# Algorithms and Data Structures for Data Science

## Graph Implementations 2

CS 277

Brad Solomon

March 27, 2024

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

Department of Computer Science

# Learning Objectives

Practice implementing complex data structures (graphs)

Compare and contrast different implementations

Review Big O concepts in the context of graphs

# Graph ADT

**Find**

getVertices() — return the list of vertices in a graph

getEdges(v) — return the list of edges that touch the vertex v

areAdjacent(u, v) — returns a bool based on if an edge from u to v exists

**Insert**

insertVertex(v) — adds a vertex to the graph

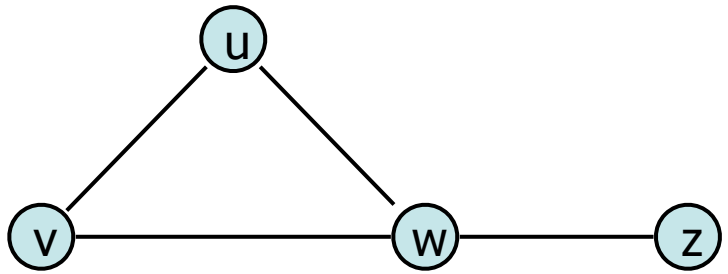
insertEdge(u, v) — adds an edge to the graph

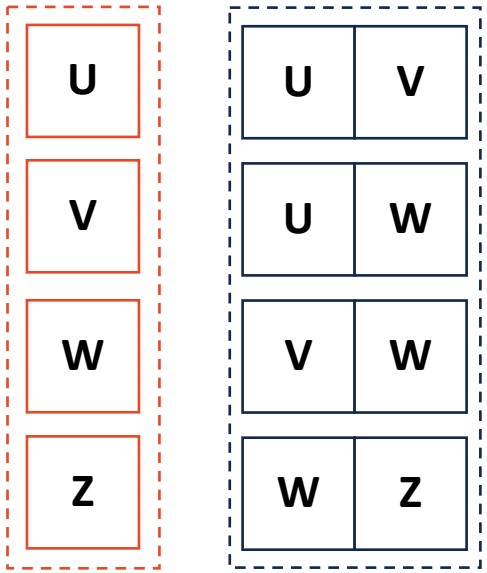
**Remove**

removeVertex(v) — removes a vertex from the graph

removeEdge(u, v) — removes an edge from the graph

# Graph Implementation: Edge List



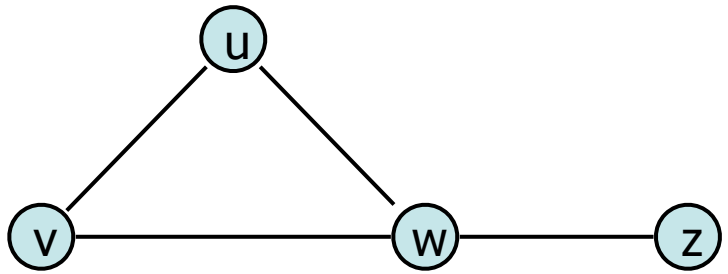



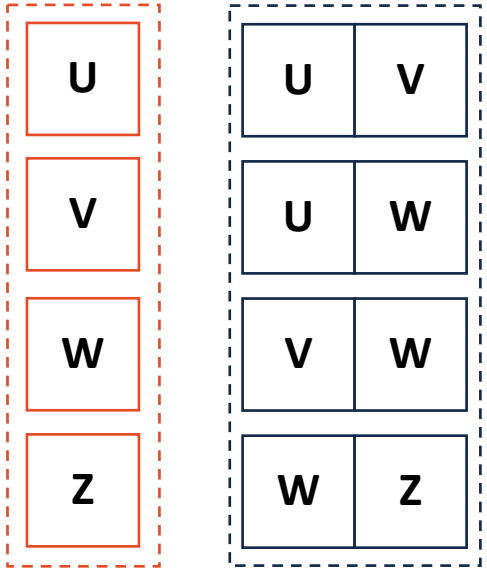




| Expressed as O(f) | Edge List |
|---|---|
| Space | n+m |
| insertVertex(v) | |
| removeVertex(v) | |
| insertEdge(u, v) | |
| removeEdge(u, v) | |
| getEdges(v) | |
| areAdjacent(u, v) | |

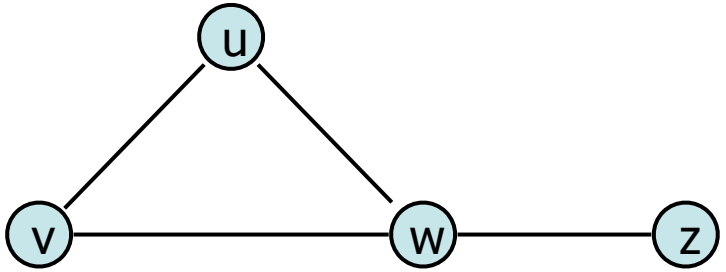
# Graph Implementation: Edge List




| Expressed as O(f) | Edge List |
|---|---|
| Space | n+m |
| insertVertex(v) | 1* |
| removeVertex(v) | m** |
| insertEdge(u, v) | 1 |
| removeEdge(u, v) | m |
| getEdges(v) | m |
| areAdjacent(u, v) | m |

