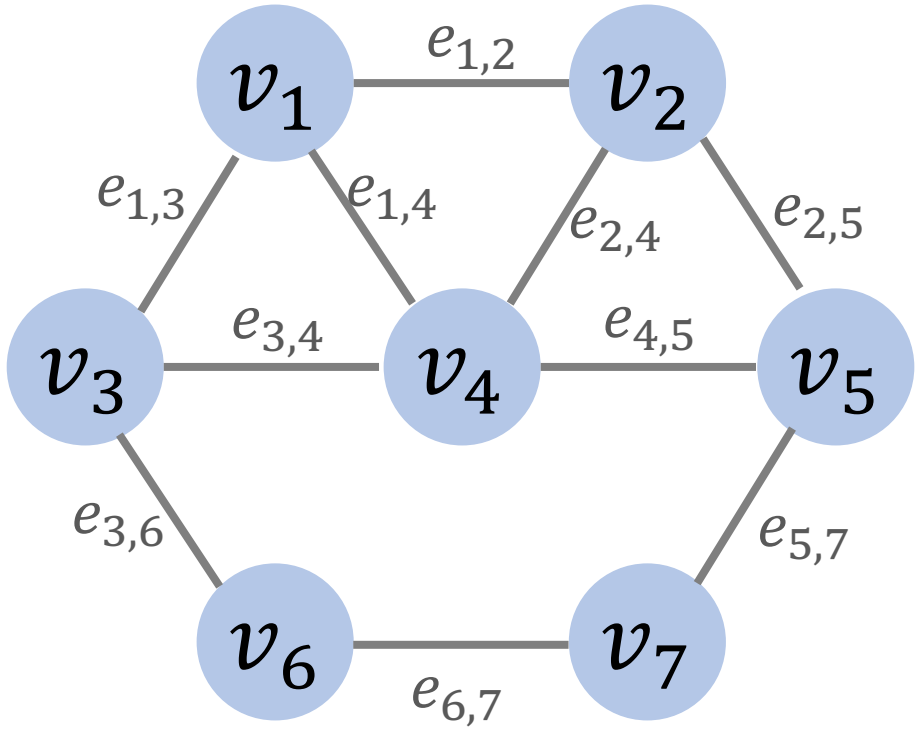
Graphs

Shusen Wang

What is graph?



Definitions

- Set of vertices:

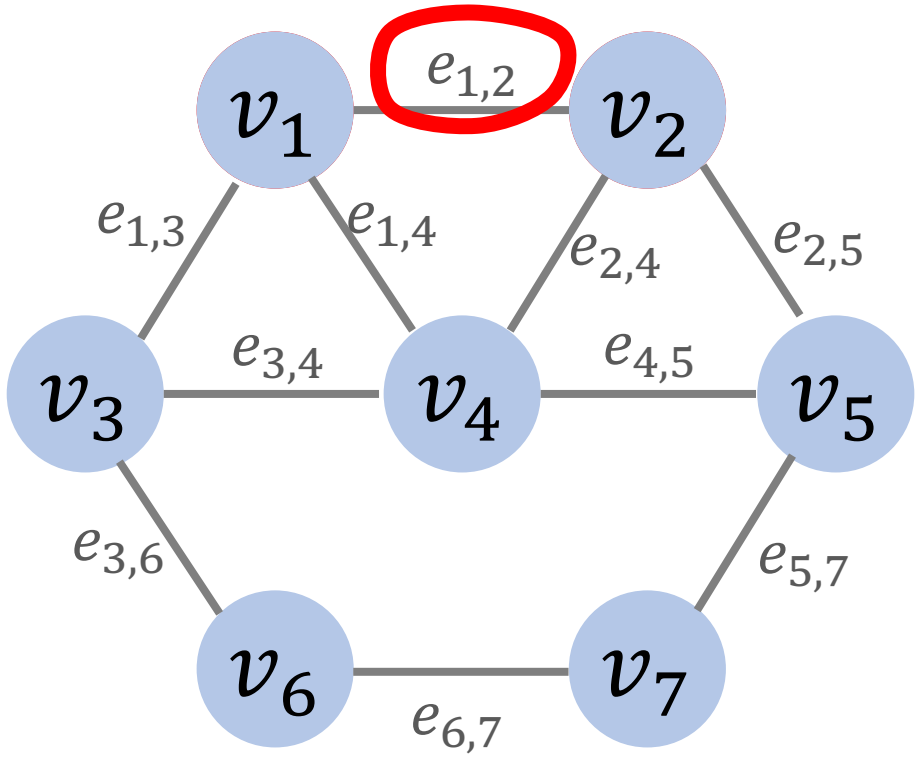
$$\mathcal{V} = \{v_1, v_2, \cdots, v_7\}.$$

- Set of edges:

$$\mathcal{E} = \{e_{1,2}, e_{1,3}, e_{1,4}, e_{2,4}, \cdots, e_{6,7}\}.$$

- Graph: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$.

What is graph?



Definitions

- Set of vertices:

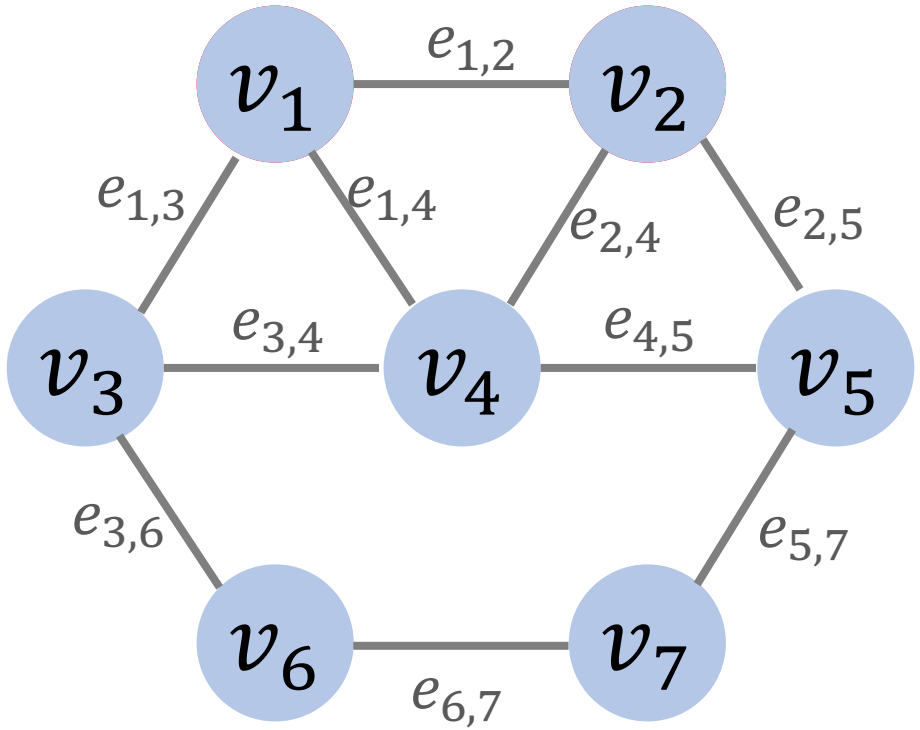
$$\mathcal{V} = \{v_1, v_2, \cdots, v_7\}.$$

- Set of edges:

$$\mathcal{E} = \{e_{1,2}, e_{1,3}, e_{1,4}, e_{2,4}, \cdots, e_{6,7}\}.$$

- Graph: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$.

What is graph?



Definitions

- Set of vertices:

$$\mathcal{V} = \{v_1, v_2, \cdots, v_7\}.$$

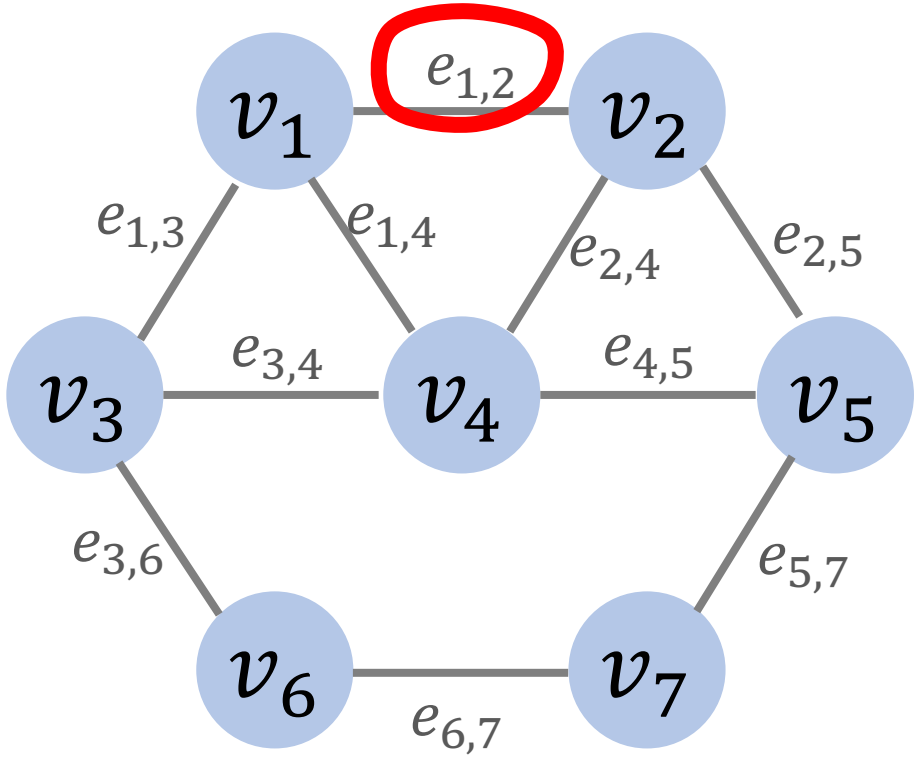
- Set of edges:

$$\mathcal{E} = \{e_{1,2}, e_{1,3}, e_{1,4}, e_{2,4}, \cdots, e_{6,7}\}.$$

- Graph: $\mathcal{G} = (\mathcal{V}, \mathcal{E})$.

