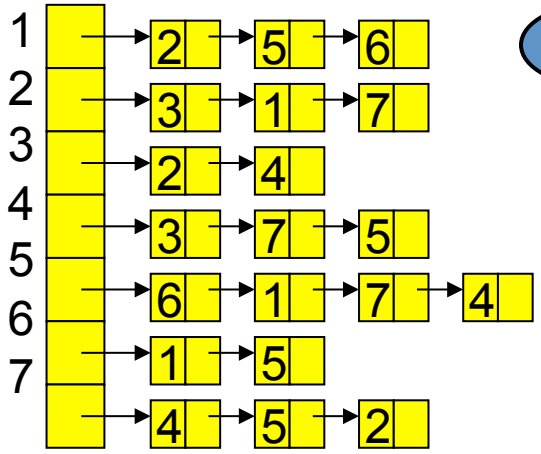
CSE 373: Data Structures and Algorithms

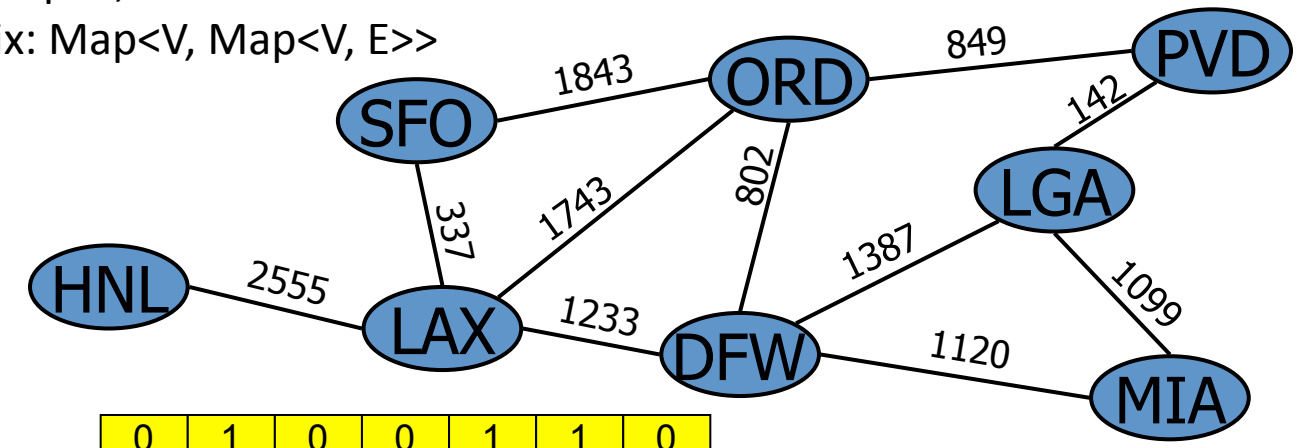
Lecture 20: Graphs IV

Practical implementation

- Not all graphs have vertices/edges that are easily "numbered"
  - how do we actually represent 'lists' or 'matrices' of vertex/edge relationships? How do we quickly look up the edges and/or vertices adjacent to a given vertex?

  - Adjacency list: Map<V, List<V>>
  - Adjacency matrix: Map<V, Map<V, E>>





0	1	0	0	1	1	0
1	0	1	0	0	0	1
0	1	0	1	0	0	0
0	0	1	0	1	0	1
1	0	0	1	0	1	1
1	0	0	0	1	0	0
0	1	0	1	1	0	0

Maps and sets within graphs

since not all vertices can be numbered, we can use:

1. adjacency list
   - each Vertex maps to a List of edges or adjacent Vertices
   - Vertex --> List of Edges
   - to get all edges adjacent to V_1, look up
     List<Edge> neighbors = map.get(V_1)

2. adjacency adjacency matrix map
   - each Vertex maps to a Hash of adjacent
   - Vertex --> (Vertex --> Edge)
   - to find out whether there's an edge from V1 to V2, call *map.get (V1).containsKey(V2)*
   - to get the edge from V1 to V2, call *map.get(V1).get(V2)*

Implementing Graph with Adjacency List

```
public interface Graph<V> {
    public void addVertex(V v);

    public void addEdge(V v1, V v2, int weight);

    public boolean hasEdge(V v1, V v2);

    public Edge<V> getEdge(V v1, V v2);

    public boolean hasPath(V v1, V v2);

    public List<V> getDFSPath(V v1, V v2);

    public String toString();
}
```

Edge class

```
public class Edge<V> {
    public V from, to;
    public int weight;

    public Edge(V from, V to, int weight) {
        if (from == null || to == null) {
            throw new IllegalArgumentException("null");
        }
        this.from = from;
        this.to = to;
        this.weight = weight;
    }

    public String toString() {
        return "<" + from + ", " + to + ", " + weight + ">";
    }
}
```

`VertexInfo` class

```
public class VertexInfo<V> {
   public V v;
   public boolean visited;

   public VertexInfo(V v) {
      this.v = v;
      this.clear();
   }

   public void clear() {
      this.visited = false;
   }
}
```