Data Structures and Algorithms

Introduction to Graphs

Chris Brooks

Department of Computer Science
University of San Francisco
Graphs are one of the most important data structures.

They allow us to specify arbitrary relations between states or objects.

More general than trees, lists, or heaps.
A graph consists of:

- A set of **nodes** or **vertices** (terms are interchangable)
- A set of **edges** or **arcs** (terms are interchangable)

Edges in graph can be either directed or undirected
Edges can be labeled or unlabeled

- Edge labels are typically the cost associated with an edge

- e.g., Nodes are cities, edges are roads between cities, edge label is the length of road
There are several problems that are “naturally” graph problems

- Networking problems
  - Measuring traffic flow or bandwidth
- Route planning
- AI search problems
- Modeling computer programs or processes
- Finding sequences of tasks

Problems that don’t seem like graph problems can also be solved with graphs
- Register allocation using graph coloring
18-4: Connected Undirected Graph

Path from every node to every other node

Connected
18-5: Connected Undirected Graph

- Path from every node to every other node

Connected
18-6: *Connected Undirected Graph*

- Path from every node to every other node

\[ \begin{align*}
&\text{1} \\
&\text{2} \quad \text{3} \\
&\text{4} \quad \text{5}
\end{align*} \]

- *Not Connected*
**18-7: Strongly Connected Graph**

- Directed Path from every node to every other node

- Strongly Connected
18-8: **Strongly Connected Graph**

- Directed Path from every node to every other node

- Strongly Connected

```
Directed Graph

1 -> 2
1 -> 3
2 -> 1
3 -> 1
4 -> 1
5 -> 4
```

Department of Computer Science — University of San Francisco — p.10/56
18-9: Strongly Connected Graph

- Directed Path from every node to every other node

- Not Strongly Connected
Directed graph w/ connected backbone

Weakly Connected

1 2 3 4 5
6 Undirected cycles

Contains an undirected cycle
6 Undirected cycles

6 Contains an undirected cycle
6 Undirected cycles

Contains *no* undirected cycle
6 Undirected cycles

Contains *no* undirected cycle
Directed cycles

Contains a directed cycle
Directed cycles

Contains a directed cycle
Directed cycles

Contains a directed cycle
Directed cycles

Contains \textit{no} directed cycle
Must a connected, undirected graph contain a cycle?
18-20: Cycles & Connectivity

6 Must a connected, undirected graph contain a cycle?
   △ No.

6 Can an unconnected, undirected graph contain a cycle?
Must a connected, undirected graph contain a cycle?
△ No.

Can an unconnected, undirected graph contain a cycle?
△ Yes.

Must a strongly connected graph contain a cycle?
Must a connected, undirected graph contain a cycle?
△ No.

Can an unconnected, undirected graph contain a cycle?
△ Yes.

Must a strongly connected graph contain a cycle?
△ Yes! (why?)
6 If a graph is weakly connected, and contains a cycle, must it be strongly connected?
If a graph is weakly connected, and contains a cycle, must it be strongly connected?

No.
If a graph is weakly connected, and contains a cycle, must it be strongly connected?
   ▲ No.

If a graph contains a cycle which contains all nodes, must the graph be strongly connected?
If a graph is weakly connected, and contains a cycle, must it be strongly connected?
- No.

If a graph contains a cycle which contains all nodes, must the graph be strongly connected?
- Yes. (why?)
Adjacency Matrix

Represent a graph with a two-dimensional array $G$
- $G[i][j] = 1$ if there is an edge from node $i$ to node $j$
- $G[i][j] = 0$ if there is no edge from node $i$ to node $j$

If graph is undirected, matrix is symmetric

Can represent edges labeled with a cost as well:
- $G[i][j] =$ cost of link between $i$ and $j$
- If there is no direct link, $G[i][j] = \infty$
6. Examples:

\[ \begin{array}{cccc}
0 & 1 & 2 & 3 \\
0 & 0 & 1 & 0 & 1 \\
1 & 1 & 0 & 1 & 1 \\
2 & 0 & 1 & 0 & 0 \\
3 & 1 & 1 & 0 & 0 \\
\end{array} \]
Examples:

\[
\begin{array}{cccc}
0 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 \\
2 & 0 & 0 & 0 \\
3 & 1 & 0 & 0 \\
\end{array}
\]
Examples:

```
  0 1 2 3
-0-0-0-0-
1-1-1-0-0-
2-0-1-0-0-
3-0-0-0-1-
```
18-31: Adjacency Matrix

6 Examples:

\[
\begin{array}{c|c|c|c}
0 & 1 & 2 & 3 \\
\hline
0 & \infty & \infty & \infty & 5 \\
1 & 4 & \infty & \infty & \infty \\
2 & \infty & 7 & \infty & \infty \\
3 & \infty & \infty & -2 & \infty \\
\end{array}
\]
Adjacency List

Maintain a linked-list of the neighbors of every vertex.

