Data Structures and Algorithms

Lab 9
Graph representation

Today’s Objectives

- Graph
- Planar graph
- Graph representations
  - Edge List Structure
  - Adjacency List Structure
  - Adjacency Matrix Structure
- Programming exercises

Graph data structure

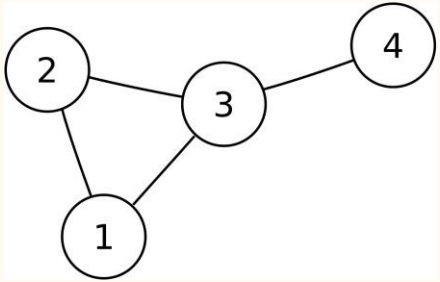
- definition:
  - A graph data structure is a collection of nodes that have data and are connected to other nodes.
  - On facebook, everything is a node. That includes User, Photo, Album, Event, Group, Page, Comment, Story, Video, Link, Note...anything that has data is a node.
  - Every relationship is an edge from one node to another. Whether you post a photo, join a group, like a page, etc., a new edge is created for that relationship.
  - More precisely, a graph is a data structure *(V, E)* that consists of
    - A collection of vertices *V*
    - A collection of edges *E*, represented as ordered pairs of vertices *(u,v)*

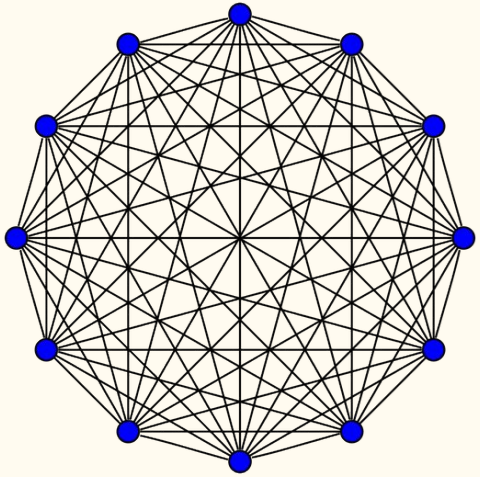
Graph basic operations:

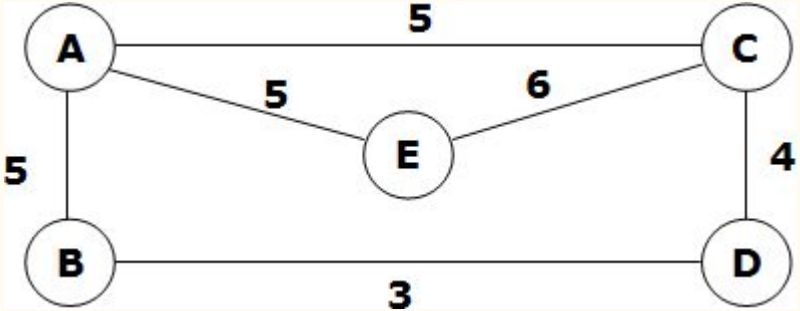
- **`insertVertex(o)`**: insert a vertex storing element `o`
- **`insertEdge(v, u, o)`**: insert an edge (`v`,`u`) storing element `o`
- **`removeVertex(v)`**: remove vertex `v` (and its incident edges)
- **`removeEdge(e)`**: remove edge `e`
- **`incidentEdges(v)`**: edges incident to `v`
- **`areAdjacent(v, u)`**: true `v` and `u` are adjacent vertices
- **`degree(v)`**: number of incident edges to `v`
- **`endVertices(o)`**: an array of the two end vertices of edge `e`
- **`opposite(v, e)`**: the vertex opposite of vertex `v` on edge `e`

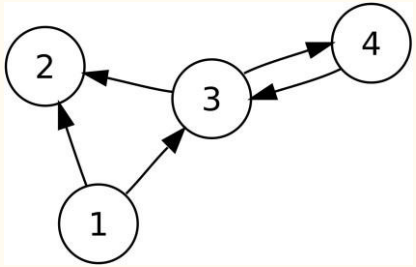
Graph types

- Directed/Undirected
- Cyclic/Acyclic
- Complete
- Weighted
- Sparse/Dense
- Planar

