Graph Vocabulary

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

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

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

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

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

Disjoint Sets Running Time:

- Worst case running time of find(k):
- Worst case running time of union(r1, r2), given roots:
- New function: “Iterated Log”:
  \[ \log^*(n) := \]
- Overall running time:
  - A total of \( m \) union/find operation runs in:

Graphs

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

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:
Graph ADT

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

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:

Adj. Matrix

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:

Adj. Matrix

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:

Adj. Matrix

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:

Adj. Matrix

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:

Adj. Matrix

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:

Adj. Matrix

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:

Adj. Matrix

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:

Adj. Matrix

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:

Adj. Matrix

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:

Adj. Matrix

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:

Adj. Matrix

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:

Adj. Matrix

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:

Adj. Matrix

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:

Adj. Matrix

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:

Adj. Matrix

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:

Adj. Matrix

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:

Adj. Matrix

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:

Adj. Matrix

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?