Graphs

Unweighted, undirected

Unweighted, directed

Weighted, directed
What type of a graph is this?
Graphs

- Seattle
  - $70 to Los Angeles
  - $90 to Denver
- Denver
  - $280 to Seattle
  - $45 to Atlanta
- New York
  - $70 to Denver
  - $50 to Atlanta
- Atlanta
  - $120 to Los Angeles
- Los Angeles
- Miami

Undirected, weighted
What is the cheapest way to get to Miami from LA?
What type of a graph is this?
Graphs

Algebra → Geometry → Trigonometry → Calculus

Statistics → Computer science → Machine Learning

Unweighted, directed
Graph representation

- Vertices
- Edges

How to connect vertices through edges?
Graph applications examples

- **Networking**: finding the shortest paths, managing the flow
- **Google maps**: building transportation systems
- **Google page ranking**: evaluating pages by the number of references they have
- **Facebook**: friends suggestion algorithm
- **Neural networks**
- **Biological networks**
Graph representation: adjacency matrix
Graph representation: adjacency matrix

<table>
<thead>
<tr>
<th></th>
<th>Alg</th>
<th>Geo</th>
<th>Tri</th>
<th>Cal</th>
<th>Sta</th>
<th>CS</th>
<th>ML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alg</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Geo</td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tri</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cal</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sta</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>ML</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Graph representation: adjacency matrix

<table>
<thead>
<tr>
<th></th>
<th>Alg</th>
<th>Geo</th>
<th>Tri</th>
<th>Cal</th>
<th>Sta</th>
<th>CS</th>
<th>ML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alg</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Geo</td>
<td></td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Tri</td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Cal</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Sta</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>CS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>ML</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>
Graph representation: adjacency matrix

<table>
<thead>
<tr>
<th></th>
<th>Alg</th>
<th>Geo</th>
<th>Tri</th>
<th>Cal</th>
<th>Sta</th>
<th>CS</th>
<th>ML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alg</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Geo</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Tri</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cal</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sta</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ML</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Diagram:
- Algebra
- Geometry
- Trigonometry
- Calculus
- Statistics
- Computer science
- Machine Learning

Each node is connected to others based on the adjacency matrix.
Graph representation: adjacency matrix

<table>
<thead>
<tr>
<th></th>
<th>Alg</th>
<th>Geo</th>
<th>Tri</th>
<th>Cal</th>
<th>Sta</th>
<th>CS</th>
<th>ML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alg</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Geo</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Tri</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Cal</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sta</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ML</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Diagram:

- Algebra
- Geometry
- Trigonometry
- Calculus
- Statistics
- Computer Science
- Machine Learning
Graph representation: adjacency matrix

<table>
<thead>
<tr>
<th></th>
<th>Alg</th>
<th>Geo</th>
<th>Tri</th>
<th>Cal</th>
<th>Sta</th>
<th>CS</th>
<th>ML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alg</td>
<td>✅ 1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Geo</td>
<td>0</td>
<td>✅ 1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Tri</td>
<td>0</td>
<td>0</td>
<td>✅ 1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Cal</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>✅ 1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Sta</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>✅ 1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>CS</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>✅ 1</td>
<td>1</td>
</tr>
<tr>
<td>ML</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>✅ 1</td>
</tr>
</tbody>
</table>
Graph representation: adjacency list

<table>
<thead>
<tr>
<th>keys</th>
<th>values</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Algebra</strong></td>
<td>Geometry, Statistics</td>
</tr>
<tr>
<td><strong>Geometry</strong></td>
<td>Trigonometry</td>
</tr>
<tr>
<td><strong>Trigonometry</strong></td>
<td>Calculus</td>
</tr>
<tr>
<td><strong>Calculus</strong></td>
<td>Machine Learning</td>
</tr>
<tr>
<td><strong>Statistics</strong></td>
<td>Machine Learning</td>
</tr>
<tr>
<td><strong>Computer Science</strong></td>
<td>Machine learning</td>
</tr>
</tbody>
</table>
Graph traversals

- The process of visiting each vertex in a graph.
  - Checking, updating vertexes during the visit.
- Is there a challenge compared to tree traversals?
  - A different structure: node can be visited multiple times
- Traversal algorithms:
  - Breadth first search
  - Depth first search
Traversal algorithms

● DFS: visiting a child before siblings
  ○ Implementation?
  ○ Adding children into a stack

● BFS: visiting a sibling before visiting children
  ○ Implementation?
  ○ Adding children into a queue

● Practice: implement BFS and DFS for a graph
Depth-first search

dfs(node=1, graph)
Depth-first search

dfs(node=1, graph)

stack
Depth-first search

dfs(node=1, graph)
Depth-first search

dfs(node=1, graph)
Depth-first search

dfs(node=1, graph)
Depth-first search

dfs(node=1, graph)
Depth-first search

dfs(node=1, graph)
Depth-first search

dfs(node=1, graph)
Depth-first search

dfs(node=1, graph)

1 3 6

1 3 6

stack

5
4
2
Depth-first search

dfs(node=1, graph)

1 3 6 5

stack

4
2

stack
Depth-first search

dfs(node=1, graph)

1 -> 3 -> 6 -> 5
Depth-first search

dfs(node=1, graph)

stack

1 3 6 5 4
Depth-first search

dfs(node=1, graph)

1 → 3 → 6 → 5 → 4

stack

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>4</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>
Depth-first search

dfs(node=1, graph)

stack

1 3 6 5 4
Depth-first search

dfs(node=1, graph)
Depth-first search

dfs(node=1, graph)
Breadth-first search

bfs(node=1, graph)
Breadth-first search

bfs(node=1, graph)
Breadth-first search

bfs(node=1, graph)
Breadth-first search

bfs(node=1, graph)
Breadth-first search

bfs(node=1, graph)
Breadth-first search

bfs(node=1, graph)
Breadth-first search

bfs(node=1, graph)

Queue

front 3 5 3 rear

Queue
Breadth-first search

bfs(node=1, graph)
Breadth-first search

bfs(node=1, graph)

Queue

front 5 3 6 rear

Queue
Breadth-first search

bfs(node=1, graph)

Queue

<table>
<thead>
<tr>
<th>front</th>
<th>3</th>
<th>6</th>
<th>rear</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>3</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Breadth-first search

bfs(node=1, graph)
Breadth-first search

bfs(node=1, graph)

front | 6 | 4 | rear
Queue
Breadth-first search

bfs(node=1, graph)
Breadth-first search

bfs(node=1, graph)

Queue

front 4 4 5 rear

Queue
Breadth-first search

bfs(node=1, graph)

Queue

front 4 5 rear

Queue
Breadth-first search

bfs(node=1, graph)

Queue

front 4 5 6 rear

Queue
Breadth-first search

bfs(node=1, graph)

Queue

front 5 6 rear
Queue
Breadth-first search

bfs(node=1, graph)
Breadth-first search

bfs(node=1, graph)
Practice problem

- Given a non-weighted directed graph with adjacency list:
  - Find the shortest path between from SRC to DST

```python
def bfs(node=1, graph):
    pass
```

![Graph Diagram](image-url)
Practice problem

- Given a non-weighted directed graph with adjacency list:
  - Find the shortest path between from SRC to DST

bfs(node=1, graph)