Graph Terminology

CSE 373
Data Structures
Lecture 13
Reading

• Reading
  › Section 9.1
What are graphs?

• Yes, this is a graph....

• But we are interested in a different kind of “graph”
Graphs

• Graphs are composed of
  › Nodes (vertices)
  › Edges (arcs)
Varieties

- Nodes
  - Labeled or unlabeled
- Edges
  - Directed or undirected
  - Labeled or unlabeled
Motivation for Graphs

- Consider the data structures we have looked at so far…

  - **Linked list**: nodes with 1 incoming edge + 1 outgoing edge

  - **Binary trees/heaps**: nodes with 1 incoming edge + 2 outgoing edges

  - **B-trees**: nodes with 1 incoming edge + multiple outgoing edges
Motivation for Graphs

• How can you generalize these data structures?
• Consider data structures for representing the following problems…
CSE Course Prerequisites at UW

Nodes = courses
Directed edge = prerequisite
Representing a Maze

Nodes = rooms
Edge = door or passage
Representing Electrical Circuits

Nodes = battery, switch, resistor, etc.
Edges = connections
Program statements

\[ x_1 = q + y \times z \]
\[ x_2 = y \times z - q \]

Naive:

\[
\begin{align*}
\text{y} \times \text{z} & \text{ calculated twice} \\
\text{common} & \text{ subexpression} \\
\text{eliminated:} & \\
\end{align*}
\]

Nodes = symbols/operators
Edges = relationships
Precedence

\[ S_1 \quad a=0; \]
\[ S_2 \quad b=1; \]
\[ S_3 \quad c=a+1 \]
\[ S_4 \quad d=b+a; \]
\[ S_5 \quad e=d+1; \]
\[ S_6 \quad e=c+d; \]

Which statements must execute before \( S_6 \)?
\( S_1, S_2, S_3, S_4 \)

Nodes = statements
Edges = precedence requirements
Information Transmission in a Computer Network

Seoul

Tokyo

Sydney

56

16

30

181

128

140

L.A.

New York

Nodes = computers
Edges = transmission rates

12/26/03
Graph Terminology - Lecture 13
Traffic Flow on Highways

Nodes = cities
Edges = # vehicles on connecting highway
Graph Definition

• A graph is simply a collection of nodes plus edges
  › Linked lists, trees, and heaps are all special cases of graphs
• The nodes are known as vertices (node = “vertex”)
• Formal Definition: A graph $G$ is a pair $(V, E)$ where
  › $V$ is a set of vertices or nodes
  › $E$ is a set of edges that connect vertices
Graph Example

• Here is a directed graph $G = (V, E)$
  › Each edge is a pair $(v_1, v_2)$, where $v_1, v_2$ are vertices in $V$
  › $V = \{A, B, C, D, E, F\}$
  $E = \{(A,B), (A,D), (B,C), (C,D), (C,E), (D,E)\}$
Directed vs Undirected Graphs

- If the order of edge pairs \((v_1, v_2)\) matters, the graph is directed (also called a digraph): \((v_1, v_2) \neq (v_2, v_1)\)

- If the order of edge pairs \((v_1, v_2)\) does not matter, the graph is called an undirected graph: in this case, \((v_1, v_2) = (v_2, v_1)\)
Undirected Terminology

- Two vertices $u$ and $v$ are adjacent in an undirected graph $G$ if $\{u,v\}$ is an edge in $G$
  - edge $e = \{u,v\}$ is incident with vertex $u$ and vertex $v$
- The degree of a vertex in an undirected graph is the number of edges incident with it
  - a self-loop counts twice (both ends count)
  - denoted with $\text{deg}(v)$
Undirected Terminology

(A,B) is incident to A and to B

B is adjacent to C and C is adjacent to B

Self-loop

Degree = 3

Degree = 0
Directed Terminology

• Vertex u is adjacent to vertex v in a directed graph G if (u,v) is an edge in G
  › vertex u is the initial vertex of (u,v)
• Vertex v is adjacent from vertex u
  › vertex v is the terminal (or end) vertex of (u,v)
• Degree
  › in-degree is the number of edges with the vertex as the terminal vertex
  › out-degree is the number of edges with the vertex as the initial vertex
Directed Terminology

B adjacent to C and C adjacent from B

In-degree = 2
Out-degree = 1

In-degree = 0
Out-degree = 0
Handshaking Theorem

• Let $G=(V,E)$ be an undirected graph with $|E|=e$ edges. Then

\[ 2e = \sum_{v \in V} \deg(v) \]

Add up the degrees of all vertices.

• Every edge contributes +1 to the degree of each of the two vertices it is incident with
  > number of edges is exactly half the sum of $\deg(v)$
  > the sum of the $\deg(v)$ values must be even
Graph Representations

- Space and time are analyzed in terms of:
  - Number of vertices = $|V|$ and
  - Number of edges = $|E|$
- There are at least two ways of representing graphs:
  - The *adjacency matrix* representation
  - The *adjacency list* representation
Adjacency Matrix

\[ M(v, w) = \begin{cases} 
1 & \text{if } (v, w) \text{ is in } E \\
0 & \text{otherwise} 
\end{cases} \]

\[
\begin{bmatrix}
\begin{array}{ccccccc}
A & B & C & D & E & F \\
A & 0 & 1 & 0 & 1 & 0 & 0 \\
B & 1 & 0 & 1 & 0 & 0 & 0 \\
C & 0 & 1 & 0 & 1 & 1 & 0 \\
D & 1 & 0 & 1 & 0 & 1 & 0 \\
E & 0 & 0 & 1 & 1 & 0 & 0 \\
F & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\end{bmatrix}
\]

\[ \text{Space} = |V|^2 \]
## Adjacency Matrix for a Digraph

The adjacency matrix for the given digraph is:

\[
\begin{pmatrix}
A & B & C & D & E & F \\
A & 0 & 1 & 0 & 1 & 0 & 0 \\
B & 0 & 0 & 1 & 0 & 0 & 0 \\
C & 0 & 0 & 0 & 1 & 1 & 0 \\
D & 0 & 0 & 0 & 0 & 1 & 0 \\
E & 0 & 0 & 0 & 0 & 0 & 0 \\
F & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{pmatrix}
\]

The space of the adjacency matrix is \( |V|^2 \):

\[
\text{Space} = |V|^2
\]

The adjacency matrix is defined as:

\[
M(v, w) = \begin{cases} 
1 & \text{if } (v, w) \text{ is in } E \\
0 & \text{otherwise}
\end{cases}
\]
Adjacency List

For each $v$ in $V$, $L(v) =$ list of $w$ such that $(v, w)$ is in $E$

![Diagram of a graph with nodes A, B, C, D, E, F and their adjacency list representation.]

Space = $a |V| + 2 b |E|$
Adjacency List for a Digraph

For each \( v \) in \( V \), \( L(v) \) = list of \( w \) such that \((v, w)\) is in \( E \)

```
A -> B
A -> C
D -> E
C -> D
C -> F
```

```
\[ \text{Space} = a \cdot |V| + b \cdot |E| \]
```