Graphs

Lecture 22
CS 2112 – Fall 2016
These are not Graphs

...not the kind we mean, anyway
These are Graphs

\[ K_5 \]

\[ K_{3,3} \]
Applications of Graphs

• Communication networks
• Routing and shortest path problems
• Commodity distribution (flow)
• Traffic control
• Resource allocation
• Geometric modeling
• ...

Graph Definitions

- A directed graph (or digraph) is a pair \((V, E)\) where
  - \(V\) is a set
  - \(E\) is a set of ordered pairs \((u, v)\) where \(u, v \in V\)
    - Usually require \(u \neq v\) (i.e., no self-loops)

- An element of \(V\) is called a vertex (pl. vertices) or node
- An element of \(E\) is called an edge or arc

- \(|V| = \text{size of } V\), often denoted \(n\)
- \(|E| = \text{size of } E\), often denoted \(m\)
Example Directed Graph (Digraph)

\[ V = \{a,b,c,d,e,f\} \]
\[ E = \{(a,b), (a,c), (a,e), (b,c), (b,d), (b,e), (c,d), (c,f), (d,e), (d,f), (e,f)\} \]

\[ |V| = 6, \ |E| = 11 \]
An **undirected graph** is just like a directed graph, except the edges are *unordered pairs (sets)* \( \{u, v\} \)

**Example:**

\[
V = \{a, b, c, d, e, f\} \\
E = \{\{a, b\}, \{a, c\}, \{a, e\}, \{b, c\}, \{b, d\}, \{b, e\}, \{c, d\}, \{c, f\}, \{d, e\}, \{d, f\}, \{e, f\}\}
\]
Some Graph Terminology

- Vertices $u$ and $v$ are called the source and sink of the directed edge $(u,v)$, respectively.
- Vertices $u$ and $v$ are called the endpoints of $(u,v)$.
- Two vertices are adjacent if they are connected by an edge.
- The outdegree of a vertex $u$ in a directed graph is the number of edges for which $u$ is the source.
- The indegree of a vertex $v$ in a directed graph is the number of edges for which $v$ is the sink.
- The degree of a vertex $u$ in an undirected graph is the number of edges of which $u$ is an endpoint.
More Graph Terminology

- A path is an alternating sequence $v_0, e_0, v_1, e_1, v_2, ..., v_n$ of vertices and edges, beginning and ending with a vertex, such that $(v_i, v_{i+1}) = e_i$, $0 \leq i \leq n - 1$
- The length of a path is its number of edges
  - In this example, the length is 5
  - A single vertex is a path of length 0
- A path is simple if it does not repeat any vertices
- A cycle is a path $v_0, ..., v_n$ with at least one edge such that $v_0 = v_n$
- A cycle is simple if it does not repeat any vertices except the first and last
- A graph is acyclic if it has no cycles
- A directed acyclic graph is called a dag
Is This a Dag?

- **Intuition:**
  - If it’s a dag, there must be a vertex with indegree zero – why?

- **This idea leads to an algorithm**
  - A digraph is a dag if and only if we can iteratively delete indegree-0 vertices until the graph disappears
Is This a Dag?

- **Intuition:**
  - If it’s a dag, there must be a vertex with indegree zero – why?

- **This idea leads to an algorithm**
  - A digraph is a dag if and only if we can iteratively delete indegree-0 vertices until the graph disappears
Is This a Dag?

• Intuition:
  ▪ If it’s a dag, there must be a vertex with indegree zero – why?

• This idea leads to an algorithm
  ▪ A digraph is a dag if and only if we can iteratively delete indegree-0 vertices until the graph disappears
Is This a Dag?

• Intuition:
  ▪ If it’s a dag, there must be a vertex with indegree zero – why?

• This idea leads to an algorithm
  ▪ A digraph is a dag if and only if we can iteratively delete indegree-0 vertices until the graph disappears
Is This a Dag?

- **Intuition:**
  - If it’s a dag, there must be a vertex with indegree zero – why?

- **This idea leads to an algorithm**
  - A digraph is a dag if and only if we can iteratively delete indegree-0 vertices until the graph disappears
Is This a Dag?

• Intuition:
  ▪ If it’s a dag, there must be a vertex with indegree zero – why?

