Graphs

A data structure for maintaining relational information

A graph $G=(V,E)$
- $V$: discrete set of **vertices** / nodes
- $E$: set of **edges** linking some pairs of vertices
For a graph $G=(V,E)$,

- an edge $e=(u,v)$ links/joins vertices $u$ and $v$
  - edges (hence graphs) may be directed or undirected
- $e$ is incident upon vertices $u$ and $v$
  - # of edges incident upon a vertex defines its degree
  - in- and out-degree for directed graphs
- two edges incident upon a vertex are adjacent
- $u$ and $v$ are neighboring vertices
- a path from $u$ to $v$ is an incident sequence of edges without any repetition
  - distance between $u$ and $v$ is the length of a shortest path between $u$ and $v$
Representation of graphs

■ Adjacency list
  - more popular (will be assumed)
  - much more efficient when $|E| \ll |V|^2$ (sparse)
  - easy to add weights for edges
  - size is $\theta(|V|+|E|)$

■ Adjacency matrix
  - could be preferred when $|E| \approx |V|^2$ (dense)
  - size is $\theta(|V|^2)$

■ Access efficiency vs memory requirements
  - to determine whether $(u,v) \in G$ is not $O(1)$ with adjacency lists
Representation of graphs
Representing attributes

- Normally need to store per node/edge attributes
  - \(v.d\) : an attribute \(d\) of vertex \(v\)
  - \((u,v).f\) : an attribute \(f\) of edge \((u,v)\)
  - associating them with graph objects might be tricky
    - use of separate data structures: \(d[1...|V|]\)
    - instance variables (e.g. of class `Vertex`)
    - others?
Breadth-first search

A simple algorithm to search a graph and basis for many useful graph algorithms

- Starts from a distinguished source vertex $s$
- Systematically explores edges to discover vertices by
  - expanding the frontier between discovered and undiscovered vertices uniformly across breadth of the frontier
  - vertices at distance $k$ from source discovered before those at distance $k+1$ from source
Breadth-first search

- Assumes adjacency lists
- Has per vertex attributes
  - $u.color$ : color of $u$
    - white, gray, and black
  - $u.\pi$ : predecessor of $u$
  - $u.d$ : distance from source
- Uses a FIFO queue $Q$

```
BFS(G, s)
1  for each vertex $u \in G.V - \{s\}$
2     $u.color = \text{WHITE}$
3     $u.d = \infty$
4     $u.\pi = \text{NIL}$
5  $s.color = \text{GRAY}$
6  $s.d = 0$
7  $s.\pi = \text{NIL}$
8  $Q = \emptyset$
9  \text{ENQUEUE}(Q, s)
10  while $Q \neq \emptyset$
11      $u = \text{DEQUEUE}(Q)$
12      for each $v \in G.Adj[u]$
13          if $v.color == \text{WHITE}$
14              $v.color = \text{GRAY}$
15              $v.d = u.d + 1$
16              $v.\pi = u$
17              \text{ENQUEUE}(Q, v)$
18      $u.color = \text{BLACK}$
```
Breadth-first search

\[
\begin{align*}
\text{Q} & \quad \begin{array}{c}
\text{Q} \quad \begin{array}{c}
\text{Q} \quad \begin{array}{c}
\end{align*}
\end{align*}
\end{align*}
\]
Breadth-first search

Ugur Dogrusoz

CS 502, Algorithms II, Bilkent Univ
Breadth-first search

Graph representation with vertex labels and queue (Q) updates.
Breadth-first search: analysis

- \( O(|V|+|E|) \) since
  - Initialization is \( \theta(|V|) \)
  - Each vertex enqueued and dequeued only once: \( O(|V|) \)
  - Each edge visited only once: \( \theta(|E|) \)
Breadth-first search

**Lemma** Let $G=(V,E)$ be directed or undirected graph, and let $s \in V$ be an arbitrary vertex. Then, for any edge $(u,v) \in E$,

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

**Proof** Consider both cases:
- $u$ is reachable from $s$,
- otherwise
Breadth-first search

- **Lemma** Let $G=(V,E)$ be directed or undirected graph, and suppose that BFS is run on $G$ from a given source vertex $s \in V$. Then, upon termination, for each vertex $v \in V$, the value of $v.d$ computed by BFS satisfies $v.d \geq \delta(s, v)$

- **Proof** Use induction on the number of `ENQUEUE` operations.
  - **Inductive step**: Consider a white vertex $v$ that is discovered during search from a vertex $u$
    \[
    v.d = u.d + 1 \\
    \geq \delta(s,u) + 1 \text{ (by Inductive Hypotheses)} \\
    \geq \delta(s,v) \text{ (by previous Lemma)}
    \]
  - $v$ is enqueued only once.
Breadth-first search

