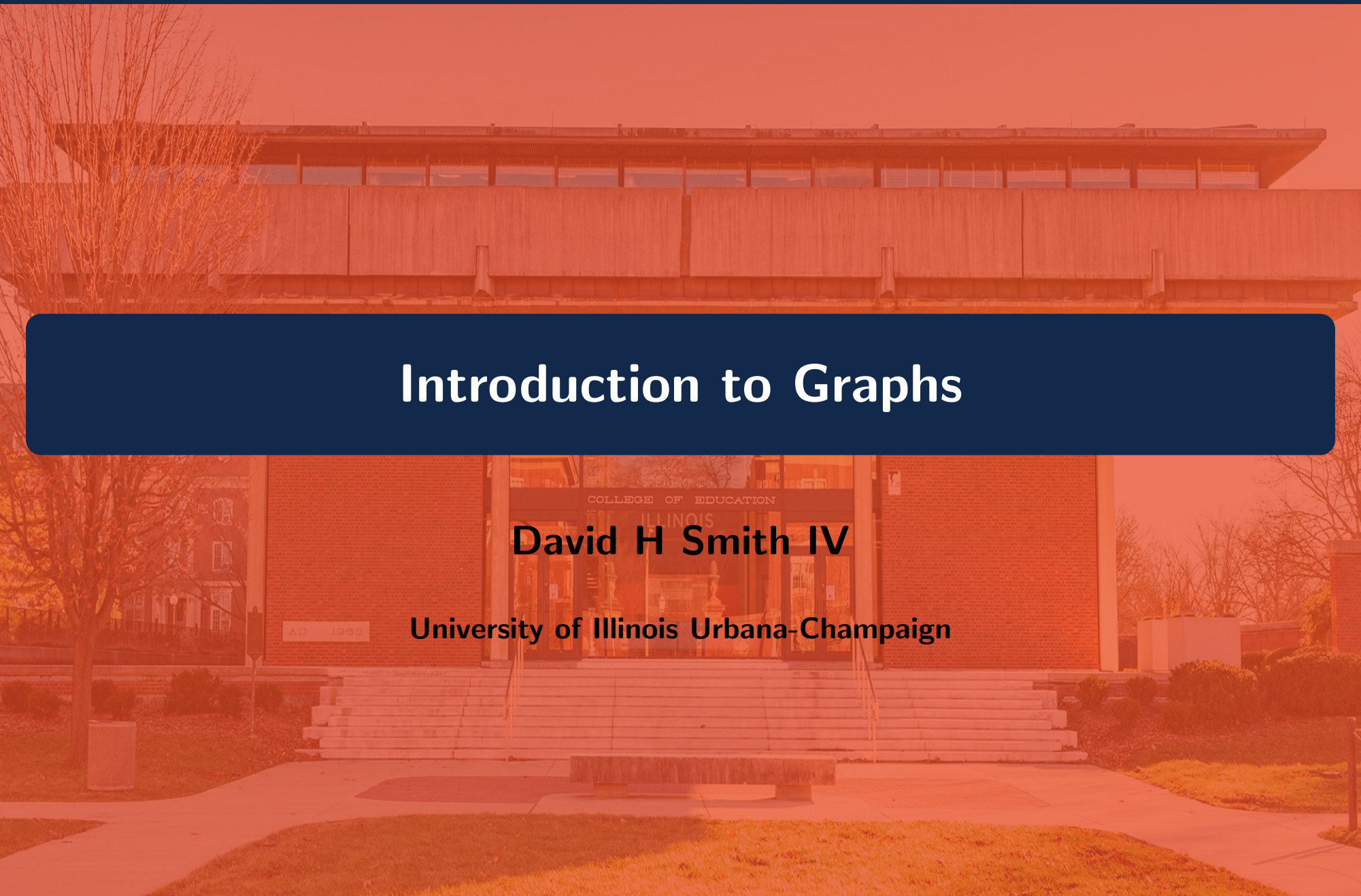


Introduction to Graphs

David H Smith IV

University of Illinois Urbana-Champaign

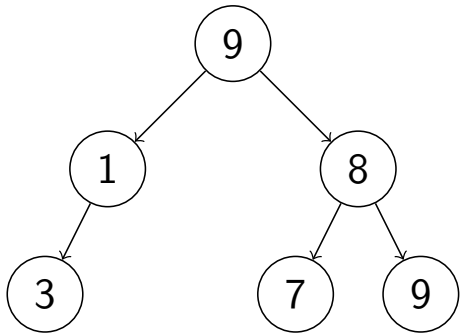
Objectives

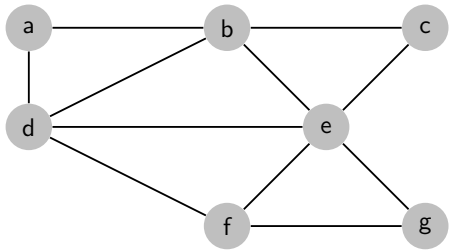
- Become familiar with graph terminology.
- How graphs are structured.
- How graphs are represented with adjacency matrices.
- Practice implementing some basic graph operations using adj matrices.

Objectives

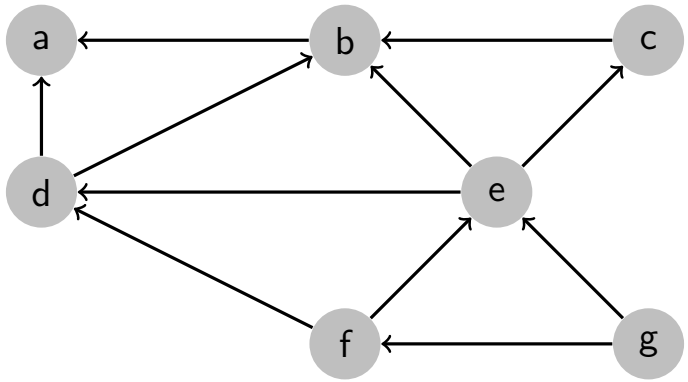
- **Week 10:** Introduction to graphs via Adj Matrices.
- **Week 11**: Introduction into Adj. Lists and Graphs Search Algorithms. Implementation 5 released.