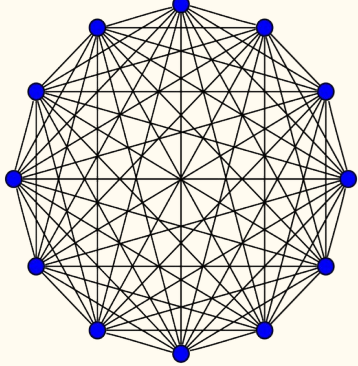




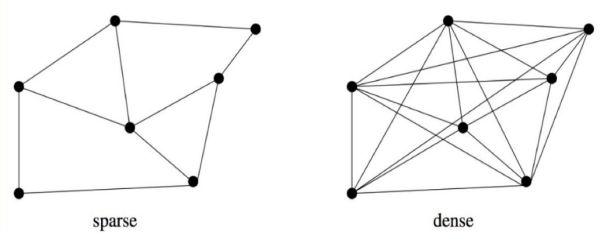




Graph types

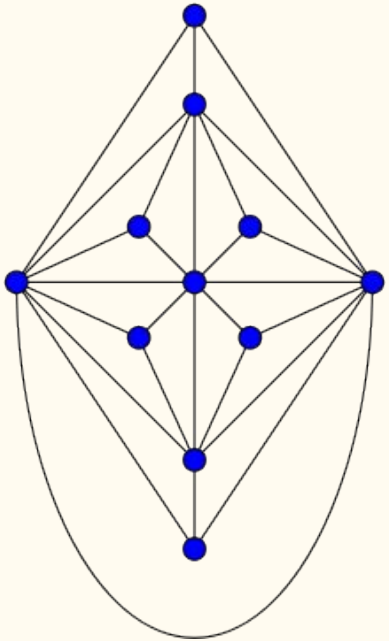


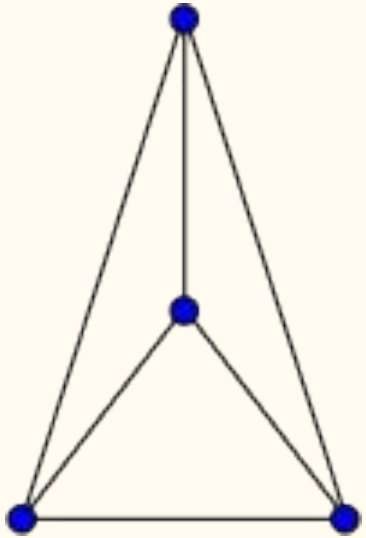
- Directed/Undirected: An edge (u,v) doesn't necessarily mean that there is an edge (v, u) as well. The edges in such a graph are represented by arrows to show the direction of the edge.
- Cyclic: A path that starts and ends at the same vertex
- Complete A graph in which every vertex is directly connected to every other vertex
- Weighted: a graph in which each edge carries a value (weight)
- Sparse/Dense: There are maximum *n(n-1)/2* total pair of vertices(edges) in an undirected graph of n vertices.

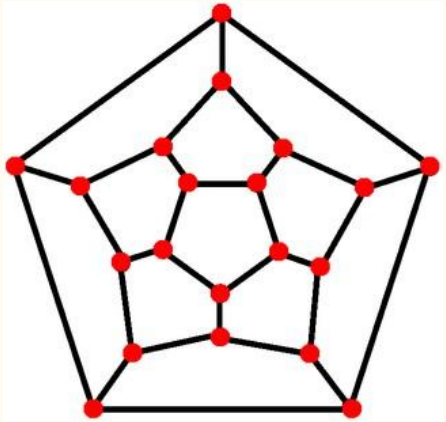


Planar graph

- When a connected graph can be drawn without any edges crossing







Planar graph: Euler’s formula

- A planar graph divides the plane into regions called faces. A face is defined to be an area of the plane that is bounded by edges and cannot be further subdivided (including the outer, infinitely large region),
- A planar graph divides the plans into one or more faces, one of them will be infinite.
- Euler's formula for planar graphs:

$$v - e + f = 2$$

Planar graph: Euler’s formula

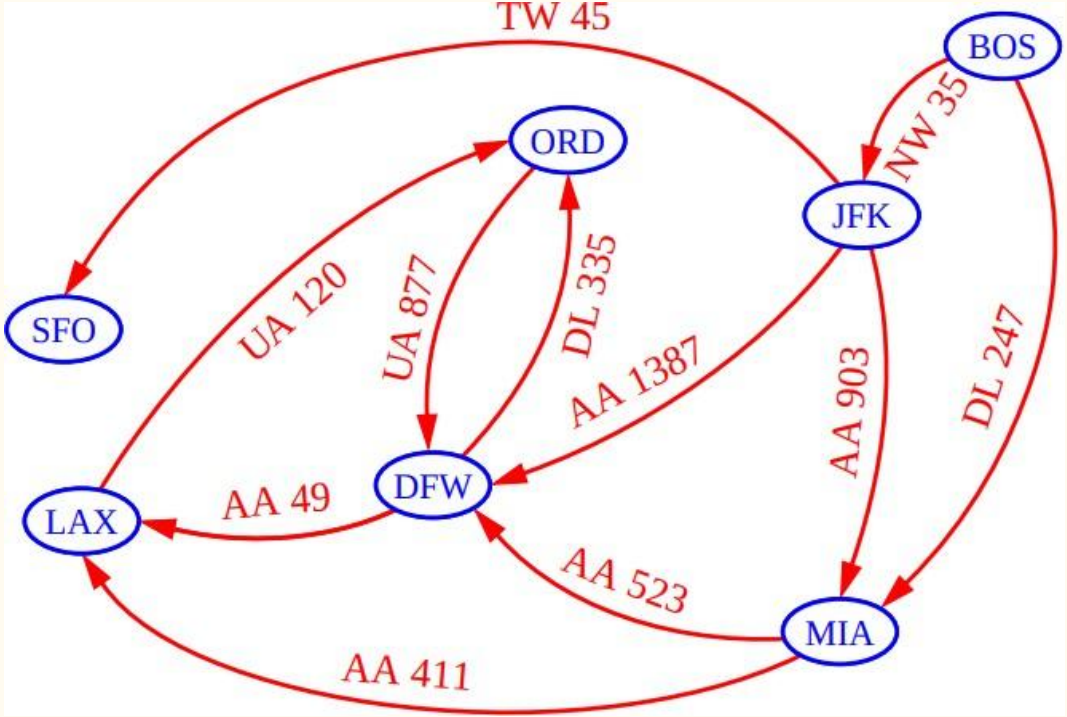
- Euler's formula can also be proved as follows:

  if the graph isn't a tree, then remove an edge which completes a cycle. This lowers both e and *f* by one, leaving *v* − *e* + *f* constant. Repeat until the remaining graph is a tree; trees have *v* = *e* + *1* and *f* = *1*, yielding *v* − *e* + *f* = *2*, i. e., the Euler characteristic is 2.