# Graph Implementation: Adjacency Matrix




| | |
|---|---|
| u | 0 |
| v | 1 |
| w | 2 |
| z | 3 |

| | u | v | w | z |
|---|---|---|---|---|
| u | 0 | 1 | 1 | 0 |
| v | 1 | 0 | 1 | 0 |
| w | 1 | 1 | 0 | 1 |
| z | 0 | 0 | 1 | 0 |

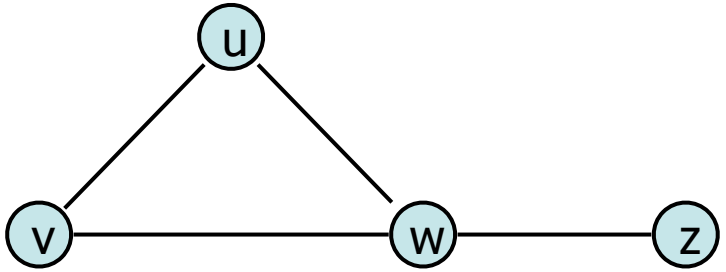
**Vertex Storage:**

A dictionary (key = label, value = index)

**Edge Storage:**

A 2D matrix storing edge existence

# Graph Implementation: Adjacency Matrix



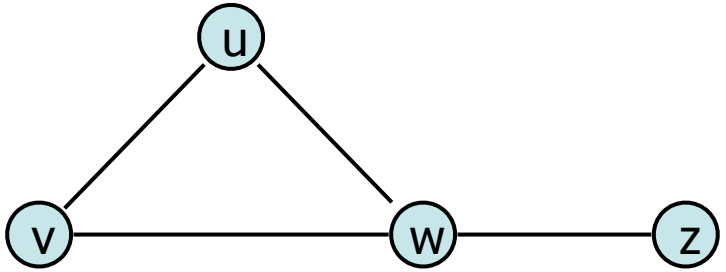

**getVertices():**

```
return self.vertices.keys()
```

| U | 0 |
|---|---|
| V | 1 |
| W | 2 |
| Z | 3 |

| | u | v | w | z |
|---|---|---|---|---|
| u | 0 | 1 | 1 | 0 |
| v | 1 | 0 | 1 | 0 |
| w | 1 | 1 | 0 | 1 |
| z | 0 | 0 | 1 | 0 |

# Graph Implementation: Adjacency Matrix

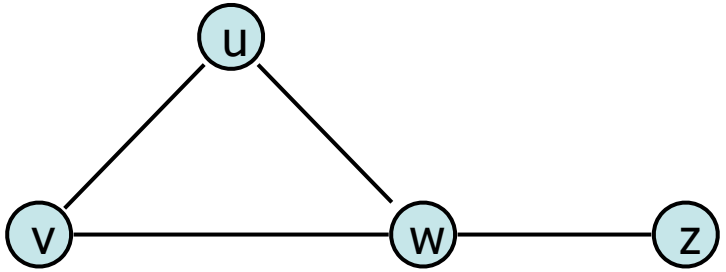



**getEdges(v):**

| U | 0 |
|---|---|
| V | 1 |
| W | 2 |
| Z | 3 |

| | u | v | w | z |
|---|---|---|---|---|
| u | 0 | 1 | 1 | 0 |
| v | 1 | 0 | 1 | 0 |
| w | 1 | 1 | 0 | 1 |
| z | 0 | 0 | 1 | 0 |

# Graph Implementation: Adjacency Matrix

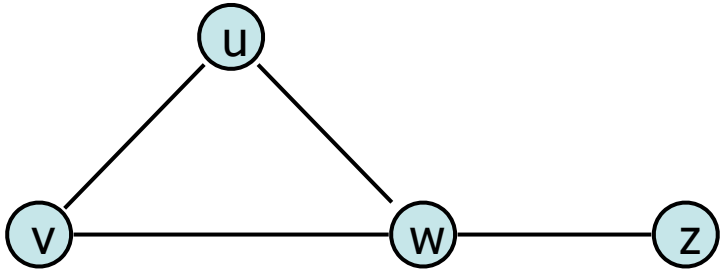



**areAdjacent(u, v):**

| U | 0 |
|---|---|
| V | 1 |
| W | 2 |
| Z | 3 |

| | u | v | w | z |
|---|---|---|---|---|
| u | 0 | 1 | 1 | 0 |
| v | 1 | 0 | 1 | 0 |
| w | 1 | 1 | 0 | 1 |
| z | 0 | 0 | 1 | 0 |

# Graph Implementation: Adjacency Matrix

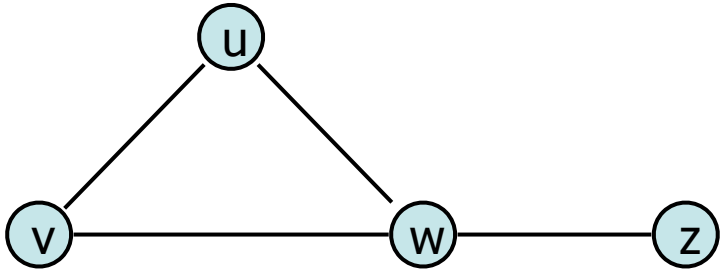



**insertVertex(v):**

| U | 0 |
|---|---|
| V | 1 |
| W | 2 |
| Z | 3 |

| | u | v | w | z |
|---|---|---|---|---|
| u | 0 | 1 | 1 | 0 |
| v | 1 | 0 | 1 | 0 |
| w | 1 | 1 | 0 | 1 |
| z | 0 | 0 | 1 | 0 |

