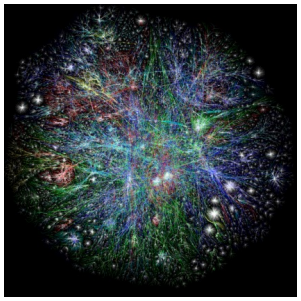
CS 225

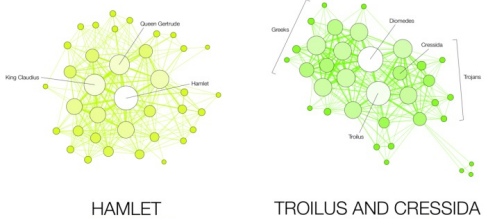
Data Structures

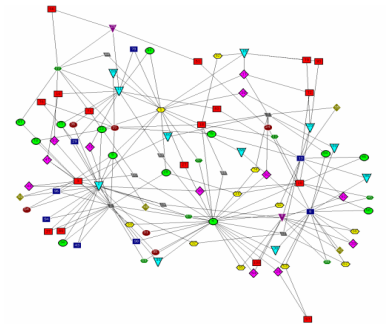
October 21 – Graph Implementations

G Carl Evans

Graphs

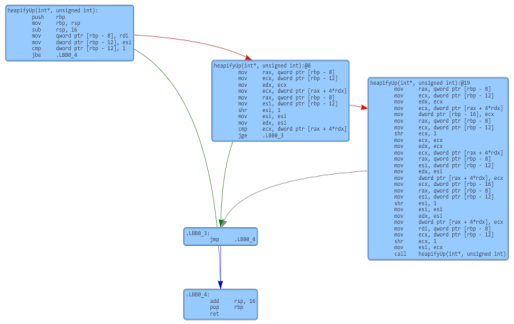


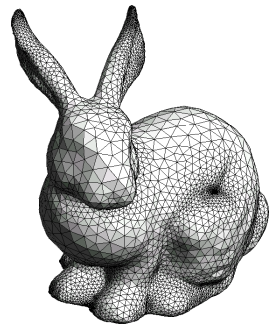


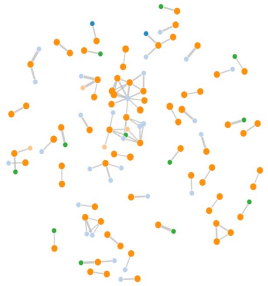


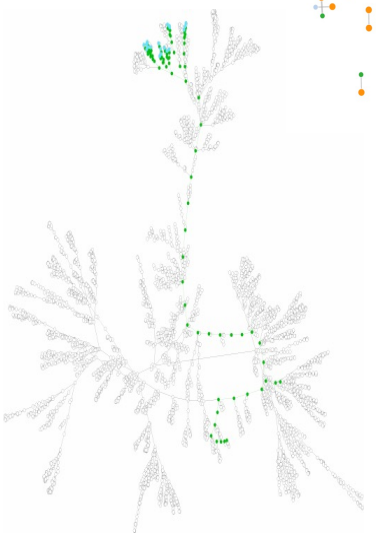
To study all of these structures:

1. A common vocabulary
2. Graph implementations
3. Graph traversals
4. Graph algorithms







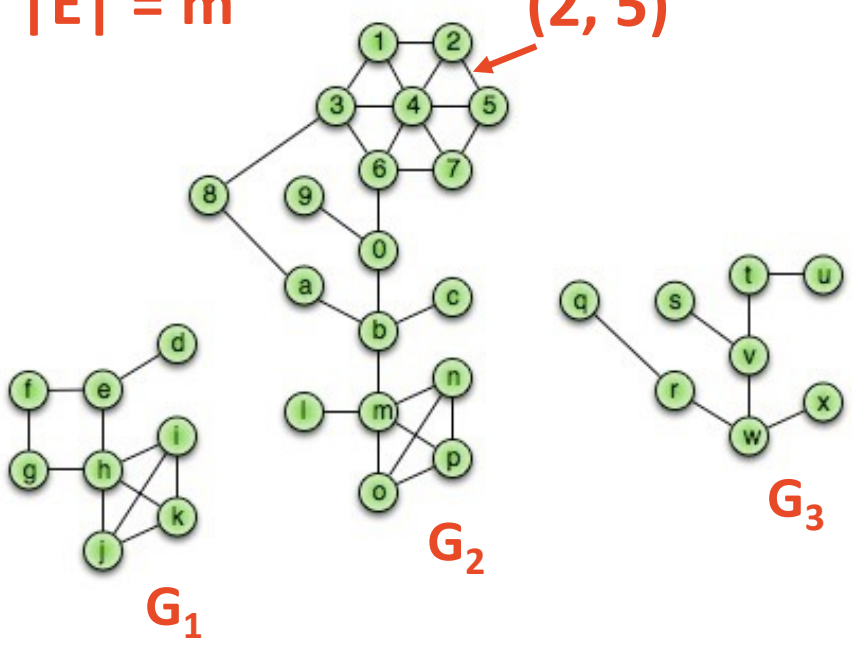


Graph Vocabulary

G = (V, E)

|V| = n

|E| = m



Incident Edges:
I(v) = { {x, v} in E }

Degree(v): |I|

Adjacent Vertices:
A(v) = { x : {x, v} in E }

Walk(G_2): Sequence of vertices connected by edges

Path(G_1): A Walk with no repeated vertices

Cycle(G_1): Path with a common begin and end vertex with at least 3 vertices.

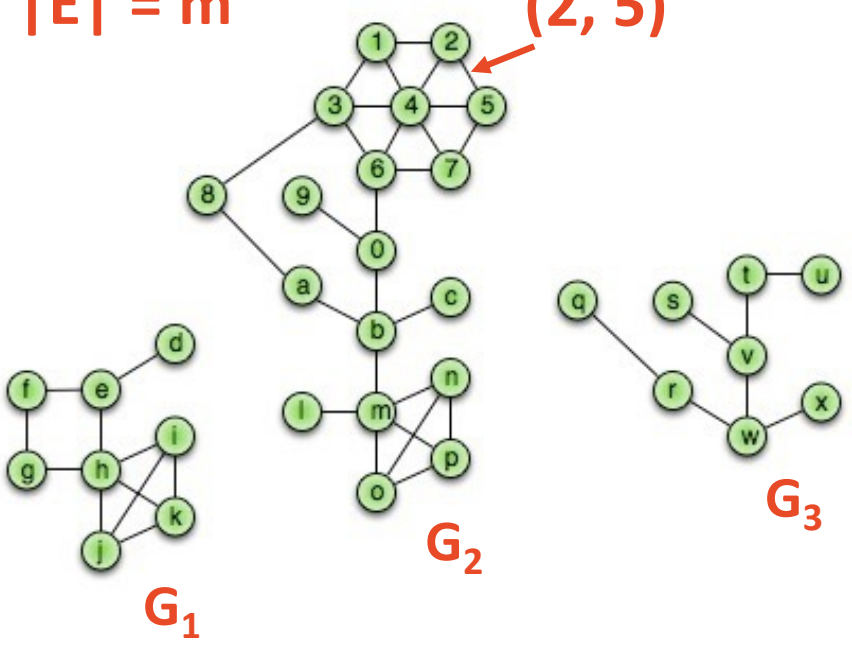
Simple Graph(G): A graph with no self loops or multi-edges.

