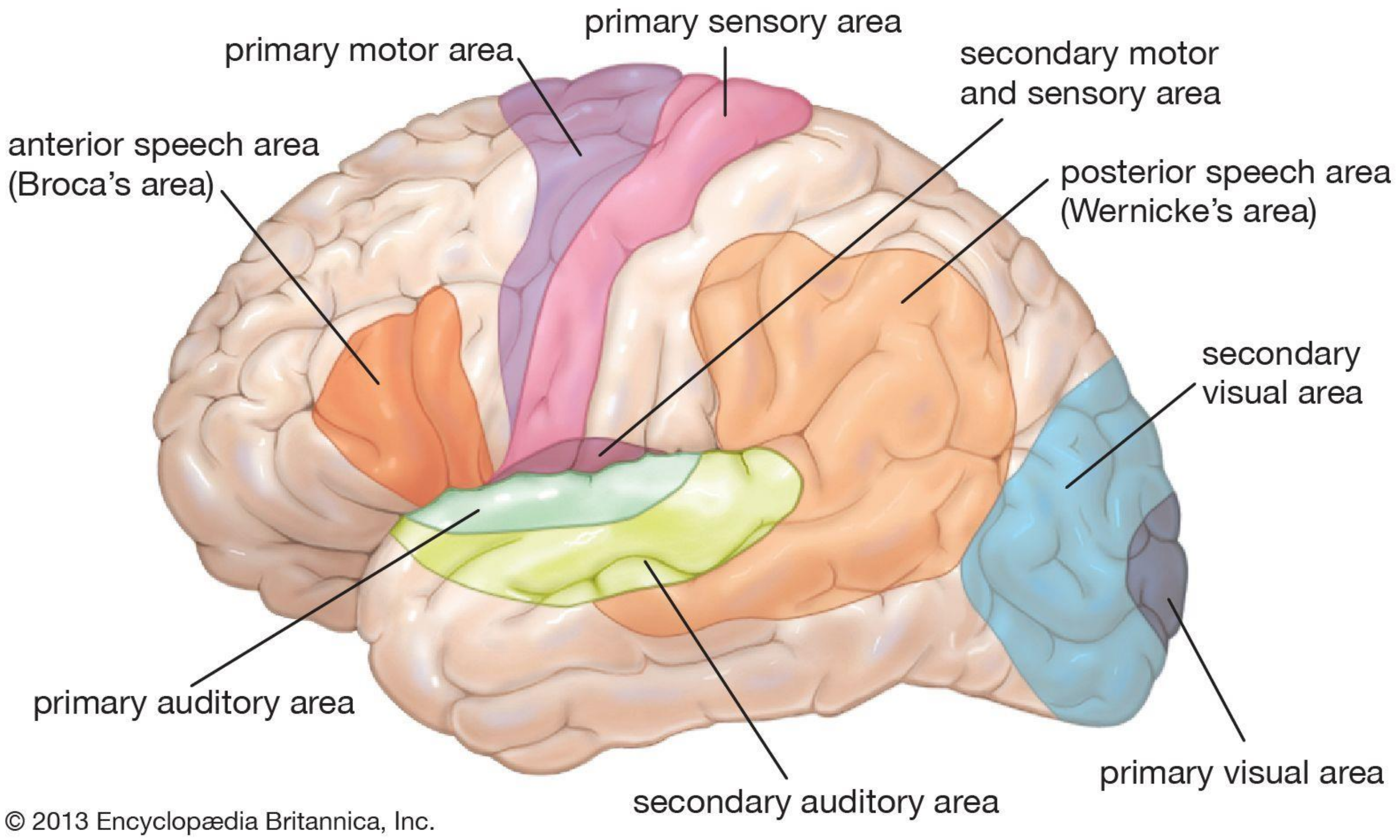
Graphs

Ch. 6

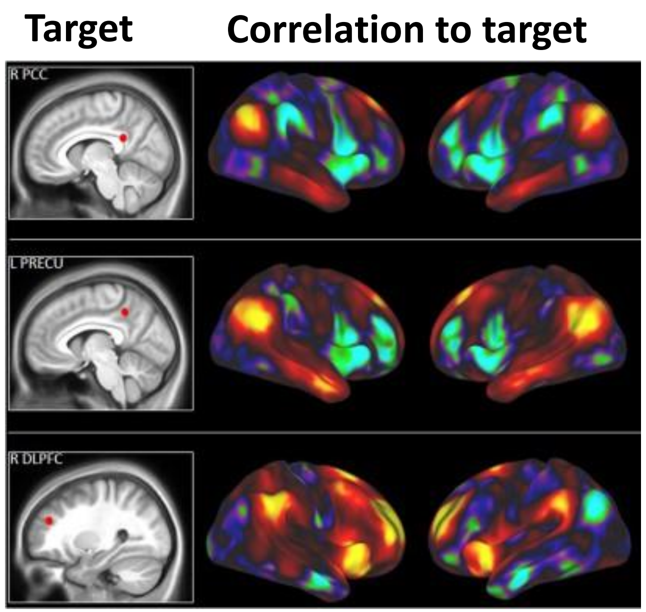
Regions in the brain

- Each region is associated with different functions.