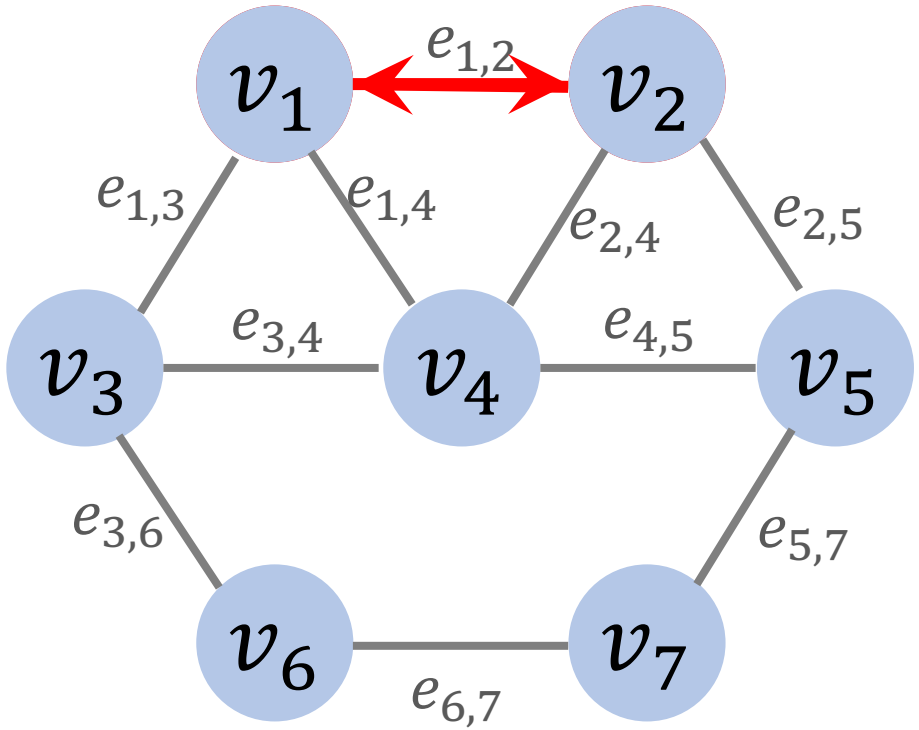
Undirected vs Directed

Undirected Graph



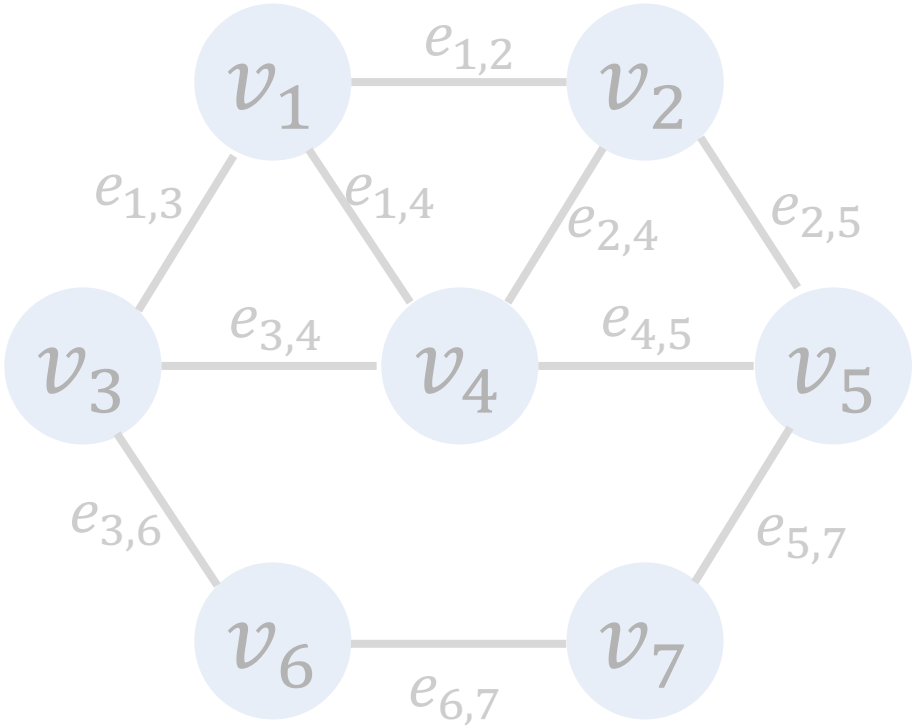
Undirected vs Directed

Undirected Graph

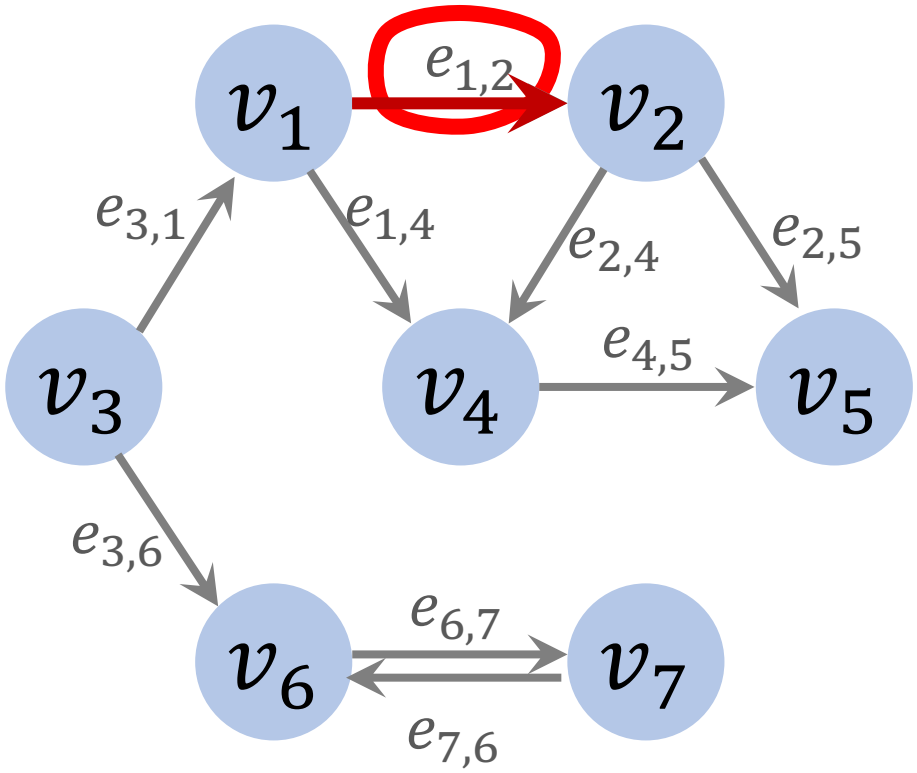


Undirected vs Directed

Undirected Graph

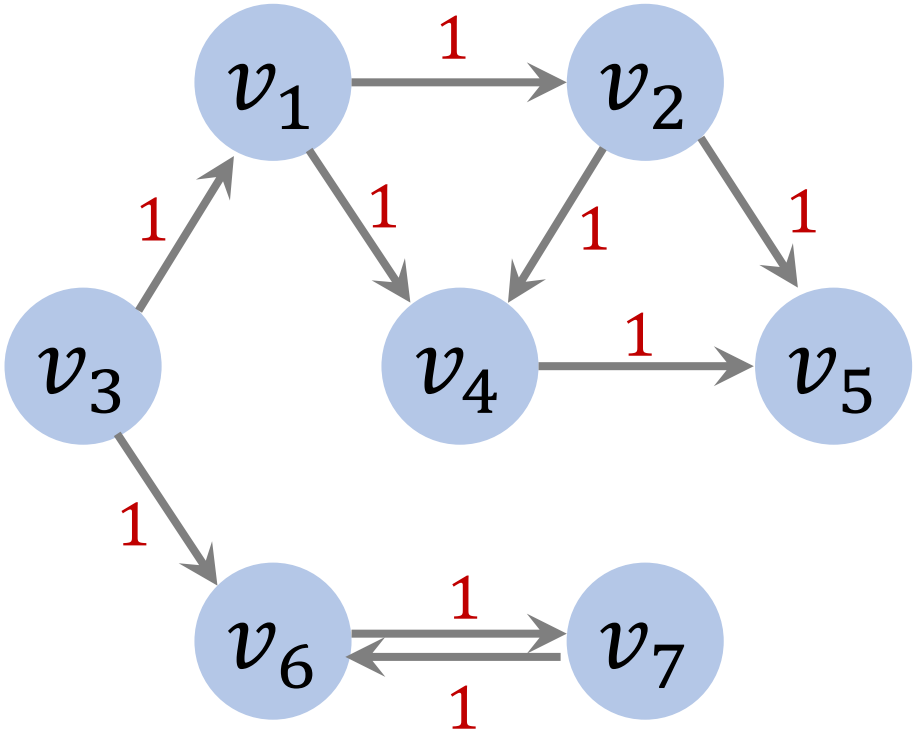


Directed Graph



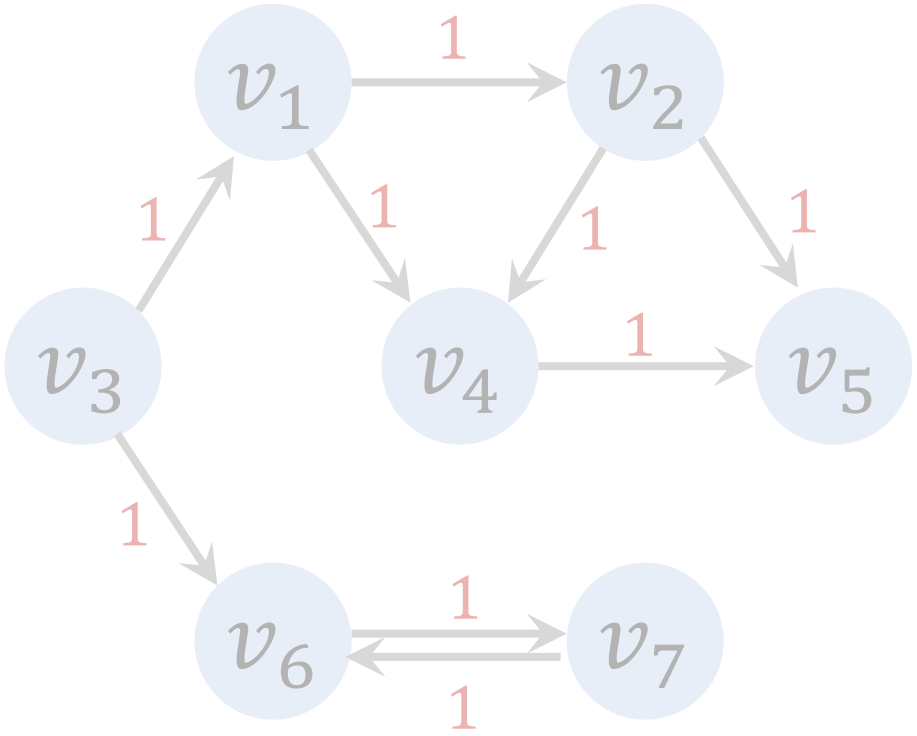
Unweighted vs Weighted

Unweighted Graph

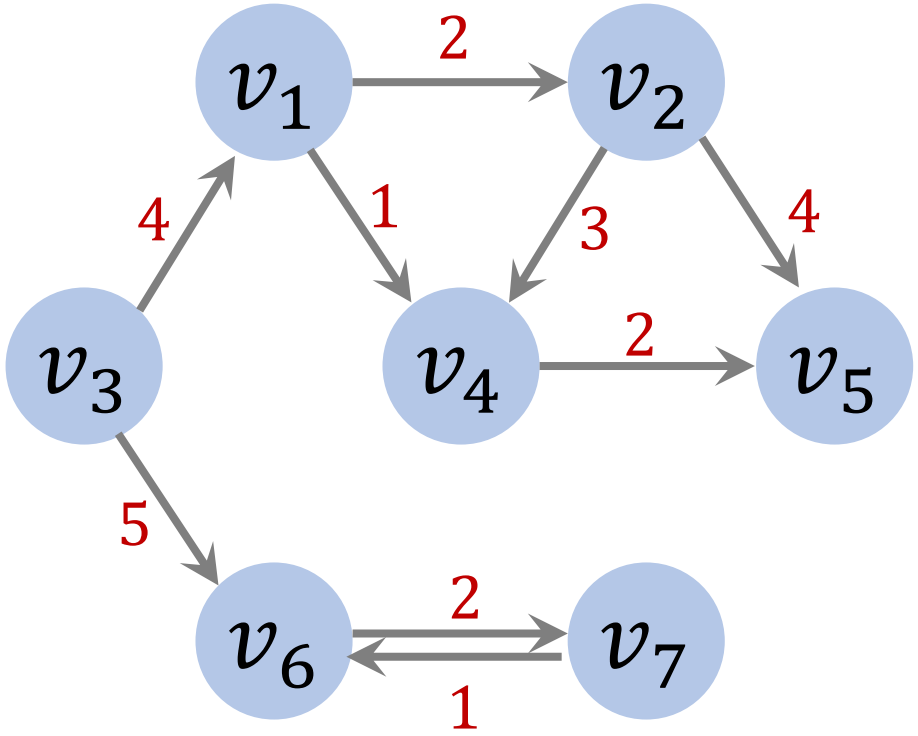


Unweighted vs Weighted

Unweighted Graph



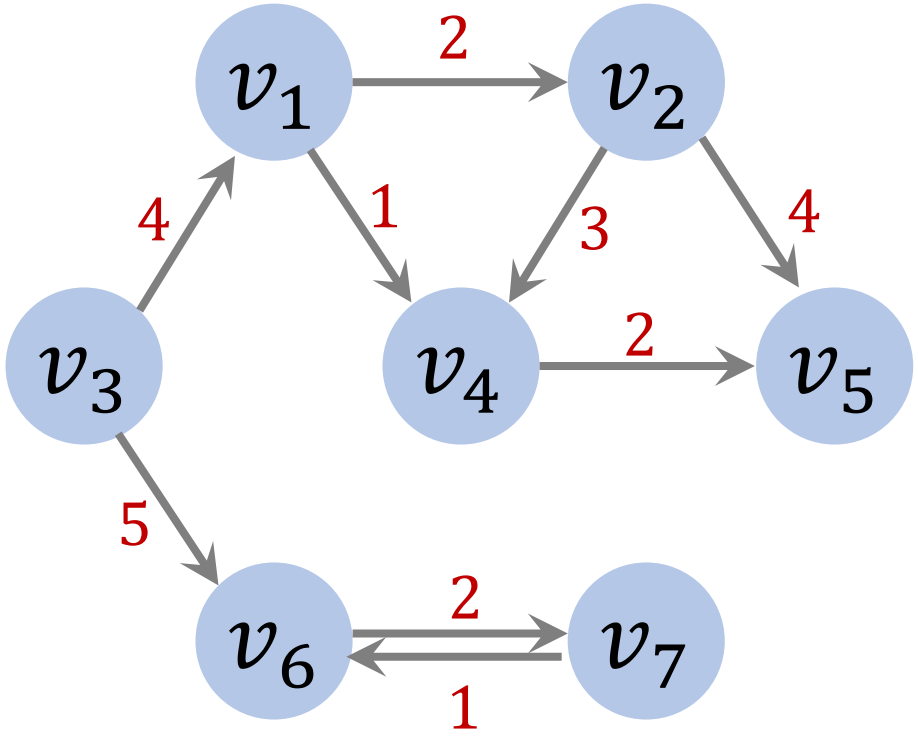
Weighted Graph



