Goals:
- Describe graphs using adjacency matrices & lists.

Ways to Represent Graphs in Computer

Adjacency Matrix

\[
\begin{array}{cccc}
1 & 2 & 3 & 4 \\
1 & 0 & 0 & 1 \\
2 & 0 & 0 & 0 & 1 \\
3 & 0 & 1 & 0 & 1 \\
4 & 1 & 1 & 1 & 0 \\
\end{array}
\]

Store as array A in memory. E.g., \(A[3,4] = 1\)
- Can learn \(A[i,j]\) in \(O(1)\) time

Which adjacency matrix represents this graph?

\[
\begin{array}{c}
1 \\
\end{array}
\]

A
\[
\begin{array}{cccc}
0 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 \\
0 & 1 & 0 & 1 \\
0 & 1 & 1 & 0 \\
\end{array}
\]

B
\[
\begin{array}{cccc}
6 & 1 & 0 & 0 \\
1 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 \\
0 & 0 & 1 & 0 \\
\end{array}
\]

C
**Adjacency List**

<table>
<thead>
<tr>
<th>Vertex</th>
<th>Adjacent Vertices</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3, 4</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>1, 4</td>
</tr>
<tr>
<td>4</td>
<td>1, 2, 3</td>
</tr>
</tbody>
</table>

Store as an array of lists

Can access $A[3]$ (the list) in $O(1)$ time, but to go through list takes time $O(L)$ where $L$ is length of list. Can learn $A[3].\text{length}$ in $O(1)$ time.


$A[3, 2] = 4$

$A.\text{length} = 4$
Edge List
Can represent graph as list of edges, but worst case time complexity bad for most applications

def: The degree of a vertex is the number of adjacent edges.

def: A vertex $v_1$ is adjacent to vertex $v_2$ if connected by an edge.
How would you represent a
- directed graph?
- graph with self-loops?
- graph with multi-edges?
- graph with weighted edges?

Using Adjacency Matrix / Adjacency List?

Give representations of this graph using both approaches:
Input: Adj Matrix $A$ for $G = (V, E)$ (undirected, unweighted, no self loops)

Output:
1. $S = 0$
2. for $u, v \in V$
   \[ S^+ = A[u, v] \]
3. return $S$

A) $|V|$  B) $|V| \times |V|$  C) $|E|$  D) $2|E|$