• This idea leads to an algorithm
  ▪ A digraph is a dag if and only if we can iteratively delete indegree-0 vertices until the graph disappears
Is This a Dag?

- **Intuition:**
  - If it’s a dag, there must be a vertex with indegree zero – why?

- **This idea leads to an algorithm**
  - A digraph is a dag if and only if we can iteratively delete indegree-0 vertices until the graph disappears.
Is This a Dag?

• Intuition:
  ▪ If it’s a dag, there must be a vertex with indegree zero – why?

• This idea leads to an algorithm
  ▪ A digraph is a dag if and only if we can iteratively delete indegree-0 vertices until the graph disappears
Is This a Dag?

• Intuition:
  ▪ If it’s a dag, there must be a vertex with indegree zero – why?

• This idea leads to an algorithm
  ▪ A digraph is a dag if and only if we can iteratively delete indegree-0 vertices until the graph disappears
Is This a Dag?

• Intuition:
  ▪ If it’s a dag, there must be a vertex with indegree zero – why?

• This idea leads to an algorithm
  ▪ A digraph is a dag if and only if we can iteratively delete indegree-0 vertices until the graph disappears
Is This a Dag?

- **Intuition:**
  - If it’s a dag, there must be a vertex with indegree zero – why?

- **This idea leads to an algorithm**
  - A digraph is a dag if and only if we can iteratively delete indegree-0 vertices until the graph disappears
Topological Sort

• We just computed a topological sort of the dag
  - This is a numbering of the vertices such that all edges go from lower- to higher-numbered vertices

• Useful in job scheduling with precedence constraints
Graph Coloring

- A coloring of an undirected graph is an assignment of a color to each node such that no two adjacent nodes get the same color.

Q: How many colors are needed to color this graph?
Graph Coloring

- A coloring of an undirected graph is an assignment of a color to each node such that no two adjacent nodes get the same color.

Q: How many colors are needed to color this graph?
A: 3
An Application of Coloring

• Vertices are jobs
• Edge \((u,v)\) is present if jobs \(u\) and \(v\) each require access to the same shared resource, and thus cannot execute simultaneously
• Colors are time slots to schedule the jobs
• Minimum number of colors needed to color the graph = minimum number of time slots required
Planarity

- A graph is **planar** if it can be embedded in the plane with no edges crossing

Q: Is this graph planar?
Planarity

• A graph is planar if it can be embedded in the plane with no edges crossing

• Q: Is this graph planar?
  • A: Yes
Planarity

- A graph is \textit{planar} if it can be embedded in the plane with no edges crossing

- Q: Is this graph planar?
- A: Yes
Detecting Planarity

- Kuratowski's Theorem

- A graph is planar if and only if it does not contain a copy of $K_5$ or $K_{3,3}$ (possibly with other nodes along the edges shown)
Every planar graph is 4-colorable
(Appel & Haken, 1976)
Bipartite Graphs

- A directed or undirected graph is bipartite if the vertices can be partitioned into two sets such that all edges go between the two sets.
Bipartite Graphs

- The following are equivalent:
  - $G$ is bipartite
  - $G$ is 2-colorable
  - $G$ has no cycles of odd length
Traveling Salesperson

- Find a path of minimum distance that visits every city
Representations of Graphs

Adjacency List

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>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Adjacency Matrix or Adjacency List?

**• Adjacency Matrix**
- Uses space $O(n^2)$
- Can iterate over all edges in time $O(n^2)$
- Can answer “Is there an edge from $u$ to $v$?” in $O(1)$ time
- Better for dense graphs (lots of edges) ($m \sim n^2$)

**• Adjacency List**
- Uses space $O(m+n)$
- Can iterate over all edges in time $O(m+n)$
- Can answer “Is there an edge from $u$ to $v$?” in $O(d(u))$ time
- Better for sparse graphs (fewer edges) ($m \ll n^2$)

• $n =$ number of vertices
• $m =$ number of edges
• $d(u) =$ outdegree of $u$
Conclusion

• Graphs are an extremely useful tool for modeling many different kinds of computational problems

• There are many efficient basic graph algorithms – depth-first search, breadth-first search, topological sort, ...

• ... and some not so efficient – traveling salesperson, coloring

• Learn about this and more in CS 4820
Remember to Vote!