- \( n \) vertices
- Array of \( n \) lists, one per vertex
- Each list \( i \) contains a list of all vertices adjacent to \( i \).
6 Examples:
Examples:

0 1
2 3

Note – lists are not always sorted
Sparse graph – relatively few edges

Dense graph – lots of edges

Complete graph – contains all possible edges

△ These terms are fuzzy. “Sparse” in one context may or may not be “sparse” in a different context
If nodes are labeled with strings instead of integers

- Internally, nodes are still represented as integers
- Need to associate string labels & vertex numbers
  - Vertex number $\rightarrow$ label
  - Label $\rightarrow$ vertex number
Vertex numbers → labels
Vertex numbers → labels

- Array
  - Vertex numbers are indices into array
  - Data in array is string label
6 Labels → vertex numbers
6. Labels $\rightarrow$ vertex numbers
   - Use a hash table
     - Key is the vertex label
     - Data is vertex number

Examples!
Directed Acyclic Graph, Vertices $v_1 \ldots v_n$

Create an ordering of the vertices

- If there a path from $v_i$ to $v_j$, then $v_i$ appears before $v_j$ in the ordering

Prerequisite chains

Scheduling jobs
Which node(s) could be first in the topological ordering?
Which node(s) could be first in the topological ordering?

- Node with no incident (incoming) edges
Pick a node \( v_k \) with no incident edges

Add \( v_k \) to the ordering

Remove \( v_k \) and all edges from \( v_k \) from the graph

Repeat until all nodes are picked.
How can we find a node with no incident edges?
Count the incident edges of all nodes
for (i=0; i < NumberOfVertices; i++)
    NumIncident[i] = 0;

for (i=0; i < NumberOfVertices; i++)
    each node k adjacent to i
    NumIncident[k]++
for (i = 0; i < NumberOfVertices; i++)
    NumIncident[i] = 0;

for (i = 0; i < NumberOfVertices; i++)
    for (tmp = G[i]; tmp != null; tmp = tmp.next())
        NumIncident[tmp.neighbor()]++
18-48: Topological Sort

6 Create NumIncident array

6 Repeat
   △ Search through NumIncident to find a vertex \( v \) with NumIncident[\( v \)] == 0
   △ Add \( v \) to the ordering
   △ Decrement NumIncident of all neighbors of \( v \)
   △ Set NumIncident[\( v \)] = -1

6 Until all vertices have been picked
In a graph with $V$ vertices and $E$ edges, how long does this version of topological sort take?
In a graph with $V$ vertices and $E$ edges, how long does this version of topological sort take?

\[ \Theta(V^2 + E) = \Theta(V^2) \]

- Since $E \in O(V^2)$
6 Where are we spending “extra” time
Where are we spending “extra” time

- Searching through NumIncident each time looking for a vertex with no incident edges
- Keep around a set of all nodes with no incident edges
- Remove an element $v$ from this set, and add it to the ordering
- Decrement NumIncident for all neighbors of $v$
  - If NumIncident[$k$] is decremented to 0, add $k$ to the set.
- How do we implement the set of nodes with no incident edges?
6 Where are we spending “extra” time
   ▲ Searching through NumIncident each time looking for a vertex with no incident edges
   ▲ Keep around a set of all nodes with no incident edges
   ▲ Remove an element $v$ from this set, and add it to the ordering
   ▲ Decrement NumIncident for all neighbors of $v$
     ▲ If NumIncident[$k$] is decremented to 0, add $k$ to the set.
   ▲ How do we implement the set of nodes with no incident edges?
     ▲ Use a stack
6 Examples!!

- Graph
- Adjacency List
- NumIncident
- Stack