Physical Meanings of Weights

- **Examples 1:**
  - Weights are distances between towns.
  - No edge means not road.
  - Weight of nonexistent edge is infinity.

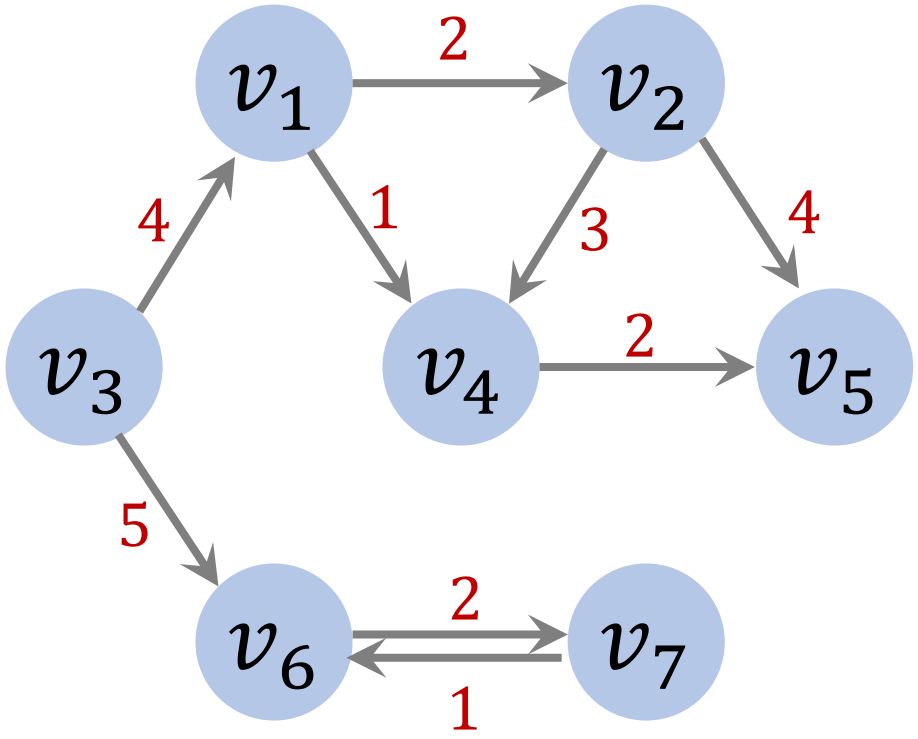
Weighted Graph



Physical Meanings of Weights

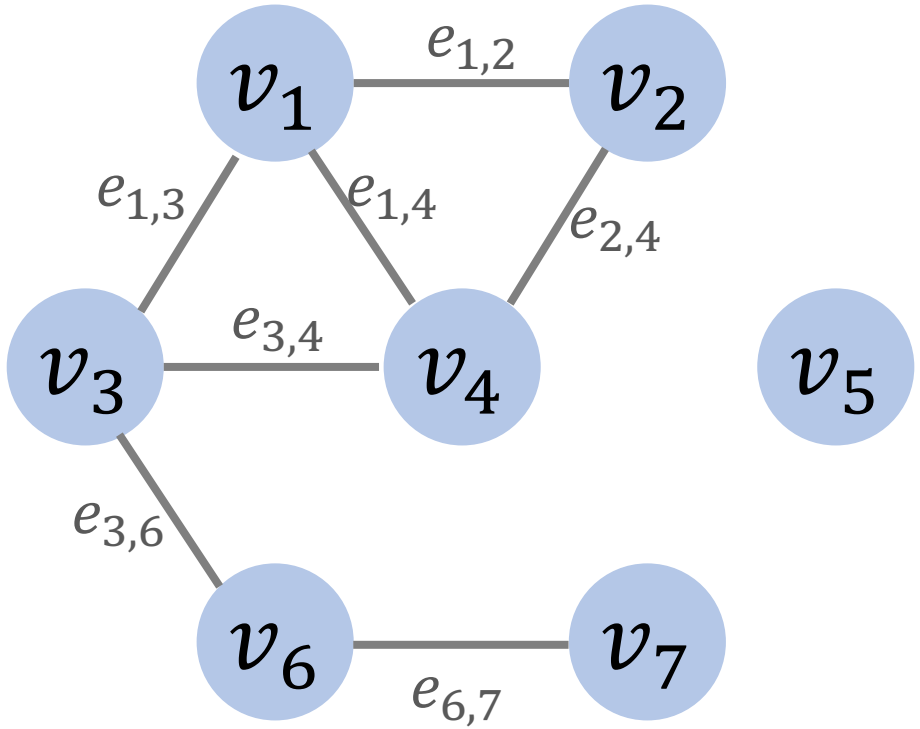
- **Examples 1:**
  - Weights are distances between towns.
  - No edge means not road.
  - Weight of nonexistent edge is infinity.
- **Example 2:**
  - Weights are widths (capacity) of roads.
  - No edge means not road.
  - Weight of nonexistent edge is zero.