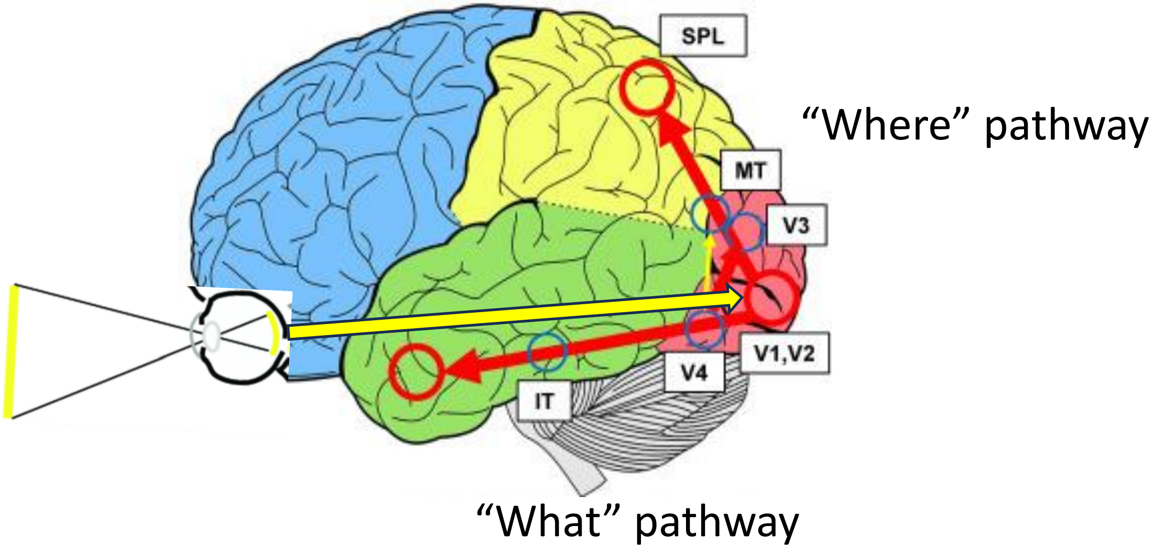


https://www.britannica.com/science/human-nervous-system/Lobes-of-the-cerebral-cortex

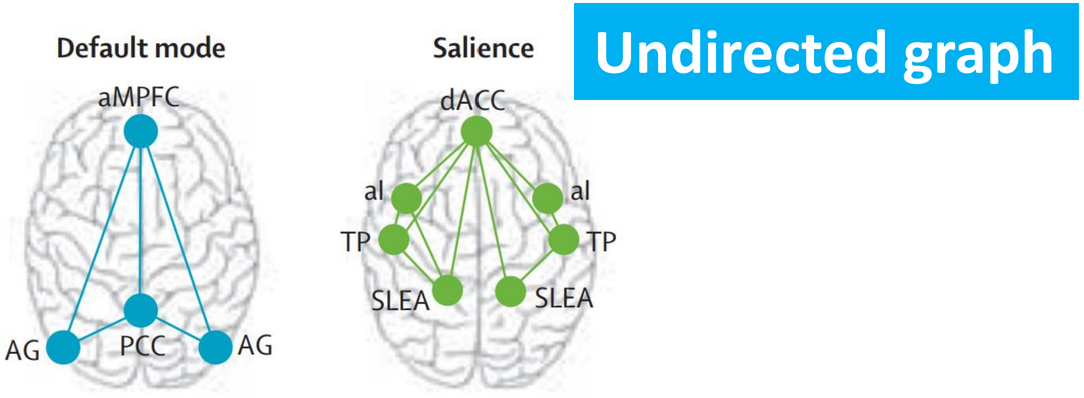
Connections between regions



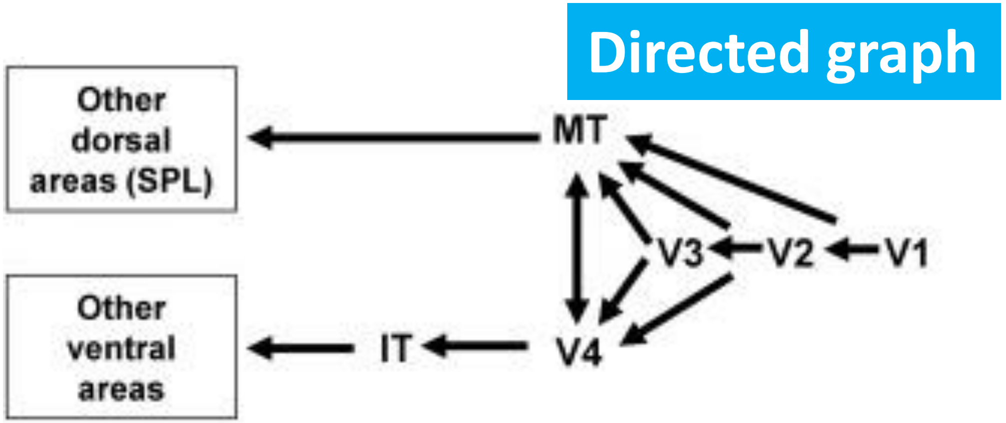
[Beaty et al., Scientific Reports 2015; 5: 10964]



- Using graphs to represent the connections for visualization and further analysis.



