Graphs

Outline and Reading

Graphs (§6.1)
- Definition
- Applications
- Terminology
- Properties
- ADT

Data structures for graphs (§6.2)
- Edge list structure
- Adjacency list structure
- Adjacency matrix structure

Graph

Graphs (§6.1)
- Definition
- Applications
- Terminology
- Properties
- ADT

Data structures for graphs (§6.2)
- Edge list structure
- Adjacency list structure
- Adjacency matrix structure

Outline and Reading

A graph is a pair \((V, E)\), where

- \(V\) is a set of nodes, called vertices
- \(E\) is a collection of pairs of vertices, called edges
- Vertices and edges are positions and store elements

Example:

- A vertex represents an airport and stores the three-letter airport code
- An edge represents a flight route between two airports and stores the mileage of the route

Edge Types

Directed edge
- ordered pair of vertices \((u, v)\)
  - first vertex \(u\) is the origin
  - second vertex \(v\) is the destination
  - e.g., a flight

Undirected edge
- unordered pair of vertices \((u, v)\)
  - e.g., a flight route

Directed graph
- all the edges are directed
  - e.g., flight network

Undirected graph
- all the edges are undirected
  - e.g., route network

Applications

Electronic circuits
- Printed circuit board
- Integrated circuit

Transportation networks
- Highway network
- Flight network

Computer networks
- Local area network
- Internet
- Web

Databases
- Entity-relationship diagram

Terminology

End vertices (or endpoints) of an edge

Adjacent vertices
- \(U\) and \(V\) are adjacent

Degree of a vertex
- \(X\) has degree 5

Parallel edges
- \(h\) and \(i\) are parallel edges

Self-loop
- \(j\) is a self-loop
**Terminology (cont.)**

- **Path**
  - sequence of alternating vertices and edges
  - begins with a vertex
  - ends with a vertex
  - each edge is preceded and followed by its endpoints
- **Simple path**
  - path such that all its vertices and edges are distinct
- **Examples**
  - \( P_1 = (V, b, X, h, Z) \) is a simple path
  - \( P_2 = (U, c, W, e, X, g, Y, f, W, d, V) \) is a path that is not simple

**Cycle**

- **Circular sequence of alternating vertices and edges**
- **Each edge is preceded and followed by its endpoints**
- **Simple cycle**
  - cycle such that all its vertices and edges are distinct
- **Examples**
  - \( C_1 = (V, b, X, g, Y, f, W, c, U, a, \ldots) \) is a simple cycle
  - \( C_2 = (U, c, W, e, X, g, Y, f, W, d, V, a, \ldots) \) is a cycle that is not simple

**Properties**

**Property 1**

\[
\sum v \text{deg}(v) = 2m
\]

**Proof:** Each edge is counted twice.

**Property 2**

In an undirected graph with no self-loops and no multiple edges, \( m \leq \binom{n}{2} \).

**Proof:** Each vertex has degree at most \( n-1 \).

What is the bound for a directed graph?

**Main Methods of the Graph ADT**

- **Vertices and edges**
  - are positions
  - store elements
- **Accessor methods**
  - \( \text{aVertex()} \)
  - \( \text{incidentEdges(v)} \)
  - \( \text{endVertices(e)} \)
  - \( \text{isDirected(e)} \)
  - \( \text{origin(e)} \)
  - \( \text{destination(e)} \)
  - \( \text{opposite(v, e)} \)
  - \( \text{areAdjacent(v, w)} \)
- **Update methods**
  - \( \text{insertVertex(o)} \)
  - \( \text{insertEdge(v, w, o)} \)
  - \( \text{insertDirectedEdge(v, w, o)} \)
  - \( \text{removeVertex(v)} \)
  - \( \text{removeEdge(e)} \)
- **Generic methods**
  - \( \text{numVertices()} \)
  - \( \text{numEdges()} \)
  - \( \text{vertices()} \)
  - \( \text{edges()} \)

**Edge List Structure**

- **Vertex object**
  - element
  - reference to position in vertex sequence
- **Edge object**
  - element
  - origin vertex object
  - destination vertex object
  - reference to position in edge sequence
- **Vertex sequence**
  - sequence of vertex objects
- **Edge sequence**
  - sequence of edge objects

**Adjacency List Structure**

- **Incidence sequence**
  - for each vertex
  - sequence of references to edge objects of incident edges
- **Augmented edge objects**
  - references to associated positions in incidence sequences of end vertices
Adjacency Matrix Structure

- Edge list structure
- Augmented vertex objects
  - Integer key (index) associated with vertex
- 2D adjacency array
  - Reference to edge object for adjacent vertices
  - Null for non-adjacent vertices
- The "old fashioned" version just has 0 for no edge and 1 for edge

Asymptotic Performance

<table>
<thead>
<tr>
<th>Operation</th>
<th>Edge List</th>
<th>Adjacency List</th>
<th>Adjacency Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Space</td>
<td>$n + m$</td>
<td>$n + m$</td>
<td>$n^2$</td>
</tr>
<tr>
<td>incidentEdges(v)</td>
<td>$m$</td>
<td>$\deg(v)$</td>
<td>$n$</td>
</tr>
<tr>
<td>areAdjacent(v, w)</td>
<td>$m$</td>
<td>$\min(\deg(v), \deg(w))$</td>
<td>1</td>
</tr>
<tr>
<td>insertVertex(v)</td>
<td>1</td>
<td>1</td>
<td>$n^2$</td>
</tr>
<tr>
<td>insertEdge(v, w, o)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>removeVertex(v)</td>
<td>$m$</td>
<td>$\deg(v)$</td>
<td>$n^2$</td>
</tr>
<tr>
<td>removeEdge(e)</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>