- **Week 12**: Election day, checkpoint 1 due.
- **Week 13**: Shortest Path and Minimum Spanning Tree Algorithms. Implementation 5 due and 6 released.
- **Week 14**: Thanksgiving break.
- **Week 15**: Conclusion and Application 3 released.

LinkedLists →Trees →Graphs



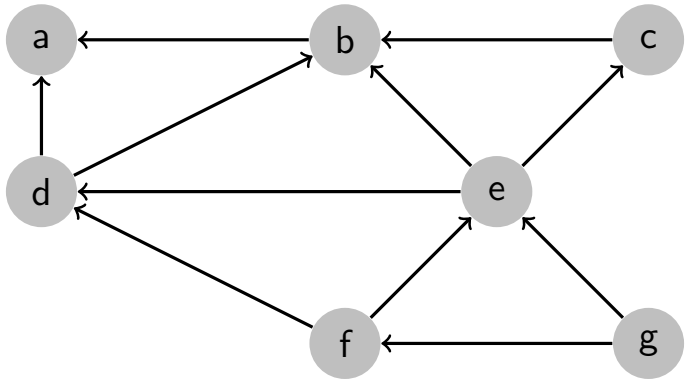


Terms



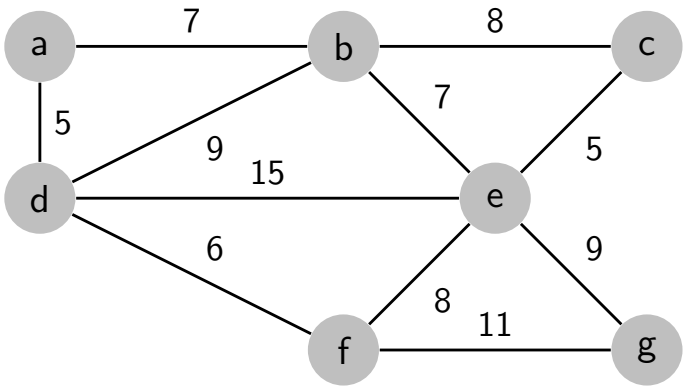
- **Vertex**: A "Node" in the graph.
- **Edge**: A connection between two vertecies.
- **Graph**: A set of edges (E) and a set of vertecies (V) and is often denoted as $G = (V, E)$.