- **Lemma** Suppose that during BFS on a graph $G=(V,E)$, the queue $Q$ contains vertices $<v_1,v_2,...,v_r>$ where $v_1$ is the head of $Q$ and $v_r$ is the tail. Then

$$v_r.d \leq v_1.d + 1 \text{ and } v_i.d \leq v_{i+1}.d \text{ for } i = 1,2,...,r-1$$

- **Proof** Use induction on # of queue operations
  - On dequeue
    
    $$v_1.d \leq v_2.d \ldots \leq v_r.d \text{ by the I.H.}$$
    $$v_r.d \leq v_1.d + 1 \text{ by the I.H.}$$
    $$\Rightarrow v_r.d \leq v_1.d + 1 \leq v_2.d + 1$$
    $$\Rightarrow v_r.d \leq v_2.d + 1 \text{ I.S. satisfied for new head}$$
  - Enqueue is similar
Corollary Suppose that vertex \( v_i \) is enqueued before vertex \( v_j \) during BFS. Then, \( v_i.d \leq v_j.d \) at the time \( v_j \) is enqueued.

Proof Immediate from previous Lemma and the property that each vertex receives a finite \( d \) value at most once during BFS.
**Breadth-first search: correctness**

- **Theorem** During execution of BFS on $G=(V,E)$ from source $s \in V$, every vertex $v \in V$ that is reachable from $s$ is discovered, and upon termination, $v.d=\delta(s,v)$ for all $v \in V$. Moreover, for any $v \neq s$ that is reachable from $s$, one of the shortest paths from $s$ to $v$ is a shortest path from $s$ to $v.\pi$ followed by the edge $(v.\pi,v)$.

- **Proof** Follows from previous lemmas
Breadth-first search

- **Lemma** When applied to a directed or undirected graph $G=(V,E)$, procedure BFS constructs $\pi$ so that predecessor subgraph $G_{\pi}=(V_{\pi}, E_{\pi})$ is a breadth-first tree.

- **Proof** Apply previous theorem inductively
Breadth-first search

- Print out vertices on a shortest path from s to v (already computed breadth-first tree)

```
PRINT-PATH(G, s, v)
1   if v == s
2       print s
3   elseif v.\pi == NIL
4       print "no path from" s "to" v "exists"
5   else PRINT-PATH(G, s, v.\pi)
6       print v
```

- Runs in time linear in the length of the path
Depth-first search

- Search deeper in the graph whenever possible
  - Explore edges out of the most recently discovered vertex \( v \) that still has unexplored edges leaving it
  - Once all of \( v \)'s edges have been explored, backtrack to explore edges leaving the vertex from which \( v \) was discovered
  - Predecessor subgraph of DFS forms a depth-first forest
  - Records when it discovers and finishes a vertex \( u \) in attributes \( u.d \) and \( u.f \)
    - \( u \): white before \( u.d \), gray between \( u.d \) & \( u.f \), and black thereafter
    - \( u.d < u.f \) for each vertex \( u \)
Depth-first search

DFS($G$)
1. for each vertex $u \in G.V$
2. $u.color = \text{WHITE}$
3. $u.\pi = \text{NIL}$
4. $time = 0$
5. for each vertex $u \in G.V$
6. \[ \text{if } u.color == \text{WHITE} \]
7. \[ \text{DFS-VISIT}(G, u) \]

DFS-VISIT($G, u$)
1. $time = time + 1$ \hspace{1cm} // white vertex $u$ has just been discovered
2. $u.d = time$
3. $u.color = \text{GRAY}$
4. for each $v \in G.\text{Adj}[u]$ \hspace{1cm} // explore edge $(u, v)$
5. \[ \text{if } v.color == \text{WHITE} \]
6. \[ v.\pi = u \]
7. \[ \text{DFS-VISIT}(G, v) \]
8. $u.color = \text{BLACK}$ \hspace{1cm} // blacken $u$; it is finished
9. $time = time + 1$
10. $u.f = time$
Depth-first search
Depth-first search
Depth-first search: analysis

- Depth-first forest mirrors the structure of \texttt{DFS-Visit}
- \(O(|V|+|E|)\) since
  - \texttt{DFS-Visit} is called exactly once per vertex
  - lines 4-7 executes \(|\text{Adj}[v]|\) times and
    \[
    \sum_{v \in V} |\text{Adj}[v]| = \Theta(|E|)
    \]
Depth-first search: analysis

