How can we represent a graph?

What are basic things we might want to do?

Is there an edge from \( V_i \) to \( V_j \)?

Less often, iterate over all edges to \( V_i \).
\[ V = \text{set of vertices in graph} \]

For Lab 4

STL → airport

3-letter acronym for airport

Set of airports
Object “STL”

city name St. Louis

time zone

location

List of edges (Flights) that originate in St. Louis

Could use a Set versus a list.

Expected constant time search by dest.

List of outgoing edges

Could also keep a list of edges that terminate in St. Louis.
**Adjacency List**

Representation of a graph where for each vertex you store a list of all outgoing edges adjacent (in a directed sense)
Graph

Typical way to draw
\[ e_1, e_3 \]
\[ a : b, e_1, e_3 \]
\[ b : e_5, c, e_2 \]
\[ c : d, e_4 \]
\[ d : \]

Incoming edges

Also we use:
\[ a : \]
\[ b : a \]
\[ c : a, b \]
\[ d : b, c \]

Implicit representation of an edge when no multi-edges
If we keep a set of outgoing edges for each vertex (with comparison defined by dest),
answer is edge from $V_i$ to $V_j$ in expected constant time.

Augmented Adjacency Set also keep a set of incoming edges for each vertex.
Adjacency Matrix

\[
\begin{array}{cccc}
  a & b & c & d \\
\hline
  x & e_1 & e_3 & x \\
  b & x & x & e_2 & e_5 \\
  c & x & x & x & e_4 \\
  d & x & x & x & x \\
\end{array}
\]

\[O(n^2)\] time to iterate over \(n\) vertices and \(m\) edges (if not multigraph)
\[ O(m+n) \text{ time} \]

to iterate over all edges

adj list

\[ a \rightarrow e_1 \rightarrow e_2 \rightarrow e_3 \rightarrow \text{null} \]
\[ b \rightarrow e_2 \rightarrow e_5 \rightarrow \text{null} \]
\[ c \rightarrow e_4 \rightarrow \text{null} \]
\[ d : \text{null} \]

# objects among all lists is \( m \)
Breadth-First Search

First we'll overview the graph representations.

Then we'll look at problem of finding a shortest path in a directed unweighted graph.
Adjacency List

vertices $\{a, b, c, d\}$

outedges
- $a \rightarrow \{e_3, e_1, e_2\}$
- $b \rightarrow \{e_4\}$
- $c \rightarrow \{e_5\}$
- $d \rightarrow \{\}$

inedges
- $a \rightarrow \{\}$
- $b \rightarrow \{e_3\}$
- $c \rightarrow \{e_1, e_2\}$
- $d \rightarrow \{e_2, e_4, e_5\}$

we've called Augmented Adj List
<table>
<thead>
<tr>
<th>Data Structure</th>
<th>storeIncomingEdges</th>
<th>TaggedBucketCollection type</th>
</tr>
</thead>
<tbody>
<tr>
<td>AdjacencyList</td>
<td>false</td>
<td>$V \rightarrow \text{List}&lt;E&gt;$</td>
</tr>
<tr>
<td>AugmentedAdjacencyList</td>
<td>true</td>
<td>$V \rightarrow \text{List}&lt;E&gt;$</td>
</tr>
<tr>
<td>Adjacency Set (no multi-edges)</td>
<td>false</td>
<td>$V \rightarrow \text{Set}&lt;E&gt;$</td>
</tr>
<tr>
<td>Augmented Adjacency Set (no multi-edges)</td>
<td>true</td>
<td>$V \rightarrow \text{Set}&lt;E&gt;$</td>
</tr>
<tr>
<td>Adjacency Set (with multi-edges)</td>
<td>false</td>
<td>$V \rightarrow \text{BucketMapping}&lt;V, \text{List}&lt;E&gt;&gt;$</td>
</tr>
<tr>
<td>Augmented Adjacency Set (with multi-edges)</td>
<td>true</td>
<td>$V \rightarrow \text{BucketMapping}&lt;V, \text{List}&lt;E&gt;&gt;$</td>
</tr>
</tbody>
</table>
Adjacency matrix

ids
a  →  0
b  →  1
c  →  2
d  →  3

edges
0  1  2  3
∅  ∅  e₁  e₂
∅  ∅  ∅  ∅
∅  ∅  ∅  e₅
∅  ∅  ∅  ∅
Representing Undirected Graph

$m$ edges

represent as $2m$ edges
Undirected graph, only need to store this part.
### Analysis (directed graph, no multi-edges)

<table>
<thead>
<tr>
<th></th>
<th>Adj List</th>
<th>Adj Set</th>
<th>Adj Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contains edge $v_i v_j$</td>
<td>$O(\text{# edges out of } V_i)$</td>
<td>expected $O(1)$</td>
<td>$O(1)$</td>
</tr>
<tr>
<td>Iterate over out edges $V$</td>
<td>$O(\text{# edges out of } V)$</td>
<td>$O(\text{# edges out of } V)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>Iterate over all edges</td>
<td>$O(n+m)$</td>
<td>$O(n+m)$</td>
<td>$O(n^2)$</td>
</tr>
<tr>
<td>Space</td>
<td>$O(n+m)$</td>
<td>$O(n+m)$</td>
<td>$O(n^2)$</td>
</tr>
</tbody>
</table>