Terms

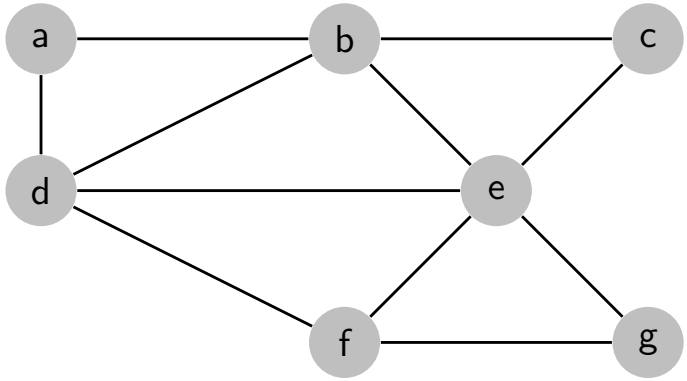


- **Directed v Undirected**: Can we move between vertecies in either direction or only in one? Examples of directed graphs include:
  - **Linked Lists**: Once you move forward you cant move back.
  - **Trees**: Once you proceed left you can *directly* move back up to the parent.
- **Weighted v Unweighted**: Does it cost us anything to move between vertecies?

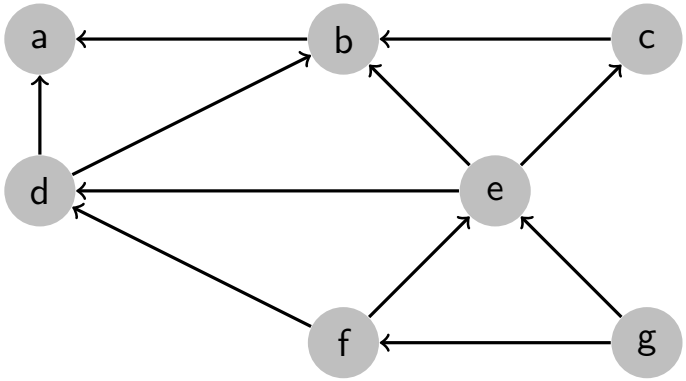
Directed vs Undirected and Weighted vs Unweighted



