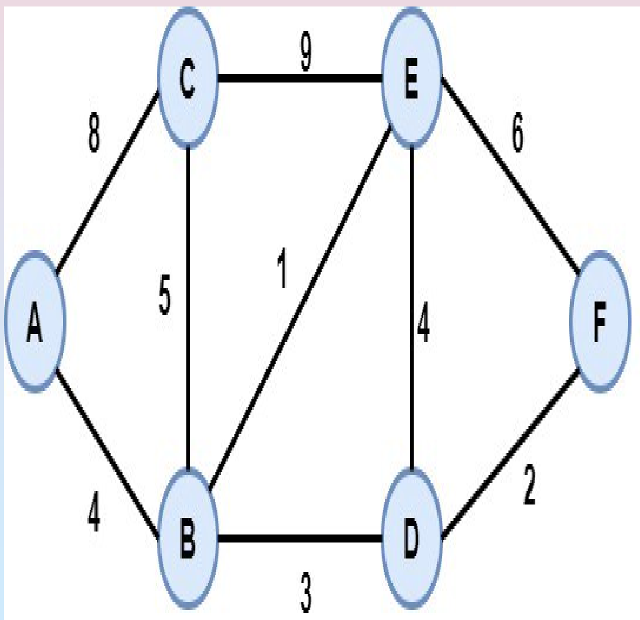
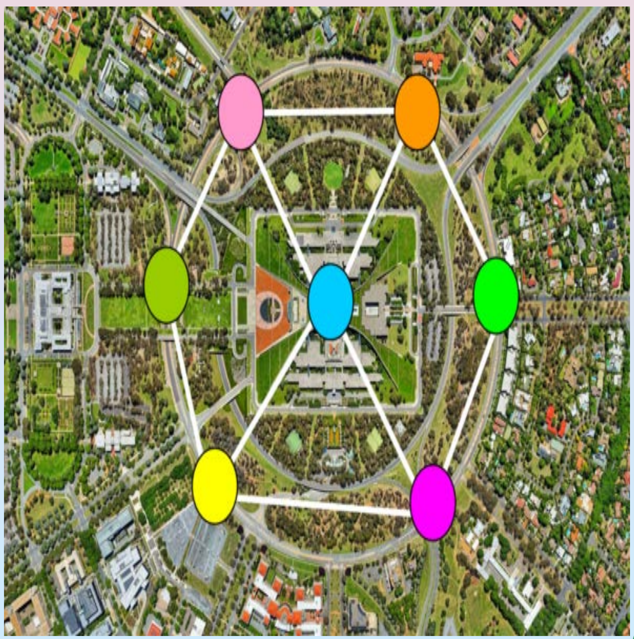
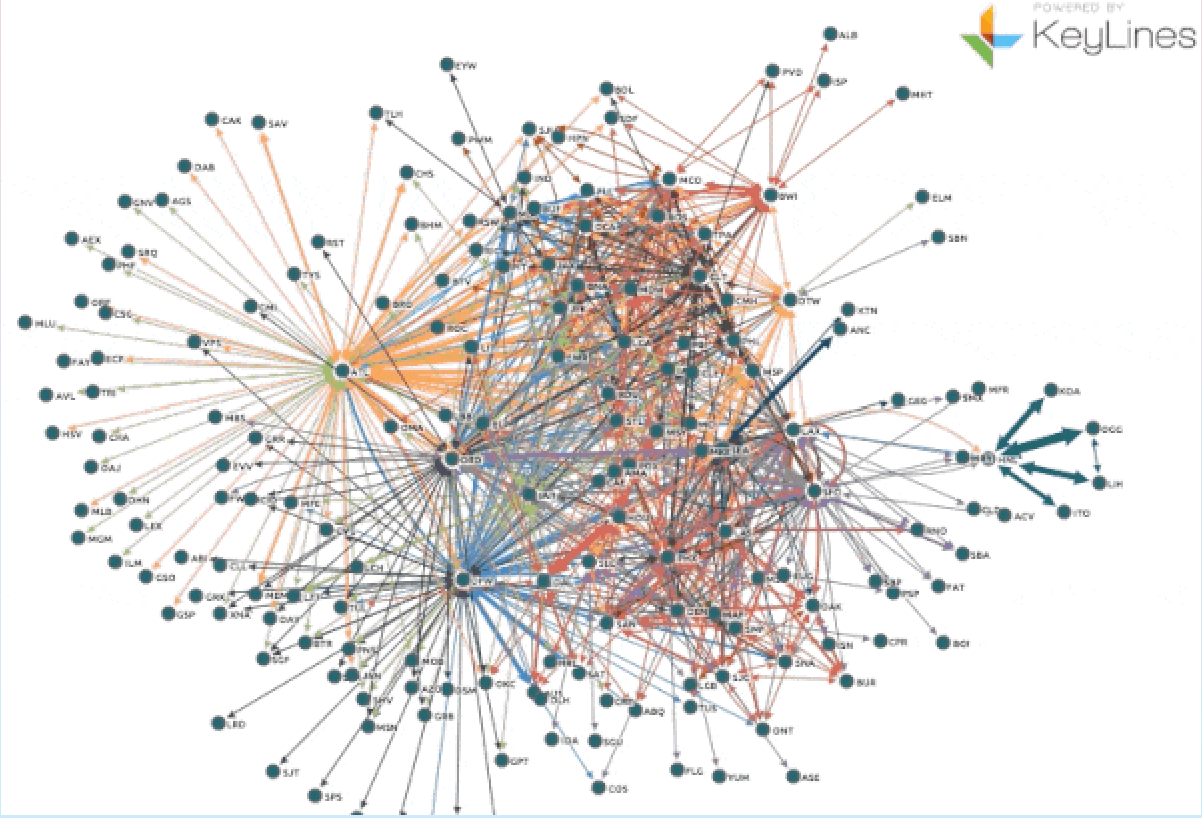
Graphs





Graphs



Graphs

- G = (V,E)
- V is the vertex set.
- Vertices are also called nodes and points.
- E is the edge set.
- Each edge connects two different vertices.
- Edges are also called arcs and lines.
- Directed edge has an orientation (u,v).

$$u \longrightarrow v$$

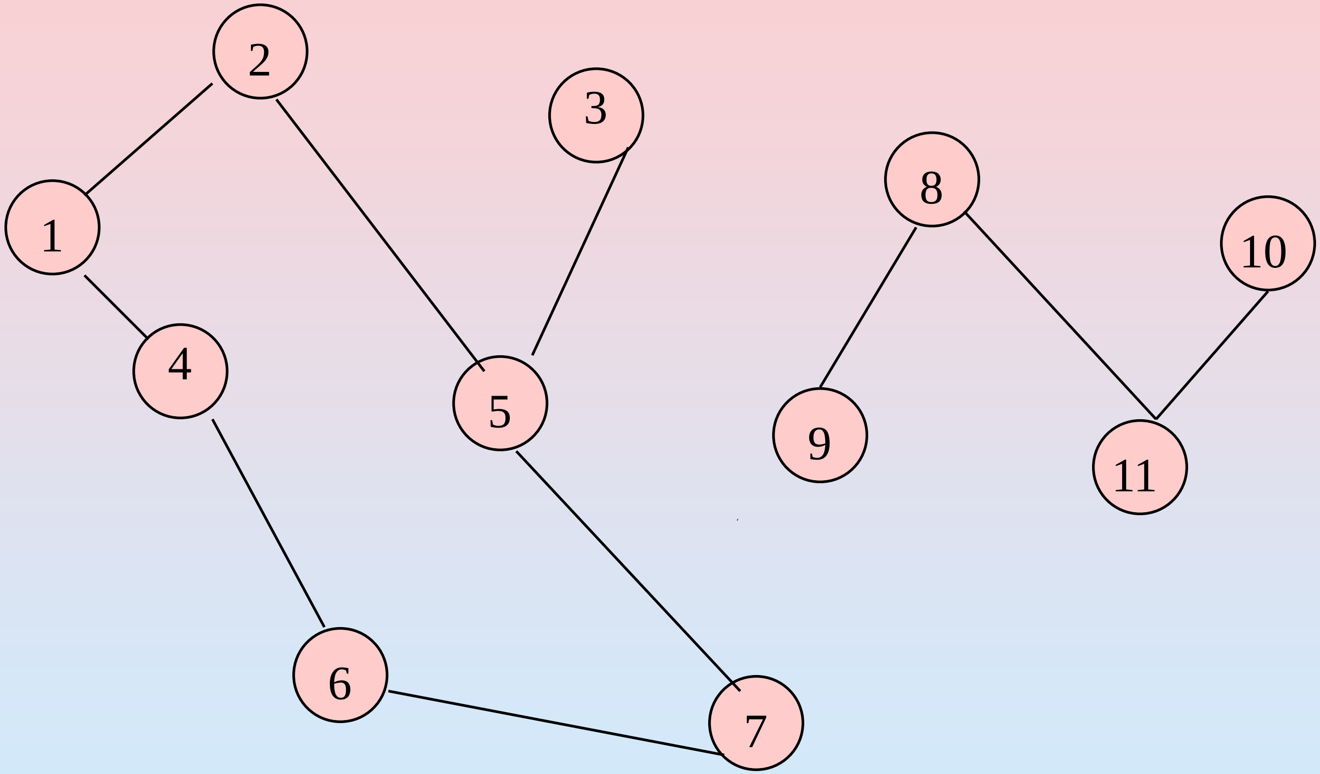
Graphs

- **Undirected edge has no orientation (u,v).**

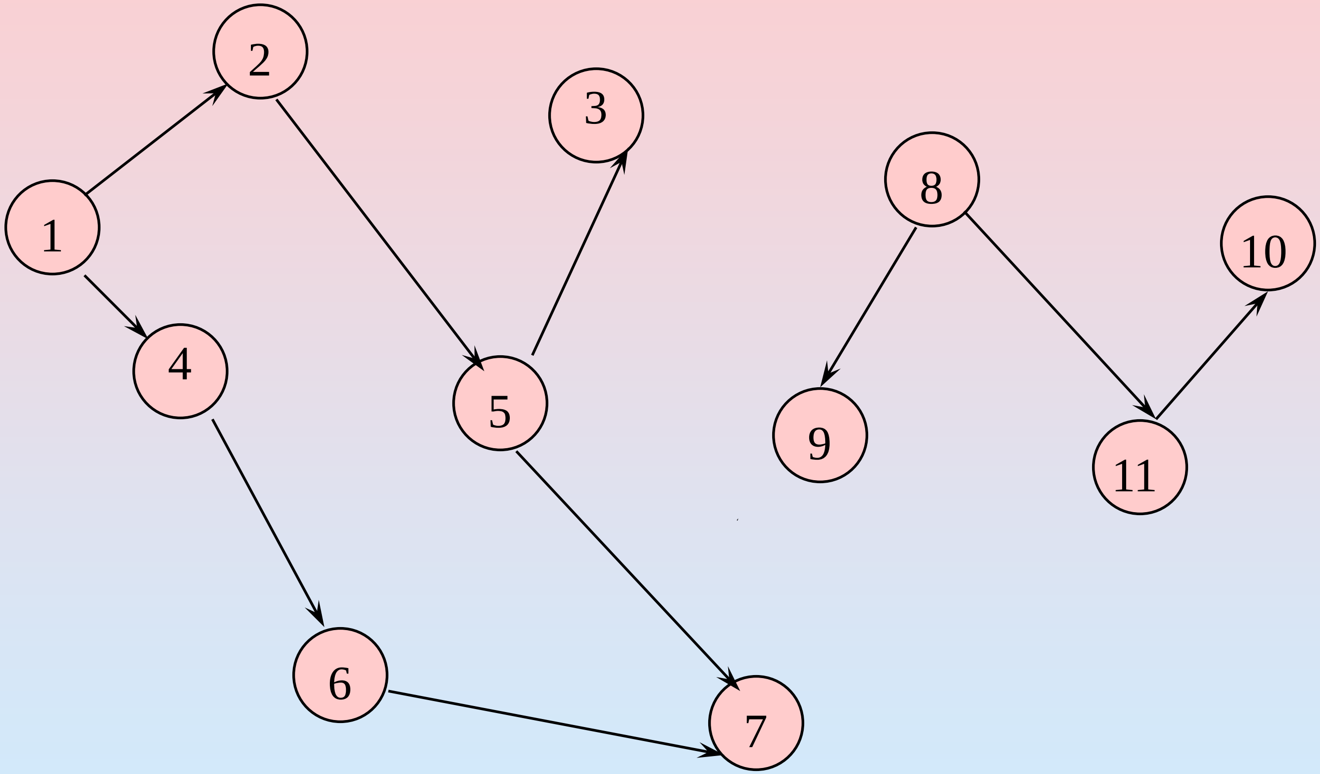
u ——— v

- Undirected graph => no oriented edge.
- Directed graph => every edge has an orientation.

