October 24 – Graph Implementations
(Adjacency Matrix)
G Carl Evans
Graphs

To study all of these structures:
1. A common vocabulary
2. Graph implementations
3. Graph traversals
4. Graph algorithms
### Graph Implementation: Edge List

- **insertVertex(K key):** $O(1)$
- **insertEdge(Vertex v1, Vertex v2, K key):** $O(1)$
- **removeVertex(Vertex v):** $O(m)$
- **incidentEdges(Vertex v):** $O(m)$
- **areAdjacent(Vertex v1, Vertex v2):** $O(m)$
Graph Implementation: Adjacency Matrix

```
0 1  a
1 2  b
0 2  c
2 3  d
```

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
Graph Implementation: Adjacency Matrix

```
<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
```
Graph Implementation: Adjacency Matrix

```
0 1 a
1 2 b
0 2 c
2 3 d
```

```
   0 1 2 3
0  0 - 0 0
1  0 - 0 -
2  0 0 - -
3  0 0 0 -
```
Adjacency Matrix

areAdjacent(Vertex v1, Vertex v2):
Adjacency Matrix

incidentEdges(Vertex v):

0
1
2
3

0 1 a
1 2 b
0 2 c
2 3 d
Adjacency Matrix

insertEdge(Vertex v1, Vertex v2, K key):
Adjacency Matrix

```
removeVertex(Vertex v1):
```

![Diagram of a graph and adjacency matrix](image)
Edge List

1. Edge List

2. Diagram with nodes and edges:
   - Node 0 connected to nodes 1 and 2
   - Node 1 connected to node 2
   - Node 2 connected to node 3

3. Adjacency list representation:

   - Node 0: 1, 2
   - Node 1: 2
   - Node 2: 0, 3
   - Node 3: 2
Graph Implementation: Adjacency List
Graph Implementation: Adjacency List
### Adjacency List

**insertVertex(K key):**

<table>
<thead>
<tr>
<th>Key</th>
<th>Vertex</th>
<th>Degree</th>
</tr>
</thead>
<tbody>
<tr>
<td>u</td>
<td>a, c</td>
<td>d=2</td>
</tr>
<tr>
<td>v</td>
<td>a, b</td>
<td>d=2</td>
</tr>
<tr>
<td>w</td>
<td>b, c, d</td>
<td>d=3</td>
</tr>
<tr>
<td>z</td>
<td>d</td>
<td>d=1</td>
</tr>
</tbody>
</table>

**Diagram:**

- **Vertices:** u, v, w, z
- **Edges:** u-v, v-w, w-z, z-d, u-a, a-c, b, d

- **Degree:**
  - u: 2
  - v: 2
  - w: 3
  - z: 1
Adjacency List

removeVertex(Vertex v):

u
a
b
v
w
z

u
d=2
v
d=2
w
d=3
z
d=1

a
b
c
d

u
v
w
a

v
w
b

u
w
z
c

w
z
d
Adjacency List

incidentEdges(Vertex v):

- u
  - incidentEdges = a, c
  - d = 2
- v
  - incidentEdges = a, b
  - d = 2
- w
  - incidentEdges = b, c, d
  - d = 3
- z
  - incidentEdges = d
  - d = 1
Adjacency List

areAdjacent(Vertex v1, Vertex v2):
Adjacency List

insertEdge(Vertex v1, Vertex v2, K key):