let $v_0, v_1, \ldots$ be the nodes listed in the order they are removed from $Q$
- We can partition the execution of BFS into epochs $0, 1, 2, \ldots$
  
  \[
  v_0, \ldots, v_{j_0}, \quad v_{j_0+1}, \ldots, v_{j_1}, \ldots
  \]
  
  \underline{epoch 0} \quad \underline{epoch 1}

- A new epoch starts at $v_j$ if $\delta(s, v_j) \neq \delta(s, v_{j-1})$
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 hypothesis, 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: $\text{Color}[v] \leftarrow \text{white}$, $\pi[v] \leftarrow \text{Nil}$
$\text{time} \leftarrow 0$
for each $v \in V$ do
  if $\text{Color}[v] = \text{white}$ then $\text{RecDFS}(v)$

Algorithm RecDFS($u$):

$\text{Color}[u] \leftarrow \text{gray}$
$d[u] \leftarrow ++\text{time}$  // discovery time
for each $v \in \text{Successor}(u)$ do:
  if $\text{Color}[v] = \text{white}$ then
    $\pi[v] \leftarrow u$, $\text{RecDFS}(v)$
$\text{Color}[u] \leftarrow \text{black}$
$f[u] \leftarrow ++\text{time}$  // 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
- grey 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. $[d[u], f[u]] \cap [d[v], f[v]] = \emptyset$, $u \not\subseteq v$, and $v \not\subseteq u$

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 $u \rightarrow \nu$

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

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

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

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

Let $u, v \in V$.

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

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

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

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

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

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

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

A topological sort of \( G \) is an ordering of the vertices \( v_1, v_2, \ldots, v_n \) such that \((v_i, v_j) \in E \implies i < j\)

“all arrows go from left to right”

Algorithm TopSort

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

So we output vertices on order of \textit{decreasing} finishing time
Lemma

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

Proof:

- $(\Leftarrow)$ 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 \( C \):

By the White Path Theorem, \( u \) is a descendent of \( v \) in the DFS forest.

\[ \therefore \text{the edge } u \to v \text{ is a back edge} \]
Theorem

Algorithm TopSort 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