$$v - e + f = 2$$

Graph representations

- Edge List Structure
- Adjacency List Structure
- Adjacency Matrix Structure



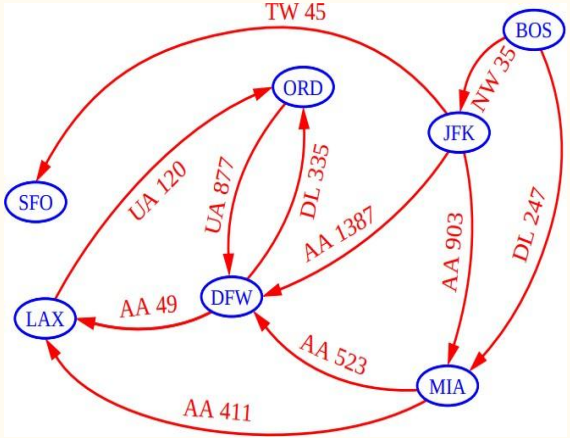
Edge List Structure

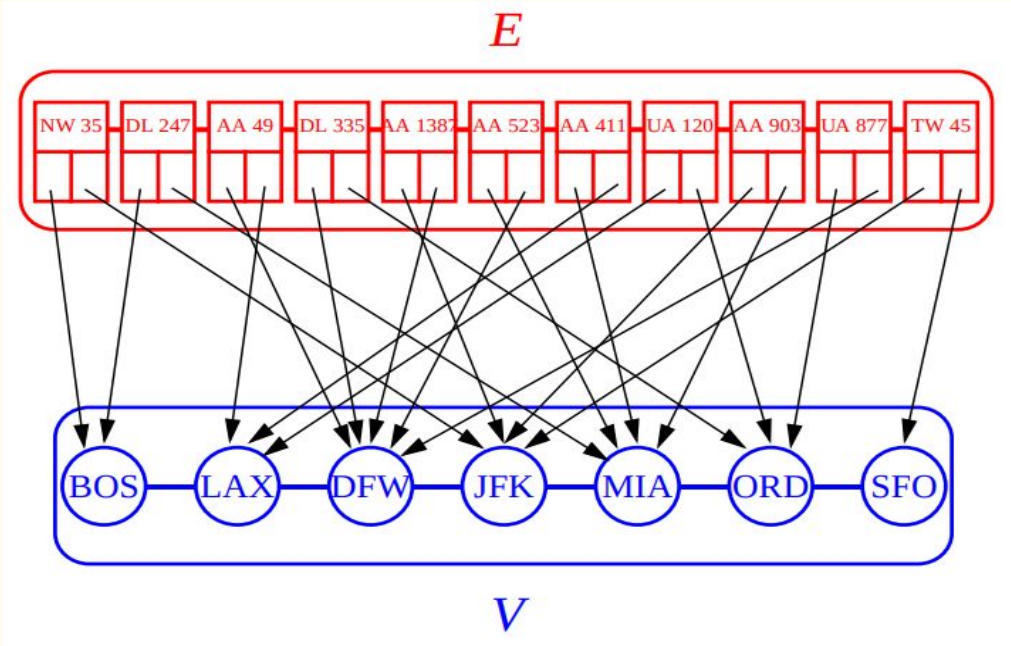
- a graph G = (V, E) where:
  - Vertex List: stores vertices
  - Edge List: stores edges

We can use singly linked lists to store vertices and edges

Edge List Structure

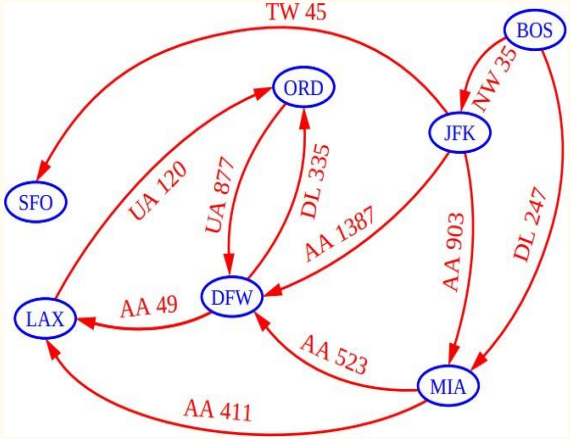
- Stores the vertices and the edges into unsorted sequences
- What is the time complexity of **`incidentEdge(v)`**?