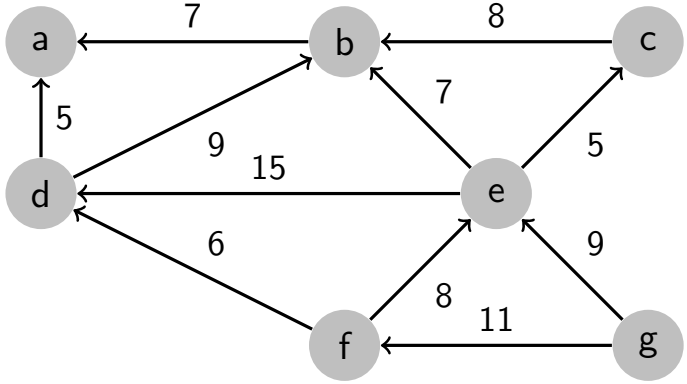
Undirected, Weighted



Undirected, Unweighted

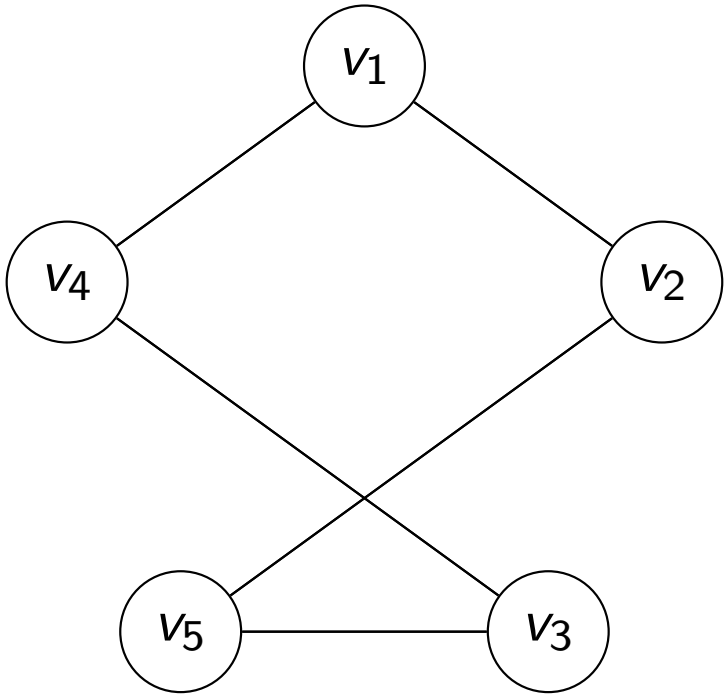


Directed, Unweighted



Directed, Weighted

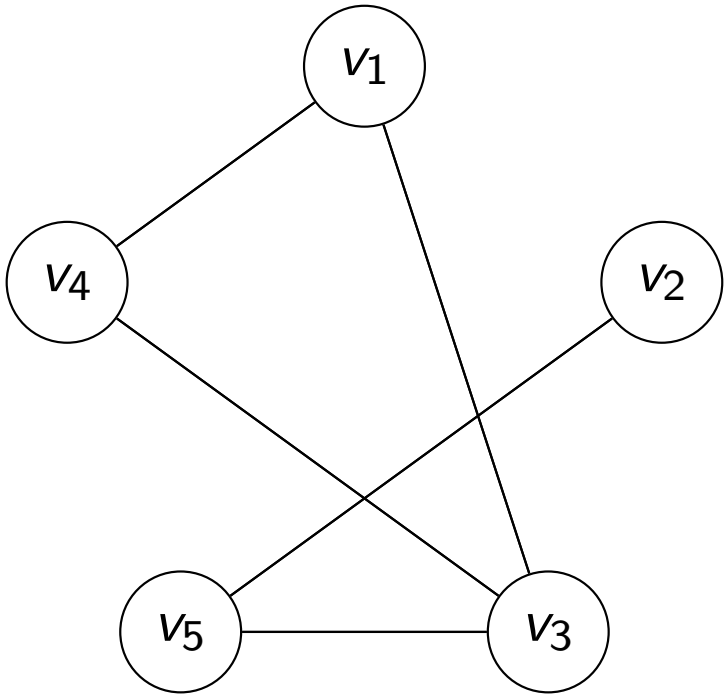
Adjacency Matrix vs Adjacency List



	1	**2**	**3**	**4**	**5**
1	0	1	0	1	0
2	1	0	0	0	1
3	0	0	0	1	1
4	1	0	1	0	0
5	0	1	1	0	0

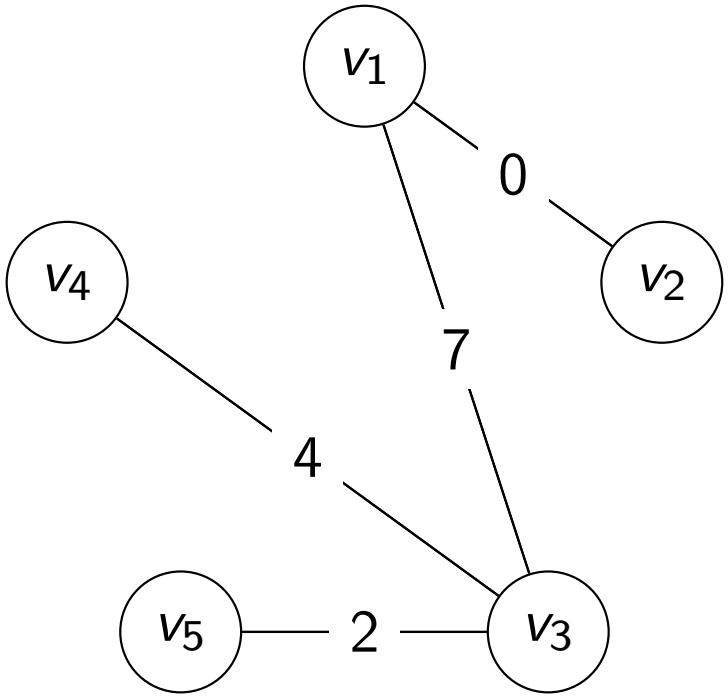
1. source vertex is the row.
2. target vertex is the column.
3. 1 for edge exist and 0 for it doesn't.

