A Review of Major Data Structures so Far

<table>
<thead>
<tr>
<th>Array-based</th>
<th>List/Pointer-based</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sorted Array</td>
<td>Singly Linked List</td>
</tr>
<tr>
<td>Unsorted Array</td>
<td>Doubly Linked List</td>
</tr>
<tr>
<td>Stacks</td>
<td>Skip Lists</td>
</tr>
<tr>
<td>Queues</td>
<td>Trees</td>
</tr>
<tr>
<td>Hashing</td>
<td>BTree</td>
</tr>
<tr>
<td>Heaps</td>
<td>Binary Tree</td>
</tr>
<tr>
<td>Priority Queues</td>
<td>Huffman Encoding</td>
</tr>
<tr>
<td>UpTrees</td>
<td>kd-Tree</td>
</tr>
<tr>
<td>Disjoint Sets</td>
<td>AVL Tree</td>
</tr>
</tbody>
</table>

Motivation:
Graphs are awesome data structures that allow us to represent an enormous range of problems. To study these problems, we need:
1. A common vocabulary to talk about graphs
2. Implementation(s) of a graph
3. Traversals on graphs
4. Algorithms on graphs

Graph Vocabulary
Consider a graph \( G \) with vertices \( V \) and edges \( E \), \( G=(V,E) \).

- Incident Edges:
  \[ I(v) = \{ (x, v) \in E \} \]

- Degree(v):
  \[ |I| \]

- Adjacent Vertices:
  \[ A(v) = \{ x : (x, v) \in E \} \]

- Path\( (G_2) \): Sequence of vertices connected by edges

Cycle\( (G_1) \): Path with a common begin and end vertex.

Simple Graph\( (G) \): A graph with no self loops or multi-edges.

Subgraph\( (G) \): \( G' = (V', E') \):
\[ V' \in V, E' \in E, (u, v) \in E \Rightarrow u \in V', v \in V' \]
Graphs that we will study this semester include:
- Complete subgraph(G)
- Connected subgraph(G)
- Connected component(G)
- Acyclic subgraph(G)
- Spanning tree(G)

**Size and Running Times**
Running times are often reported by \( n \), the number of vertices, but often depend on \( m \), the number of edges.

For arbitrary graphs, the **minimum** number of edges given a graph that is:

- Not Connected:
  
  Minimally Connected*:

The **maximum** number of edges given a graph that is:

- Simple:

- Not Simple:

The relationship between the degree of the graph and the edges:

---

**Proving the Size of a Minimally Connected Graph**

**Theorem:** Every connected graph \( G=(V, E) \) has at least \( |V|-1 \) edges.

**Proof of Theorem**
Consider an arbitrary, connected graph \( G=(V, E) \).

**Suppose** \( |V| = 1 \):

---

**Graph ADT**

<table>
<thead>
<tr>
<th>Data</th>
<th>Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Vertices</td>
<td>insertVertex(K key); insertEdge(Vertex v1, Vertex v2, K key);</td>
</tr>
<tr>
<td>2. Edges</td>
<td>removeVertex(Vertex v); removeEdge(Vertex v1, Vertex v2);</td>
</tr>
<tr>
<td>3. Some data structure maintaining the structure between vertices and edges.</td>
<td>incidentEdges(Vertex v); areAdjacent(Vertex v1, Vertex v2);</td>
</tr>
<tr>
<td></td>
<td>origin(Edge e); destination(Edge e);</td>
</tr>
</tbody>
</table>

---

**CS 225 – Things To Be Doing:**

1. Exam 4 next Friday; **Practice Exam Available Today!**
2. mp_mazes EC due Monday
3. Daily POTDs are ongoing!