Weighted Graph

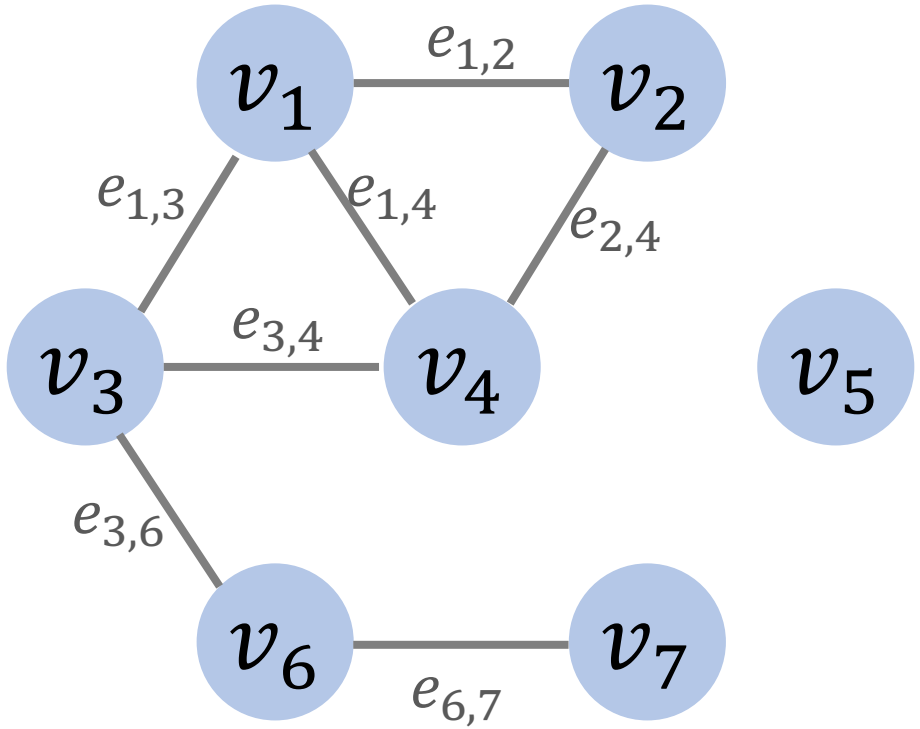


Undirected Unweighted Graphs

Undirected Unweighted Graph



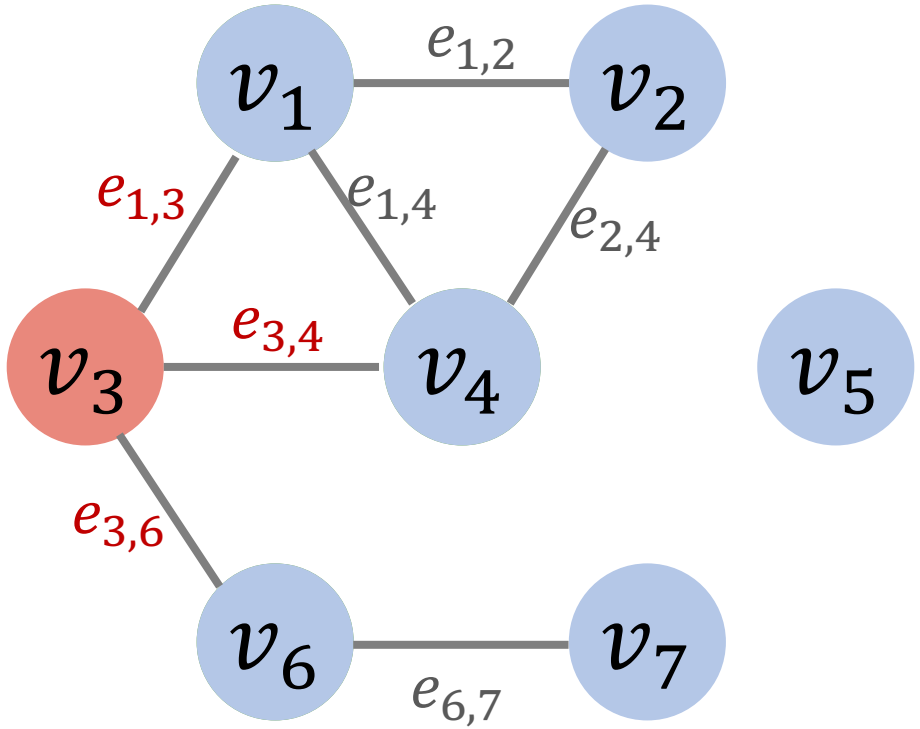
Undirected Unweighted Graph



Adjacency list:

Vertex	Neighbors
1	2, 3, 4
2	1, 4
3	1, 4, 6
4	1, 2, 3
5	empty
6	3, 7
7	6

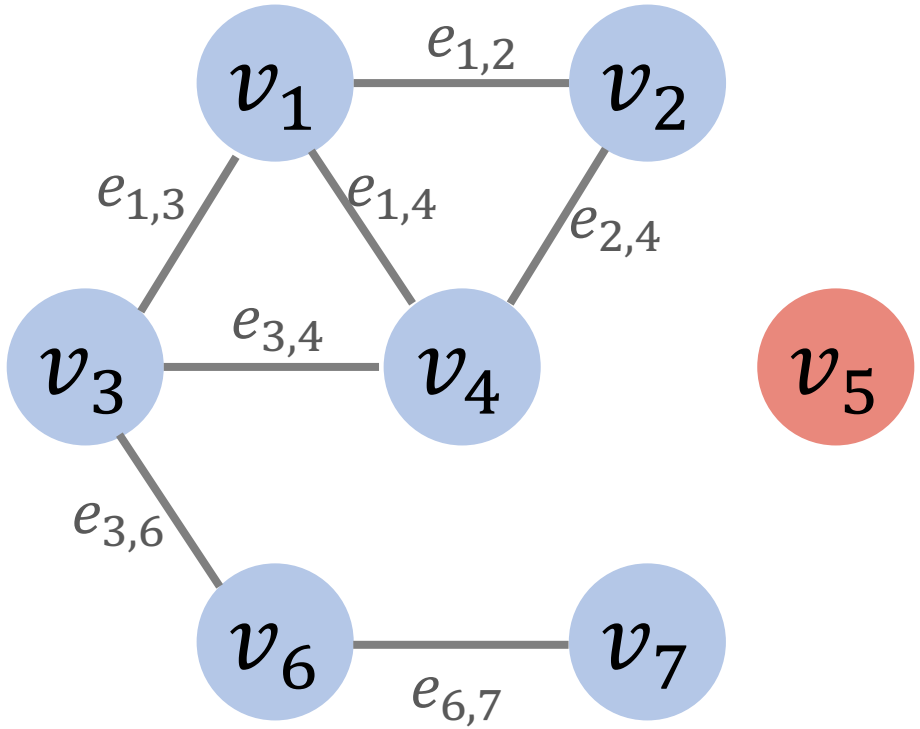
Undirected Unweighted Graph



Adjacency list:

Vertex	Neighbors
1	2, 3, 4
2	1, 4
3	**1, 4, 6**
4	1, 2, 3
5	empty
6	3, 7
7	6

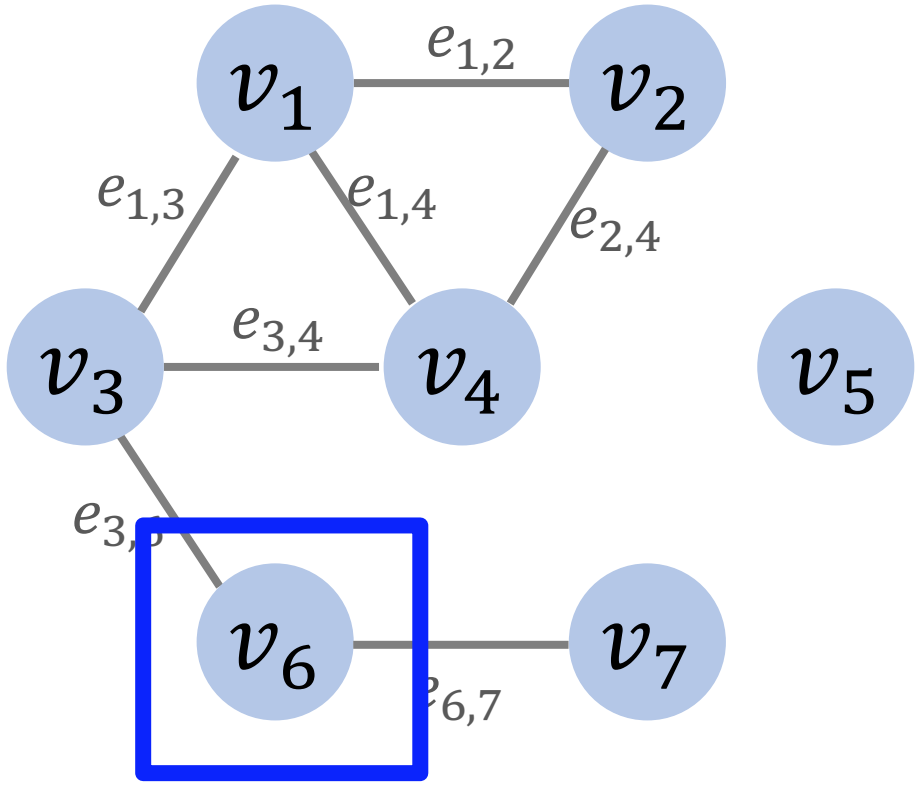
Undirected Unweighted Graph



Adjacency list:

Vertex	Neighbors
1	2, 3, 4
2	1, 4
3	1, 4, 6
4	1, 2, 3
5	**empty**
6	3, 7
7	6

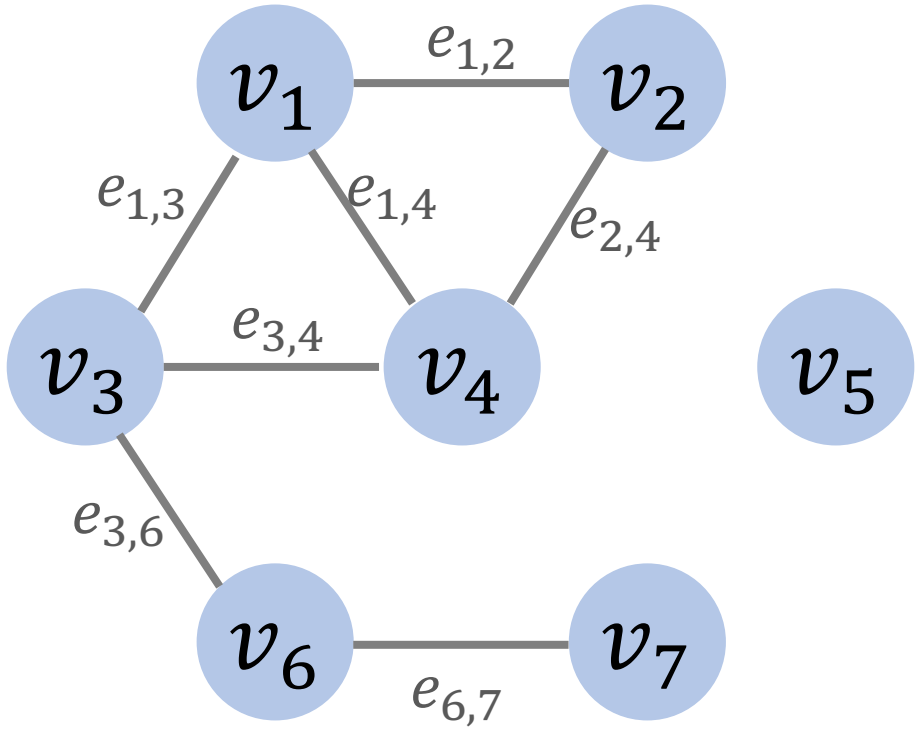
Undirected Unweighted Graph



Adjacency matrix:

	v_1	v_2	v_3	v_4	v_5	v_6	v_7
v_1	0	1	1	1	0	0	0
v_2	1	0	0	1	0	0	0
v_3	1	0	0	1	0	1	0
v_4	1	1	1	0	0	0	0
v_5	0	0	0	0	0	0	0
v_6	0	0	1	0	0	0	1
v_7	0	0	0	0	0	1	0

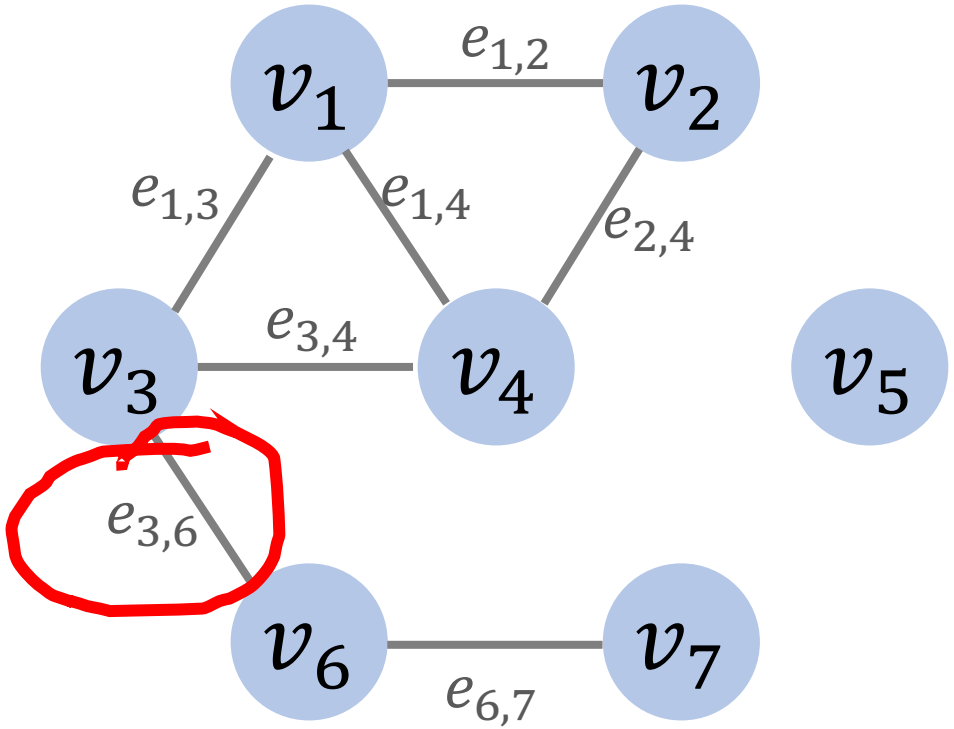
Undirected Unweighted Graph



Adjacency matrix:

	v_1	v_2	v_3	v_4	v_5	v_6	v_7
v_1	0	**1**	**1**	**1**	0	0	0
v_2	**1**	0	0	**1**	0	0	0
v_3	**1**	0	0	**1**	0	**1**	0
v_4	**1**	**1**	**1**	0	0	0	0
v_5	0	0	0	0	0	0	0
v_6	0	0	**1**	0	0	0	**1**
v_7	0	0	0	0	0	**1**	0

Undirected Unweighted Graph

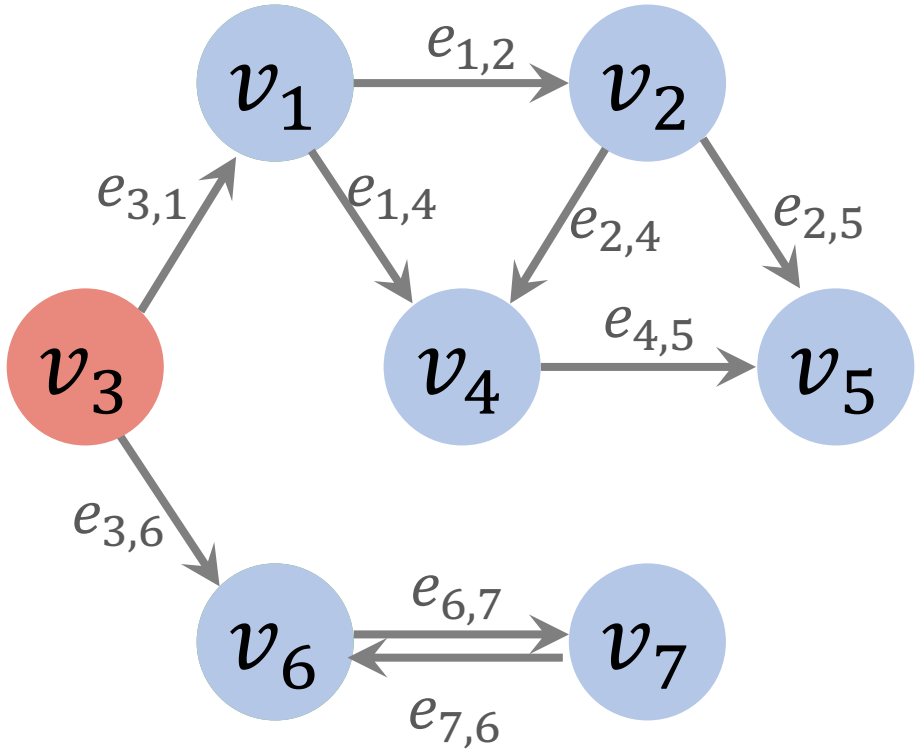


Adjacency matrix:

	v_1	v_2	v_3	v_4	v_5	v_6	v_7
v_1	0	1	1	1	0	0	0
v_2	1	0	0	1	0	0	0
v_3	1	0	0	1	0	1	0
v_4	1	1	1	0	0	0	0
v_5	0	0	0	0	0	0	0
v_6	0	0	1	0	0	0	1
v_7	0	0	0	0	0	1	0