**Theorem (Parenthesis theorem)** In any DFS of a graph $G=(V,E)$, for any two vertices $u$ and $v$, exactly one of following holds:

- intervals $[u.d,u.f]$ and $[v.d,v.f]$ are entirely disjoint, and neither $u$ nor $v$ is a descendant of the other in the depth-first forest,
- interval $[u.d,u.f]$ is contained entirely within interval $[v.d,v.f]$, and $u$ is a descendant of $v$ in a depth-first tree, or vice versa.
Theorem (Parenthesis theorem)

Proof

W.l.o.g. suppose $u.d < v.d (< v.f)$. Then we have two cases:

- $v.d < u.f$: $v$ was discovered while $u$ was gray, thus $v$ is a descendant of $u$, thus $v$’s interval entirely contained within $u$’s
- $u.f < v.d$: means $u.d < u.f < v.d < v.f$, making two intervals disjoint
Depth-first search
Depth-first search
Depth-first search: analysis

- **Corollary (Nesting of descendants’ intervals)** Vertex $v$ is a proper descendant of vertex $u$ in the depth-first forest for a graph $G$ if and only if $u.d < v.d < v.f < u.f$.

- **Proof** Follows from Parenthesis theorem
Theorem (White path theorem) In a depth-first forest of a graph G=(V,E), vertex v is a descendant of vertex u if and only if at the time \(u.d\) that the search discovers u, there is a path from u to v consisting entirely of white vertices.

Proof

\(\Rightarrow\): if v is a proper descendant of u, then \(u.d < v.d\) and v is white at time \(u.d\) (by previous Corollary)

\(\Rightarrow\): similar argument
Depth-first search: analysis

1. **Tree edges**: edges \((u,v)\) in depth-first forest; \(v\) was first discovered by exploring edge \((u,v)\).

2. **Back edges**: edges \((u,v)\) connecting a vertex \(u\) to an ancestor \(v\) in a depth-first tree. Self-loops of directed graphs are back edges.

3. **Forward edges**: non-tree edges \((u,v)\) connecting a vertex \(u\) to a descendant \(v\) in a depth-first tree.

4. **Cross edges**: all other edges; they go between vertices in the same depth-first tree, as long as one vertex is not an ancestor of the other, or they can go between vertices in different depth-first trees.
Depth-first search: analysis

When we first explore an edge \((u,v)\), the color of vertex \(v\) tells us something about the edge:

- WHITE indicates a tree edge,
- GRAY indicates a back edge, and
- BLACK indicates a forward or cross edge.
Depth-first search: analysis

- **Theorem** In a depth-first search of an *undirected* graph $G$, every edge of $G$ is either a tree edge or a back edge.

- **Proof** Suppose w.l.o.g. $u.d < v.d$ for an edge $(u,v)$. Search must discover and finish $v$ before it finishes $u$ (since $v$ is on $u$’s adjacency list)
  - First time $(u,v)$ is explored from $u$ to $v$: $v$ is undiscovered (white), hence a tree edge
  - First time $(u,v)$ is explored from $v$ to $u$: $u$ is gray, hence a back edge
Topological sort

- A **linear ordering** of all vertices of a directed acyclic graph (dag) $G=(V,E)$ such that if $(u,v)$ in $V$, then $u$ appears before $v$ in the ordering.

- Not unique (partial vs. total order)
Topological sort

- Takes $O(V+E)$ since a straightforward DFS with $O(V)$ ($O(1)$ per vertex) extra processing performed

**TOPOLOGICAL-SORT($G$)**

1. call DFS($G$) to compute finishing times $v.f$ for each vertex $v$
2. as each vertex is finished, insert it onto the front of a linked list
3. **return** the linked list of vertices
Lemma A directed graph $G$ is acyclic if and only if a depth-first search of $G$ yields no back edges

Proof

- $\Rightarrow$: A back edge $(u,v)$ produced by a DFS implies $v$ is an ancestor of vertex $u$ in the depth-first forest, resulting in a path from $v$ to $u$, and the back edge $(u,v)$ completes a cycle, contradiction.

- $\Leftarrow$: Suppose $G$ contains a cycle $c$ and let $v$ be the first vertex discovered in $c$. Let $(u,v)$ be the preceding edge in $c$. At time $v.d$, the vertices of $c$ form a path of white vertices from $v$ to $u$. By the white-path theorem, vertex $u$ becomes a descendant of $v$ in the depth-first forest; hence $(u,v)$ is a back edge.
Theorem \texttt{Topological-Sort} produces a topological sort of the directed acyclic graph provided as its input.

