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
- \(a \rightarrow \{\}\)
- \(b \rightarrow \{e_3\}\)
- \(c \rightarrow \{e_1\}\)
- \(d \rightarrow \{e_2, e_4, e_5\}\)

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>$V \rightarrow \text{BucketMapping}&lt;V,\text{List}&lt;E&gt;$</td>
</tr>
<tr>
<td>Augmented Adjacency Set (with multi-edges)</td>
<td>true</td>
<td>$V \rightarrow \text{BucketMapping}&lt;V,\text{List}&lt;E&gt;$</td>
</tr>
</tbody>
</table>
Adjacency matrix

\[
\begin{array}{c}
\text{ids} \\
a \rightarrow 0 \\
c \rightarrow 2 \\
d \rightarrow 3 \\
b \rightarrow 1
\end{array}
\]

\[
\begin{array}{cccc}
\text{edges} \\
0 & 1 & 2 & 3 \\
\emptyset & \emptyset & e_1 & e_2 \\
\emptyset & \emptyset & \emptyset & \emptyset \\
\emptyset & \emptyset & \emptyset & e_5 \\
\emptyset & \emptyset & \emptyset & \emptyset
\end{array}
\]
Representing Undirected Graph

m edges → represent as 2m edges
1 edge 0 no edge

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>$O(\text{# edges of } v_i)$</td>
<td>$O(1)$</td>
<td>$O(1)$</td>
</tr>
<tr>
<td>Iterate over out edges $V$</td>
<td>$O(\text{# edges of } V)$</td>
<td>$O(\text{# edges of } V)$</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>
Finding Shortest Paths in Unweighted Directed Graph

Diagram:
- DEN → STL
- DFW → STL
- STL → BOS, PVD, ORL
- ATL → PVD, ORL
- BOS → PVD, ORL
- PVD → ORL

Symbols:
- DEN, DFW, STL, BOS, PVD, ATL, ORL
Single source shortest path - find path to source vertex (start)
directly reachable from S

Keep going until the desired destination is reached
How do we represent the solution?

Represent implicitly by storing the edge from parent.
Shortest path tree

- STL
  - Flight from STL to BOS
  - Parent
  - BOS
    - Flight from BOS to PVD
    - Parent
    - PVD
Source a

parent

Queue: ❌❌❌❌❌❌❌
Time complexity

\[ \text{adj list: iterate over outedges of each vertex at most once each} \]

\[ O(n + m) \]

\[ O(n^2) \text{ matrix} \]

\[ O(n + m) \text{ adj list} \]