Constructing Adj. Matrix - Unweighted, Undirected



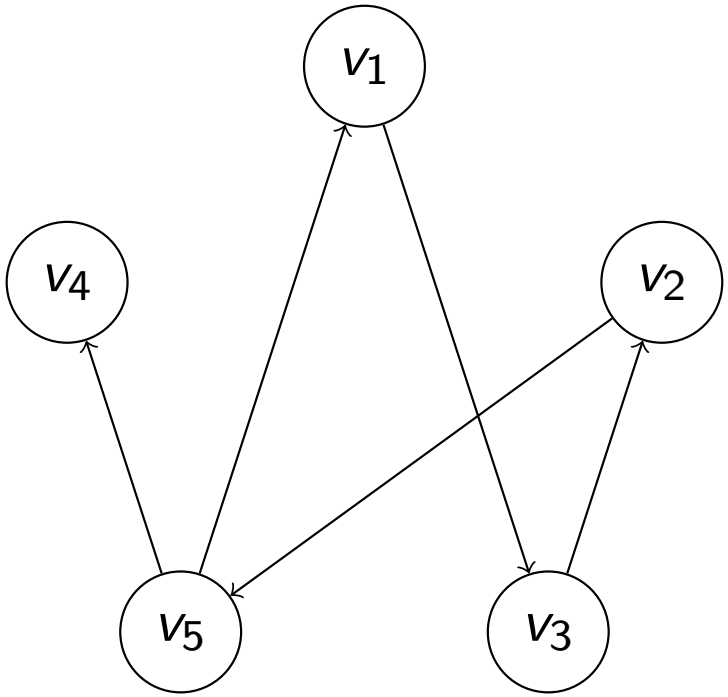
	1	2	3	4	5
1					
2					
3					
4					
5					

Constructing Adj. Matrix - Weighted, Undirected



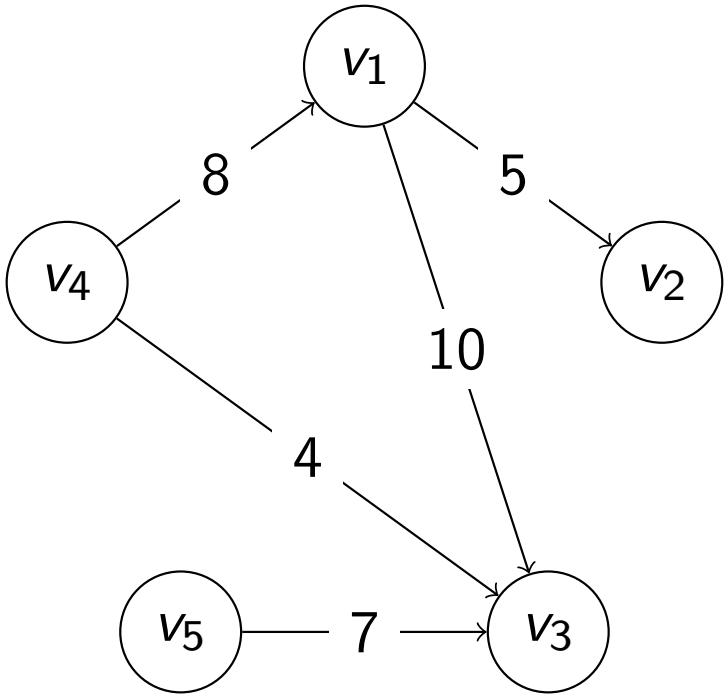
	1	2	3	4	5
1					
2					
3					
4					
5					

Constructing Adj. Matrix - Unweighted, Directed



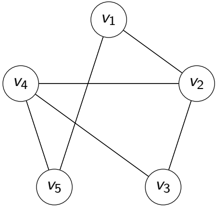
	1	2	3	4	5
1					
2					
3					
4					
5					

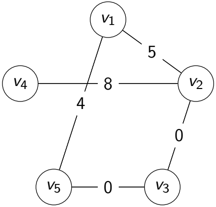
Constructing Adj. Matrix - Weighted, Undirected

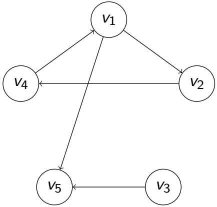


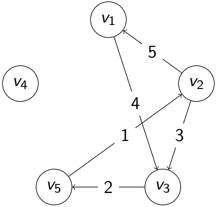
	1	2	3	4	5
1					
2					
3					
4					
5					

Adj Matrix - Practice









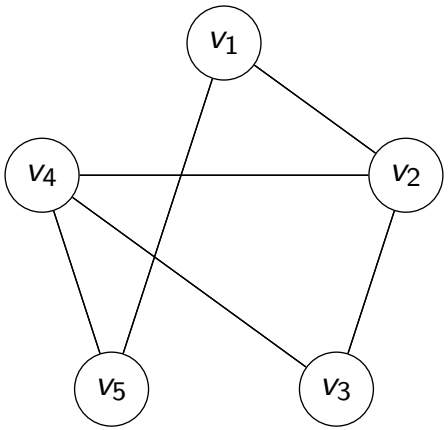
	1	2	3	4	5
1					
2					
3					
4					
5					