[Leanne M Williams, Lancet Psychiatry 2016; 3: 472–80]



[Choi et al., NeuroImage 2020; 220: 117145]

Graphs

$$G = (V, E)$$

- *V*: Set of vertices
  - Vertices are also called nodes or points.
- *E*: Set of edges
  - Each edge connects two different vertices.
  - Edges are called arcs and lines.
  - In **undirected** graph, (u, v) and (v, u) represent the same edge.

u —— v *Edge without orientation*

(v, u) or (u, v)

  - In **directed** graph, $<u, v>$ and $<v, u>$ represent different edges.

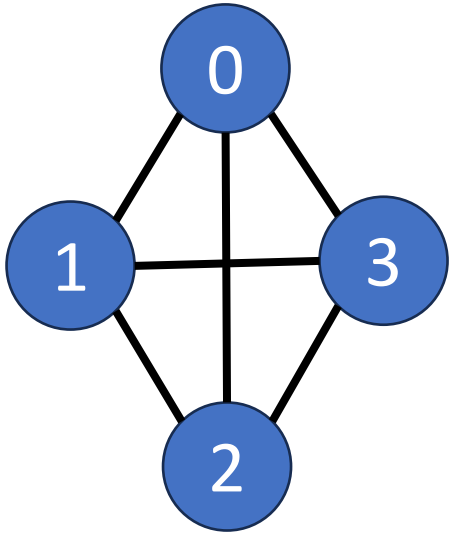
$u \longrightarrow v$ $u \longleftarrow v$ *Edge with orientation*

$<u, v>$ $<v, u>$

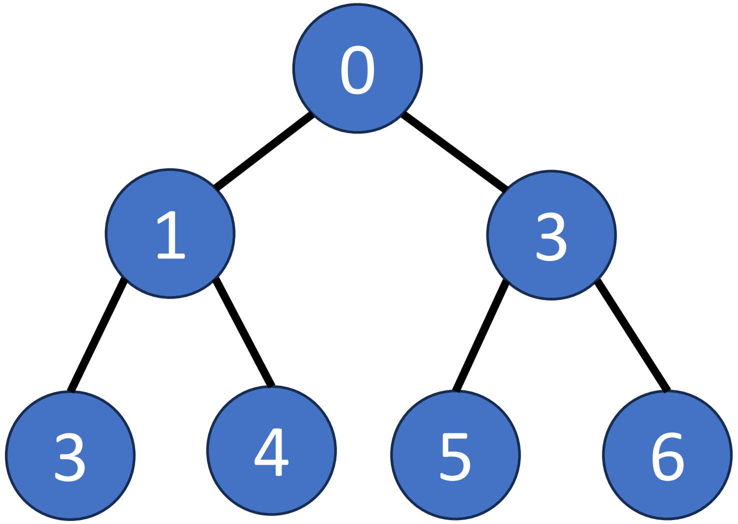
Sample graphs

Undirected graph

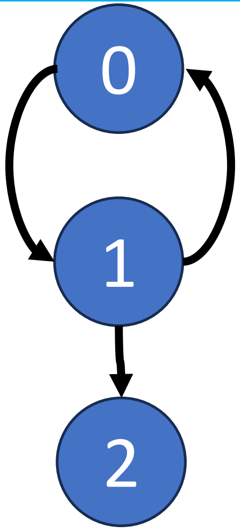
Directed graph (Digraph)



$\boldsymbol{G}_1$



$\boldsymbol{G}_2$



$\boldsymbol{G}_3$

Set representation:

$V(G_1) = \{0,1,2,3\}$
$E(G_1) = \{(0,1),(1,2),(2,3),(0,3),(0,2),(1,3)\}$

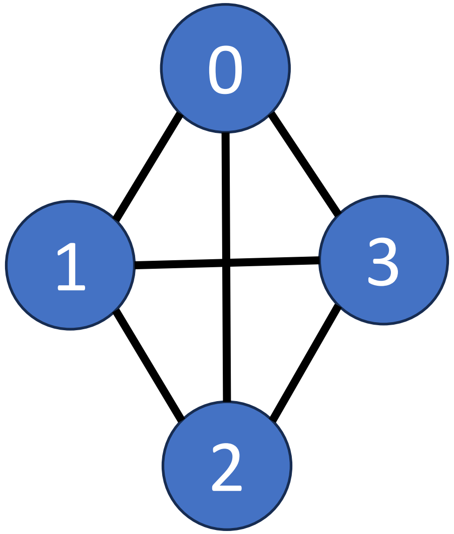
Tree is a type of graph.