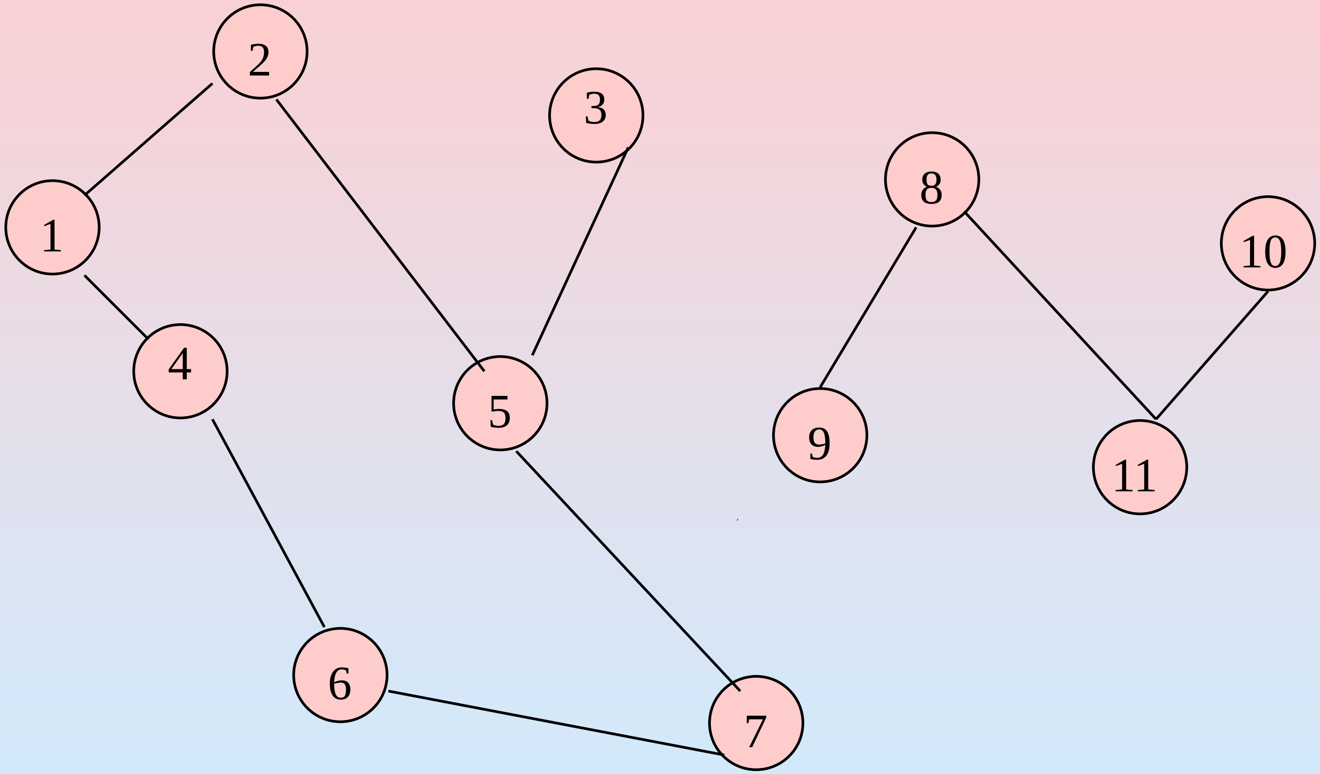
Undirected Graph



Directed Graph (Digraph)

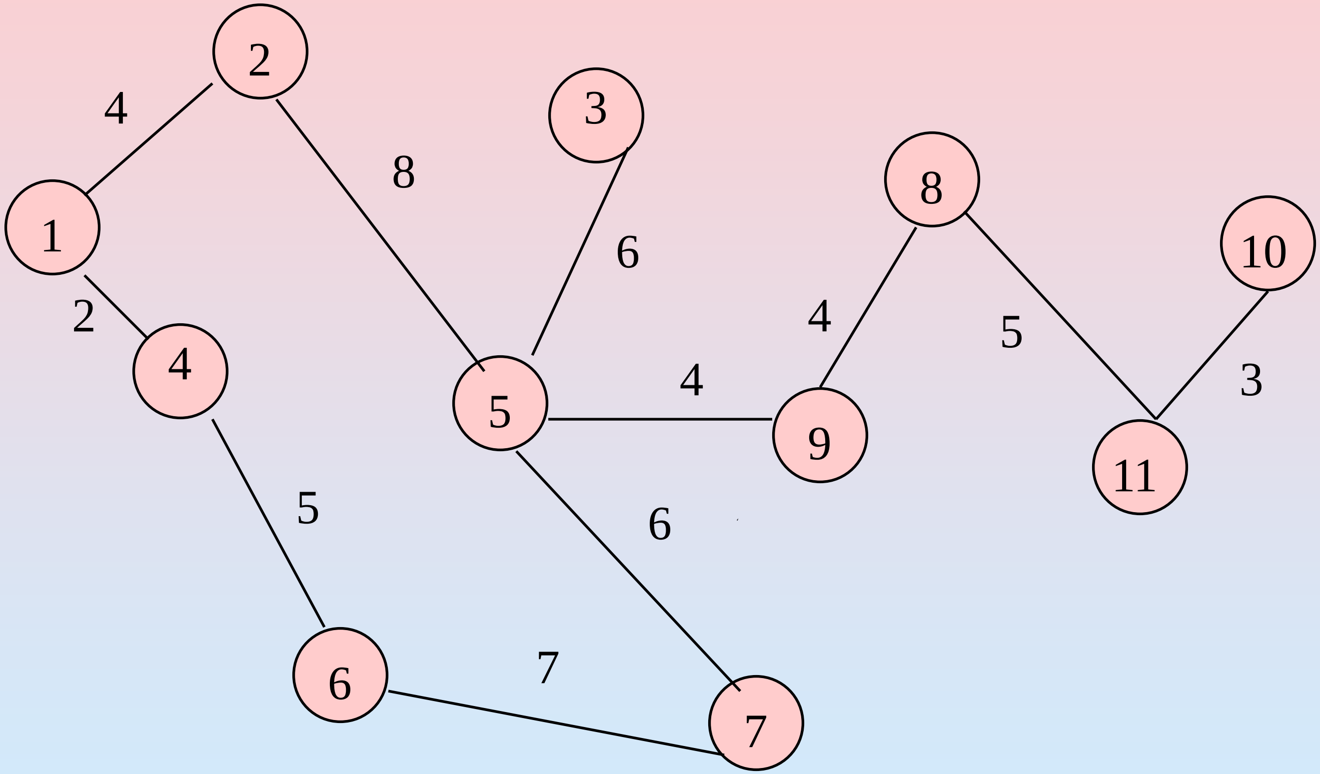


Applications—Communication Network



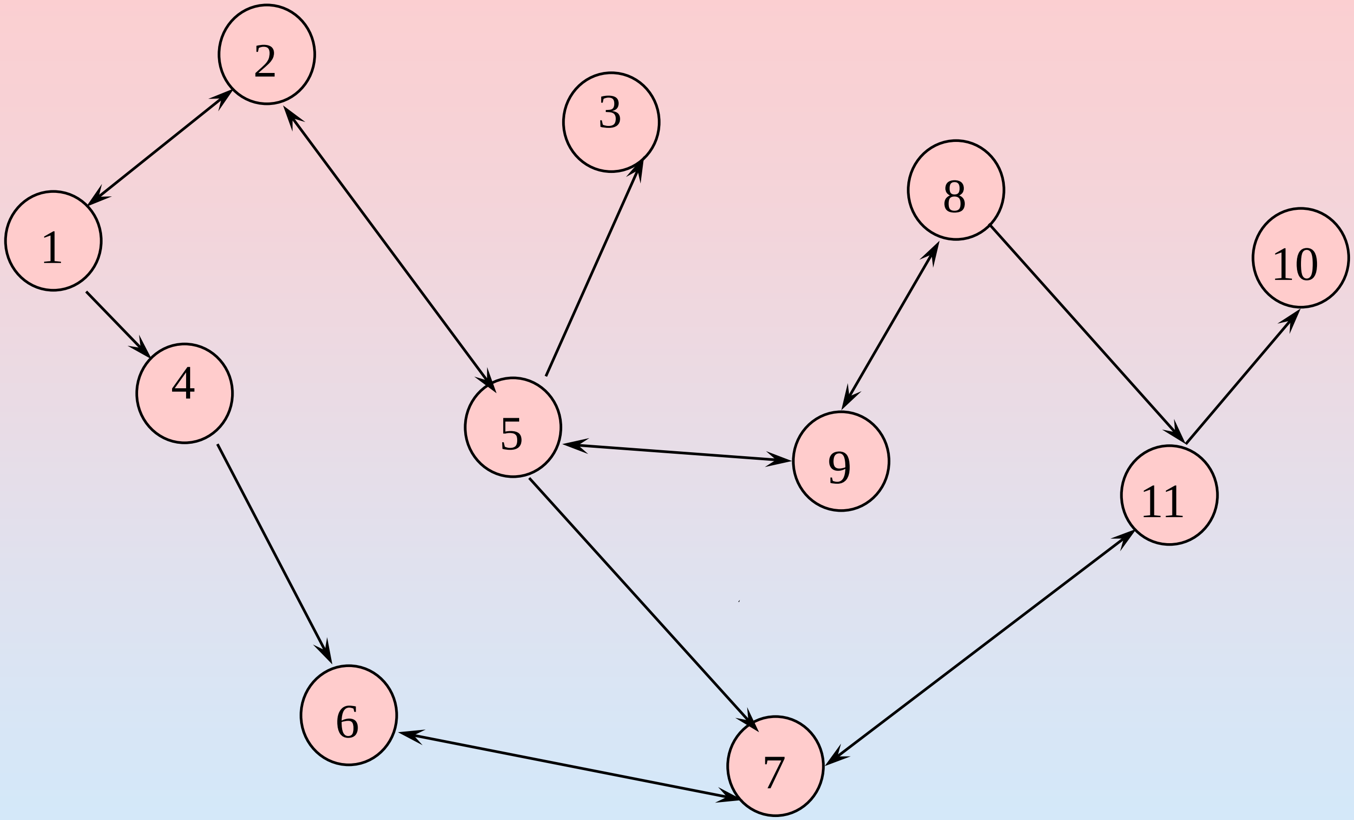
- Vertex = city, edge = communication link.

Driving Distance/Time Map



- Vertex = city, edge weight = driving distance/time.

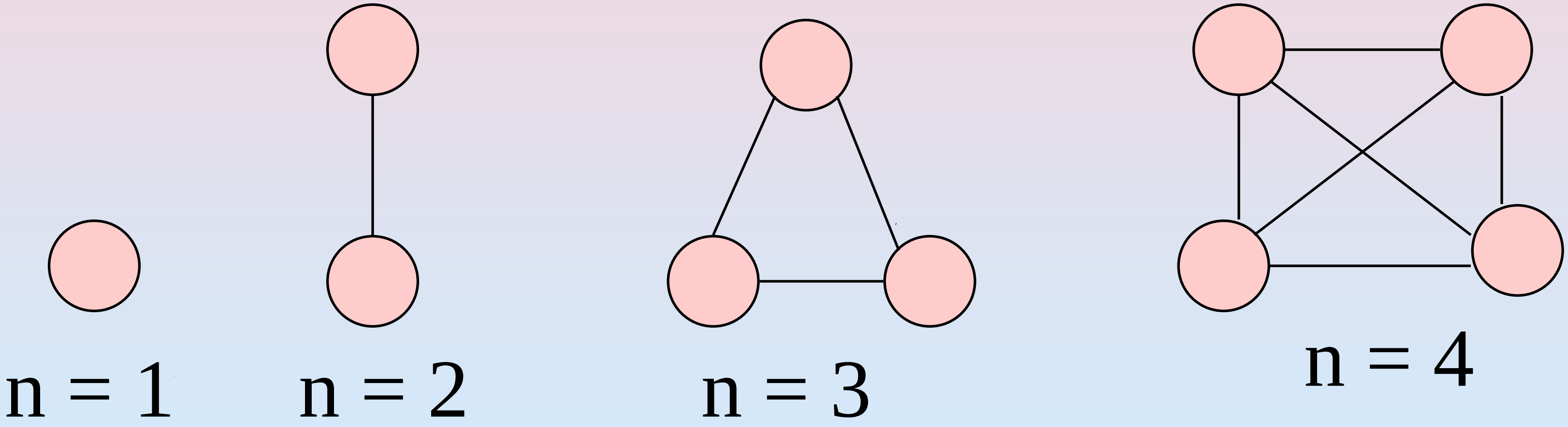
Street Map



❖ **Some streets are one way.**

Complete Undirected Graph

Has all possible edges.



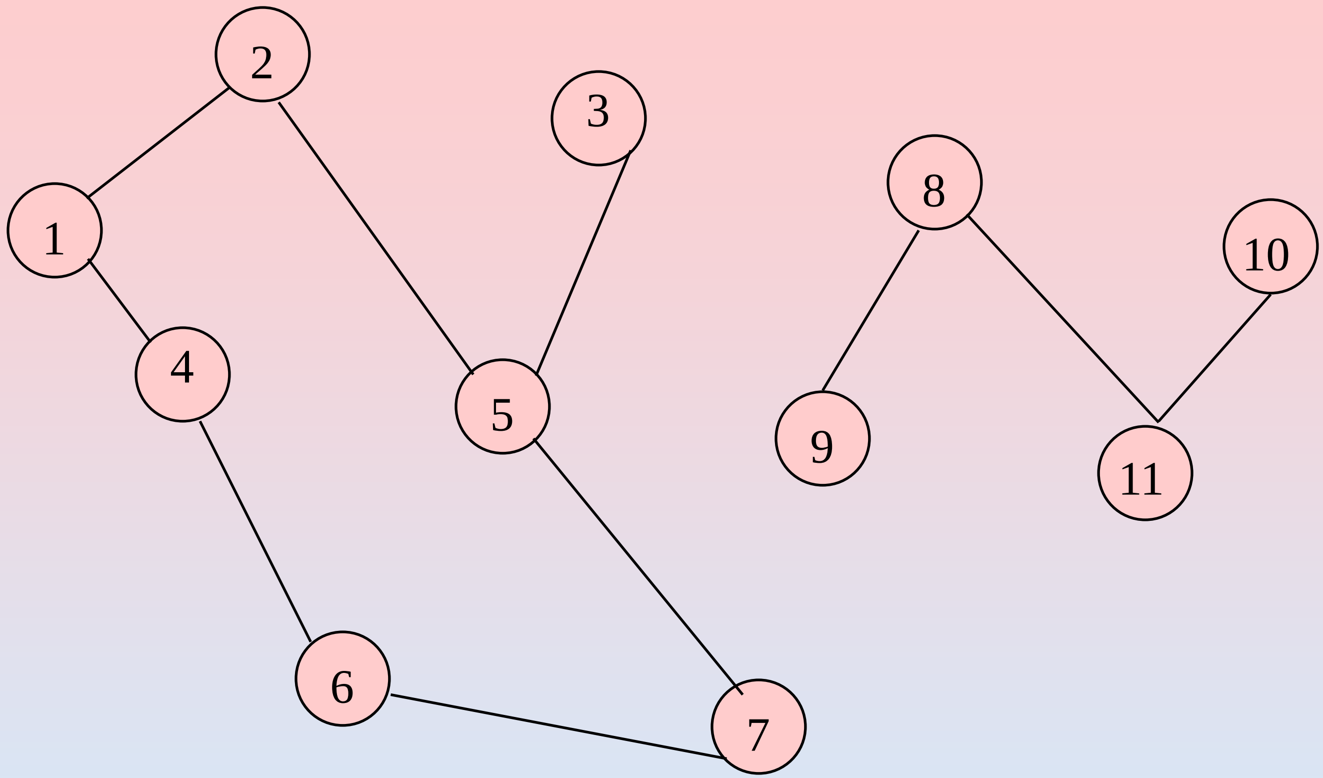
Number Of Edges—Undirected Graph

- Each edge is of the form (u,v), u != v.
- Number of such pairs in an n vertex graph is n(n-1).
- Since edge (u,v) is the same as edge (v,u), the number of edges in a complete undirected graph is n(n-1)/2.
- Number of edges in an undirected graph is <= n(n-1)/2.

