Lecture 16: Directed Graphs
Outline

- Directed Graphs
  - Properties
  - Algorithms for Directed Graphs
    - DFS and BFS
    - Strong Connectivity
    - Transitive Closure
    - DAG and Topological Ordering
Digraphs

- A **digraph** is a graph whose edges are all directed
  - short for “directed graph”
- Applications
  - one-way streets
  - flights
  - task scheduling
Digraph Properties

- Each edge goes in one direction:
  - edge \((A,B)\) goes from \(A\) to \(B\)
  - edge \((B,A)\) goes from \(B\) to \(A\)
- If \(G\) is simple, \(m \leq n^*(n-1)\).
- Keep in-edges and out-edges in separate adjacency lists, can perform listing of in-edges and out-edges in time proportional to their size
  - \(\text{outgoingEdges}(A)\): \((A,C), (A,D), (A,B)\)
  - \(\text{ingoingEdges}(A)\): \((E,A),(B,A)\)
- Vertex \(w\) is reachable from vertex \(v\) if there is a \textbf{directed} path from \(v\) to \(w\)
  - \(E\) is reachable from \(A\)
  - \(E\) is not reachable from \(D\)
Directed DFS

- DFS and BFS traverse edges only along their direction
- In the directed DFS, have four types of edges:
  - discovery edges
  - back edges
  - forward edges
  - cross edges

- Directed DFS, starting at vertex \( s \) visits all vertices reachable from \( s \)
  - if \( u \) is reachable from \( s \) does not mean that \( s \) is reachable from \( u \)

Algorithm DFS(\( G, v \))

Input: digraph \( G \) and a start vertex \( v \) of \( G \)
Output: traverses vertices in \( G \) reachable from \( v \)

setLabel(\( v \), VISITED)
for all \( e \in G.outgoingEdges(v) \)
  \( w \leftarrow \text{opposite}(v, e) \)
  if getLabel(\( w \)) = UNEXPLORED
    DFS(\( G, w \))
Reachability

- **DFS tree** rooted at $v$: vertices reachable from $v$ via directed paths

  - DFS($G$, C)

- **BFS tree** rooted at $v$: vertices reachable from $v$ via directed paths

  - BFS($G$, B)
A graph is strongly connected if from any fixed vertex can reach all other vertices.
stronglyConnected = true
for every vertex v in G
    DFS(G,v)
    if some vertex w is not visited
        stronglyConnected = false

- DFS(G,v) visits every vertex reachable from v
- Simple, yet inefficient, $O((n+m)n)$
1) Pick a vertex \( v \) in \( G \)

2) Perform a DFS from \( v \) in \( G \)
   - If there is vertex \( w \) not visited, print “no” and terminate
   - else from \( v \) can reach any other graph vertex

3) Let \( G' \) be \( G \) with edges reversed

4) Perform DFS from \( v \) in \( G' \).
   - If there is path from \( v \) to \( w \) in \( G' \), then there is path from \( w \) to \( v \) in original graph \( G \)
   - If every vertex is reachable from \( v \) in \( G' \), then there is a path from any \( w \) to \( v \) in original graph \( G \)
     - then \( G \) is strongly connected
     - to find path between any \( b \) and \( d \): first go from \( b \) to \( v \), then go from \( v \) to \( d \)
   - Running time is \( O(n + m) \), perform DFS 2 times
Transitive Closure

- Given a digraph $G$, the transitive closure of $G$ is the digraph $G^*$ such that
  - $G^*$ has the same vertices as $G$
  - if $G$ has a directed path from $u$ to $v$ ($u \neq v$), $G^*$ has a directed edge from $u$ to $v$
- Transitive closure summarizes reachability in a digraph
Computing Transitive Closure

- Perform $\text{DFS}(G,v)$ for each vertex $v$

  \begin{verbatim}
  for all vertices $v$ in graph $G$
  $\text{DFS}(G,v)$
  for all vertices $w$ in graph $G$
  if $w$ is reachable from $v$
  put edge $(v,w)$ in $G^*$
  \end{verbatim}

