Ways to Represent Graphs in Computer

Adjacency Matrix

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>00</td>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>00</td>
<td>01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>01</td>
<td>01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>11</td>
<td>10</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Graph:

- 1
- 2
- 3
- 4

Edges:
- 1 to 2
- 1 to 3
- 2 to 3
- 3 to 4
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

Ex: $A[4] = (1, 2, 3)$

$A[3, 2] = 4$

$4$ is the 2nd vertex connected to vertex 3

[Plucker Question]

$O(1)$ operations

Adjacent Matrix

$A[i,j] \leftarrow$ edge $(i,j) \in E$?

columns(A) $\leftarrow$ # vertices

Adjacent List

$A[i,j] \leftarrow$ $j$th neighbor of $i$

length(A) $\leftarrow$ # vertices

length($A[i]$) $\leftarrow$ degree of $i$
How would you represent a
- directed graph?
- graph with self-loops?
- graph with multiedges?
- graph with weighted edges?

Using Adjacency Matrix / Adjacency List?

Give representations of this graph using both approaches:

<table>
<thead>
<tr>
<th></th>
<th>V</th>
<th>List</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(2, 1/3), (3, 1/3), (4, 1/3)</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>(1, 1/2), (4, 1/2)</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>(4, 1)</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>(4, 1)</td>
<td></td>
</tr>
</tbody>
</table>
Representing Adjacency Matrices + Lists in Computer

- **Matrix**
  - List of Lists / Array
  - In $O(1)$ time can learn $A[i,j]$

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

  $$A[2,3] = 0$$

- **List**
  - List of Lists
  - In $O(1)$ time can learn $A[i,j]$~

  \[ A[i,j] \]

  \[ j_m \text{ neighbor of } i \]

  \[ \text{length}(A[i]) \]
### 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>

**Ex.**

\[
A[4] = (1, 2, 3)
\]

\[
A[3, 2] = 4
\]

4 is the 2nd vertex connected to vertex 3

[Plicker Question]
Write pseudocode to learn degree of vertex v in an unweighted graph, and give big-O bound on time complexity.

**Adjacency Matrix**

Input: vertex v, adj. matrix A for G = (V, E)
Output: degree of v

- `deg = 0`
- for `i ∈ V`
  - `deg = deg + A[i, v]`  
    - alternate: for `i = 1` to `|V|`
    - `deg = deg + A[i, v]`
      - alternate: if `(A[i, v] = 1)`: `deg++`
- return `deg`

Time complexity: `O(|V|)`

**Adjacency List**

Input: vertex v, adj. list A for G = (V, E)
Output: degree of v

- return `A[v].length`
  - alternate: length `(A[v])`

Time complexity: `O(1)`  

Much faster for this problem