# Graph Implementation: Adjacency Matrix

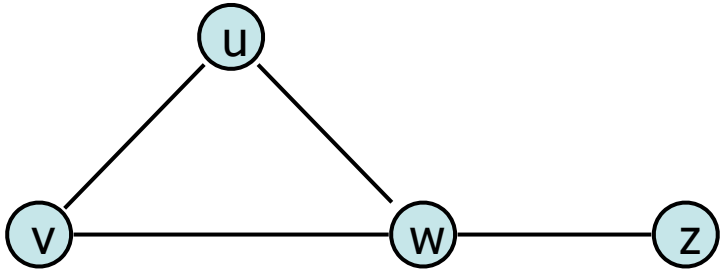



**insertEdge(u, v):**

| U | 0 |
|---|---|
| V | 1 |
| W | 2 |
| Z | 3 |

| | u | v | w | z |
|---|---|---|---|---|
| u | 0 | 1 | 1 | 0 |
| v | 1 | 0 | 1 | 0 |
| w | 1 | 1 | 0 | 1 |
| z | 0 | 0 | 1 | 0 |

# Graph Implementation: Adjacency Matrix



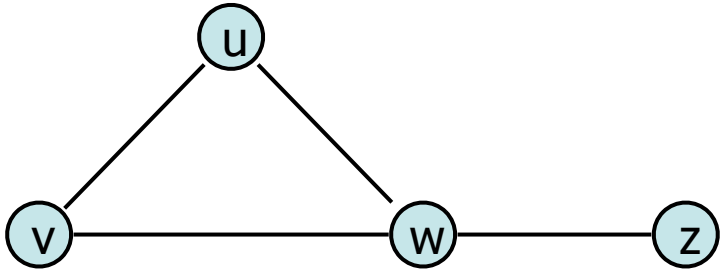

| U | 0 |
|---|---|
| V | 1 |
| W | 2 |
| Z | 3 |

| | u | v | w | z |
|---|---|---|---|---|
| u | 0 | 1 | 1 | 0 |
| v | 1 | 0 | 1 | 0 |
| w | 1 | 1 | 0 | 1 |
| z | 0 | 0 | 1 | 0 |

**removeVertex(v):**

**removeEdge(u, v):**

# Graph Implementation: Adjacency Matrix




| U | 0 |
|---|---|
| V | 1 |
| W | 2 |
| Z | 3 |

| | u | v | w | z |
|---|---|---|---|---|
| u | 0 | 1 | 1 | 0 |
| v | 1 | 0 | 1 | 0 |
| w | 1 | 1 | 0 | 1 |
| z | 0 | 0 | 1 | 0 |

**Pros:**

**Cons:**

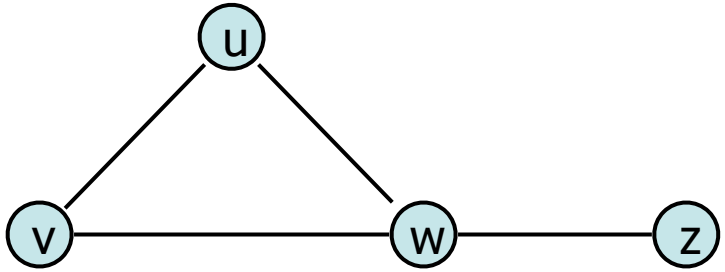
# Graph Implementations

We want something…

Faster than an edge list

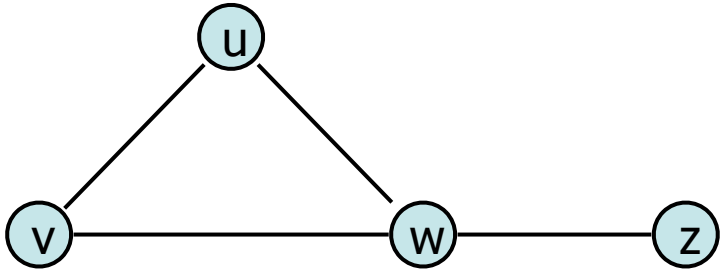
Less space than an adjacency matrix

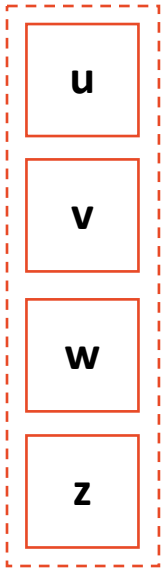
Particularly good at finding adjacent elements

