Graphs

- set of vertices V
- set of edges E: V x V

Edges must be between vertices in the defined vertex set.

Many different types of relationships can be represented as graphs.

Example: Sorting.

Directed graph

Vertices connected by an edge are adjacent
**Weighted edges**
are edges with a value associated with them.

Example: distance in miles

![Diagram: undirected graph with labeled vertices and edges.]

**Path**: list of sequentially connected edges.

Path from A to B exists $\rightarrow$ B is reachable from A.

**Cycle**: path that starts and ends in the same place.

**Hamiltonian path**: visit every vertex exactly once.

**Eulerian Path**: visit every edge exactly once.
Graph Representation

![Graph Diagram]

1. Adjacency list representation
   - Good for sparse graphs
     - (when $|E| < \sqrt{|V|}$)

```
A -> B -> D
B -> C
C -> D
D -> A
E -> D -> E
```

Total memory consumed:

- $|V|$ linked lists
- Total number of elements in all linked lists: $|E|
- So memory is $O(\max(|V|, |E|) = O(|V| + |E|)$
Adjacency Matrix Representation:

good when $|E| \approx |V|^2$

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>E</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

amt of memory: $O(|V|^2)$

note: search for whether edge is present is fast.

If undirected, you can store approx. half the matrix.
Set representation.

\[ V = \{A, B, C, D, E\} \]

\[ E = \{ (A, B), (A, D), (B, C), (C, D), (D, A), (E, D), (E, E) \} \]

Which representation is best for finding all neighbors of a vertex? adjacency list

Which representation is best to test for existence of a specific edge? matrix
Undirected Graphs

Adjacency Matrix:
If undirected, you only need to store approx. half the matrix.

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>E</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Can you save space if using an adjacency list?
How do you prove things about graphs?

Example

**in-degree**: # edges going into a vertex
**out-degree**: # edges coming out of a vertex.

**Theorem:**

\[
\sum_{i=1}^{1\nu} in\text{-}degree(v_i) = \sum_{i=1}^{1\nu} out\text{-}degree(v_i)
\]

<table>
<thead>
<tr>
<th></th>
<th>in-deg</th>
<th>out-deg</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>∅</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>D</td>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>

\[4\]
**Induction Proof**

**Base case**

\[ \sum_{i=1}^{2} \text{out degree}(v_i) = \sum_{i=1}^{2} \text{in degree}(v_i) = 1 \]

\[ \text{A} \rightarrow \text{B} \]

**Inductive Hypothesis**
- Assume theorem holds when \( |E| = k \).

**Inductive Step**

\( |E| = k+1 \)

Let \( G \) be a graph with \( k+1 \) edges.

Select an edge \( e \) and remove it from \( G \) to make \( G' \).

\[ |G'\cdot V| \leq \sum_{i=1}^{\infty} \text{in degree}(v_i) = \sum_{i=1}^{\infty} \text{in degree}(v_i)+1 \]
\[
\frac{|G \cdot v|}{i = 1} \geq \text{outdegree}(v_i) = \frac{|G \cdot v|}{i = 1} \geq \text{outdegree}(v_i) + 1
\]

From IH,

\[
\frac{|G \cdot v|}{i = 1} \geq \text{outdegree}(v_i) = \frac{|G \cdot v|}{i = 1} \geq \text{indegree}(v_i)
\]

so

\[
\frac{|G \cdot v|}{i = 1} \geq \text{outdegree}(v_i) = \frac{|G \cdot v|}{i = 1} \geq \text{indegree}(v_i) + 1
\]

\[
\frac{|G \cdot v|}{i = 1} \geq \text{indegree}(v_i)
\]
Thought Question

degree of a vertex: number of edges incident to it

Prove the theorem: \[ \sum_{v \in V} \text{degree}(v) \] is even.
Another Classic Graph Problem

Königsberg Bridge Problem.
Kalininingrad, Russia

Can you cross all seven bridges without re-using any?
Express as a graph
Find the Eulerian Path
What about this? Eulerian path?

\[\text{no}\]
What about this? Eulerian path?

Yes
How to look for an eulerian path

- Start with one edge
- Try an adjacent edge
- Keep trying until you reach an edge you've seen before
- Backtrack if this happens

If you reach all nodes, you have an eulerian path!
If you've tried all possible paths then no eulerian path exist's.

Lower bound: you never backtrack!

$O(1E1)$

Upper bound: you visit some edges back track
you visit some edges back track

Worst case $\rightarrow$ you visit all paths!
worst case \# of paths

= (|E|!)(|E| - 1)(|E| - 2) \ldots

= |E|! \in \Theta(|E|^{\left|E\right|})

|E|! \in \Omega(2^{\left|E\right|})
Say we don't have to know what exactly the Eulerian path is. We just need to know whether one exists. What is an algorithm for that?
Euler proved: if all vertices are of even degree, then an Eulerian cycle exists.

If there are exactly two vertices of odd degree, then an Eulerian path exists.

Once and you return where you start.
Can we find the Eulerian Path in less than $|E|!$ time?

**Fleury's Algorithm**

1. Start at a vertex with odd degree
2. Choose an edge out of that vertex whose removal will not cut you off from any vertices that still have edges to them.
   - Reach ability $O(|V| + |E|)$ done $O(|E|)$ times
3. Traverse and delete the edge.
   - Repeat $|E|$ times.
   - $O(|E| + |E|(|V| + |E|))$