Proof Need to show $v.f < u.f$ for any edge $(u,v)$ discovered by DFS. $v$ cannot be gray since $(u,v)$ cannot be a back edge (by previous Lemma):

- $v$ is white: $v$ is a descendant of $u$, so $v.f < u.f$
- $v$ is black: $v$ has been finished and $v.f$ has been set; still exploring from $u$, yet to assign a timestamp to $u$, thus we will have $v.f < u.f$
Another application of DFS to decompose a directed graph into strongly connected components, a maximal set of vertices $C$ in $V$ such that for every vertex pair $u$ and $v$ are reachable from each other in $C$. 
Strongly connected components

The transpose of a graph $G$ is $G^T=(V,E^T)$, where $E^T=\{(u,v) \mid (v,u) \text{ in } E\}$, edges of $G$ with their directions reversed.

Acyclic component graph $G^{\text{SCC}}$ obtained by contracting all edges within each strongly connected component of $G$ so that only a single vertex remains in each component.
**Strongly connected components**

\texttt{STRONGLY-CONNECTED-COMPONENTS}(G)

1. call DFS\((G)\) to compute finishing times \(u.f\) for each vertex \(u\)
2. compute \(G^T\)
3. call DFS\((G^T)\), but in the main loop of DFS, consider the vertices in order of decreasing \(u.f\) (as computed in line 1)
4. output the vertices of each tree in the depth-first forest formed in line 3 as a separate strongly connected component
Lemma 22.13 Let C and C’ be distinct strongly connected components in directed graph G=(V,E), with u and v in C and u’ and v’ in C’. Suppose G contains a path u -> u’. Then G cannot also contain a path v’ -> v.

Proof If G contains a path v’ -> v, then it contains paths u -> u’ -> v’ and v’ -> v -> u. Thus, u and v’ are reachable from each other, thereby contradicting the assumption that C and C’ are distinct strongly connected components.
Lemma 22.14 Let $C$ and $C'$ be distinct strongly connected components in directed graph $G=(V,E)$. Suppose that there is an edge $(u,v)$ in $E$, where $u$ in $C$ and $v$ in $C'$. Then $f(C) > f(C')$.

Proof

$\text{d}(C) < \text{d}(C')$: Let $x$ be the first vertex discovered in $C$. At time $x.d$, all vertices in $C$ and $C'$ are white. At that time, $G$ contains a path from $x$ to each vertex in $C$ consisting only of white vertices. Because $(u,v)$ in $E$, for any vertex $w$ in $C'$, there is also a path in $G$ at time $x.d$ from $x$ to $w$ consisting only of white vertices: $x \rightarrow u \rightarrow v \rightarrow w$. By the white-path theorem, all vertices in $C$ and $C'$ become descendants of $x$ in the depth-first tree. By previous corollary, $x$ has the latest finishing time of any of its descendants, and so $x.f = f(C) > f(C')$.
Strongly connected components

Proof cntd

- $d(C) > d(C')$: Let $y$ be the first vertex discovered in $C'$. At time $y . d$, all vertices in $C'$ are white and $G$ contains a path from $y$ to each vertex in $C'$ consisting only of white vertices. By the white-path theorem, all vertices in $C'$ become descendants of $y$ in the depth-first tree, and by previous corollary (nesting of descendants’ intervals), $y . f = f(C')$. At time $y . d$, all vertices in $C$ are white. Since there is an edge $(u, v)$ from $C$ to $C'$, Lemma 22.13 implies that there cannot be a path from $C'$ to $C$. Hence, no vertex in $C$ is reachable from $y$. At time $y . f$, therefore, all vertices in $C$ are still white. Thus, for any vertex $w$ in $C$, we have $w . f > y . f$, which implies that $f(C) > f(C')$. 
Corollary 22.15 Let \( C \) and \( C' \) be distinct strongly connected components in directed graph \( G=(V,E) \). Suppose that there is an edge \((u,v)\) in \( E^T \), where \( u \) in \( C \) and \( v \) in \( C' \). Then \( f(C) < f(C') \).

Proof Since \((u,v)\) in \( E^T \), we have \((v,u)\) in \( E \) (the strongly connected components of \( G \) and \( G^T \) are the same), Lemma 22.14 implies that \( f(C) < f(C') \).
Theorem 22.16 The Strongly-Connected-Components procedure correctly computes the strongly connected components of the directed graph G provided as its input.

Proof Use induction on the number of depth-first trees found in the depth-first search of $G^T$ in line 3:

- I.H.: First k trees produced in line 3 are strongly connected components
- Basis: k=0 is trivial
- I.S.: Consider the $(k+1)^{st}$ tree produced