Graph Vocabulary

G = (V, E)

|V| = n

|E| = m



Subgraph(G):

G' = (V', E'):

V' ∈ V, E' ∈ E, and

(u, v) ∈ E' → u ∈ V', v ∈ V'

Connected subgraph(G)

Connected component(G)

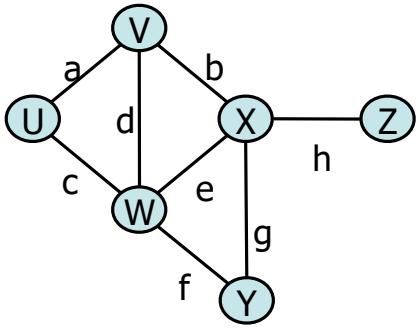
Acyclic subgraph(G)

Spanning tree(G)

Running times are often reported by **n**, the number of vertices, but often depend on **m**, the number of edges.

How many edges? **Minimum edges:**

Not Connected:



Connected*:

Maximum edges:

Simple:

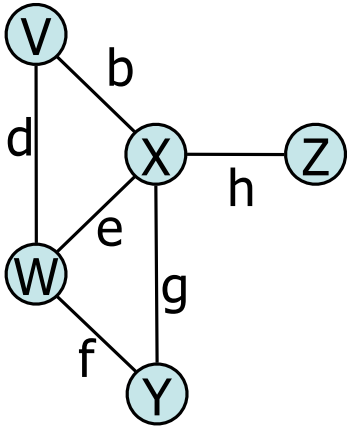
Not simple:

$$\sum_{v \in V} \deg(v) =$$

Graph ADT

Data:

- **Vertices**
- **Edges**



Functions:

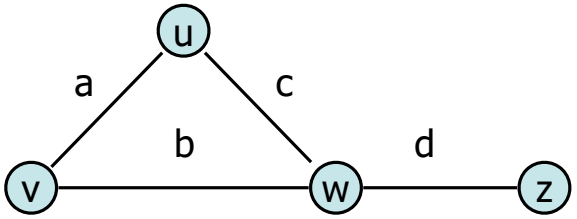
- **insertVertex(K key);**
- **insertEdge(Vertex v1, Vertex v2, K key);**

- **removeVertex(Vertex v);**
- **removeEdge(Vertex v1, Vertex v2);**

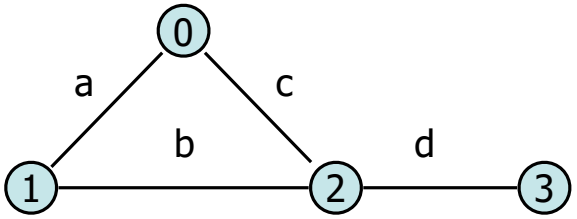
- **incidentEdges(Vertex v);**
- **areAdjacent(Vertex v1, Vertex v2);**

- **origin(Edge e);**
- **destination(Edge e);**

Graph Implementation Idea



Graph Implementation: Edge List



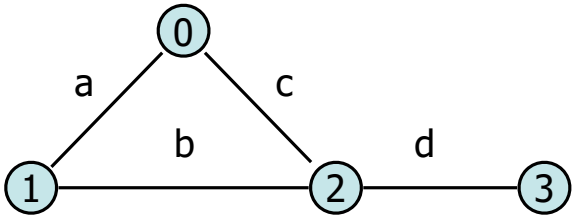
Vertex Collection:

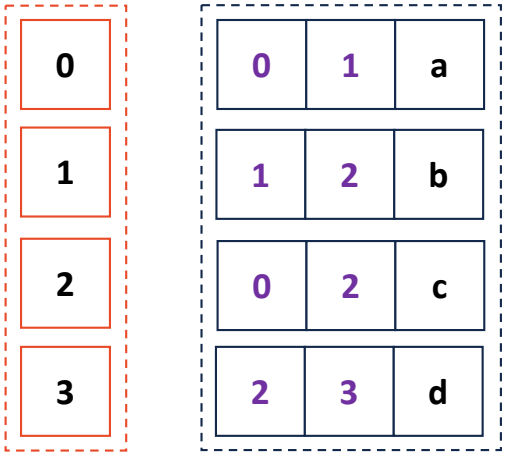
0
1
2
3

Edge Collection:

0	1	a
1	2	b
0	2	c
2	3	d

Graph Implementation: Edge List

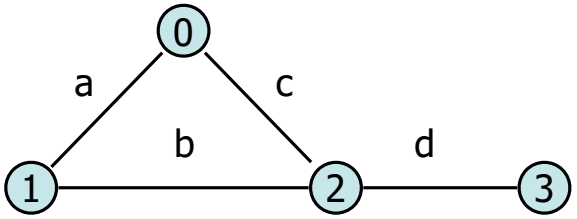


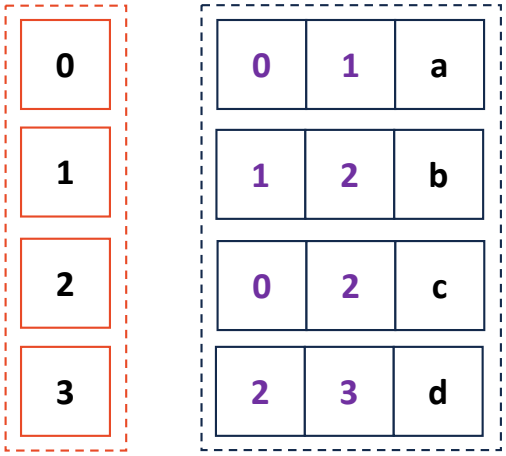


insertVertex(K key):

removeVertex(Vertex v):

Graph Implementation: Edge List



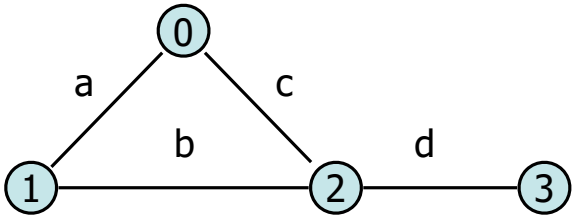


incidentEdges(Vertex v):

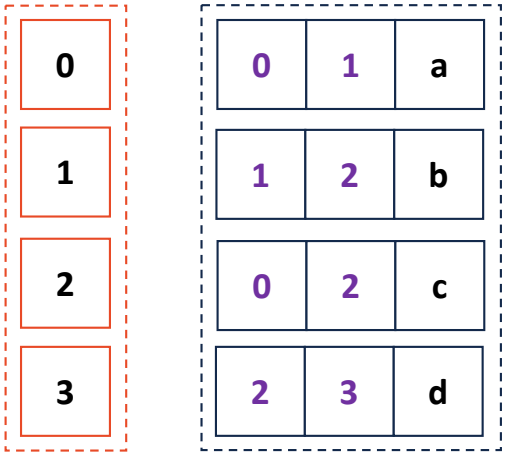
areAdjacent(Vertex v1, Vertex v2):

```
G.incidentEdges(v1).contains(v2)
```

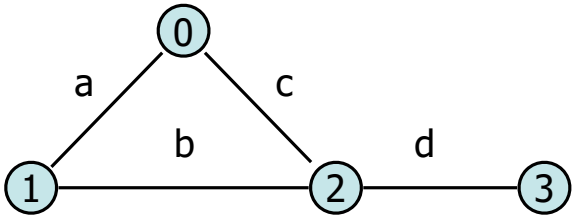
Graph Implementation: Edge List



insertEdge(Vertex v1, Vertex v2, K key):