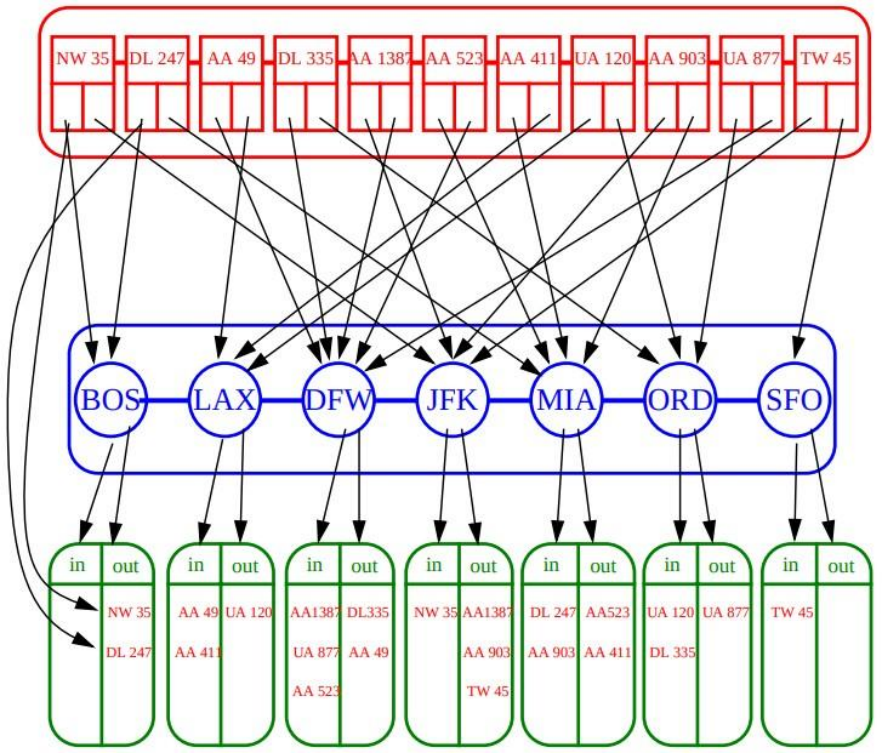




Adjacency List Structure

- Extends the edge list structure by adding incidence containers to each vertex

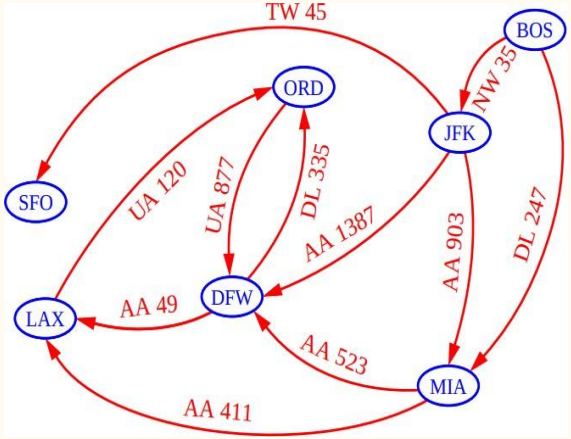




Adjacency Matrix Structure

- Augments the edge list structure with a matrix where each row and column corresponds to a vertex

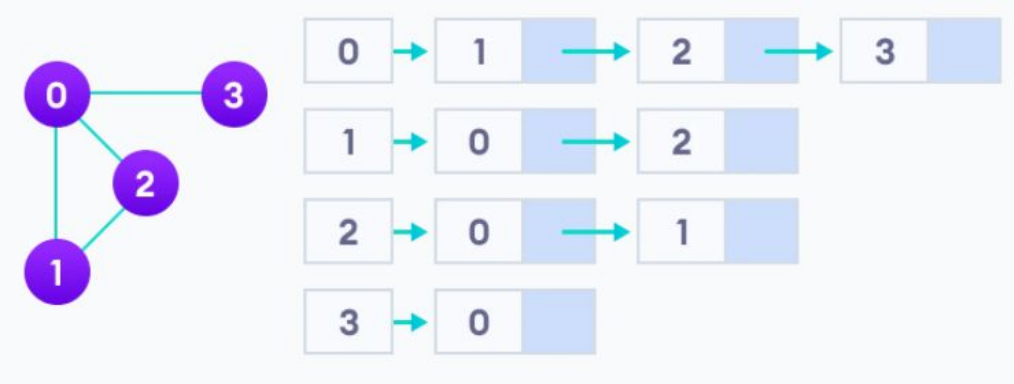
BOS	DFW	JFK	LAX	MIA	ORD	SFO
0	1	2	3	4	5	6