Directed Unweighted Graph

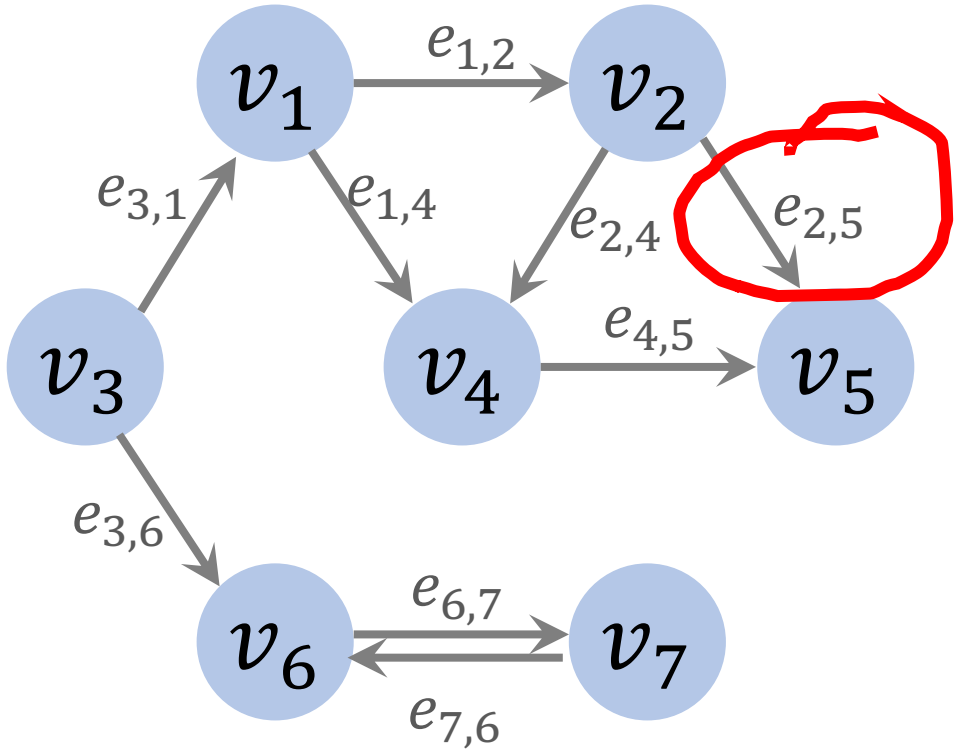
Directed Unweighted Graph



Adjacency list:

From	To
1	2, 4
2	4, 5
3	**1, 6**
4	5
5	empty
6	7
7	6

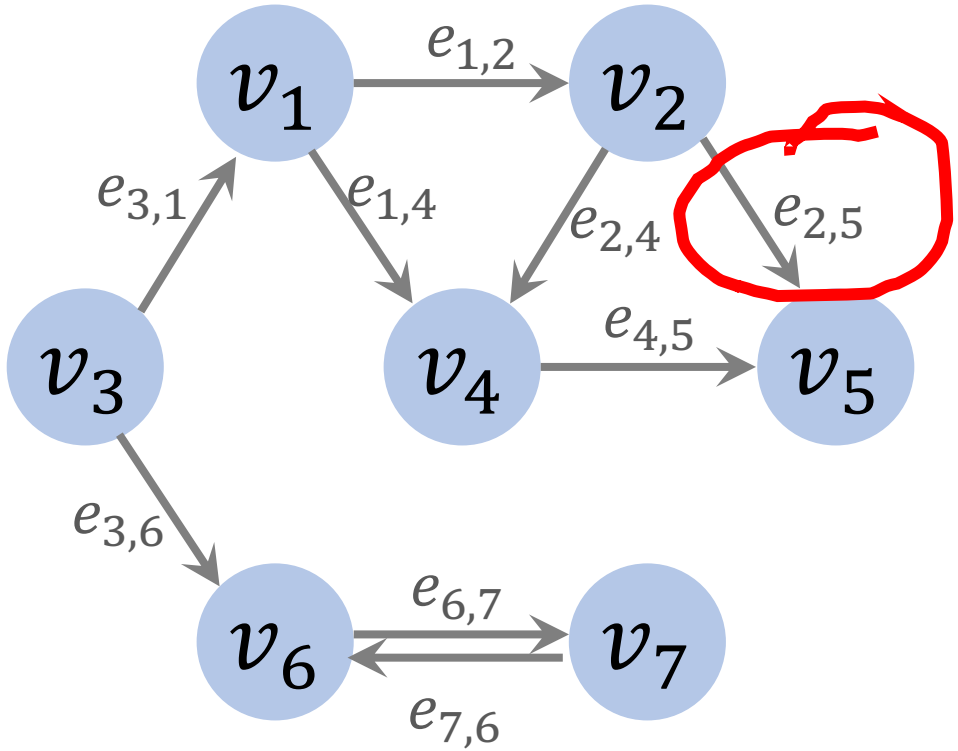
Directed Unweighted Graph



Adjacency matrix:

From \ To	v_1	v_2	v_3	v_4	v_5	v_6	v_7
v_1	0	**1**	0	**1**	0	0	0
v_2	0	0	0	**1**	**1**	0	0
v_3	**1**	0	0	0	0	**1**	0
v_4	0	0	0	0	**1**	0	0
v_5	0	0	0	0	0	0	0
v_6	0	0	0	0	0	0	**1**
v_7	0	0	0	0	0	**1**	0

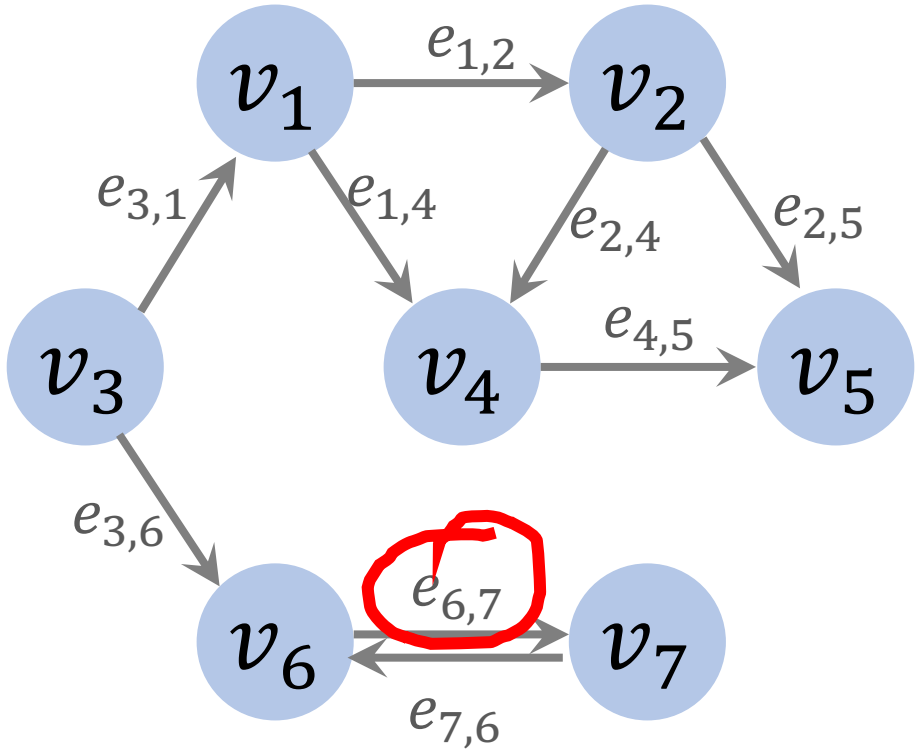
Directed Unweighted Graph



Adjacency matrix:

From \ To	v_1	v_2	v_3	v_4	v_5	v_6	v_7
v_1	0	**1**	0	**1**	0	0	0
v_2	0	0	0	**1**	**1**	0	0
v_3	**1**	0	0	0	0	**1**	0
v_4	0	0	0	0	**1**	0	0
v_5	0	0	0	0	0	0	0
v_6	0	0	0	0	0	0	**1**
v_7	0	0	0	0	0	**1**	0

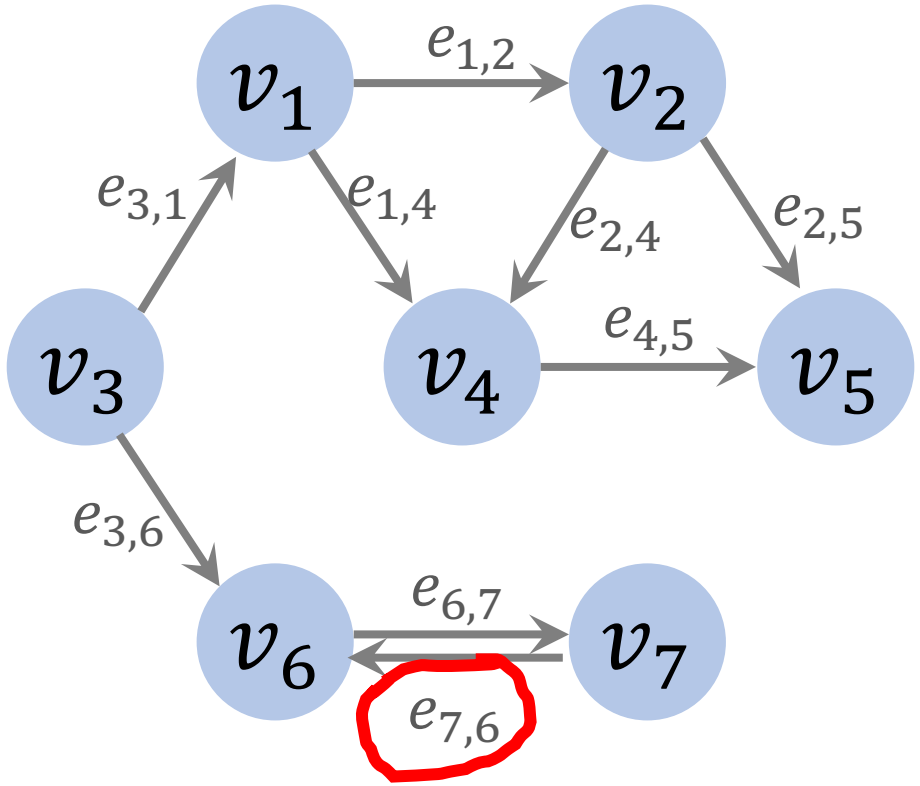
Directed Unweighted Graph



Adjacency matrix:

From \ To	v_1	v_2	v_3	v_4	v_5	v_6	v_7
v_1	0	**1**	0	**1**	0	0	0
v_2	0	0	0	**1**	**1**	0	0
v_3	**1**	0	0	0	0	**1**	0
v_4	0	0	0	0	**1**	0	0
v_5	0	0	0	0	0	0	0
v_6	0	0	0	0	0	0	**1**
v_7	0	0	0	0	0	**1**	0

