Graphs

\[ G = (V, E), \quad V = \text{set of nodes (a.k.a., vertices)} \quad E = \text{set of edges} \]

\( G \) is usually assumed to be \textit{directed}, so that an edge is a pair of nodes \((u, \nu)\) (graphically, \(u \rightarrow \nu\))

If \((u, \nu) \in E\), let’s call \(\nu\) a \textit{successor} of \(u\), and \(u\) a \textit{predecessor} of \(\nu\)

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

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

One usually assumes an undirected graph contains no \textit{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 $d[v] =$ length of shortest path from $s$ to $v$
- a “breadth first search” tree $T$, represented as an array $\pi$ indexed by $V$
  
  $\pi[v] = u$ means $u$ is $v$’s parent in $T$

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

for each $v \in V$

$\text{Color}[v] \leftarrow \text{white} \quad \text{// undiscovered}$

$d[v] \leftarrow \infty$, $\pi[v] \leftarrow \text{Nil}$

$\text{Color}[s] \leftarrow \text{gray} \quad \text{// discovered}$

$d[s] \leftarrow 0$, $\pi[s] \leftarrow \text{Nil}$

$Q \leftarrow \text{NewQueue}() \quad \text{// a FIFO queue}$

$Q.\text{enqueue}(s)$

while not $Q.\text{empty}()$ do

$u \leftarrow Q.\text{dequeue}()$

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

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

$\text{Color}[v] \leftarrow \text{gray} \quad \text{// discovered}$

$d[v] \leftarrow d[u] + 1$, $\pi[v] \leftarrow u$

$Q.\text{enqueue}(v)$

$\text{Color}[u] \leftarrow \text{black} \quad \text{// finished}$
Example:

BFS Tree:
Invariant:

- At the beginning of each loop iteration, $Q$ contains all nodes that are colored gray.

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 $= O(|V| + |E|)$
Correctness

Notation: computed distance, $\delta(s, v) = \text{length of shortest path from } s \text{ to } v$

**Shortest Path Lemma.** if $\delta(s, v) = m > 0$, then $v$ is the successor of some node $u$ with $\delta(s, u) = m - 1$

- **Proof.** Consider a shortest path from $s$ to $v$:

  $$s \sim u \rightarrow v$$

  \[m-1\]

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

Prove by induction on \( m \):

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

\( m = 0 \): 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 $v \in V$

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* $0, 1, 2, \ldots$

\[
\underbrace{v_0, \ldots, v_{j_0}}_{\text{epoch 0}} \quad \underbrace{v_{j_0+1}, \ldots, v_{j_1}}_{\text{epoch 1}} \ldots
\]

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 $\delta[v] = i$ 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: $Color[v] \leftarrow \text{white}$, $\pi[v] \leftarrow \text{Nil}$

$\text{time} \leftarrow 0$

for each $v \in V$ do

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

Algorithm $RecDFS(u)$:

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

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

for each $v \in Succ(u)$ do:

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

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

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

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

- **Tree edge**
- **Forward edge**
- **Back edge**
- **Cross edge**
Invariant:

- At the beginning of each loop iteration, the gray nodes are the ancestors of \( u \) in the DFS forest, and these are also the nodes currently on the “recursion stack”

Running Time Analysis:

- Each node is discovered once
- Each edge is traversed once
- Running time = \( O(|V| + |E|) \)
For \( u, v \in V \), “\( u \sqsubseteq v \)” means that \( u \) is a descendent of \( v \) in the DFS forest (possibly \( u = v \)), and “\( u \sqsubset v \)” 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 \( u \sqsubseteq v \)
3. \([d[u], f[u]] \supseteq [d[v], f[v]]\), and \( u \sqsupseteq v \)
Classification of edge $u \rightarrow v$

- **Tree edge:** in the DFS forest ($u \cong v$)
  - $v$ was **white** when $u \rightarrow v$ was explored;
    $$(d[u] < d[v] < f[v] < f[u])$$
- **Back edge:** $u \sqsubseteq v$ (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, $u \cong v$
  - $v$ 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 $u \not\sqsupseteq v$
  - $v$ 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)
$u$ discovered

$u$ finished

Some Back, Forward, and Cross edges
White Path Theorem. Let $u, \nu \in V$.

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

(⇒) Assume $u \equiv \nu$
**White Path Theorem.** Let $u, v \in V$.

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

($\Leftarrow$) 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 $u \not\supseteq v_i \text{ (} i > 0 \text{)} \Rightarrow \Leftarrow$
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 $(v_i, v_j) \in E \Rightarrow i < j$

“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

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

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