Graph Implementation #1: Edge List

Data Structures:
- Vertex Collection:
- Edge Collection:

Operations on an Edge List implementation:
- insertVertex(K key):
  - What needs to be done?
- removeVertex(Vertex v):
  - What needs to be done?
- incidentEdges(Vertex v):
  - What needs to be done?
- areAdjacent(Vertex v1, Vertex v2):
  - Can this be faster than `G.incidentEdges(v1).contains(v2)`?
- insertEdge(Vertex v1, Vertex v2, K key):
  - What needs to be done?

Graph Implementation #2: Adjacency Matrix

Data Structures:

Operations on an Adjacency Matrix implementation:
- insertVertex(K key):
  - What needs to be done?
- removeVertex(Vertex v):
  - What needs to be done?
- incidentEdges(Vertex v):
  - What needs to be done?
- areAdjacent(Vertex v1, Vertex v2):
  - Can this be faster than `G.incidentEdges(v1).contains(v2)`?
- insertEdge(Vertex v1, Vertex v2, K key):
  - What needs to be done?
Graph Implementation #3: Adjacency List

Operations on an Adjacency Matrix implementation:
insertVertex(K key):
removeVertex(Vertex v):
incidentEdges(Vertex v):
areAdjacent(Vertex v1, Vertex v2):
insertEdge(Vertex v1, Vertex v2, K key):

Running Times of Classical Graph Implementations

<table>
<thead>
<tr>
<th>Operation</th>
<th>Edge List</th>
<th>Adj. Matrix</th>
<th>Adj. List</th>
</tr>
</thead>
<tbody>
<tr>
<td>Space</td>
<td>n+m</td>
<td>n²</td>
<td>n+m</td>
</tr>
<tr>
<td>insertVertex</td>
<td>1</td>
<td>n</td>
<td>1</td>
</tr>
<tr>
<td>removeVertex</td>
<td>m</td>
<td>n</td>
<td>deg(v)</td>
</tr>
<tr>
<td>insertEdge</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>removeEdge</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>incidentEdges</td>
<td>m</td>
<td>n</td>
<td>deg(v)</td>
</tr>
<tr>
<td>areAdjacent</td>
<td>m</td>
<td>1</td>
<td>min( deg(v), deg(w) )</td>
</tr>
</tbody>
</table>

Q: If we consider implementations of simple, connected graphs, what relationship between n and m?

- On connected graphs, is there one algorithm that underperforms the other two implementations?

Q: Is there clearly a single best implementation?

- Optimized for fast construction:
- Optimized for areAdjacent operations:

CS 225 – Things To Be Doing:

1. Optional Exam: Thursday, April 11 – Sunday, April 14
2. lab_dict released this week; due on Sunday.
3. MP6 EC+5 due tonight; final due date on Monday, April 15
4. Very special POTD today!