Elementary Graph Algorithms
What is a graph?

A graph $G$ is represented as $G = (V,E)$

- $V$ is the set of vertices
- $E$ is the set of edges where an edge connects two vertices $(v_i,v_j)$ together
- An edge may be directed or undirected
Representation of Graphs

• 2 Standard Representations:
  – Adjacency Lists
    Compact way to represent \textit{sparse} graphs.
    Sparse - $|E| << |V|^2$
  – Adjacency Matrix
    Good representation for \textit{dense} graphs
    Dense - $|E| \sim |V|^2$
Figure 22.1 Two representations of an undirected graph. (a) An undirected graph $G$ having five vertices and seven edges. (b) An adjacency-list representation of $G$. (c) The adjacency-matrix representation of $G$. 
Figure 22.2  Two representations of a directed graph.  (a) A directed graph $G$ having six vertices and eight edges.  (b) An adjacency-list representation of $G$.  (c) The adjacency-matrix representation of $G$. 
Weighted Graphs

• *Weight function* $w: E \rightarrow \mathbb{R}$

• Weight can be stored in the adjacency list

• Weight can also be stored in the matrix with a weight 0 representing the non-existence of an edge
Breadth-first search

• Given G = (V,E) and a source vertex s
• Goal – Discover every vertex in G ‘reachable’ from s
• Compute the distance from s to each reachable vertex
• Produce a Breadth-first tree with s as the root, containing all reachable vertices
• The path from s to v represents the shortest path from s to v
Use of Colors

BFS uses colors to keep track of the discovery process:

• White – Undiscovered vertex
• Black – Discovered vertex all of whose adjacent vertices have also been discovered
• Grey – Discovered vertex some of whose adjacent vertices are white
More definitions

• When scanning a discovered node $v$’s adjacency list, if we come across a white node $w$
• Add edge $(v, w)$ to the tree.
• Call $v$ the predecessor of $\pi(w) \leftarrow v$
• $d(w)$ – the distance from $s$
BFS($G$, $s$)

1. for each vertex $u \in V[G] - \{s\}$
2. \hspace{1em} do $color[u] \leftarrow$ WHITE
3. \hspace{2em} $d[u] \leftarrow \infty$
4. \hspace{2em} $\pi[u] \leftarrow$ NIL
5. \hspace{1em} $color[s] \leftarrow$ GRAY
6. \hspace{1em} $d[s] \leftarrow 0$
7. \hspace{1em} $\pi[s] \leftarrow$ NIL
8. \hspace{1em} $Q \leftarrow \emptyset$
9. \hspace{1em} ENQUEUE($Q$, $s$)
10. while $Q \neq \emptyset$
11. \hspace{1em} do $u \leftarrow$ DEQUEUE($Q$)
12. \hspace{2em} for each $v \in \text{Adj}[u]$
13. \hspace{3em} do if $color[v] = \text{WHITE}$
14. \hspace{4em} then $color[v] \leftarrow$ GRAY
15. \hspace{4em} $d[v] \leftarrow d[u] + 1$
16. \hspace{4em} $\pi[v] \leftarrow u$
17. \hspace{4em} ENQUEUE($Q$, $v$)
18. \hspace{1em} $color[u] \leftarrow$ BLACK
Figure 22.3  The operation of BFS on an undirected graph. Tree edges are shown shaded as they are produced by BFS. Within each vertex $u$ is shown $d[u]$. The queue $Q$ is shown at the beginning of each iteration of the while loop of lines 10–18. Vertex distances are shown next to vertices in the queue.
Running Time

- Every vertex is enqueued at most once and so dequeued at most once.
- Total $O(V)$
- Each adjacency list scanned at most once
- Sum of the lists – $O(E)$
- Total time $O(V+E)$
Shortest Paths

• $\delta(s,v)$ shortest path from $s$ to $v$
• Minimum number of edges

Lemma – For any edge $(u,v)$ in $E$,

$$\delta(s,v) \leq \delta(s,u) + 1$$

• We want to show that BFS computes $d[v] = \delta(s,v)$ for every vertex in $V$
Lemma: Upon termination of BFS, for each vertex \( v \) in \( V \), \( d[v] \geq \delta(s,v) \)

Proof: Induction on number of ENQUEUE’s

Basis: After \( s \) in enqueued \( d[s]=0=\delta(s,s) \)

Inductive Step: Consider a white vertex \( v \) discovered during a search from some \( u \).

From hypothesis, \( d[u] \geq \delta(s,u) \)

And from the other lemma, \( d[v]=d[u]+1 \)

\[
\geq \delta(s,u)+1 \\
\geq \delta(s,v)
\]
Shortest Path

Lemma: Suppose the BFS queue contains \( <v_1, v_2, ..., v_r> \), then \( d[v_i] \leq d[v_{i+1}] \)

Monotonically increasing \( d \) values are in the queue over time
Correctness

Theorem: BFS run at $s$ will discover every reachable vertex $v$ and upon termination, $d[v]=\delta(s,v)$. Moreover, for any vertex $v$ reachable from $s$, one of the shortest from $s$ to $v$ is the shortest path from $s$ to $\pi(v)$ followed by the edge $(\pi(v),v)$.
PRINT-PATH\((G, s, v)\)

1 \hspace{1em} \textbf{if} \hspace{0.5em} v = s
2 \hspace{1em} \textbf{then} \hspace{0.5em} \text{print } s
3 \hspace{1em} \textbf{else} \hspace{0.5em} \textbf{if} \hspace{0.5em} \pi[v] = \text{NIL}
4 \hspace{1em} \textbf{then} \hspace{0.5em} \text{print "no path from" } s \text{ "to" } v \text{ "exists"}
5 \hspace{1em} \textbf{else} \hspace{0.5em} \text{PRINT-PATH}\((G, s, \pi[v])\)
6 \hspace{1em} \text{print } v