Graph Implementation: Adjacency Matrix

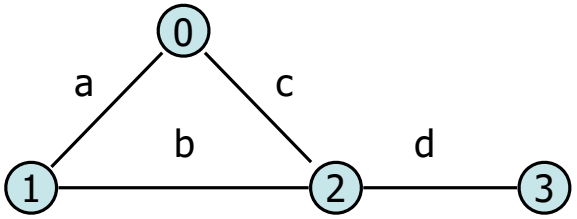


0
1
2
3

0	1	a
1	2	b
0	2	c
2	3	d

	0	1	2	3
0				
1				
2				
3				

Graph Implementation: Adjacency Matrix

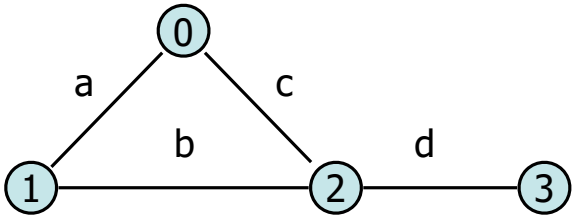


0
1
2
3

0	1	a
1	2	b
0	2	c
2	3	d

	0	1	2	3
0				
1				
2				
3				

Graph Implementation: Adjacency Matrix

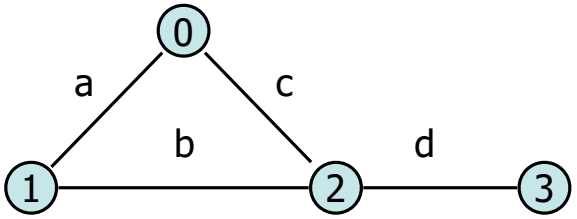


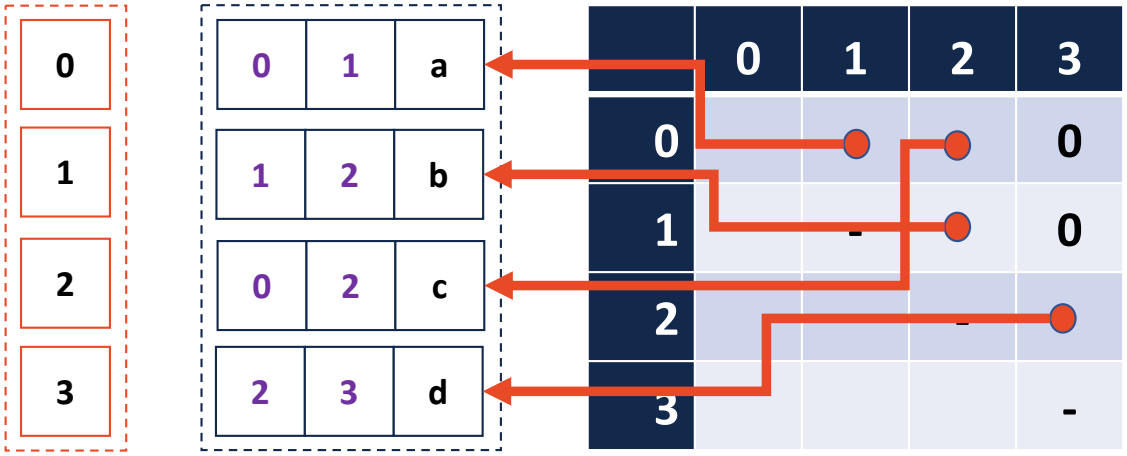
0
1
2
3

0	1	a
1	2	b
0	2	c
2	3	d

	0	1	2	3