Number Of Edges—Directed Graph

- Each edge is of the form (u,v), u != v.
- Number of such pairs in an n vertex graph is n(n-1).
- Since edge (u,v) is not the same as edge (v,u), the number of edges in a complete directed graph is n(n-1).
- Number of edges in a directed graph is <= n(n-1).

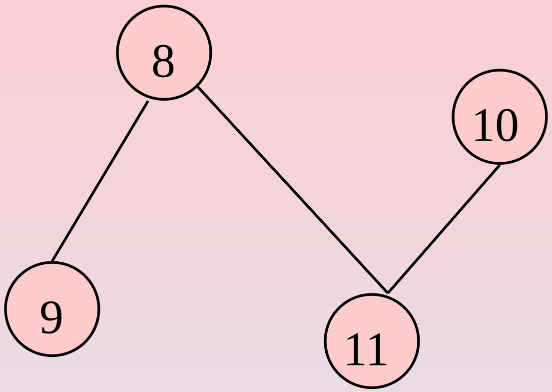
Vertex Degree



Number of edges incident to vertex.

degree(2) = 2, degree(5) = 3, degree(3) = 1

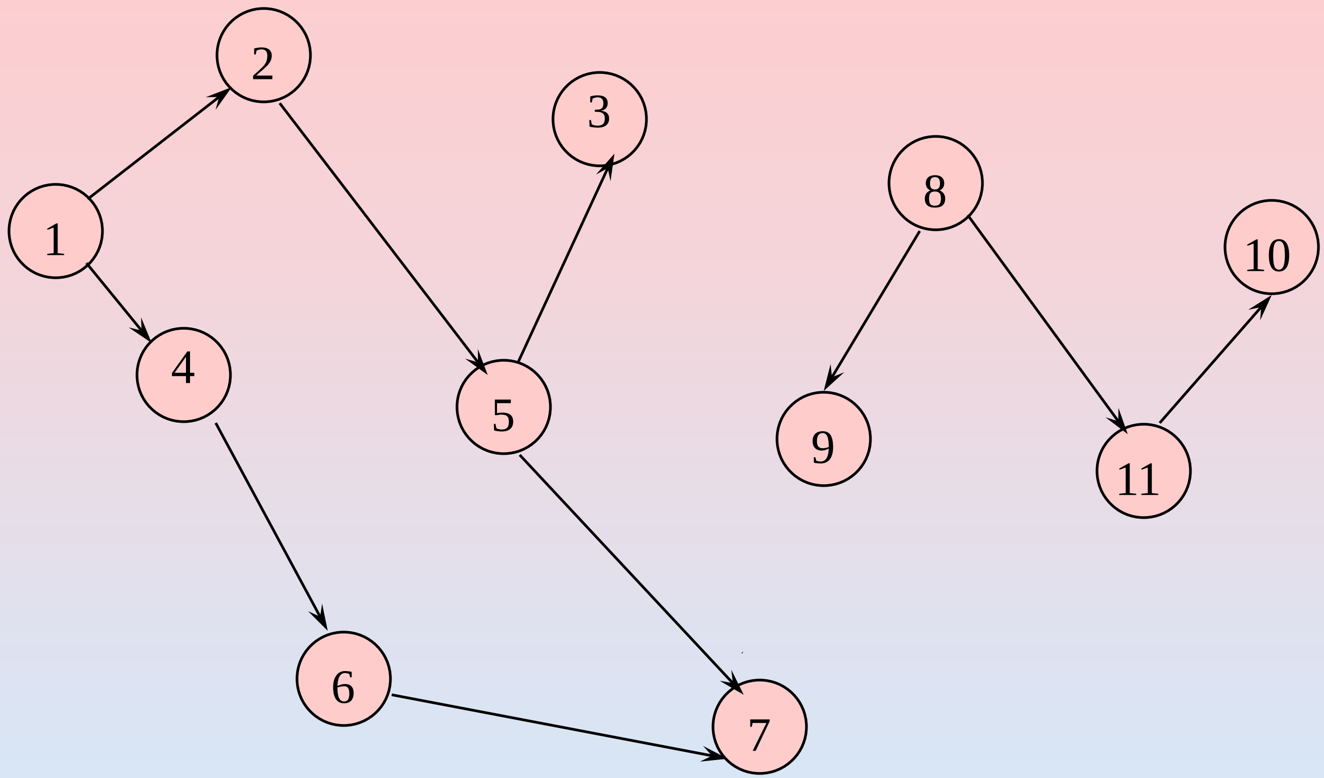
Sum Of Vertex Degrees



Sum of degrees = 2e (e is number of edges)

= 2 * 3 = 6

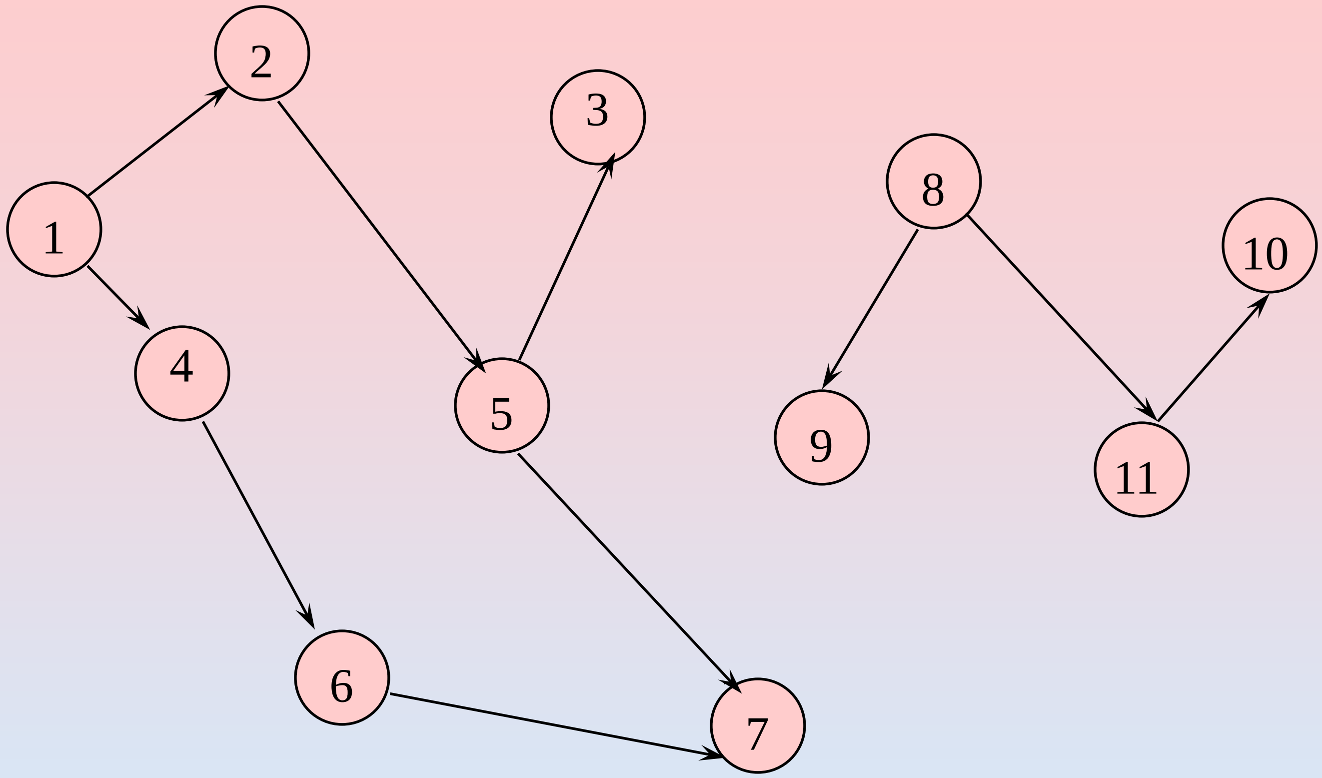
In-Degree Of A Vertex



In-degree is number of incoming edges

indegree(2) = 1, indegree(8) = 0, indegree(7) = 2

Out-Degree of a Vertex



out-degree is number of outbound edges

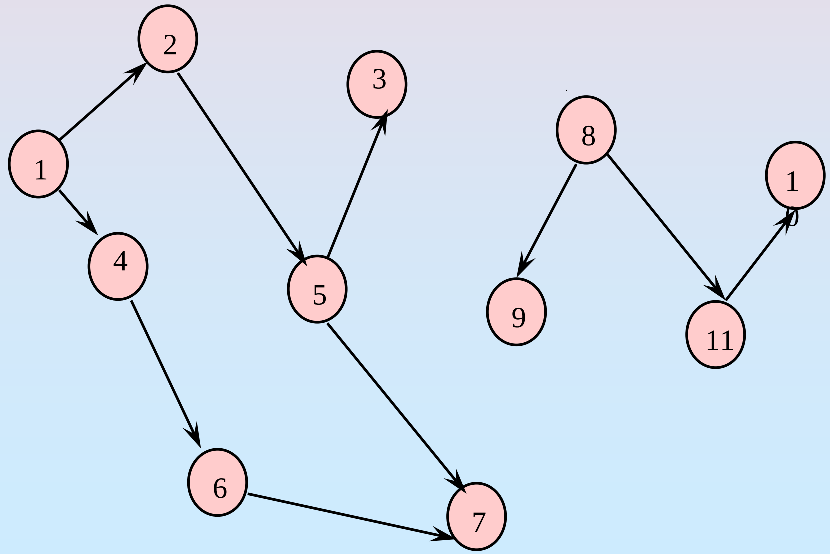
outdegree(2) = 1, outdegree(8) = 2, outdegree(7) = 0

Sum of In- and Out-Degrees

Each edge contributes 1 to the in-degree of some vertex and 1 to the out-degree of some other vertex

sum of in-degrees = sum of out-degrees = e,

where e is the number of edges in the digraph



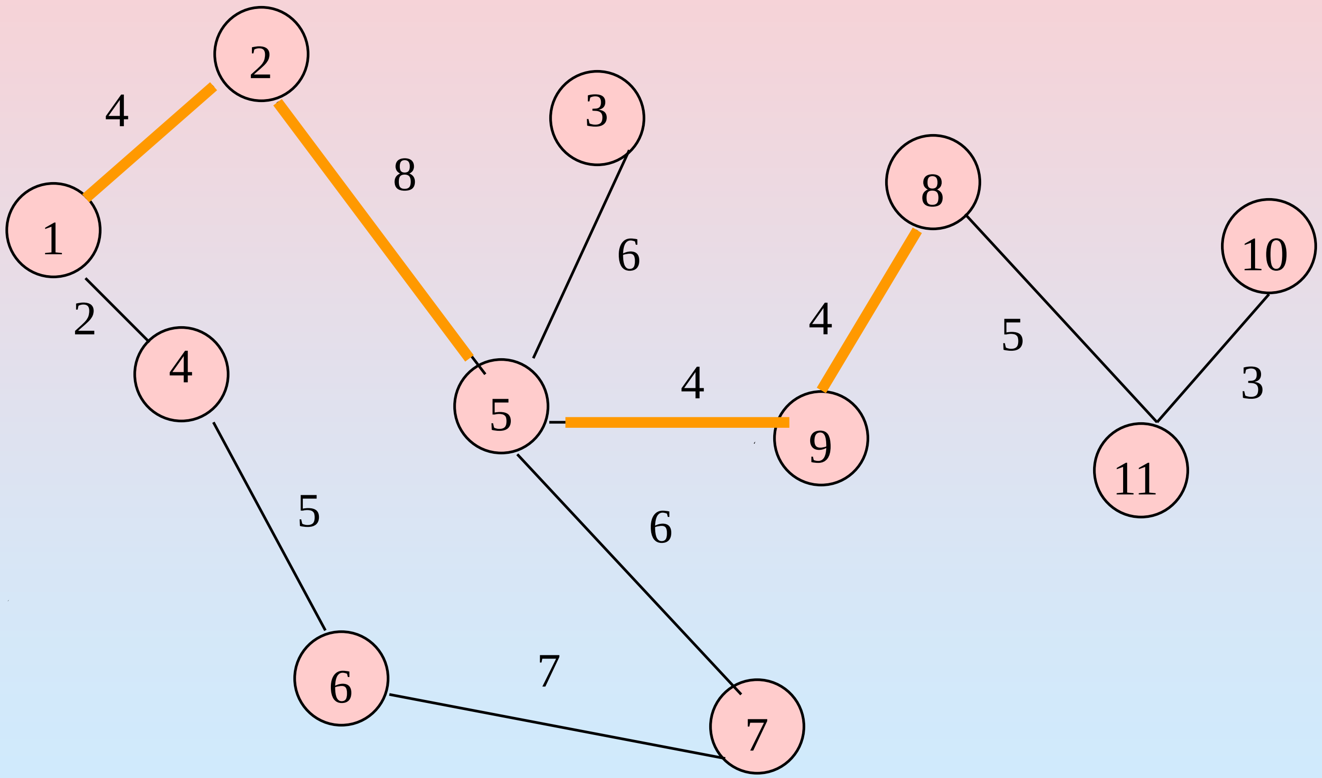
Graph Operations and Representation

Sample Graph Problems

- Path problems.
- Connectedness problems.
- Spanning tree problems.