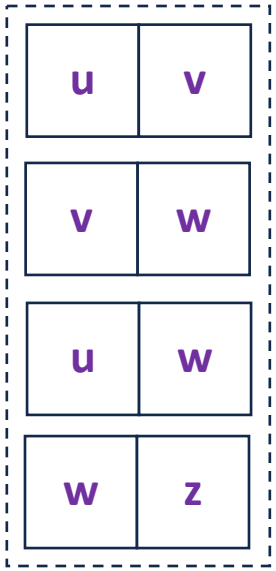


# Graph Implementation: Edge List + ?

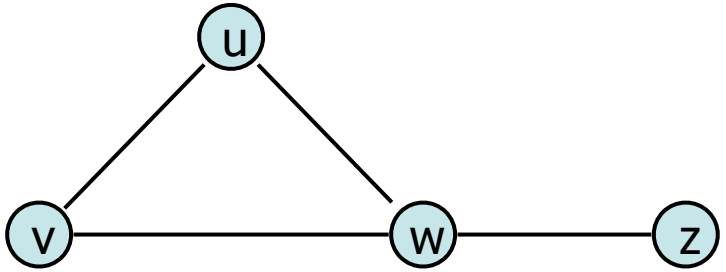
**|V|= n, |E|= m**



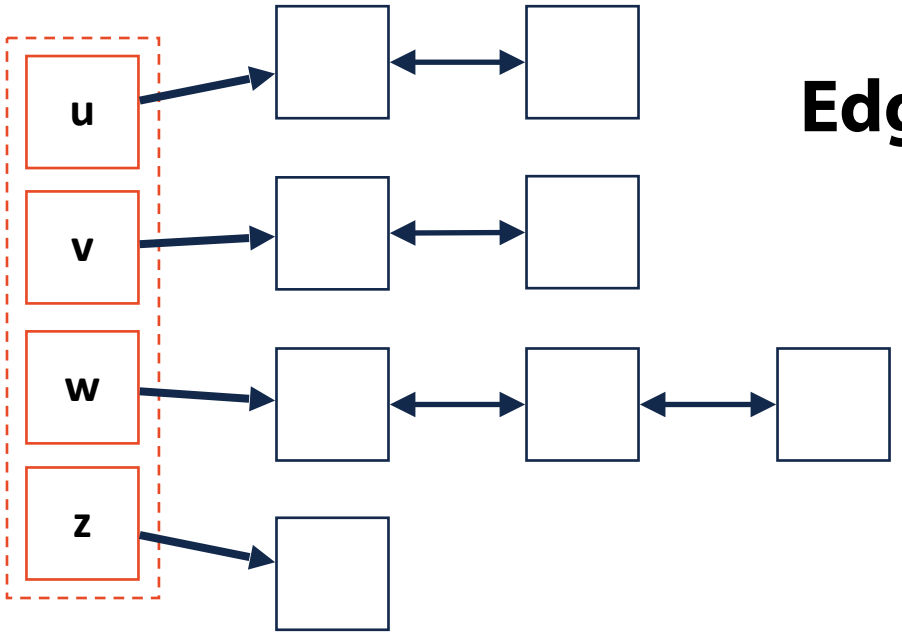



# Adjacency List

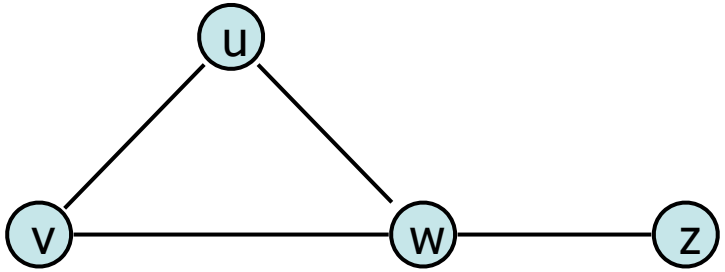



**Vertex Storage:**

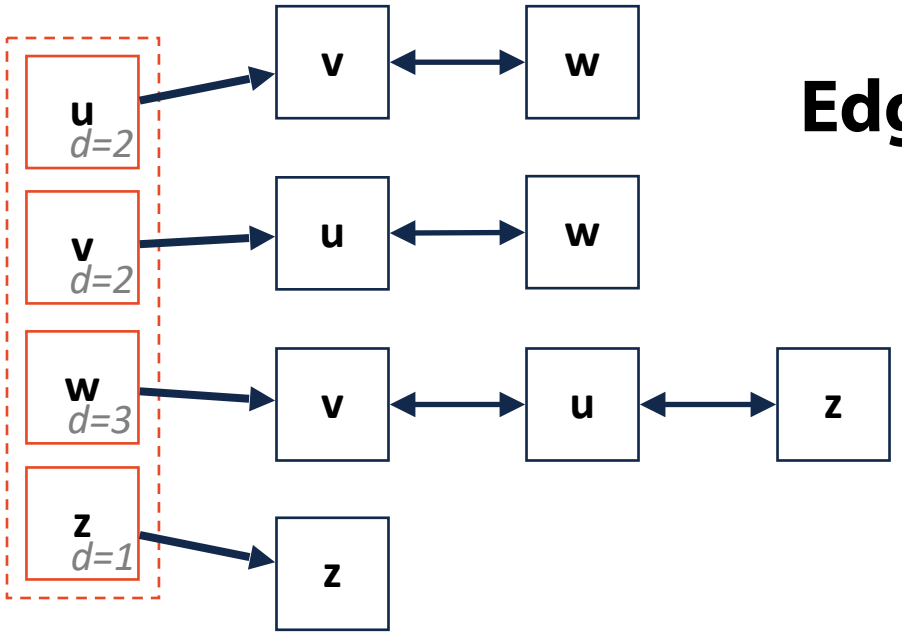



