Object-oriented programming and data-structures

CS/ENGRD 2110
SUMMER 2018
These aren't the graphs we're looking for
A graph is a data structure

A graph has
- a set of vertices
- a set of edges between vertices

Graphs are a generalization of trees
This is a graph
Another transport graph
This is a graph

The internet’s undersea world

The vast majority of the world’s telecommunications are carried by submarine cables, not by air. These cables are laid at depths of more than once the diameter of the ship that deploys them. If a ship accidentally steps on a submarine cable, the damage can take months to repair. reconstructing a network of cables is now common, as the number of core submarine cables is growing rapidly.

- **Fiber optic submarine cable systems**
  - **Present**
  - **Planned**

Alexandria, Wednesday
A ship on route to another country accidentally stepped on a submarine cable, disrupting global internet traffic for days.
Viewing the map of states as a graph

Each state is a point on the graph, and neighboring states are connected by an edge.

Do the same thing for a map of the world showing countries

http://www.cs.cmu.edu/~bryant/boolean/maps.html
A circuit graph (Intel 4004)
This is a graph
This is a graph(ical model) that has learned to recognize cats.
Graphs

$K_5$

$K_{3,3}$
Undirected graphs

- A undirected graph is a pair \((V, E)\) where
  - \(V\) is a (finite) set
  - \(E\) is a set of pairs \((u, v)\) where \(u,v \in V\)
    - Often require \(u \neq v\) (i.e. no self-loops)

- Element of \(V\) is called a vertex or node
- Element of \(E\) is called an edge or arc

- \(|V| = \text{size of } V\), often denoted by \(n\)
- \(|E| = \text{size of } E\), often denoted by \(m\)
A **undirected graph** is a pair \((V, E)\) where
- \(V\) is a (finite) set
- \(E\) is a set of pairs \((u, v)\) where \(u, v \in V\)
  - Often require \(u \neq v\) (i.e. no self-loops)

- Element of \(V\) is called a **vertex** or **node**
- Element of \(E\) is called an **edge** or **arc**

- \(|V| = \text{size of } V\), often denoted by \(n\)
- \(|E| = \text{size of } E\), often denoted by \(m\)
Directed graphs

- A directed graph (digraph) is a lot like an undirected graph
  - \( V \) is a (finite) set
  - \( E \) is a set of ordered pairs \((u, v)\) where \( u, v \in V \)

- Every undirected graph can be easily converted to an equivalent directed graph via a simple transformation:
  - Replace every undirected edge with two directed edges in opposite directions

- ... but not vice versa

\( V = \{A, B, C, D, E\} \)
\( E = \{(A, C), (B, A), (B, C), (C, D), (D, C)\} \)
\(|V| = 5\)
\(|E| = 5\)
Graph terminology

- Vertices $u$ and $v$ are called
  - the source and sink of the directed edge $(u, v)$, respectively
  - the endpoints of $(u, v)$ or $\{u, v\}$

- Two vertices are adjacent if they are connected by an edge
Graph terminology

- 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 a sequence $v_0, v_1, v_2, \ldots, v_p$ of vertices such that for $0 \leq i < p$,
  - $(v_i, v_{i+1}) \in E$ if the graph is directed
  - $\{v_i, v_{i+1}\} \in E$ if the graph is undirected

- The **length of a path** is its number of edges

- A path is **simple** if it doesn’t repeat any vertices
More graph terminology

- **A cycle** is a path $v_0, v_1, v_2, ..., v_p$ such that $v_0 = v_p$
- 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**
A directed or undirected graph is **bipartite** if the vertices can be partitioned into two sets such that no edge connects two vertices in the same set.

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

Adjacency List

1 → 2 → 4
2 → 3
3
4 → 2 → 3

Adjacency Matrix

\[
\begin{bmatrix}
1 & 0 & 0 & 1 & 1 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0
\end{bmatrix}
\]
Graph Quiz

Which of the following two graphs are DAGs?

Directed Acyclic Graph

Graph 1:

Graph 2:

Which of the following two graphs are DAGs?
Graph Quiz

1 → 3 → 2

1 → 3

2 → 3

3 → 1

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Adjacency matrix or adjacency list?

- \( v = \) number of vertices
- \( e = \) number of edges
- \( d(u) = \) degree of \( u = \) no. edges leaving \( u \)

**Adjacency Matrix**

- Uses space \( O(v^2) \)
- Enumerate all edges in time \( O(v^2) \)
- Answer “Is there an edge from \( u_1 \) to \( u_2 \)?” in \( O(1) \) time
- Better for dense graphs (lots of edges)
Adjacency matrix or adjacency list?

- $v =$ number of vertices
- $e =$ number of edges
- $d(u) =$ degree of $u =$ no. edges leaving $u$

### Adjacency List
- Uses space $O(v + e)$
- Enumerate all edges in time $O(v + e)$
- Answer “Is there an edge from $u1$ to $u2$?” in $O(d(u1))$ time
- Better for sparse graphs (fewer edges)
What can we do on graphs?

- **Search**
  - Depth-first search
  - Breadth-first search
- **Shortest paths**
  - Dijkstra's algorithm
- **Minimum spanning trees**
  - Jarnik/Prim/Dijkstra algorithm
  - Kruskal's algorithm