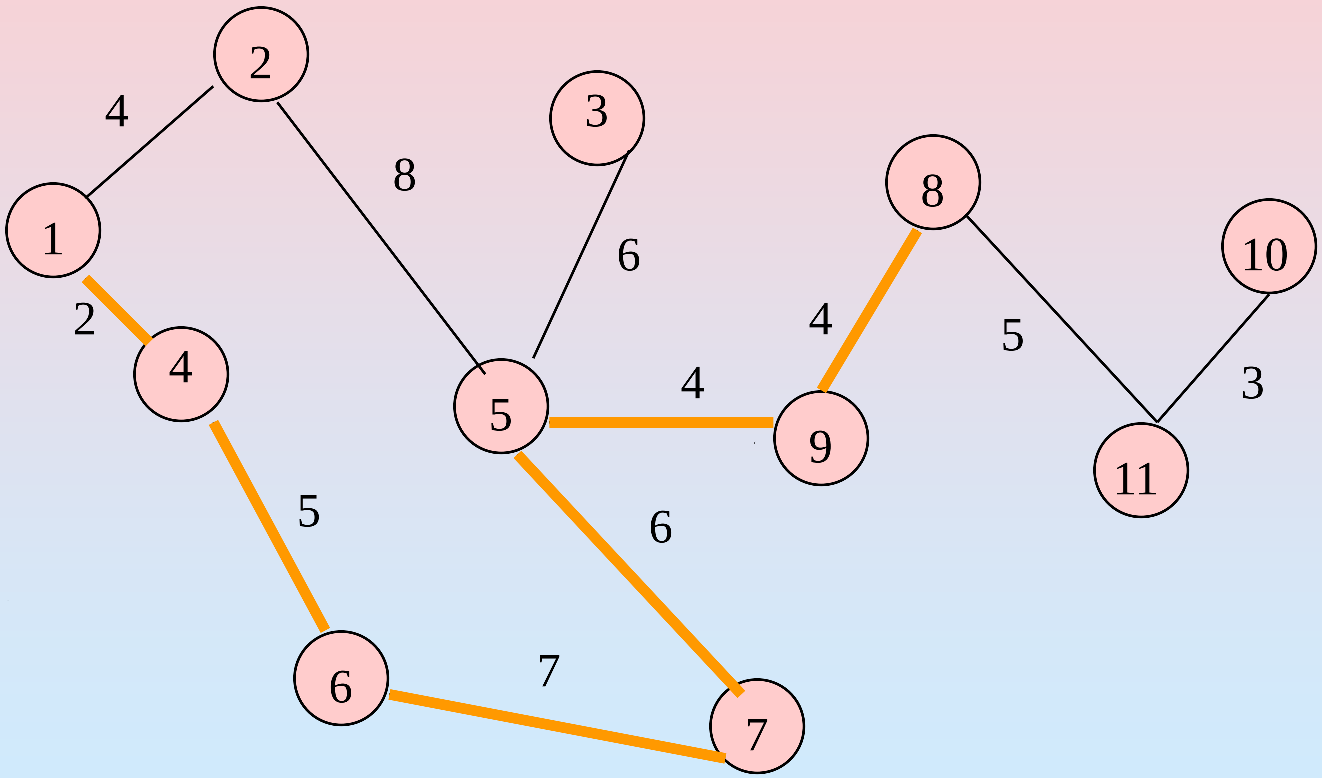
Path Finding

Path between 1 and 8.



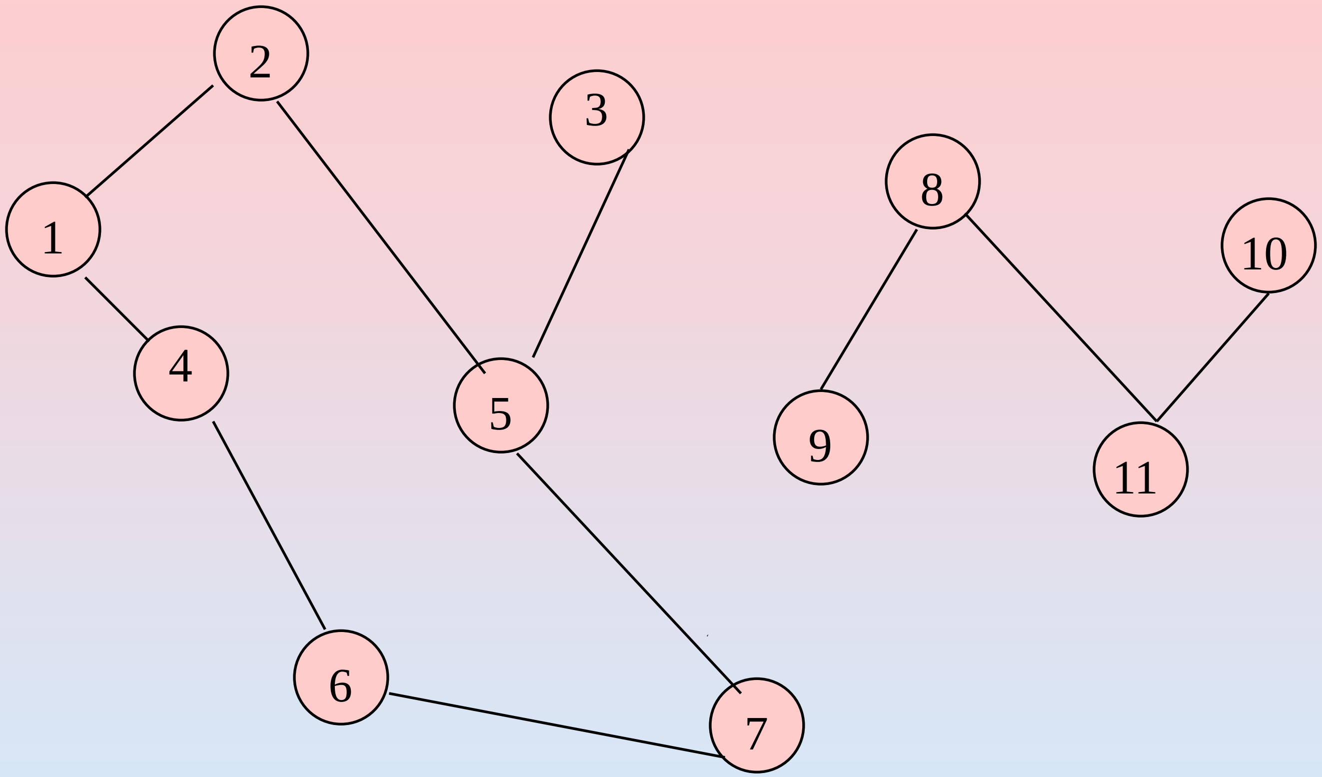
Path length is 20.

Another Path Between 1 and 8



Path length is 28.

Example of No Path

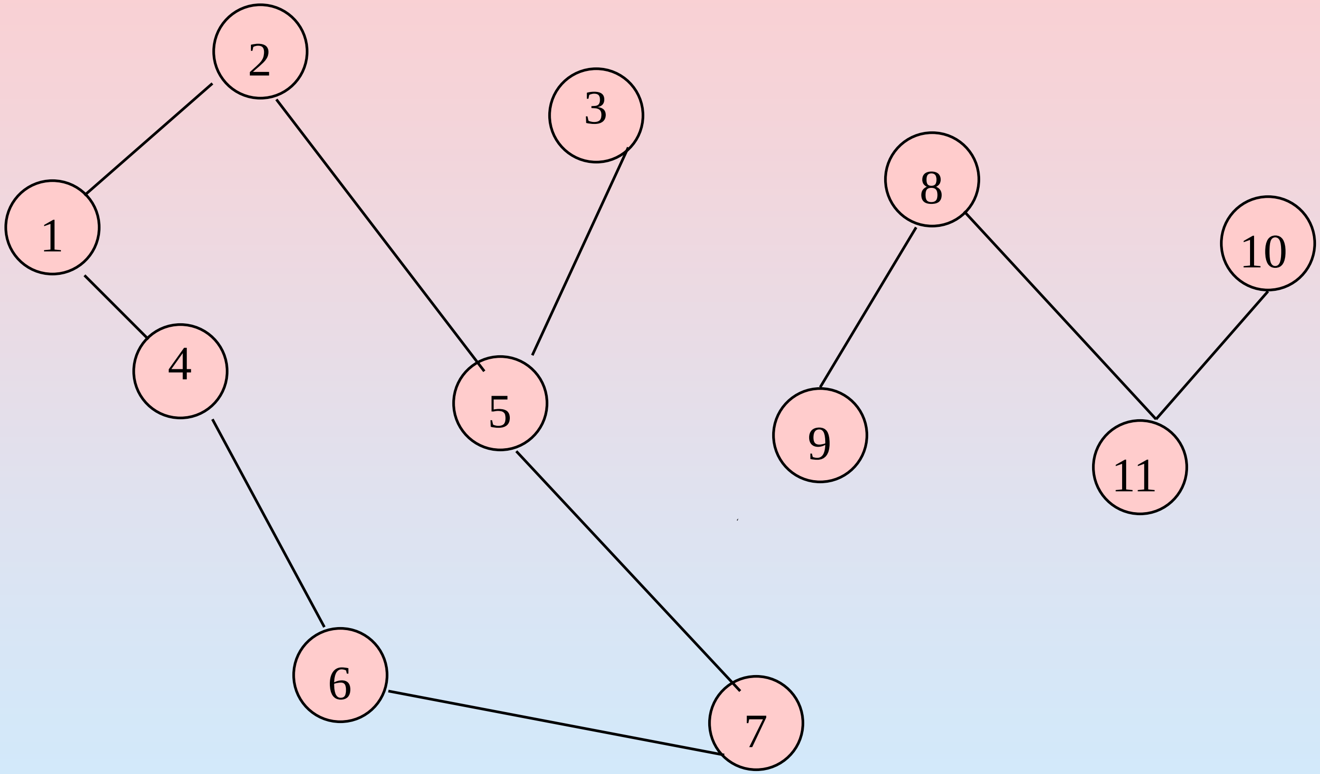


No path between 2 and 9.

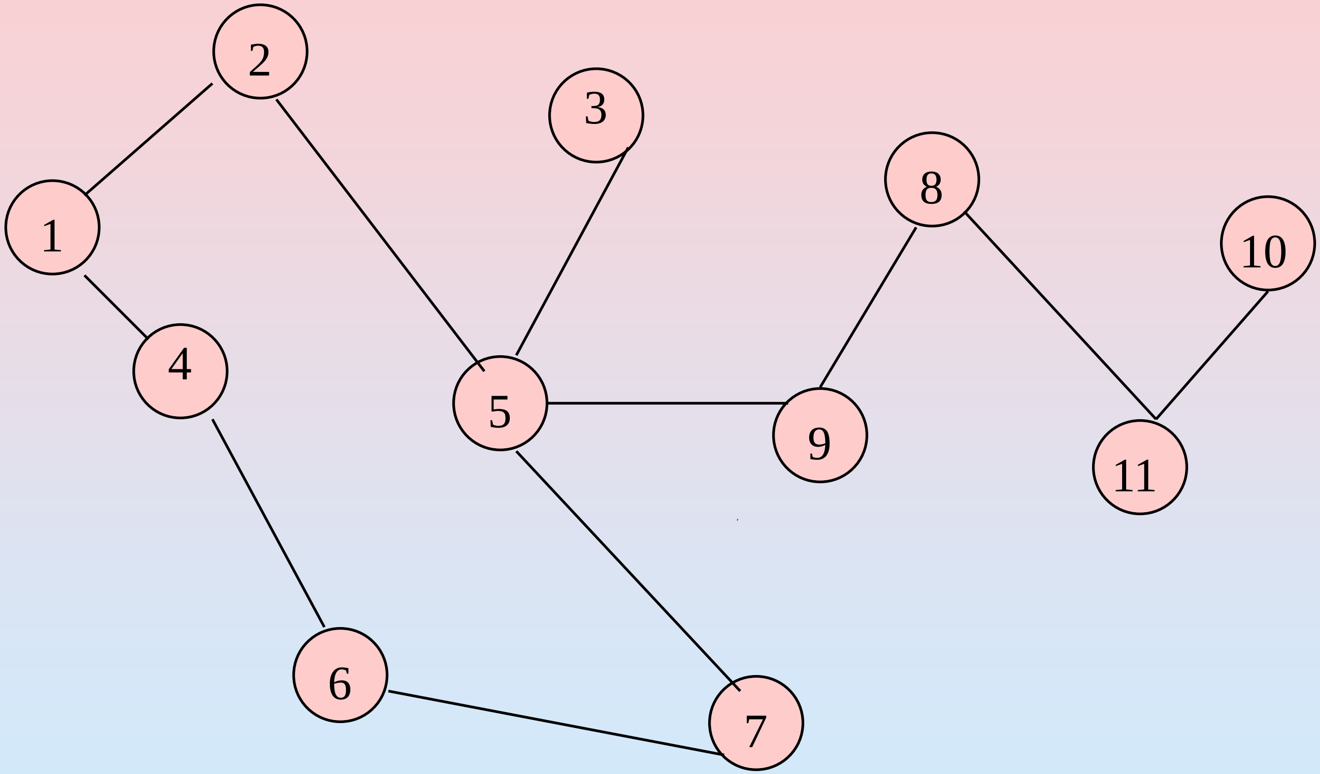
Connected Graph

- Undirected graph.
- There is a path between every pair of vertices.