**Edge Storage:**

# Adjacency List

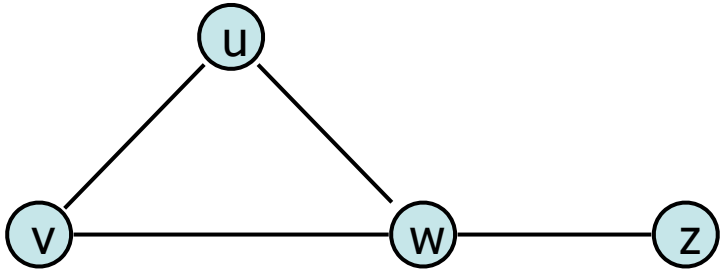



**Vertex Storage:**

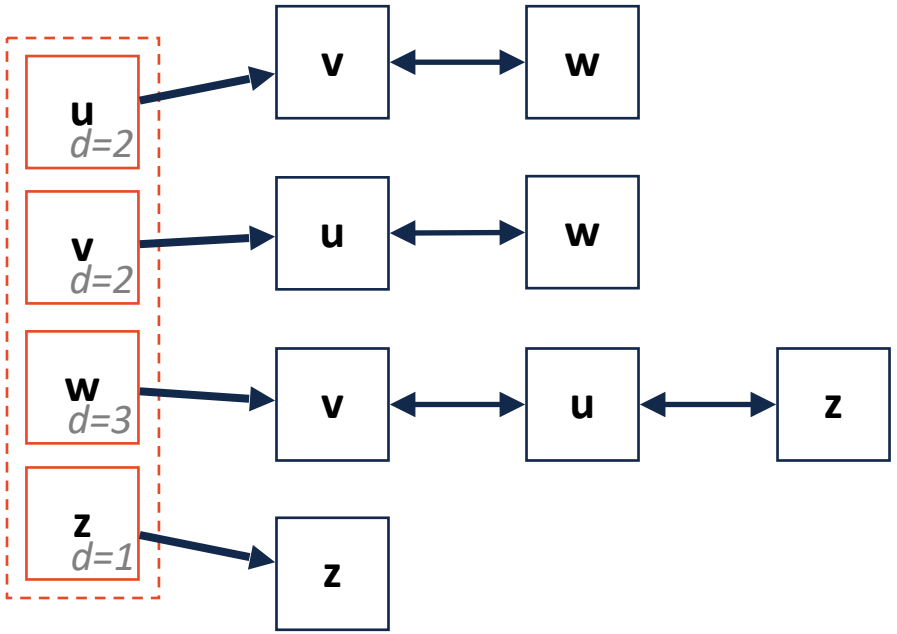



**Edge Storage:**

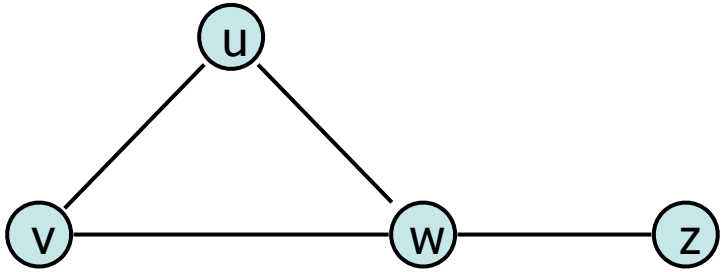
# Adjacency List



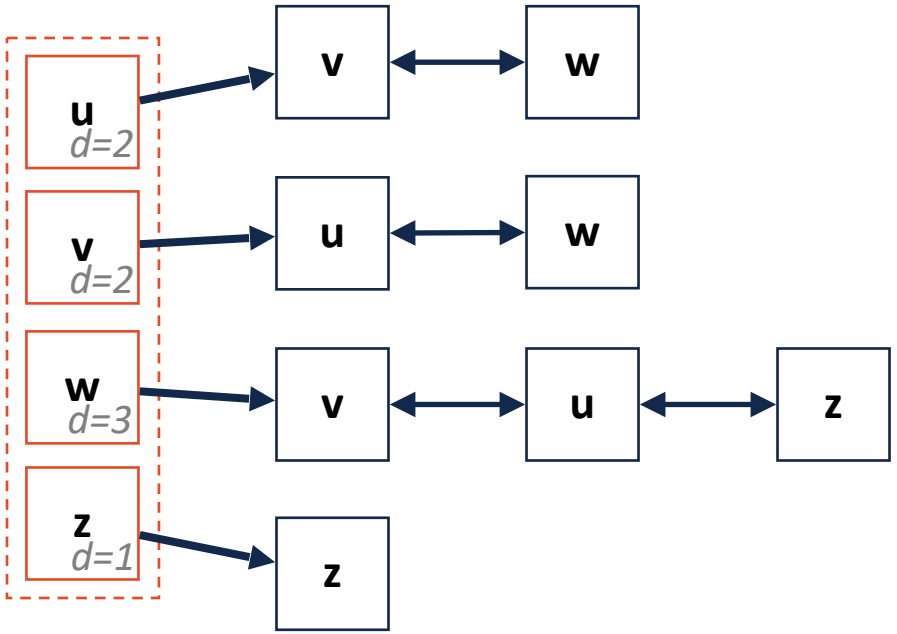

**getVertices():**



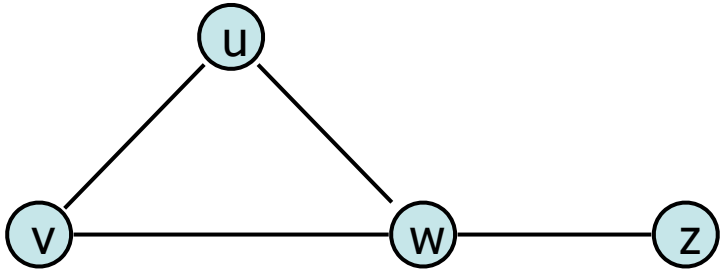
