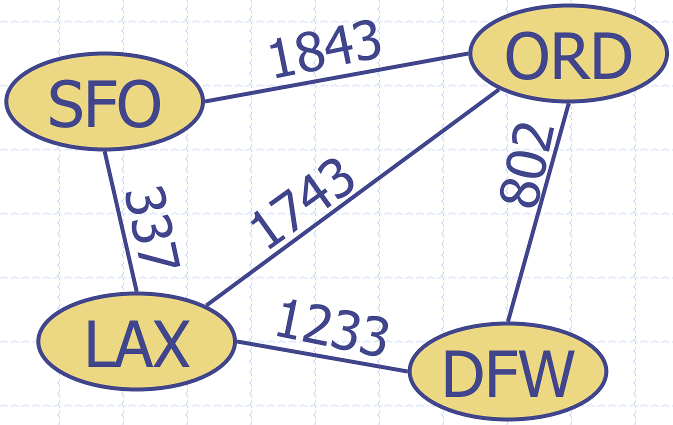
Graphs: Recap



Graphs: Basics

- Directed vs. undirected graph
  - e.g. edges have a direction associated with them
- (Non-uniformly) Weighted graph
  - e.g. edges have a weight associated with them
- Properties
  - $\Sigma_n \deg(\boldsymbol{v}_n) = 2\boldsymbol{m}$
  - $\boldsymbol{m} \leq \boldsymbol{n}\,(\boldsymbol{n} - 1)/2$
- Representation
  - Edge list structure,
  - Adjacency list structure, or
  - Adjacency matrix structure

Graphs: Traversals

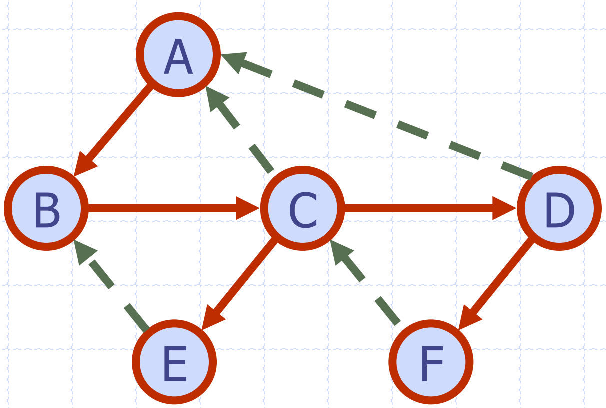
- Depth-first Search
  - Traverse deeply first
  - O(n+m)
- Breadth-first Search
  - Traverse broadly first (“breadth”)
  - O(n+m)

Graphs: Terminology

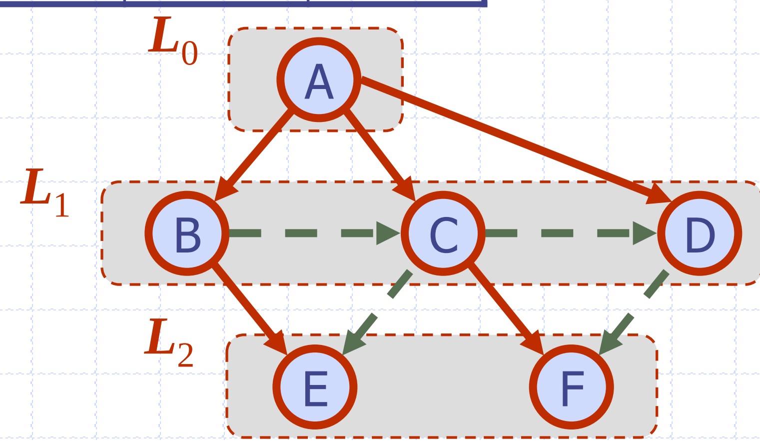
- Path
- Connected
- Subgraph
- Spanning
- Biconnected
  - e.g., separation edges or vertices
- What are these and how do you find them?
  - Connected component
  - Spanning Subgraph
  - Maximally-connected Subgraph
  - Spanning Tree
  - Spanning Forest
  - Biconnected Components

Graphs: DFS vs. BFS

Applications	DFS	BFS
Spanning forest, connected components, paths, cycles	√	√
Shortest paths (for uniformly weighted graphs)		√
Biconnected components	√	



DFS



BFS

Directed Graphs: Terminology

- Reachability
  - Is a vertex u "reachable" from v
    - e.g., can you get to MIA from HNL?
- Strongly-Connected Components
  - Can you get to any city from any city
- Transitive Closure
  - If I can get to MIA from HNL and to JFK from MIA, then I can get to JFK from HNL
  - The transitive closure graph of a graph G extracts this information
  - Algorithms:
    - Naïve: $O(n(n+m))$ to $O(n^3)$
    - Floyd-Warshall Algorithm: $O(n^3)$ but low-cost

Directed Graphs: Terminology

- Directed Acyclic Graph or DAG
  - A directed graph with no cycles
  - Permits a topological sorting
    - e.g., a sorting of the nodes from beginning to end
    - A topological sorting can be done using a modified DFS traversal

Graphs: Shortest Path

- Given a weighted graph and two vertices u and v, we want to find a path of minimum total weight between u and v
  - BFS gives us shortest paths for a uniformly weighted graph – this is the concept generalized to weighted graphs
- Note: related to Traveling Salesman problem which is finding the shortest path that visits all vertices (an NP-complete problem)

Graphs: Shortest Path

- Algorithms for finding shortest paths from a start vertex
  - Dijkstra's
    - Naively grow a "cloud of connected vertices"
    - Assumes non-negative weights
    - O(m log n)
  - Bellman-Ford's
    - Extend's Dijsktra's by carrying along the total weight so far during the expansion
    - Supports negative weights
    - O(nm)
  - DAG-based
    - Assumes a DAG
    - Uses topological sorting
    - O(n+m)

Graphs: Shortest Path

- All shortest path pairs
  - Dijkstra's
    - O(nmlogn)
  - Bellman Ford's
    - $O(n^2m)$
  - Modified Floyd-Warshall
    - $O(n^3)$

Graphs: Minimum Spanning Tree

- A spanning tree of a weighted graph with minimum total edge weight
  - e.g. the lowest cost network uniting all clients

Graphs: Minimum Spanning Tree

- Algorithms
  - Prim-Jarnik's
    - Similar to Dijkstra's: grows a cloud of connected vertices
    - O(mlogn)
  - Kruskal's
    - Maintains a forest of growing clouds of vertices
    - O((n+m)logn)
  - Baruvka's
    - Similar to Kruskal's but at each iteration halves the number of connected components
    - O(mlogn)

Fun Fact: Fastest MST Algorithm

- $O(m\alpha)$
  - α is function of (m, n) but in practice is ≤ 4
- Based on B. Chazelle's "Soft Heap"
  - Amortized cost of operations is O(1) except insert, which is O(log 1/e), for $e \in [0,1/2]$
  - At expense of e*n of keys being "corrupted", faster heap is obtained