Example of Not Connected



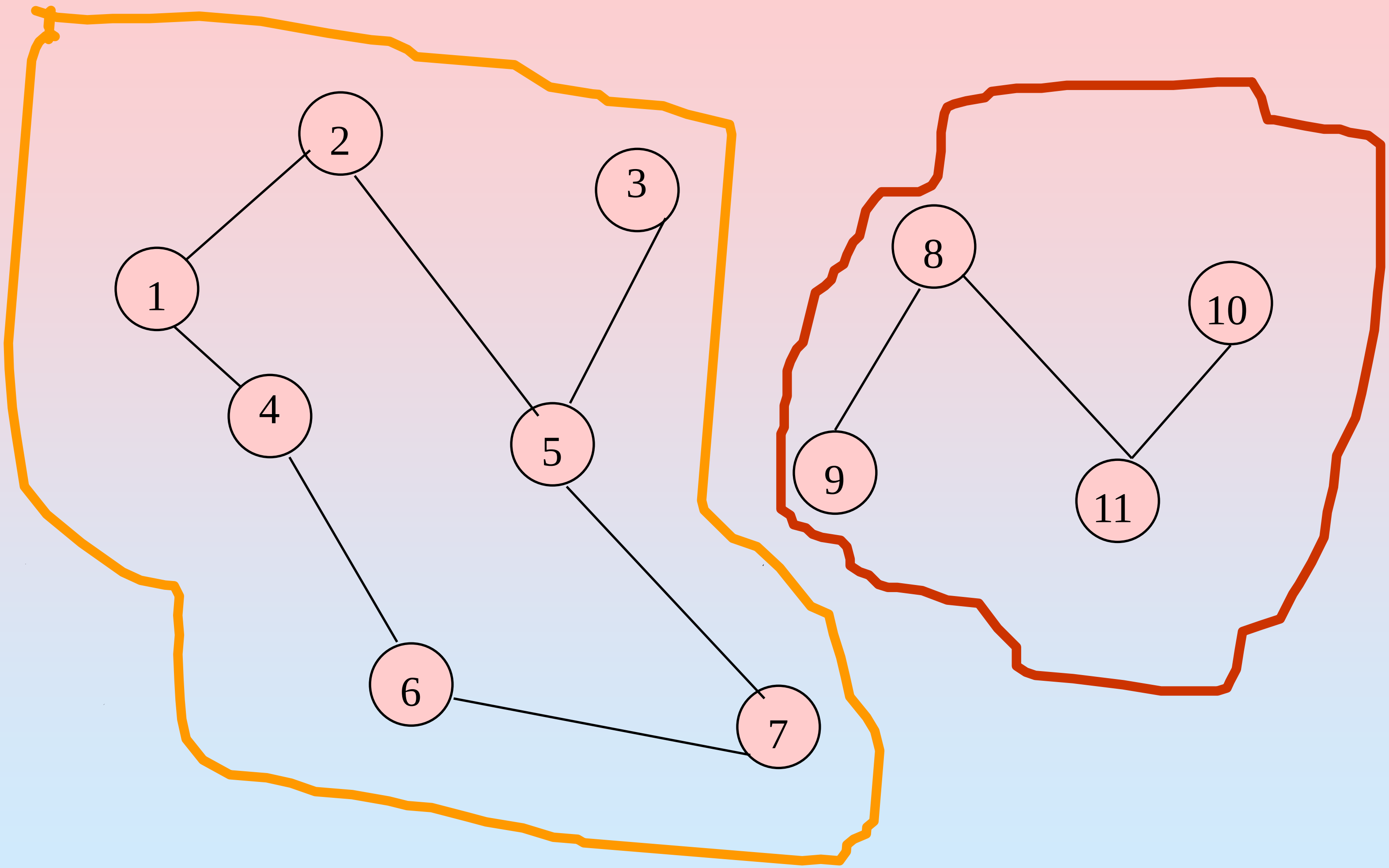
Connected Graph Example



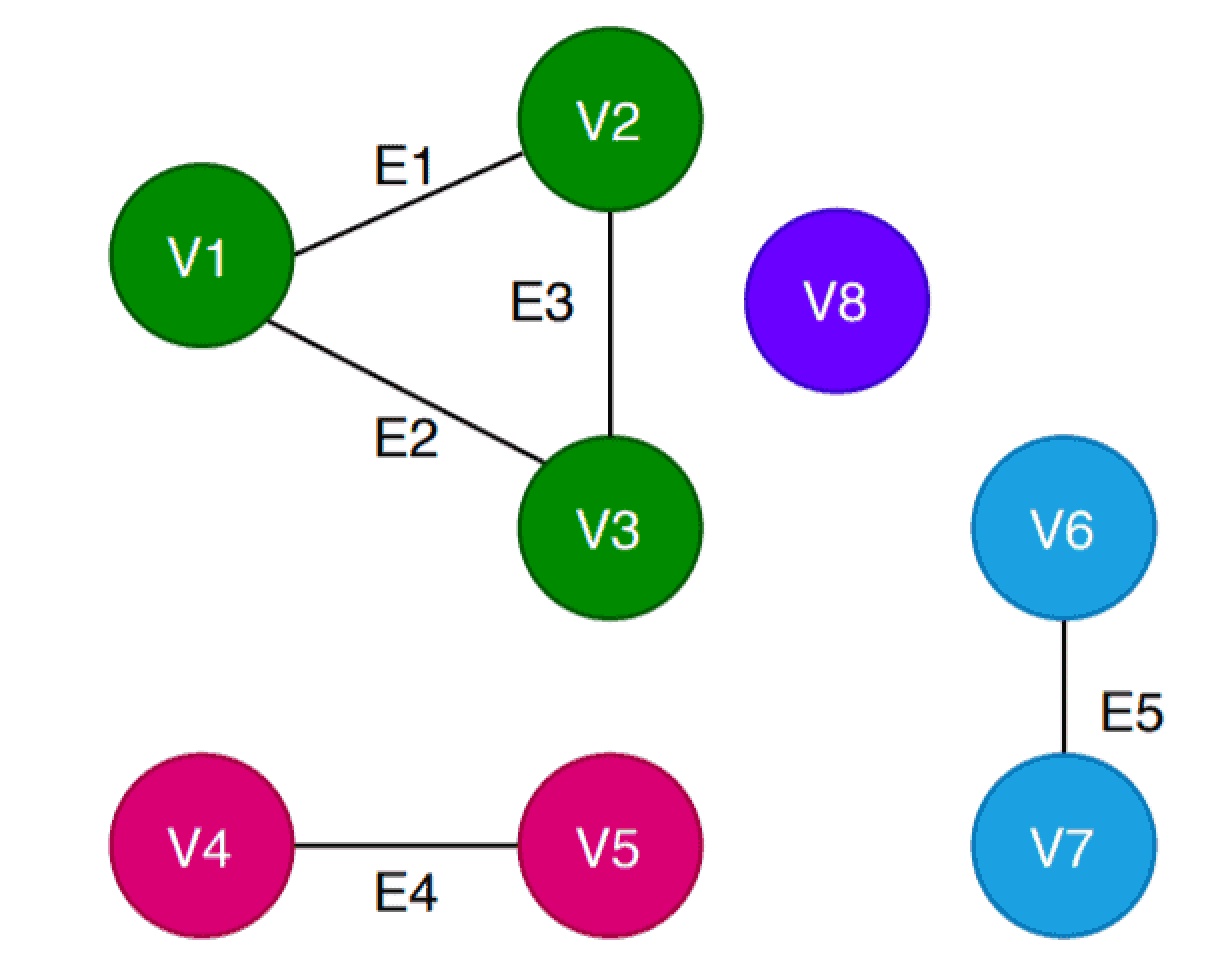
Connected Component

- **A connected component is a set of vertices in a graph that are connected to each other.**
- **Inside a component, each vertex is reachable from every other vertex in that component.**
- **A maximal subgraph that is connected.**
- **A graph can have multiple connected components.**
- **A connected graph has exactly 1 component.**

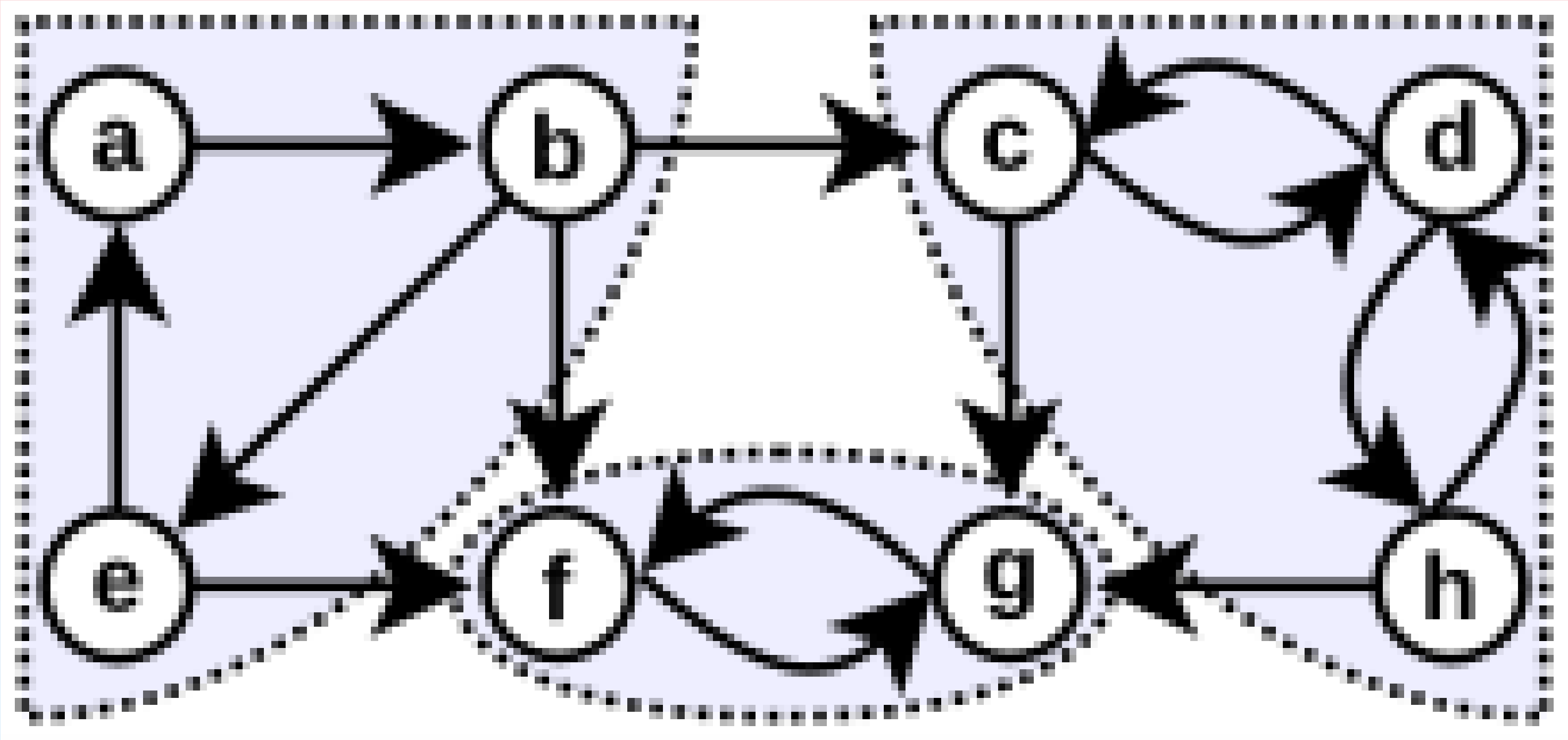
Connected components in the Graph - Example