# Adjacency List



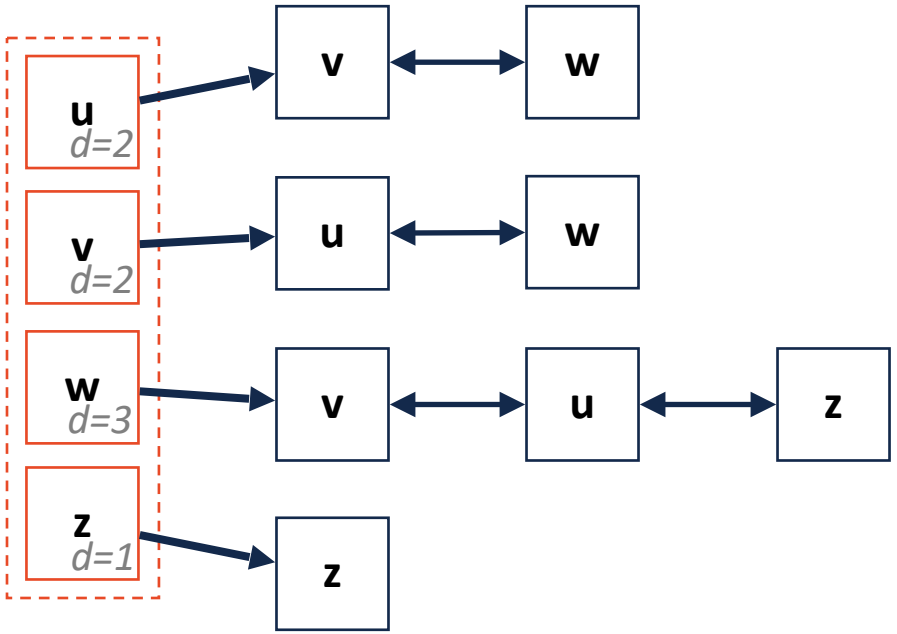

**getEdges(v):**



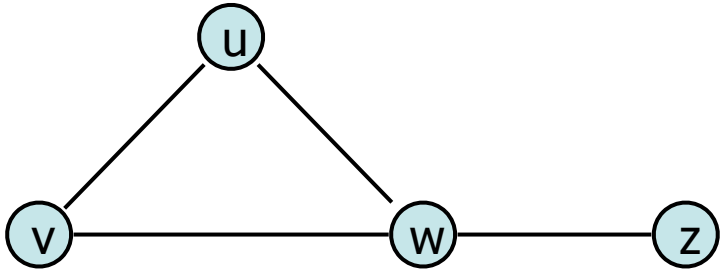
# Adjacency List



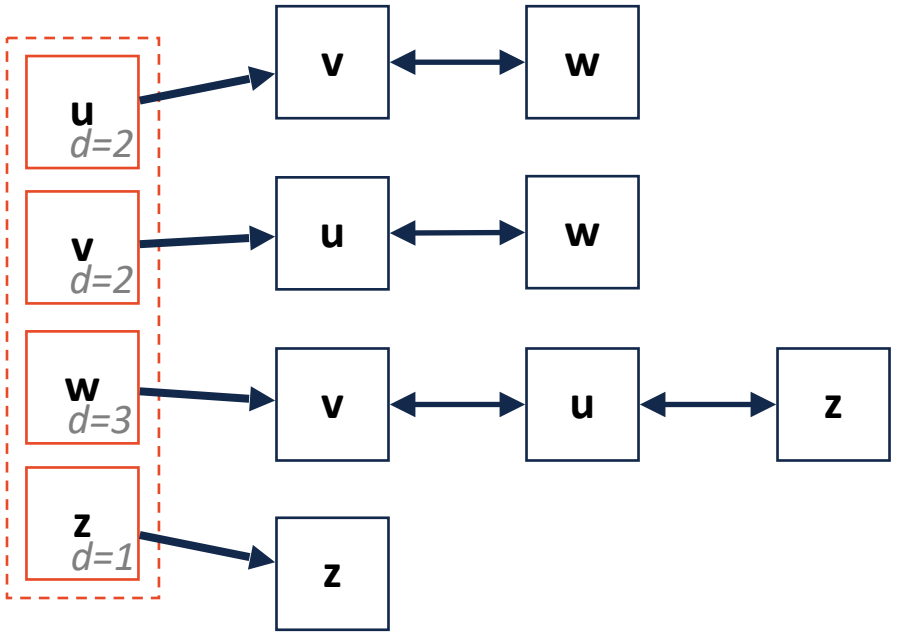

**areAdjacent(u, v):**



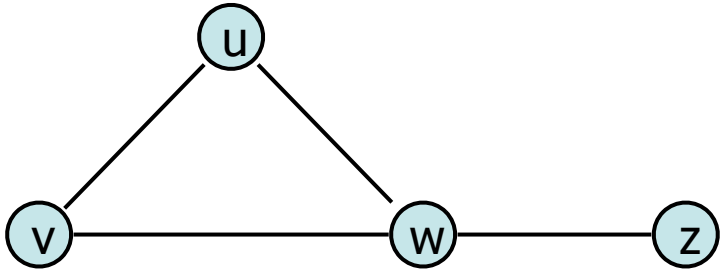
# Adjacency List



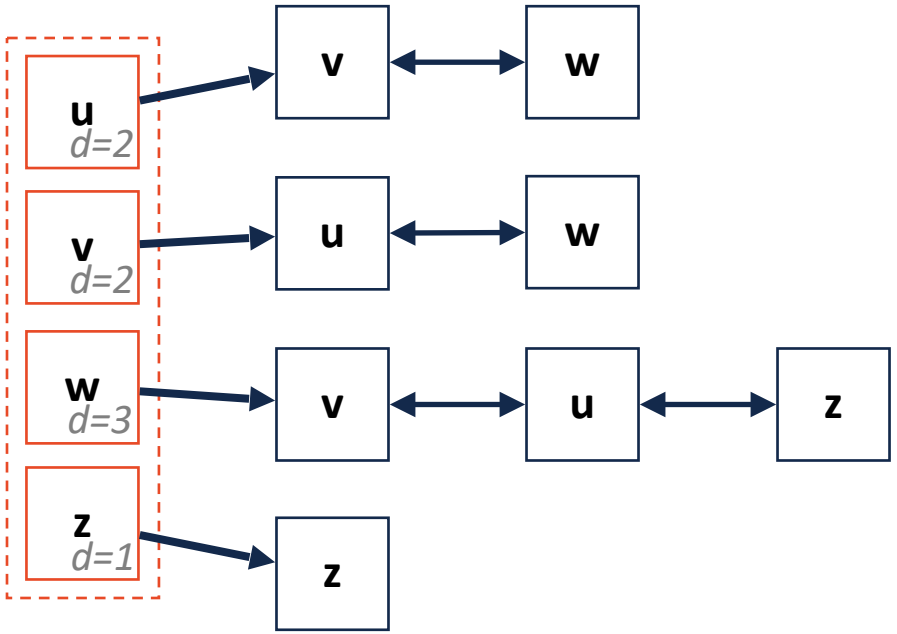

**insertVertex(v):**



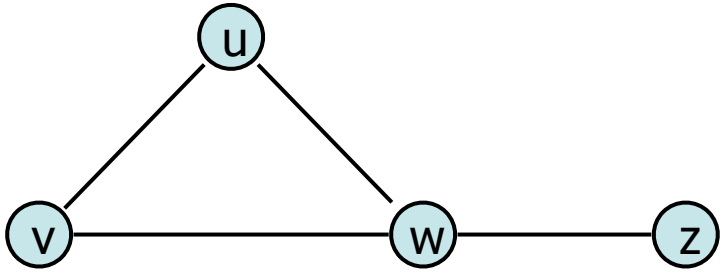