Set representation:

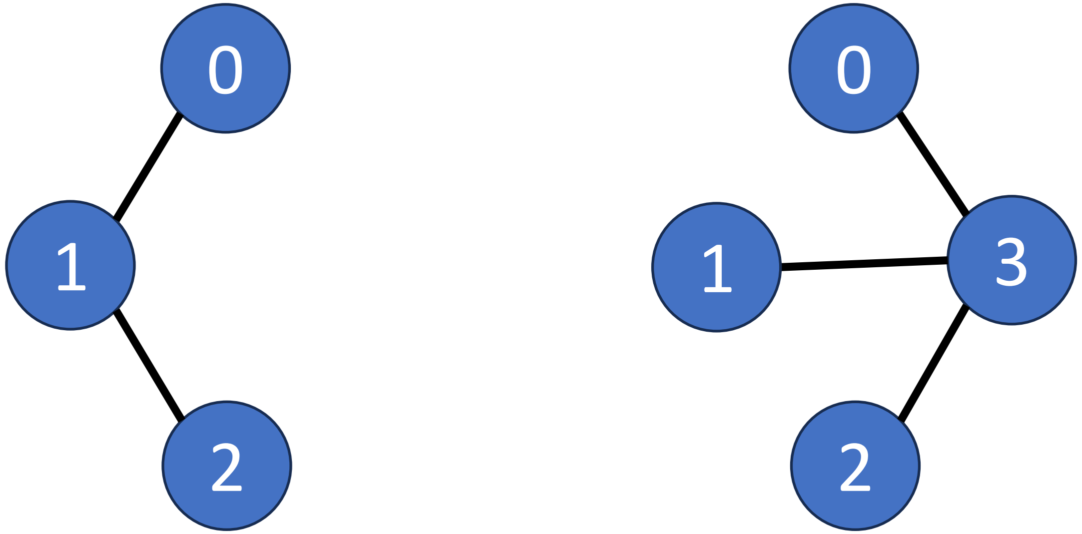
$V(G_3) = \{0,1,2\}$
$E(G_3) = \{\langle 0,1\rangle,\langle 1,0\rangle,\langle 1,2\rangle\}$

Terminology: subgraph

- Subgraph of G: a graph is composed of a subset of vertices G and a subset of edges E.



$\boldsymbol{G}_1$



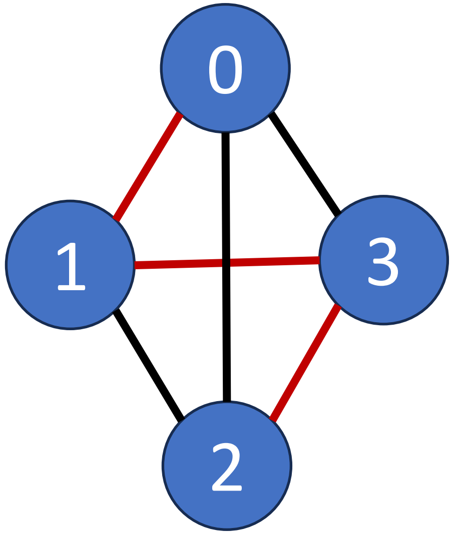
Some subgraphs of $\boldsymbol{G}_1$

Terminology: path

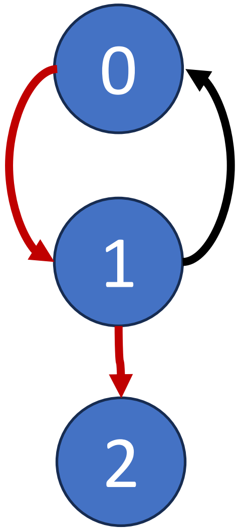
- Path
  - <u>Length</u> of a path is the number of edges on it.

(0,1),(1,3),(3,2) or 0,1,3,2 is a path from 0 to 2 in G_1.
The length of path is 3.

<0,1>,<1,2> or 0,1,2 is a path from 0 to 2 in G_3.
The length of path is 2.



G_1



G_3

Terminology: simple path and cycle

- Simple path: No repeating vertices (except for the case that the 1st and the last are the same).

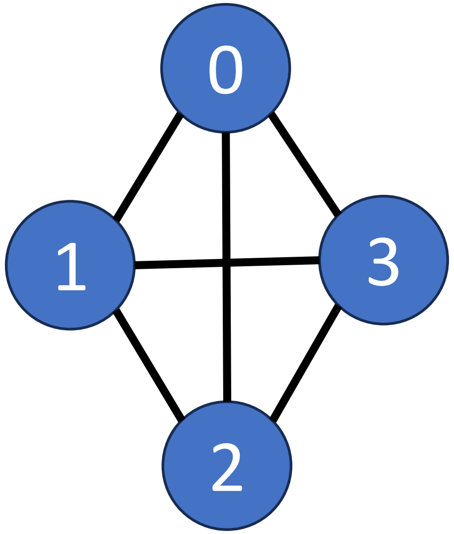
(0,1),(1,3),(3,2) or 0,1,3,2. Simple path