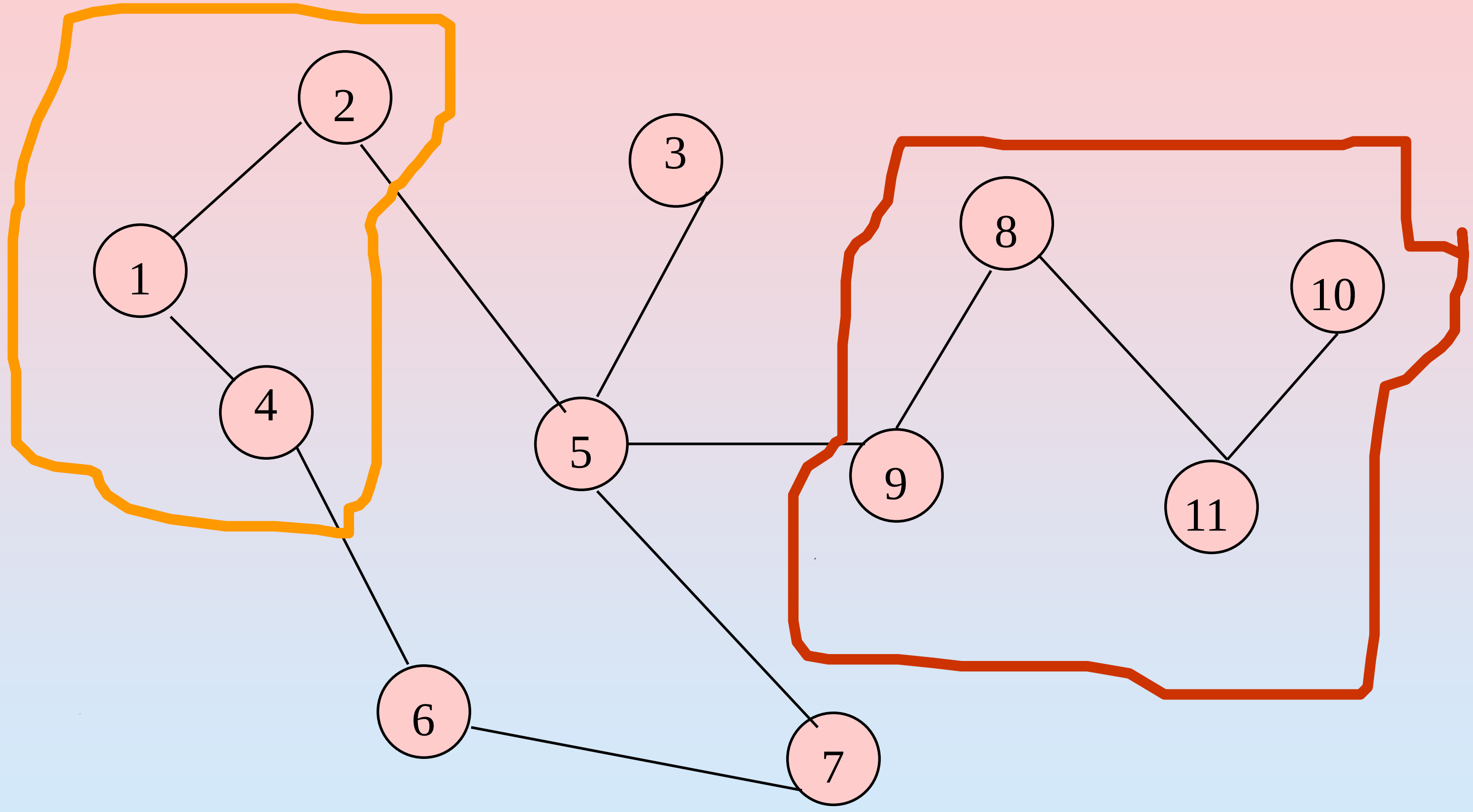
Connected components in the Graph - Example



Connected components in a Directed Graph - Example



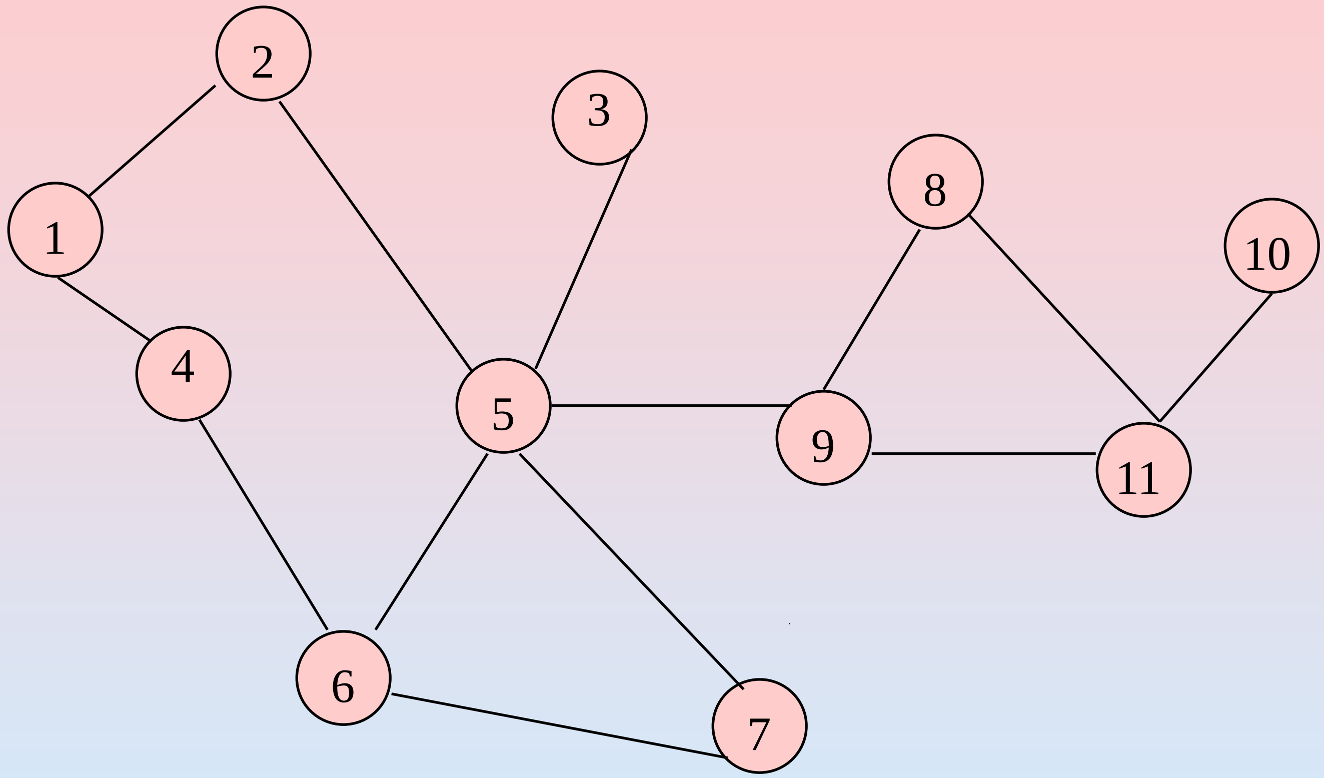
Not a Component



Applications of Connected Component

- **Graph Theory:** It is used to find subgraphs or clusters of nodes that are connected to each other.
- **Computer Networks:** It is used to discover clusters of nodes or devices that are linked and have similar qualities, such as bandwidth.
- **Image Processing:** Connected components also have usage in automated image analysis applications.

Communication Network

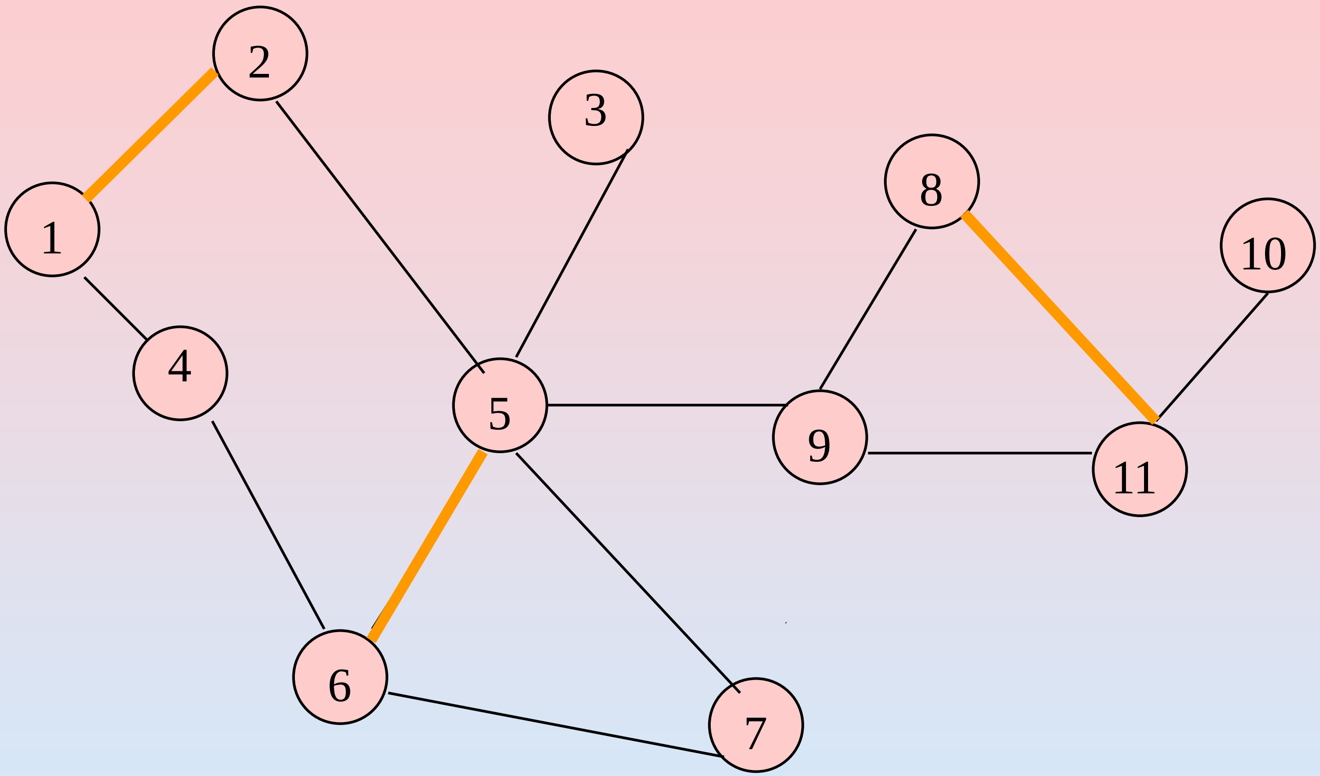


Each edge is a link that can be constructed (i.e., a feasible link).

Communication Network Problems

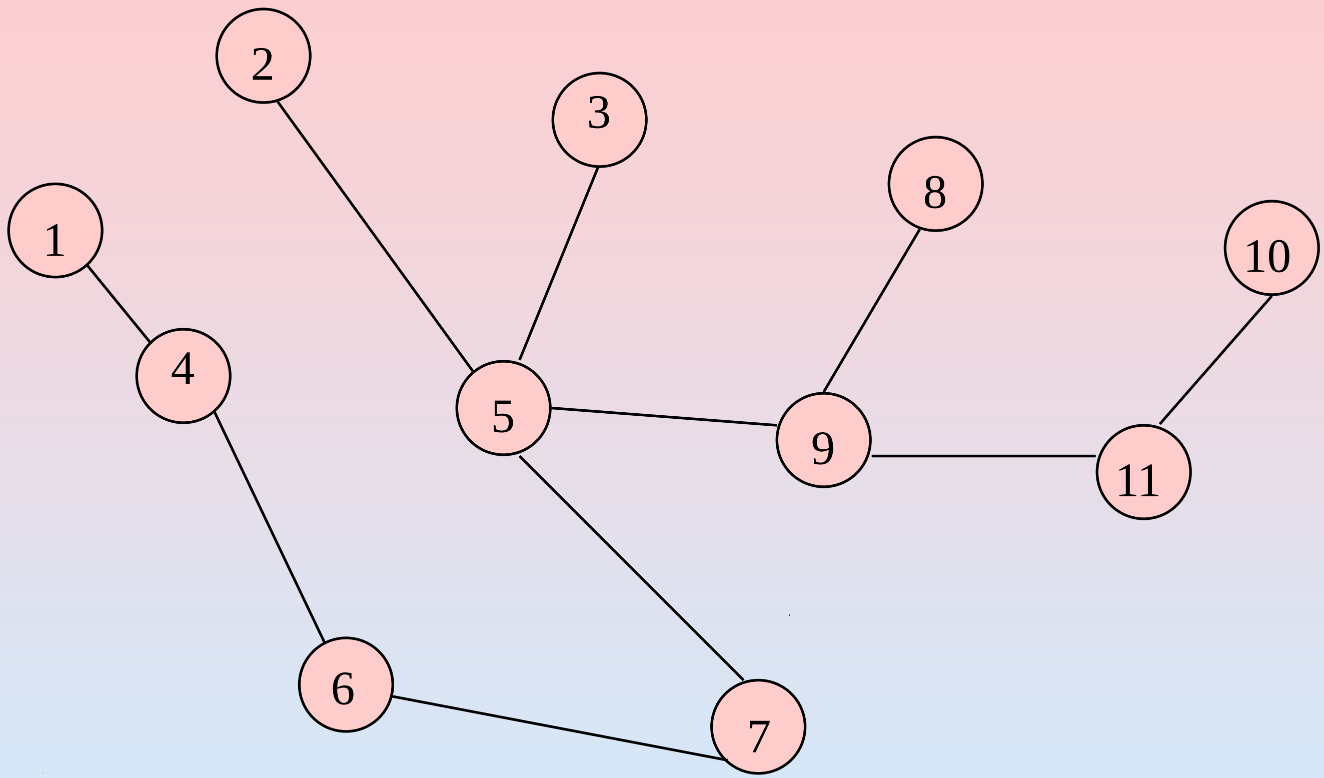
- **Is the network connected?**
  - **Can we communicate between every pair of cities?**
- **Find the components.**
- **Want to construct smallest number of feasible links so that resulting network is connected.**

Cycles and Connectedness



Removal of an edge that is on a cycle does not affect connectedness.

Cycles and Connectedness



Connected subgraph with all vertices and minimum number of edges has no cycles.