# Adjacency List



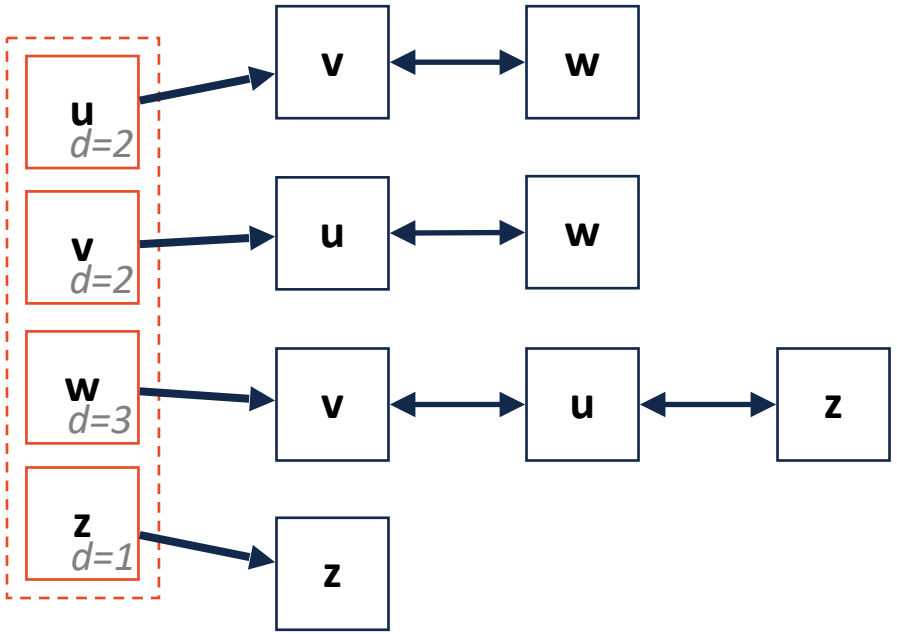

**removeVertex(v):**



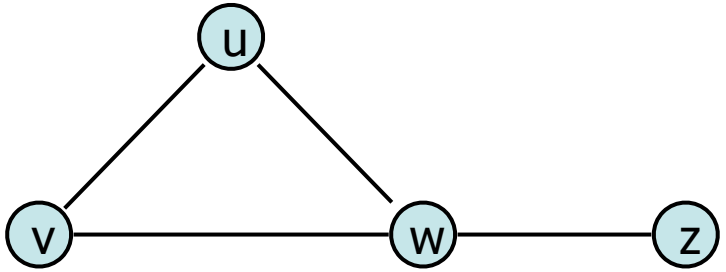
# Adjacency List



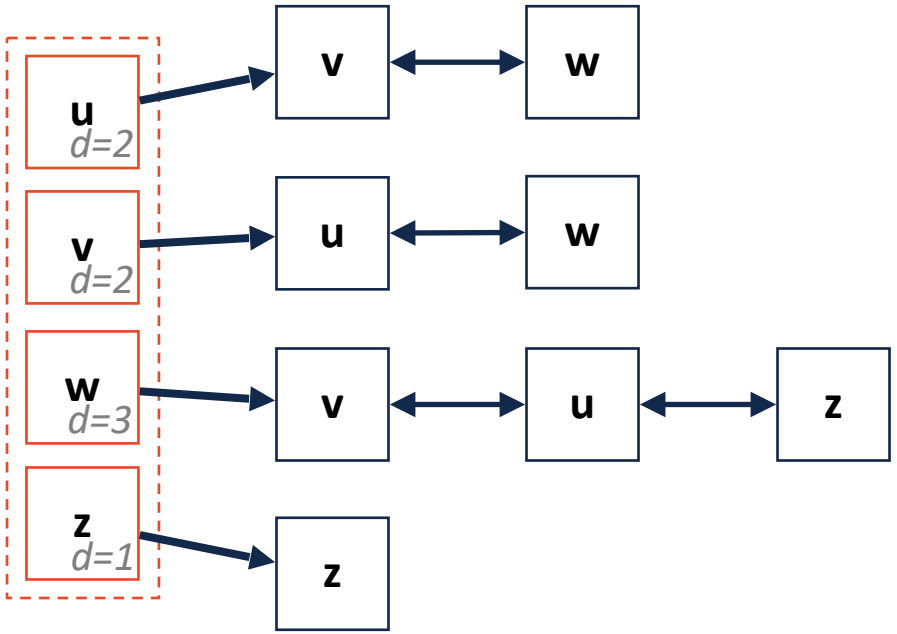

**insertEdge(u, v):**



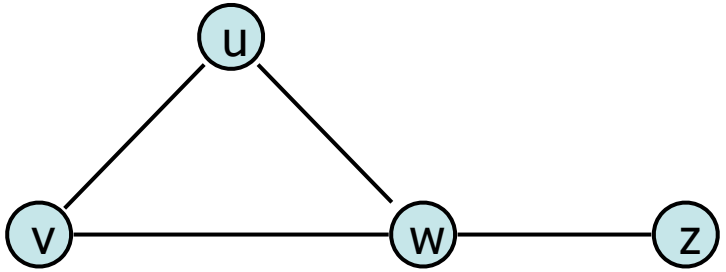
# Adjacency List

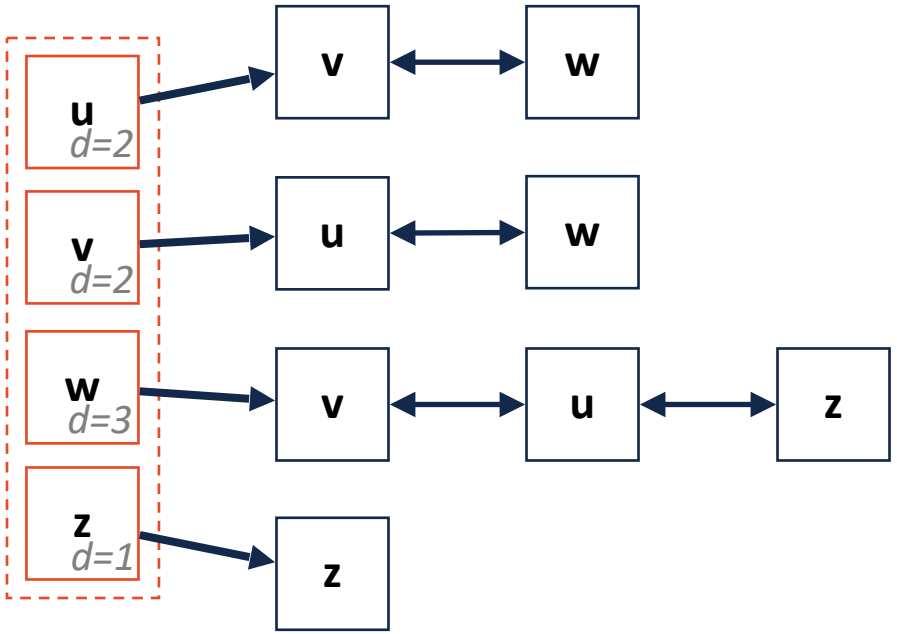



**removeEdge(u, v):**



# Adjacency List




**Pros:**

**Cons:**

$|V| = n, |E| = m$

| Expressed as O(f) | Edge List | Adjacency Matrix | Adjacency List |
|---|---|---|---|
| Space | | | |
| insertVertex(v) | | | |
| removeVertex(v) | | | |
| insertEdge(u, v) | | | |
| removeEdge(u, v) | | | |
| getEdges(v) | | | |
| areAdjacent(u, v) | | | |