(0,1),(1,3),(3,1) or 0,1,3,1. Not simple path

Repeated vertices

(0,1),(1,2),(2,0) or 0,1,2,0. Simple path Cycle

Repeated vertices located at 1st and last positions

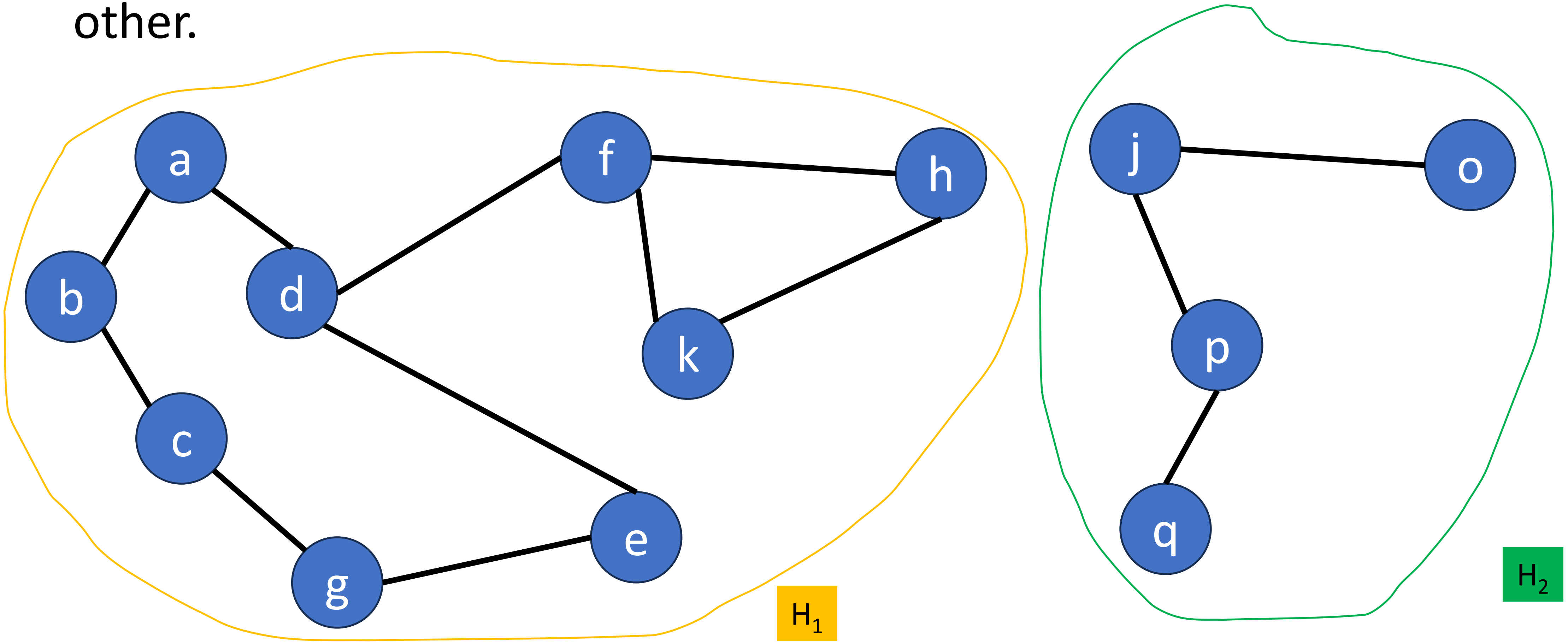


G_1

- Cycle:
  - Simple path
  - 1st and the last vertices are the same.