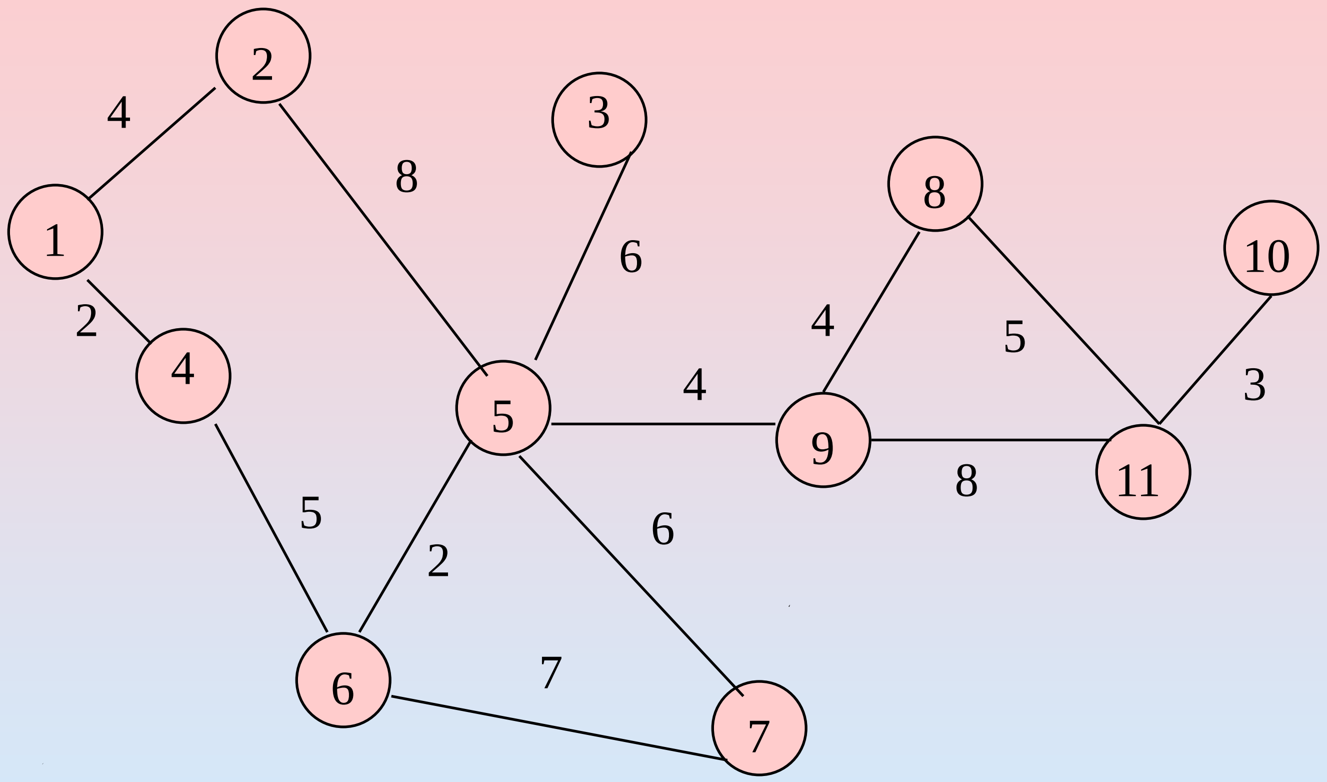
Tree

- Connected graph that has no cycles.
- **n** vertex connected graph with **n-1** edges.

Spanning Tree

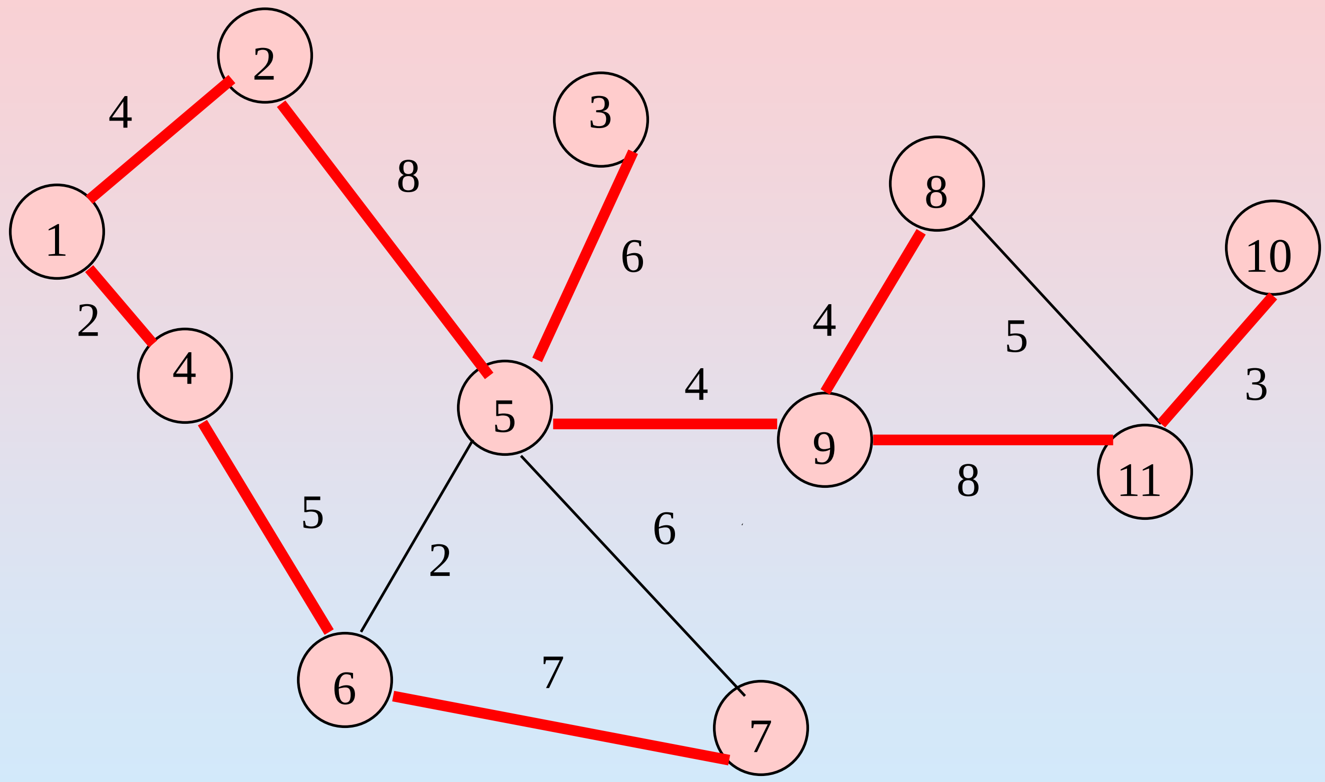
- Subgraph that includes all vertices of the original graph.
- Subgraph is a tree.
  - If original graph has n vertices, the spanning tree has n vertices and n-1 edges.

Minimum Cost Spanning Tree



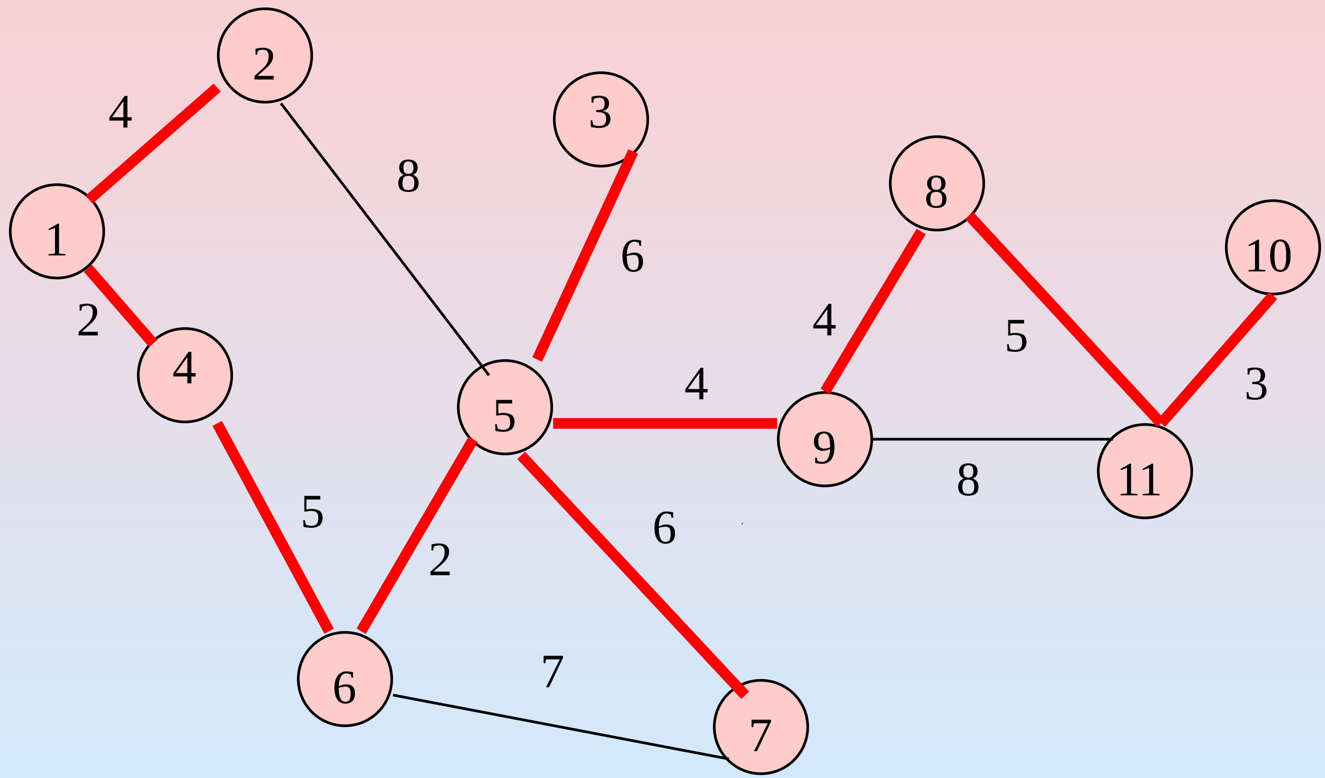
❖ **Tree cost is sum of edge weights/costs.**

A Spanning Tree



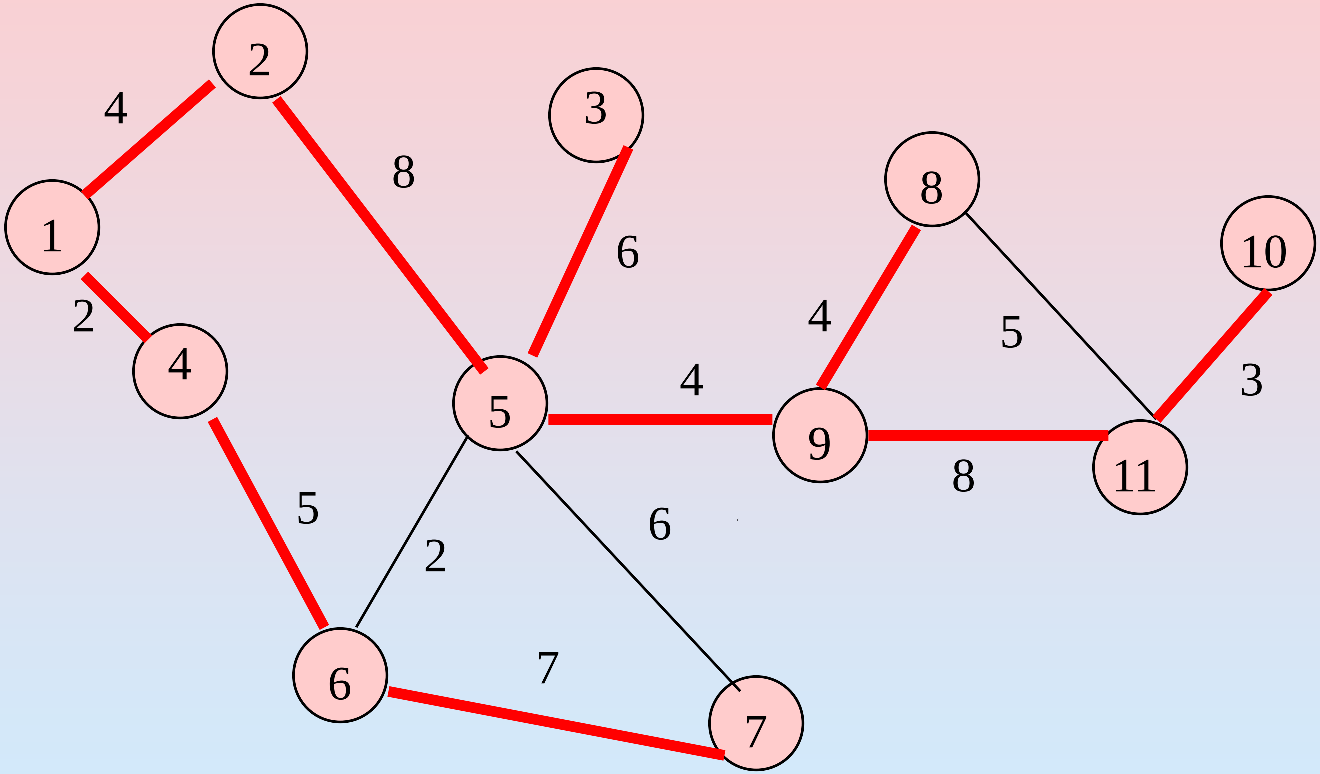
Spanning tree cost = 51.

Minimum Cost Spanning Tree



Spanning tree cost = 41.

A Wireless Broadcast Tree



Source = 1, weights = needed power.