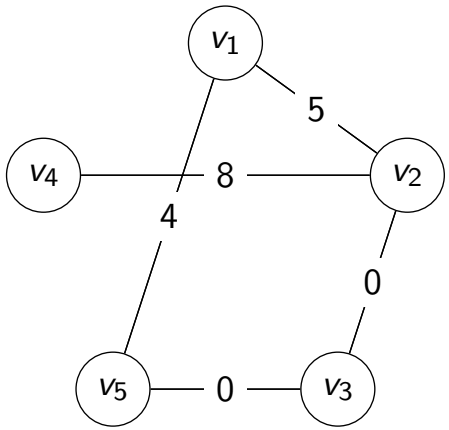
	1	2	3	4	5
1					
2					
3					
4					
5					

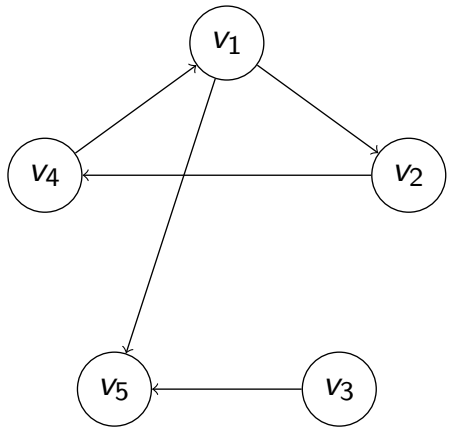
	1	2	3	4	5
1					
2					
3					
4					
5					

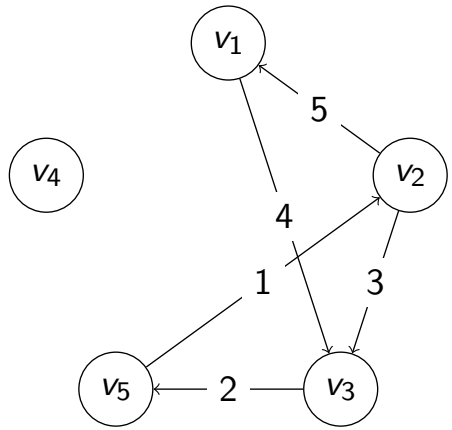
	1	2	3	4	5
1					
2					
3					
4					
5					

Adj Matrix - Practice









	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	0
3	0	1	0	1	0
4	0	1	1	0	1
5	1	0	0	1	0

	1	2	3	4	5
1		5			4
2	5		0	8	
3		0			0
4		8			
5	4		0		

	1	2	3	4	5
1	0	1	0	0	1
2	0	0	0	1	0
3	0	0	0	0	1
4	1	0	0	0	0
5	0	0	0	0	0

	1	2	3	4	5
1			4		
2	5		3		
3					2
4					
5		1			

Adj Matrix - Vertex Count and Edge List to Adj Matrix

```
public void printAdjMatrix(List<Integer[]> edges, int vertexCount){
    //Step 1: Create a NxN matrix
    int[][] matrix = ??

    //Step 2: Populate the matrix using vertex list
    for(Integer[] pair: edges){

    }
    //Step 3: Print the matrix
}
```

1. **Input:** Vertex count and list of edges
2. Steps:
   - Step 1: Initialize the 2d Array to be an NxN matrix where N is the vertex count.
   - Step 2: Iterate over the list of edges (source, target) and set
     - Recall that target is the row and source is the column
   - Step 3: Iterate over the matrix in order to display its contents.

Adj Matrix - Class

```
class MatrixGraph {
    private Integer[][] graph;
    private boolean isDirected;

    // Step 1: Initialize an NxN graph with all 0s
    MatrixGraph(int numVertecies, boolean isDirected){ ... }

    //Step 2: Construct an AddEdge Method
    public void addEdge(int v1, int v2, int edgeWeight){ ... }
}
```

1. **Step 1 - Constructor:** The constructor should take a vertexCount and an isDirected. It should initialize the graph to be an NxN array and populate it
2. **Step 2 - Add Edge:** This method should add populate the cell are the row/column as specified by the source/target. If isDirected is false, it should also add the edge going in the other direction.