Directed Unweighted Graph

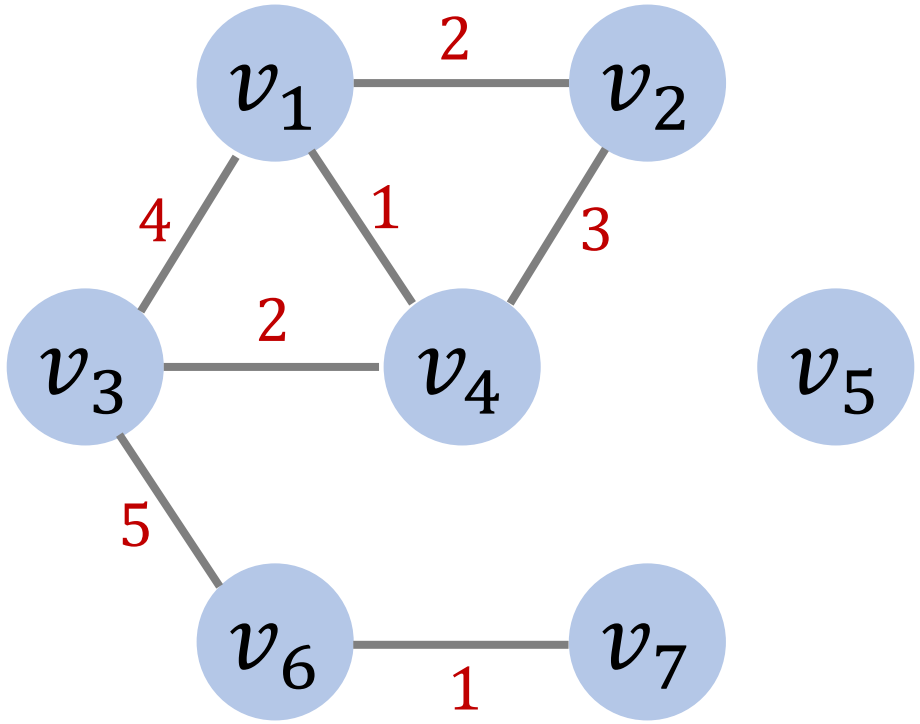


Adjacency matrix:

From \ To	v_1	v_2	v_3	v_4	v_5	v_6	v_7
v_1	0	**1**	0	**1**	0	0	0
v_2	0	0	0	**1**	**1**	0	0
v_3	**1**	0	0	0	0	**1**	0
v_4	0	0	0	0	**1**	0	0
v_5	0	0	0	0	0	0	0
v_6	0	0	0	0	0	0	**1**
v_7	0	0	0	0	0	**1**	0

Weighted Graphs

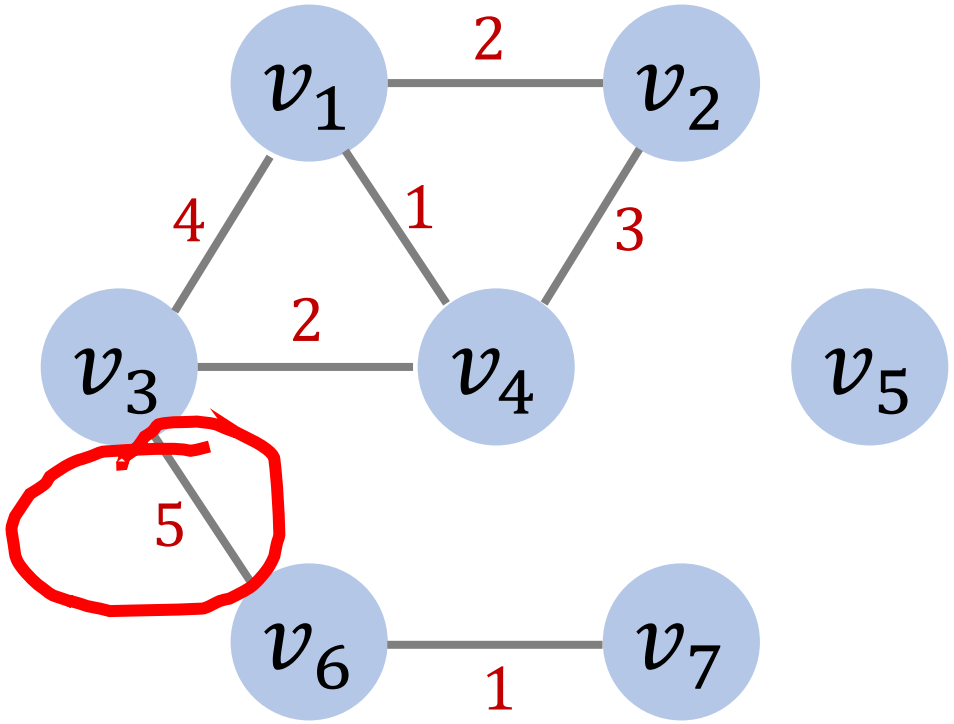
Undirected Weighted Graph



Adjacency matrix:

	v_1	v_2	v_3	v_4	v_5	v_6	v_7
v_1	0	**2**	**4**	**1**	0	0	0
v_2	**2**	0	0	**3**	0	0	0
v_3	**4**	0	0	**2**	0	**5**	0
v_4	**1**	**3**	**2**	0	0	0	0
v_5	0	0	0	0	0	0	0
v_6	0	0	**5**	0	0	0	**1**
v_7	0	0	0	0	0	**1**	0

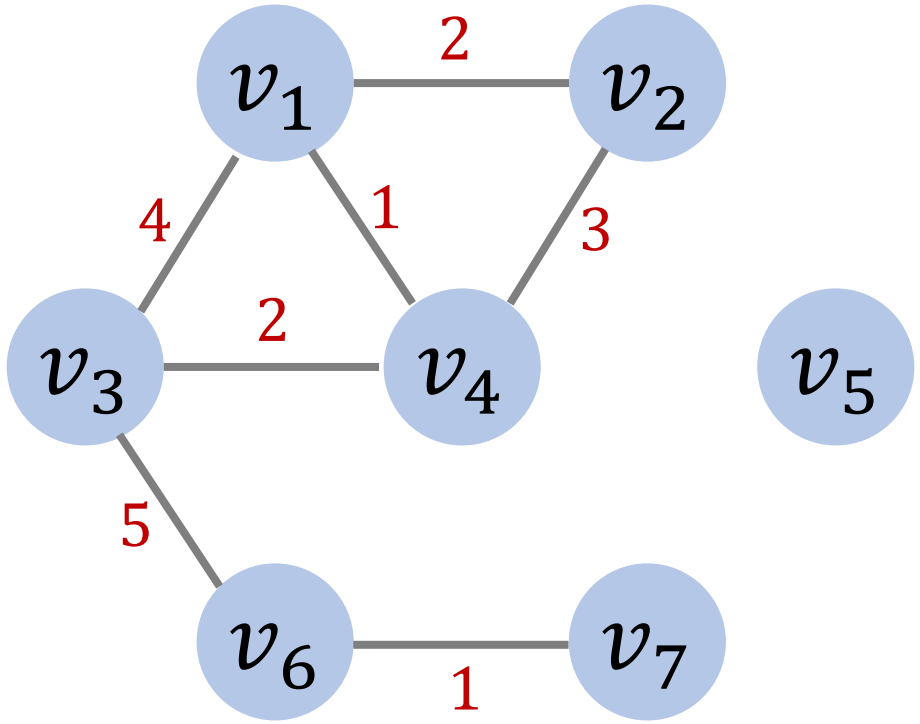
Undirected Weighted Graph



Adjacency matrix:

	v_1	v_2	v_3	v_4	v_5	v_6	v_7
v_1	0	**2**	**4**	**1**	0	0	0
v_2	**2**	0	0	**3**	0	0	0
v_3	**4**	0	0	**2**	0	**5**	0
v_4	**1**	**3**	**2**	0	0	0	0
v_5	0	0	0	0	0	0	0
v_6	0	0	**5**	0	0	0	**1**
v_7	0	0	0	0	0	**1**	0

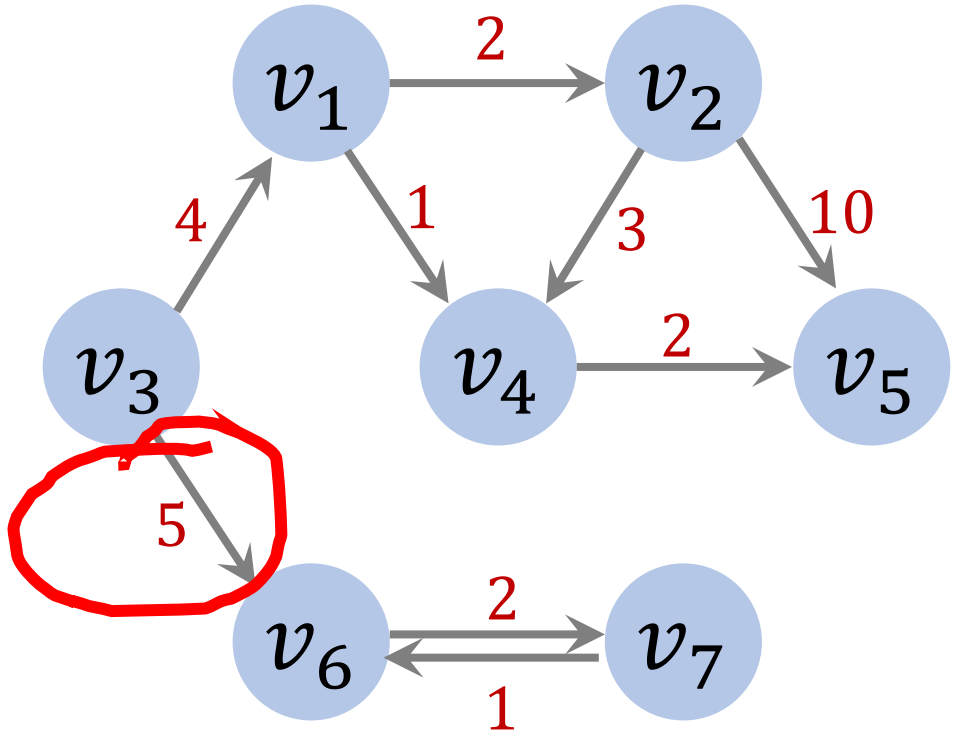
Undirected Weighted Graph



Adjacency matrix:

	v_1	v_2	v_3	v_4	v_5	v_6	v_7
v_1	∞	**2**	**4**	**1**	∞	∞	∞
v_2	**2**	∞	∞	**3**	∞	∞	∞
v_3	**4**	∞	∞	**2**	∞	**5**	∞
v_4	**1**	**3**	**2**	∞	∞	∞	∞
v_5	∞	∞	∞	∞	∞	∞	∞
v_6	∞	∞	**5**	∞	∞	∞	**1**
v_7	∞	∞	∞	∞	∞	**1**	∞