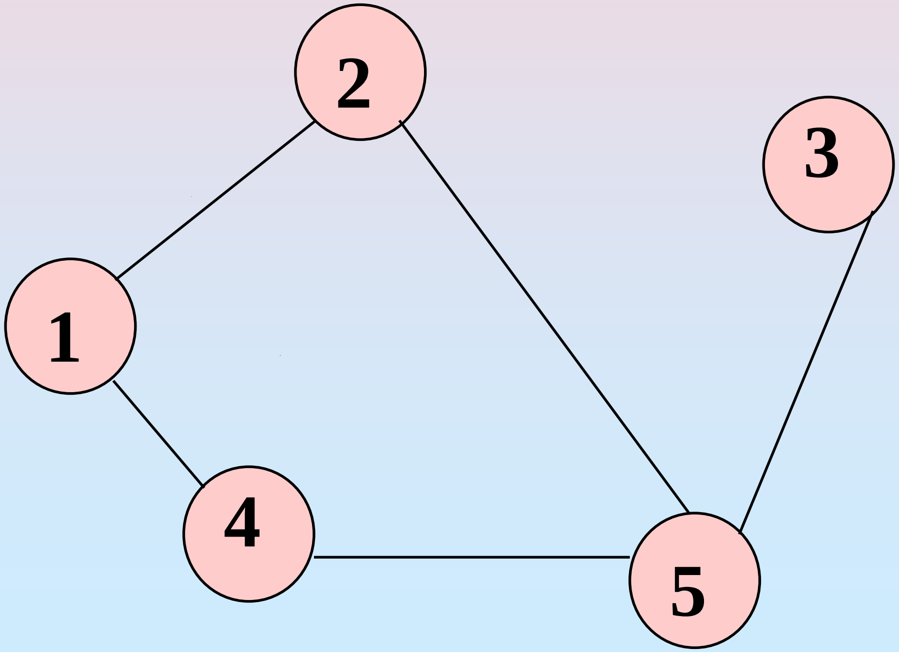
Cost = 4 + 8 + 5 + 6 + 7 + 8 + 3 = 41.

Graph Representation

- **Adjacency Matrix**
- **Adjacency Lists**
  - **Linked Adjacency Lists**
  - **Array Adjacency Lists**

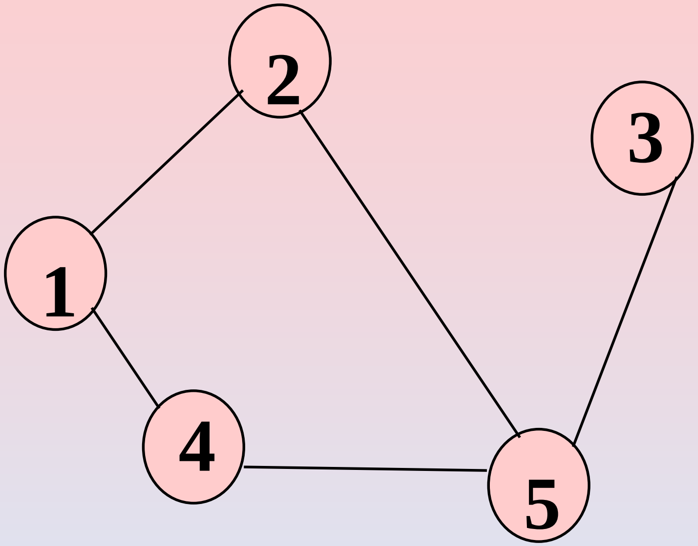
Adjacency Matrix

- **0/1 n x n matrix, where n is number of vertices**
- **A(i,j) = 1 iff (i, j) is an edge**



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

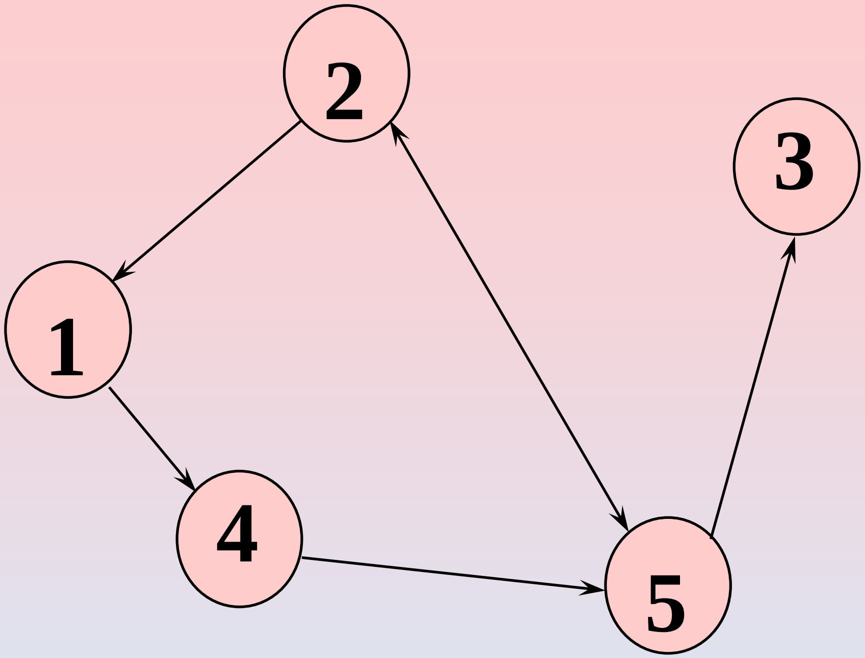
Adjacency Matrix Properties



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

- Diagonal entries are zero.
- Adjacency matrix of an undirected graph is symmetric.
  - A(i,j) = A(j,i) for all i and j.

Adjacency Matrix (Digraph)



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

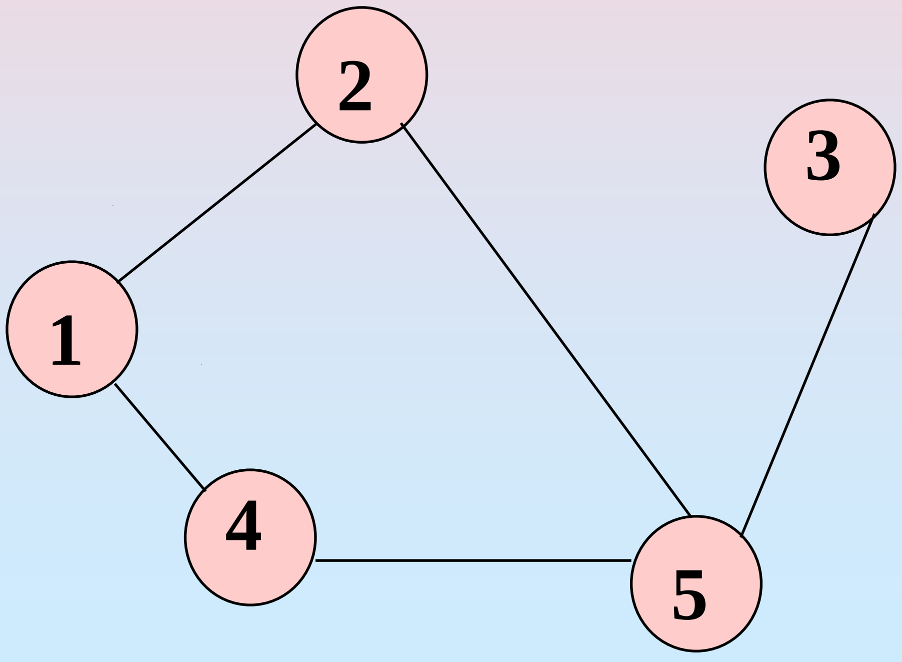
- Diagonal entries are zero.
- Adjacency matrix of a digraph need not be symmetric.

Adjacency Matrix

- **n^2 bits of space**
- **For an undirected graph, may store only lower or upper triangle (exclude diagonal).**
  - **$(n-1)n/2$ bits**
- **$O(n)$ time to find vertex degree and/or vertices adjacent to a given vertex.**

Adjacency Lists

- Adjacency list for vertex i is a linear list of vertices adjacent from vertex i.
- An array of n adjacency lists.



aList[1] = (2,4)

aList[2] = (1,5)

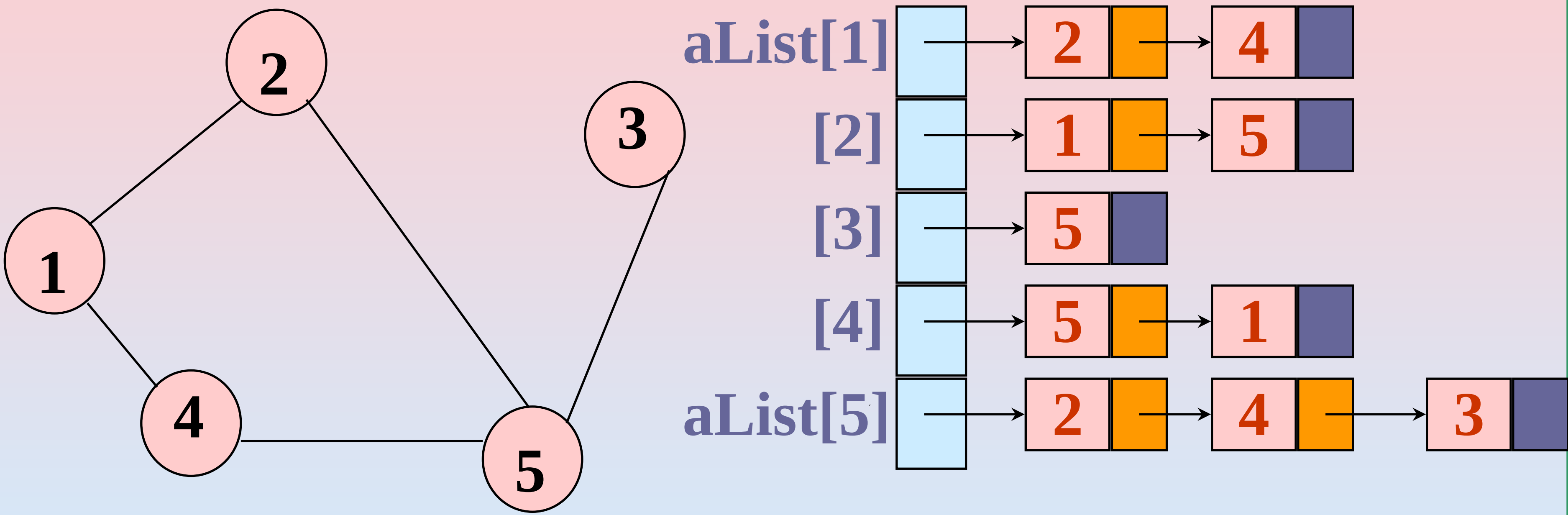
aList[3] = (5)

aList[4] = (5,1)

aList[5] = (2,4,3)

Linked Adjacency Lists

- Each adjacency list is a chain.



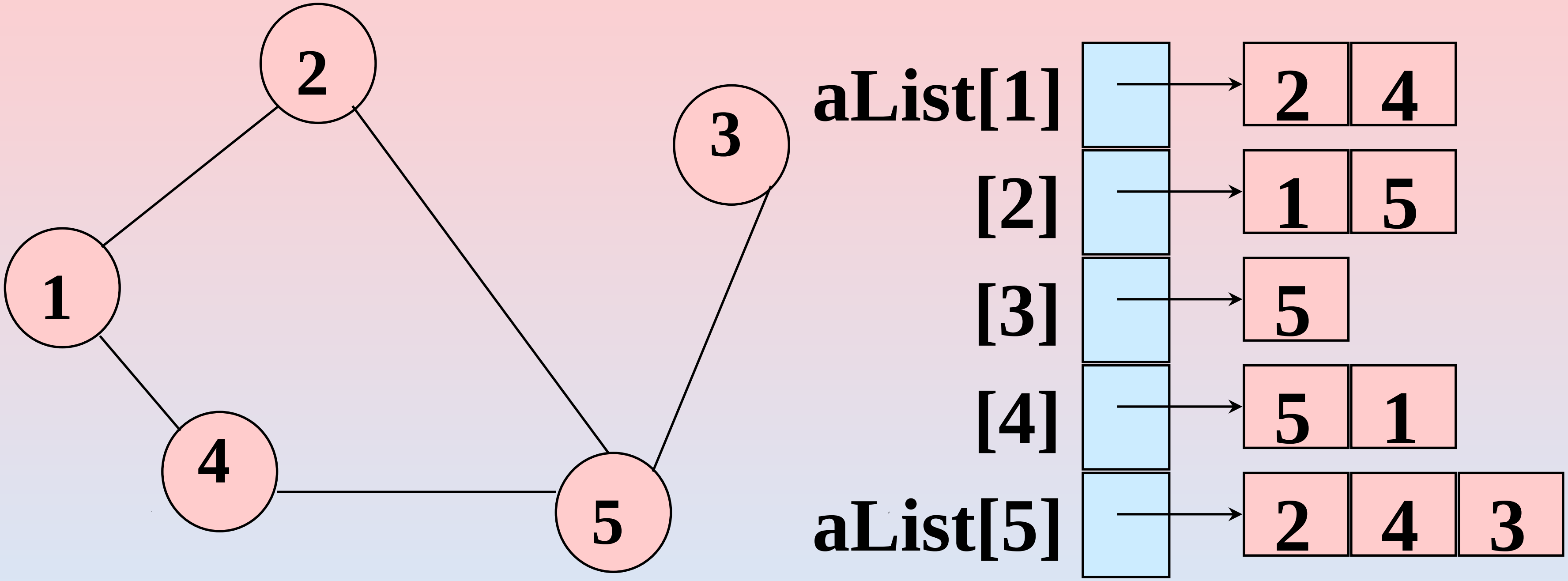
Array Length = n

# of chain nodes = 2e (undirected graph)

# of chain nodes = e (digraph)

Array Adjacency Lists

- Each adjacency list is an array list.



Array Length = n

# of list elements = 2e (undirected graph)

# of list elements = e (digraph)

Weighted Graphs

- **Cost adjacency matrix.**
  - **C(i, j) = cost of edge (i, j)**
- **Adjacency lists => each list element is a pair (adjacent vertex, edge weight)**