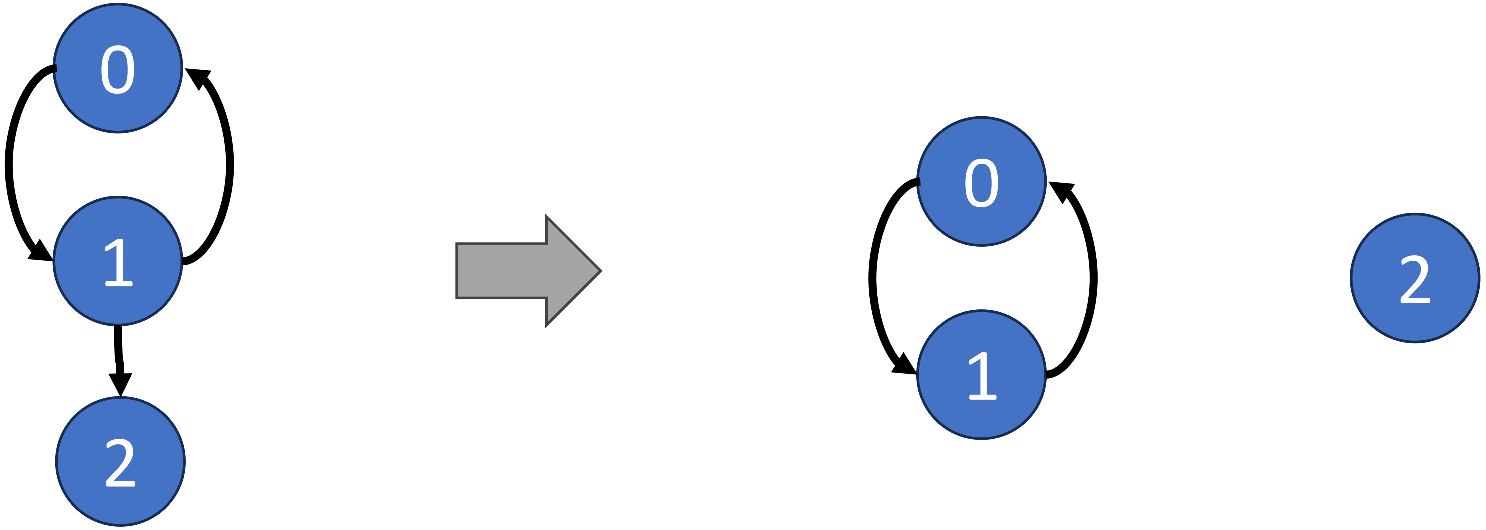
More applications of graphs

- Find connected components:
  - Two vertices *u* and *v* are connected iff there is a path from *u* to *v*.
  - Connected component: a maximum subgraph that are connected to each other.



Strongly connected components

- In a directed graph, every pair of vertices u and v has a directed path from u to v and also from v to u.



$\boldsymbol{G}_3$

Two strongly connected components

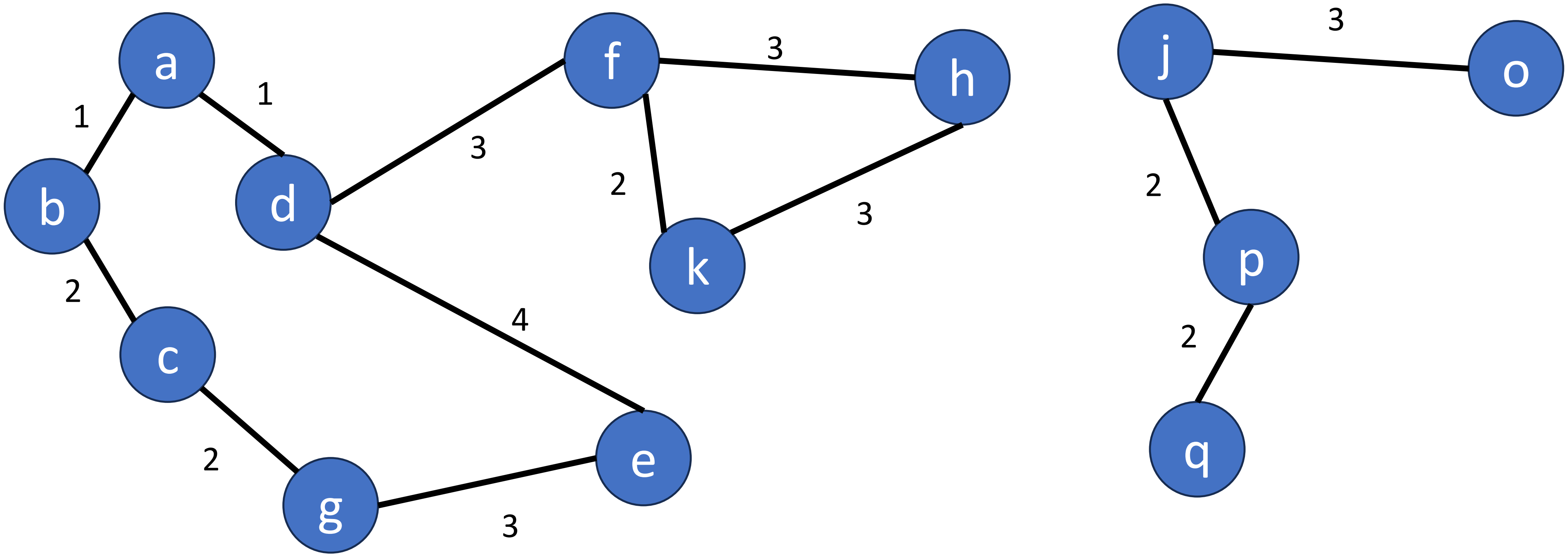
More applications of graphs

- Plan the route from city *b* to city *k*.