- Running time is $O(n(n+m))$
  - $O(n^2)$ for sparse graphs, i.e. for a graph with $O(n)$ edges
  - $O(n^3)$ for dense graphs, i.e. for a graph with $O(n^2)$ edges
DAGs and Topological Ordering

- A **directed acyclic graph** (DAG) is a digraph that has no directed cycles.
- A **topological ordering** of a digraph is a numbering of vertices $1, 2, \ldots, n$
  
  such that for every edge $(v, w)$,
  
  $\text{number}(v) < \text{number}(w)$

- Example: task scheduling
Topological Ordering

- **Scheduling problem:**
  put edge \((a, b)\) if task \(a\) must be completed before \(b\) can be started.

- **Number vertices so that**
  \(u.\text{number} < v.\text{number}\)
  for any edge \((u, v)\)

A typical student day:

- wake up
- study computer sci.
- eat
- nap
- more c.s.
- play
- write c.s. program
- make cookies for professors
- sleep
- dream about graphs
- work out
Topological Ordering Theorem

**Theorem:** Digraph admits topological ordering if and only if it is a DAG

**Proof:**

Suppose graph is not DAG, then it has cycle \( v_1, v_2, ..., v_{n-1}, v_n, v_1 \). If topological ordering exists, then

\[
v_1.num < v_2.num < ... < v_{n-1}.num < v_n.num < v_1.num
\]

- If graph is DAG, there is a vertex \( v \) with no outgoing edges
  - if every vertex has an outgoing edge, during DFS we will encounter a back edge, which means cycle
  - This vertex can be the last one in topological ordering
  - Removing this vertex from the graph with incoming and outgoing edges leaves the graph acyclic

```plaintext
counter = n
repeat until counter = 0
  1. Find vertex \( v \) with no outgoing edges
  2. Set topological label of \( v \) to \( counter \)
  3. Remove \( v \) from the graph together with all incoming and outgoing edges.
  4. \( counter = counter - 1 \)
```
Topological Sorting Algorithm Using DFS

- Simulate the algorithm by using depth-first search
- \textit{counter} is a an instance (global) variable

Algorithm \textit{topologicalDFS}(G)

\begin{itemize}
  \item \textbf{Input} dag \( G \)
  \item \textbf{Output} topological ordering of \( G \)
\end{itemize}

\begin{verbatim}
counter \leftarrow G.numVertices()
for all \( u \in G.\text{vertices}() \)
  \text{setLabel}(u, \text{UNEXPLORED})
for all \( v \in G.\text{vertices}() \)
  if \text{getLabel}(v) = \text{UNEXPLORED}
    \text{topologicalDFS}(G, v)
\end{verbatim}

\begin{itemize}
  \item \( O(n + m) \) time.
\end{itemize}

Algorithm \textit{topologicalDFS}(G, v)

\begin{itemize}
  \item \textbf{Input} graph \( G \) and a start vertex \( v \)
  \item \textbf{Output} labeling of the vertices of \( G \) in the connected component of \( v \)
\end{itemize}

\begin{verbatim}
setLabel(v, \text{VISITED})
for all \( e \in G.\text{outgoingEdges}(v) \)
  \text{w} \leftarrow \text{opposite}(v, e)
  if \text{getLabel}(w) = \text{UNEXPLORED}
    \text{topologicalDFS}(G, w)
\end{verbatim}

\begin{verbatim}
v.\text{topologicalNumber} = \text{counter}
counter \leftarrow counter - 1
\end{verbatim}
Topological Sorting Example
Topological Sorting Example
Topological Sorting Example
Topological Sorting Example
Topological Sorting Example
Topological Sorting Example
Topological Sorting Example
Topological Sorting Example
Topological Sorting Example
Topological Sorting Example