	0	1	2	3	4	5	6
0	Ø	Ø	NW 35	Ø	DL 247	Ø	Ø
1	Ø	Ø	Ø	AA 49	Ø	DL 335	Ø
2	Ø	AA 1387	Ø	Ø	AA 903	Ø	TW 45
3	Ø	Ø	Ø	Ø	Ø	UA 120	Ø
4	Ø	AA 523	Ø	AA 411	Ø	Ø	Ø
5	Ø	UA 877	Ø	Ø	Ø	Ø	Ø
6	Ø	Ø	Ø	Ø	Ø	Ø	Ø

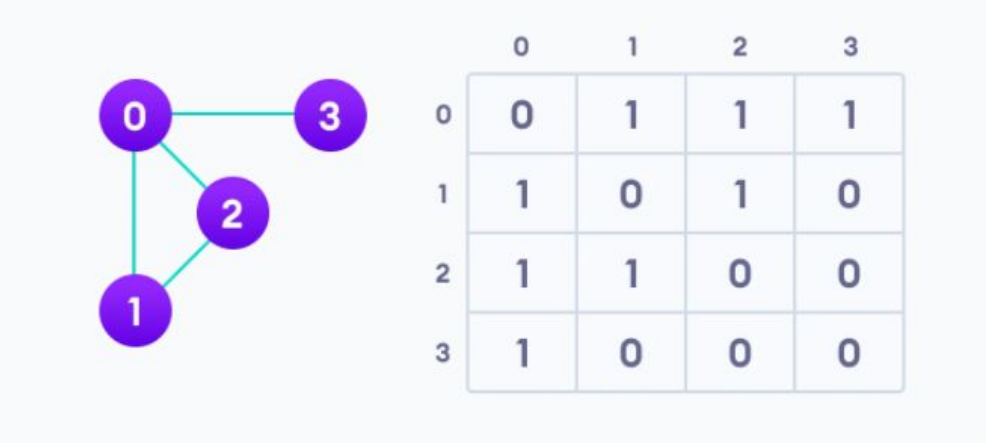
Adjacency List Structure

- An adjacency list represents a graph as an array of linked lists.
- The index of the array represents a vertex and each element in its linked list represents the other vertices that form an edge with the vertex.
- We can either use *HashMaps* and *LinkedLists* to implement graph using adjacency list.



Adjacency Matrix Structure

- An adjacency matrix is a 2D array of V x V vertices. Each row and column represent a vertex.
- If the value of any element a[i][j] is 1, it represents that there is an edge connecting vertex i and vertex j.

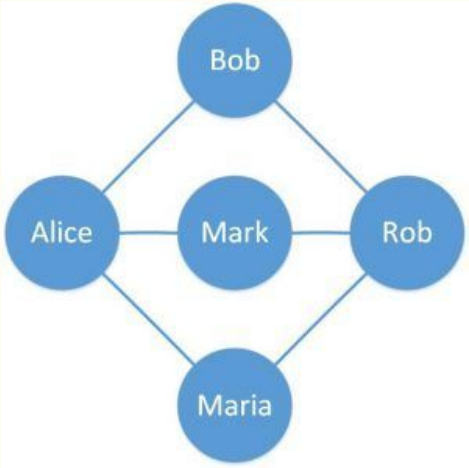


Graph representations

▪ n vertices, m edges ▪ no parallel edges ▪ no self-loops	Edge List	Adjacency List	Adjacency Matrix
Space	$n + m$	$n + m$	n^2
incidentEdges(v)	m	$\deg(v)$	n
areAdjacent (v, w)	m	$\min(\deg(v), \deg(w))$	1
insertVertex(o)	1	1	n^2
insertEdge(v, w, o)	1	1	1
removeVertex(v)	m	$\deg(v)$	n^2
removeEdge(e)	1	1	1

Coding exercise

- Implement an Adjacency List graph which will represent the friendship relationship below



See You next week!