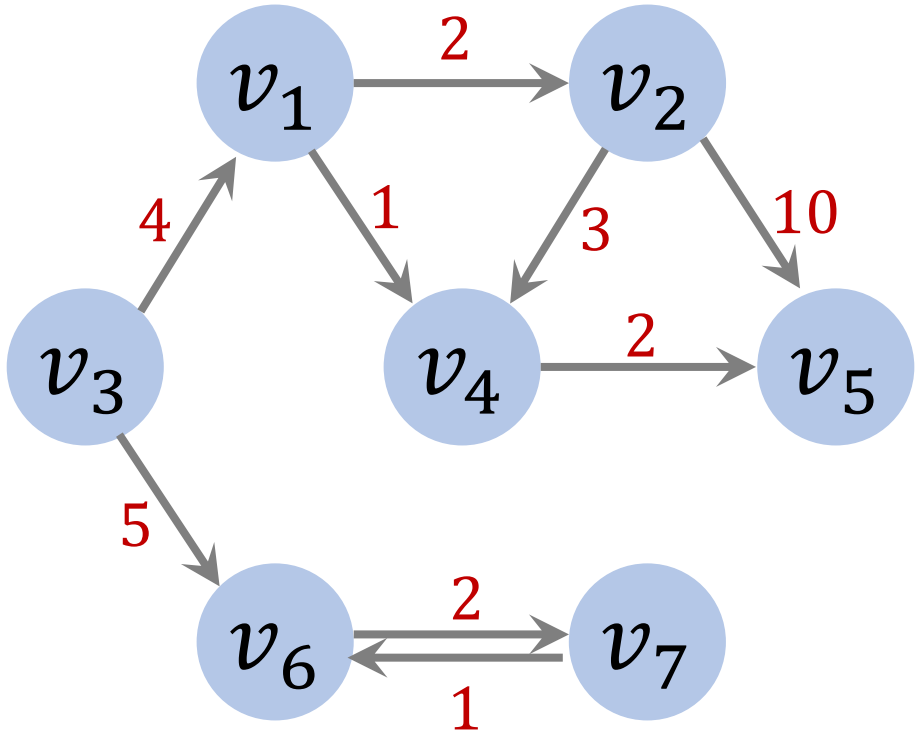
Directed Weighted Graph



Adjacency matrix:

From \ To	v_1	v_2	v_3	v_4	v_5	v_6	v_7
v_1	0	**2**	0	**1**	0	0	0
v_2	0	0	0	**3**	**10**	0	0
v_3	**4**	0	0	0	0	**5**	0
v_4	0	0	0	0	**2**	0	0
v_5	0	0	0	0	0	0	0
v_6	0	0	0	0	0	0	**2**
v_7	0	0	0	0	0	**1**	0

Directed Weighted Graph



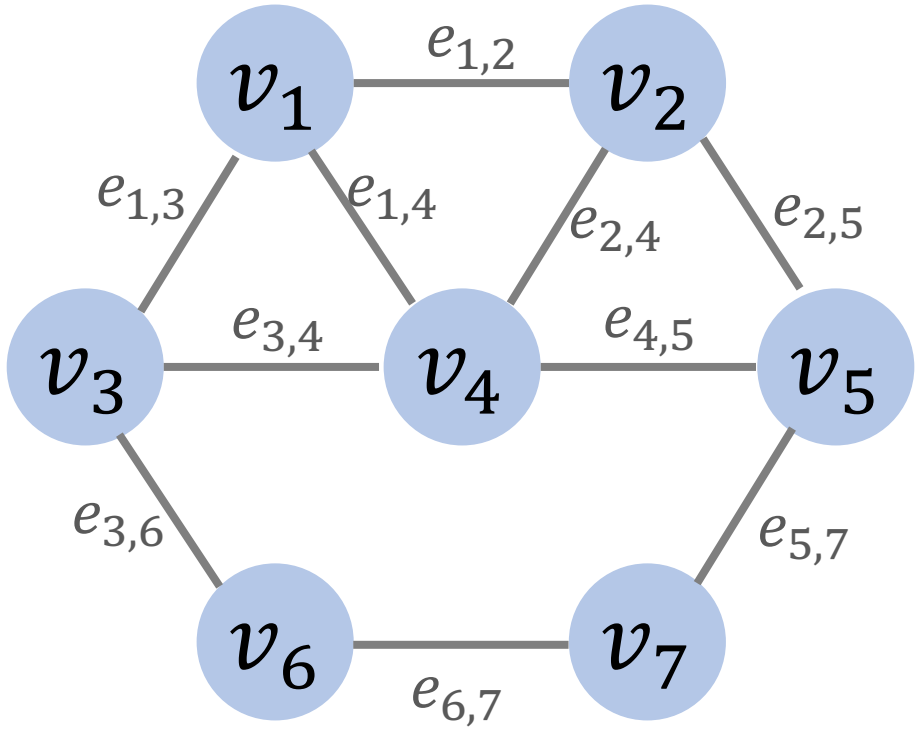
Adjacency matrix:

From \ To	v_1	v_2	v_3	v_4	v_5	v_6	v_7
v_1	∞	**2**	∞	**1**	∞	∞	∞
v_2	∞	∞	∞	**3**	**10**	∞	∞
v_3	**4**	∞	∞	∞	∞	**5**	∞
v_4	∞	∞	∞	∞	**2**	∞	∞
v_5	∞	∞	∞	∞	∞	∞	∞
v_6	∞	∞	∞	∞	∞	∞	**2**
v_7	∞	∞	∞	∞	∞	**1**	∞

Questions

Question 1: Fill in the adjacency list

Unweighted graph:

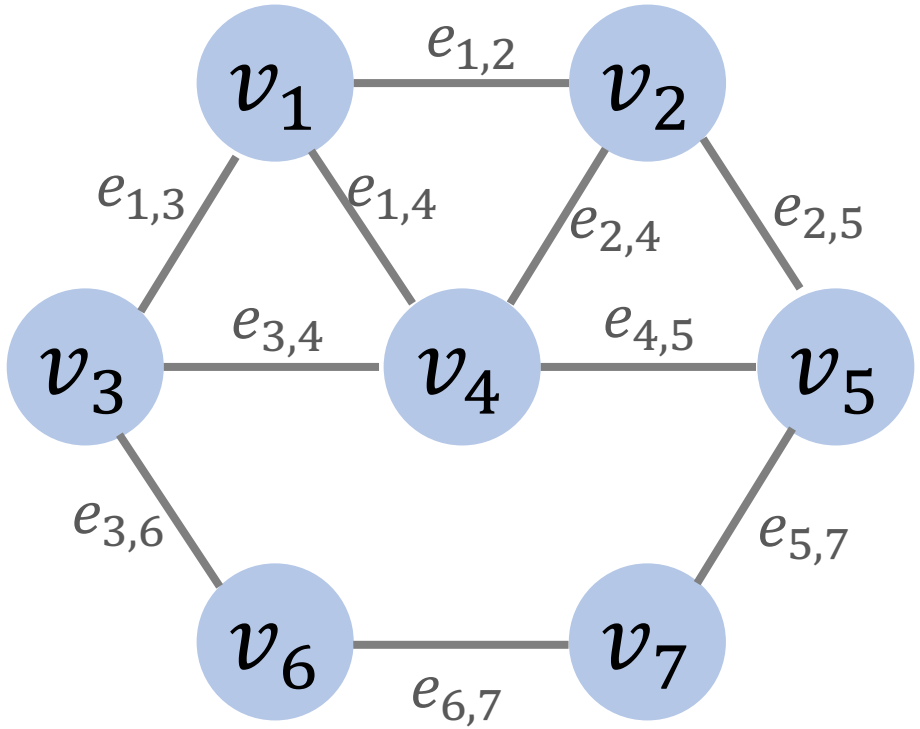


Adjacency list:

Vertex	Neighbors
1	
2	
3	
4	
5	
6	
7	

Question 2: Fill in the adjacency matrix

Unweighted graph:



Adjacency matrix:

	v_1	v_2	v_3	v_4	v_5	v_6	v_7
v_1							
v_2							
v_3							
v_4							
v_5							
v_6							
v_7							

Thank You!

Implementation

```
#include <set>

struct Graph {
    int V;
    set<int>* adjList;
};

Graph* createGraph(int n) { // n is # of vertices
    Graph* graph = new Graph;
    graph->n = n;
    graph->adjList = new set<int>[n];
    return graph;
}
```

```
void addEdge(Graph* graph, int src, int dest) {
        graph->adjList[src].insert(dest);
        graph->adjList[dest].insert(src);
}
```

```
void printGraph(Graph* graph) {
      for (int i = 0; i < graph->V; ++i) {
            set<int> lst = graph->adjList[i];
            cout << endl << "Adjacency list of vertex "
                  << i << endl;
            for (auto itr = lst.begin();
                        itr != lst.end(); ++itr) {
                  cout << *itr << " " << endl;
            }
      }
}
```