    Vertex = city

    Edge = roads

    Edge weight = distance or time

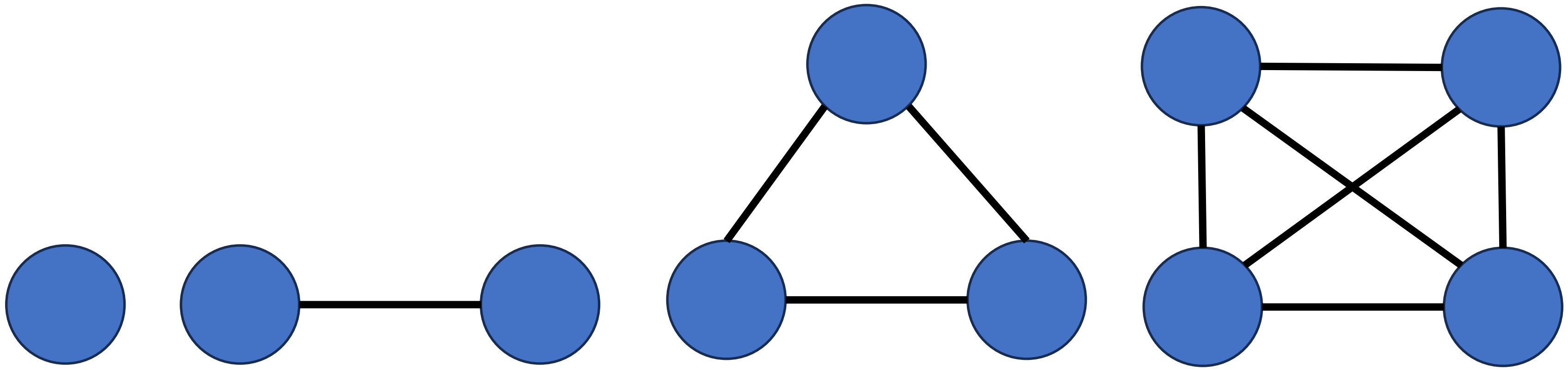


Complete graph

- Having the maximum number of edges.

n = 1 n = 2 n = 3 n = 4

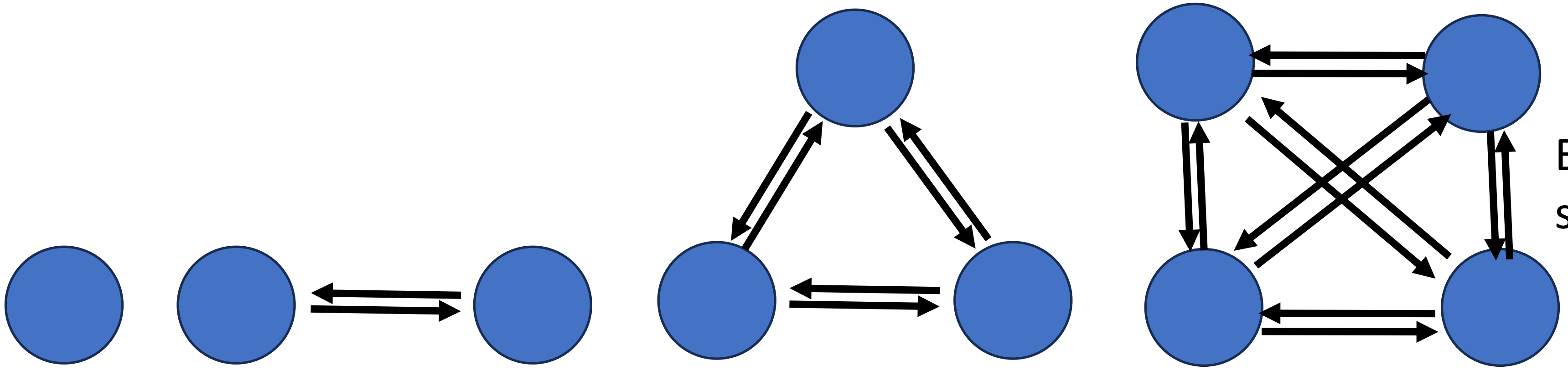
Undirected graph



Edge (u,v) is the same as edge (v,u).

Number of edges 0 1 3 6 ➡ $\frac{n(n-1)}{2}$

Digraph



Edge (u,v) is **not** the same as edge (v,u).

