Graph ADT (Adjacency List)

Adjacency List Representations
- Creates a linked-list of vertices adjacent to a given vertex
  - Maintain a vector of lists – one for each vertex
  - Add new vertices in constant time
- Best for a sparse graph - Why?

- For undirected graph
  - Add v to w's list
  - Add w to v's list
    - Important for efficiency

Pros and Cons of Adjacency Lists
- Pro
  - Space is proportional to $V+E$ rather than $V^2$
- Con
  - Time required to test for the existence of an edge
    - What is the time difference?

  - Why is it important?

Worst case cost of graph processing operations

<table>
<thead>
<tr>
<th></th>
<th>array of edges</th>
<th>adjacency matrix</th>
<th>adjacency lists</th>
</tr>
</thead>
<tbody>
<tr>
<td>space</td>
<td>$E$</td>
<td>$V^2$</td>
<td>$V+E$</td>
</tr>
<tr>
<td>initialize empty</td>
<td>1</td>
<td>$V^2$</td>
<td>$V$</td>
</tr>
<tr>
<td>copy</td>
<td>$E$</td>
<td>$V^2$</td>
<td>$E$</td>
</tr>
<tr>
<td>destroy</td>
<td>1</td>
<td>$V$</td>
<td>$E$</td>
</tr>
<tr>
<td>insert edge</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>isEdge &amp; remove edge</td>
<td>$E$</td>
<td>1</td>
<td>$V$</td>
</tr>
</tbody>
</table>

- When do we get worst case performance?

- Is anything missing from this analysis of the graph?
Exercise
Write a representation-independent ADT function that returns a pointer to a vertex-indexed array giving the degree of each vertex in the graph.