0	-	1	1	0
1		-	1	0
2			-	1
3				-

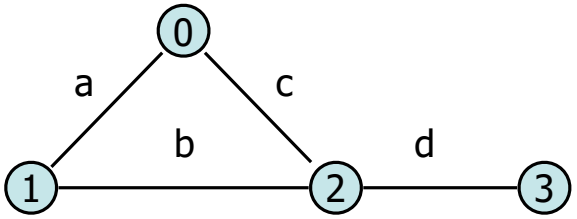
Graph Implementation: Adjacency Matrix

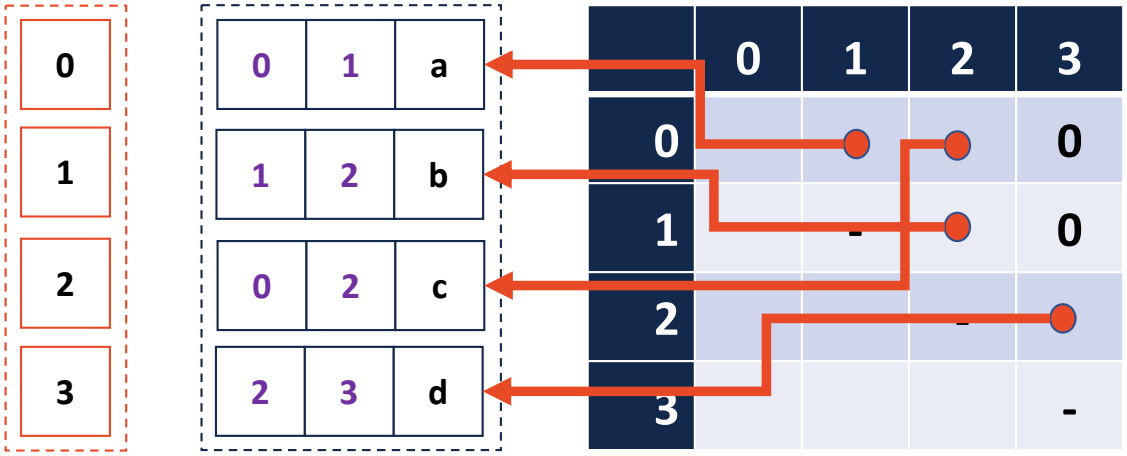




Graph Implementation: Adjacency Matrix

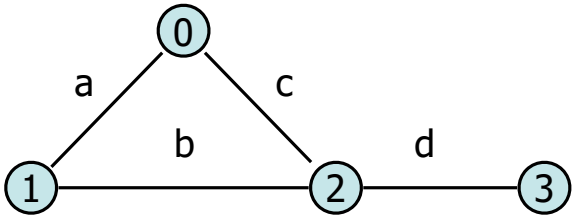
incidentEdges(Vertex v):

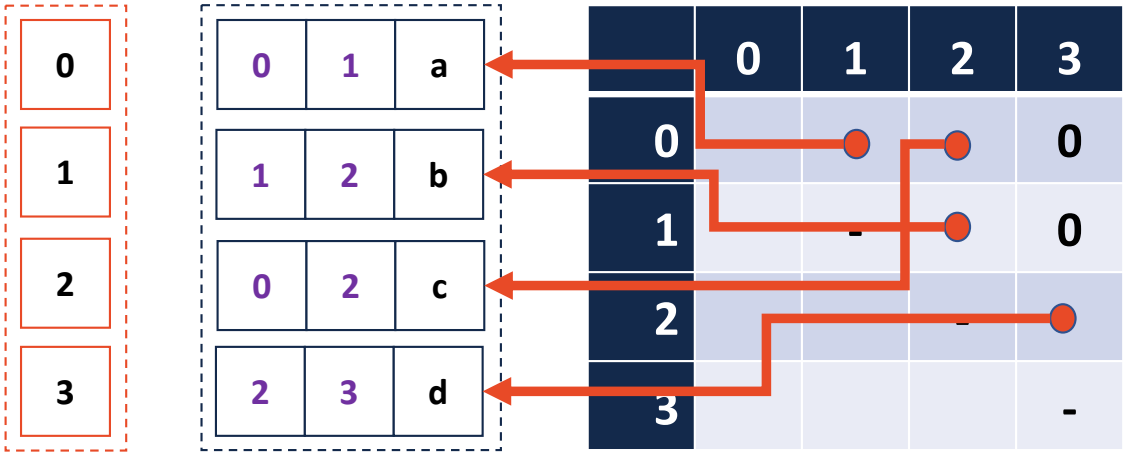




Graph Implementation: Adjacency Matrix

areAdjacent(Vertex v1, Vertex v2):





Graph Implementation: Adjacency Matrix

insertEdge(Vertex v1, Vertex v2, K key):