COSC 2011 Section N

Thursday, May 3 2001

Overview

- Graphs
  - Review
    - Definitions, Terminology
  - ADT
  - Data Structures for Graphs

Graphs - Definitions: (1)

- endVertices, endPoints:
  - Two vertices joined by an edge.
- origin:
  - First endpoint of a directed edge.
- destination:
  - Second endpoint of a directed edge.

Graphs - Definitions: (2)

- adjacent:
  - Two vertices are “adjacent” if they are connected by the same edge.
- incident:
  - An edge is “incident” on a vertex if the vertex is an endpoint of the edge.
- Outgoing edges of a vertex:
  - Directed edges whose origin is that vertex.

Graphs - Definitions: (3)

- Incoming edges of a vertex:
  - Directed edges whose destination is that vertex.
- Degree of a vertex v:
  - Number of incident edges of v.
  - Denoted by $\deg(v)$.
- in-degree of a vertex v:
  - Number of incoming edges of v. Denoted by $\text{indeg}(v)$.
Graphs - Definitions: (4)

- **out-degree of a vertex** \(v\):  
  - Number of outgoing edges of \(v\). Denoted by \(\text{outdeg}(v)\).
Graphs – ADT:

- Many, many methods!

- Three Main Categories of Methods:
  - General methods
  - Methods dealing with directed edges.
  - Methods for updating and modifying graphs.

The Graph ADT

- The Graph ADT is a positional container whose positions are the vertices and the edges of the graph.
  - size() Return the number of vertices plus the number of edges of G.
  - isEmpty() - elements() - position() - graph() - replaceElement()

Notation: Graph G; Vertices v, w; Edge e; Object o
  - numVertices() Return the number of vertices of G.
  - numEdges() Return the number of edges of G.
  - vertices() Return an enumeration of the vertices of G.
  - edges() Return an enumeration of the edges of G.

More Methods ...

- adjacent(Vertice(v)) Return an enumeration of the vertices adjacent to v.
- inAdjacentVertices(v) Return an enumeration of the vertices adjacent to v along incoming edges.
- outAdjacentVertices(v) Return an enumeration of the vertices adjacent to v along outgoing edges.
- areAdjacent(v,w) Return whether vertices v and w are adjacent.
- endVertices(e) Return an array of size 2 storing the end vertices of e.
- origin(e) Return the end vertex from which e leaves.
- destination(e) Return the end vertex at which e arrives.
- isDirected() Return true if e is directed.
Graphs – Data Structures:

- Three Main Approaches:
  - Edge List.
  - Adjacency List.
  - Adjacency Matrix.

Graphs – Edge List: (1)

- Simplest but not efficient.
- Vertex \( v \) storing element \( o \) is represented by vertex object.
- Vertex Objects:
  - Stored in container \( V \).
  - A reference to \( o \).
  - Counters for number of incident undirected edges, incoming directed edges, outgoing directed edges.
Graphs – Edge List: (2)

- Distinguishing feature is how it represents edges!
- Edge $e$ storing element $o$ is represented by edge object.
  - Edge objects are also stored in a container $E$.
- Edge Objects:
  - Reference to $o$.
  - Boolean indicator tells if edge is directed.

Graphs – Edge List: (3)

- References to the vertex objects in $V$ associated with the endpoint vertices of $e$ (undirected) or references to the $origin$ and $destination$ (directed).

Graphs – Edge List: (4)

- Provides direct access to the edges and to the two vertices it is adjacent to.
- Allows for simple and efficient algorithms for edge based methods of the ADT:
  - $endVertices$, $origin$ and $destination$.
- Not efficient when we want to access edges that are incident to some vertex.

Graphs – Edge List: (5)

- Need to look through all the edges!
  - $incidentEdges(v)$ requires a search to look for all edges incident to $v$.
  - $areAdjacent(v, w)$ also requires a search!
  - $removeVertex$ also requires a search of edges.
Graphs – Adjacency List:

1. Extends the edge list structure except it adds extra information to support direct access to incident edges of each vertex.

2. Includes all structures of edge list including:
   - Each vertex object $v$ holds reference to container $I(v)$ that stores references to the edges incident on $v$.

Graphs – Adjacency List:

- If directed edges, the $I(v)$ splits into $I_{in}(v)$, $I_{out}(v)$ and $I_{un}(v)$.

- Provides direct access from the edges to the vertices and from the vertices to the edges.

- Allows for speed up for several methods.
Graphs – Adjacency Matrix: (1)

- Extends the edge structure with additional component:
  - Matrix $A$ allows to determine adjacencies between pairs of vertices in constant time.
  - Think of the vertices as being integers in the set \{0, 1, \ldots, n-1\}.
  - Edges are pairs of these integers.
Graphs – Adjacency Matrix: (1)

- Vertex object \( v \) also stores a distinct integer key in the range \( \{0, 1, \ldots, n-1\} \), called the *index* of \( v \).

- Keep a 2D \( n \times n \) array \( A \) such that cell \( A[i,j] \) holds reference to edge \( e \) that goes from vertex \( i \) to vertex \( j \) if such an edge exists.
  - If \( e \) is undirected store reference to \( A[i,j] \) and \( A[j,i] \)!

**Performance of the Adjacency Matrix Structure**

<table>
<thead>
<tr>
<th>Operation</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>size, isEmpty, replaceElement, swap</td>
<td>( O(1) )</td>
</tr>
<tr>
<td>numVertices, numEdges</td>
<td>( O(1) )</td>
</tr>
<tr>
<td>vertices</td>
<td>( O(n) )</td>
</tr>
<tr>
<td>edges, directedEdges, undirectedEdges</td>
<td>( O(m) )</td>
</tr>
<tr>
<td>elements, positions</td>
<td>( O(n+m) )</td>
</tr>
<tr>
<td>inVertices, opposite, origin, destination, inDirected, degree, inDegree, outDegree</td>
<td>( O(1) )</td>
</tr>
<tr>
<td>incidentEdges, inIncidentEdges, outIncidentEdges, adjacentVertices, inAdjacentVertices, outAdjacentVertices</td>
<td>( O(n) )</td>
</tr>
<tr>
<td>areAdjacent</td>
<td>( O(1) )</td>
</tr>
<tr>
<td>insertEdge, insertDirectedEdge, removeEdge, makeUndirected, reverseDirection, setDirectionTo</td>
<td>( O(1) )</td>
</tr>
<tr>
<td>insertVertex, removeVertex</td>
<td>( O(n^2) )</td>
</tr>
</tbody>
</table>