Number of edges 0 2 6 12 ➡ $n(n-1)$

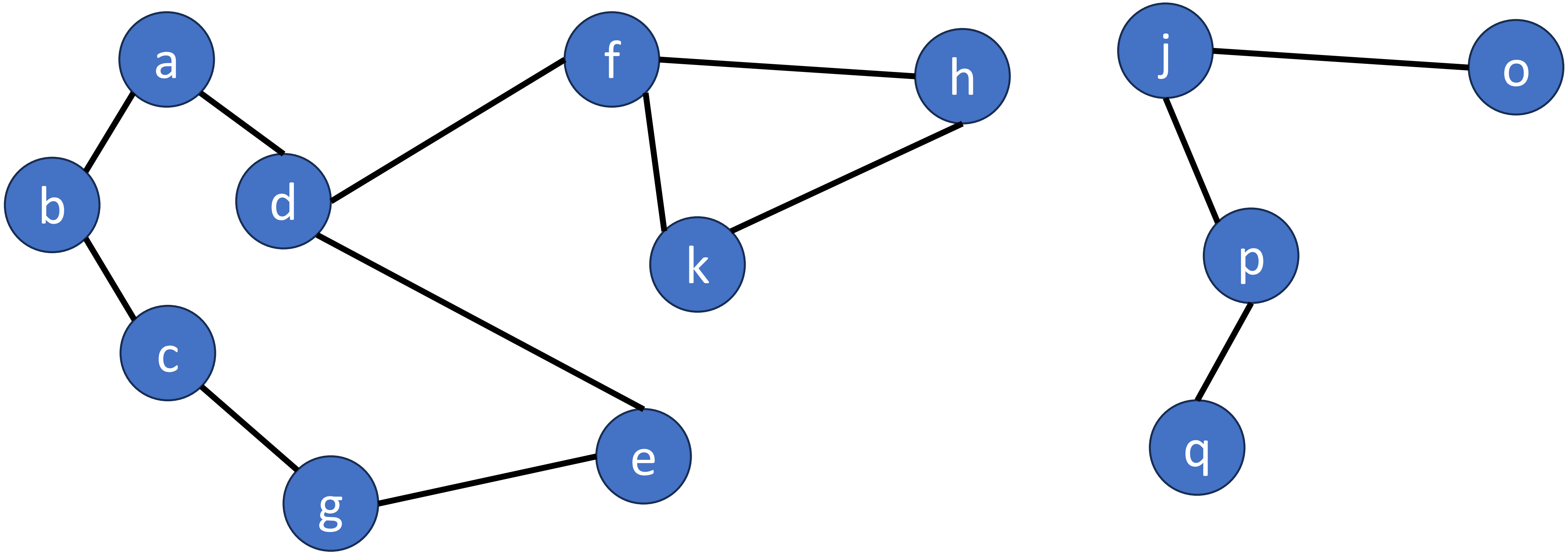
Property: Number of edges

n: number of vertices

- Number of edges in an undirected graph is $\leq n(n\text{-}1)/2$.
- Number of edges in a directed graph is $\leq n(n\text{-}1)$.

Property: Vertex degree

- Number of edges incident to vertex.

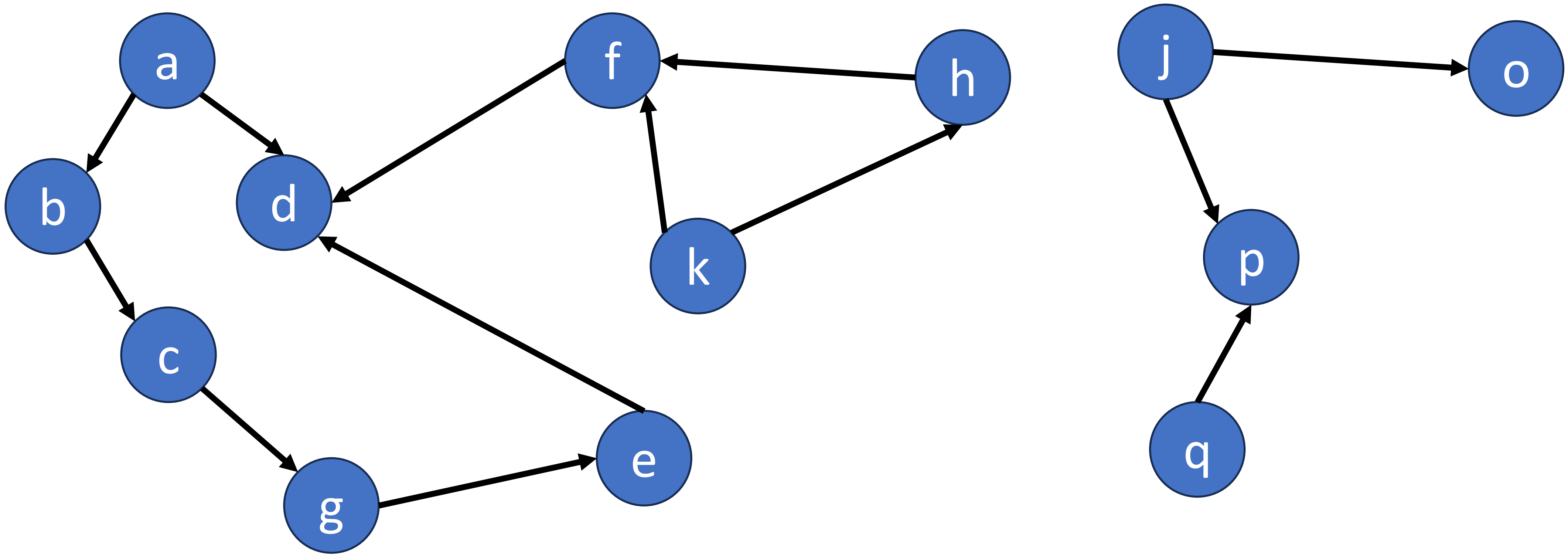


Example:

degree(d) = 3, degree(f) = 3, degree(h) = 2, degree(q) = 1

Property: In-degree of a vertex

- Number of incoming edges.

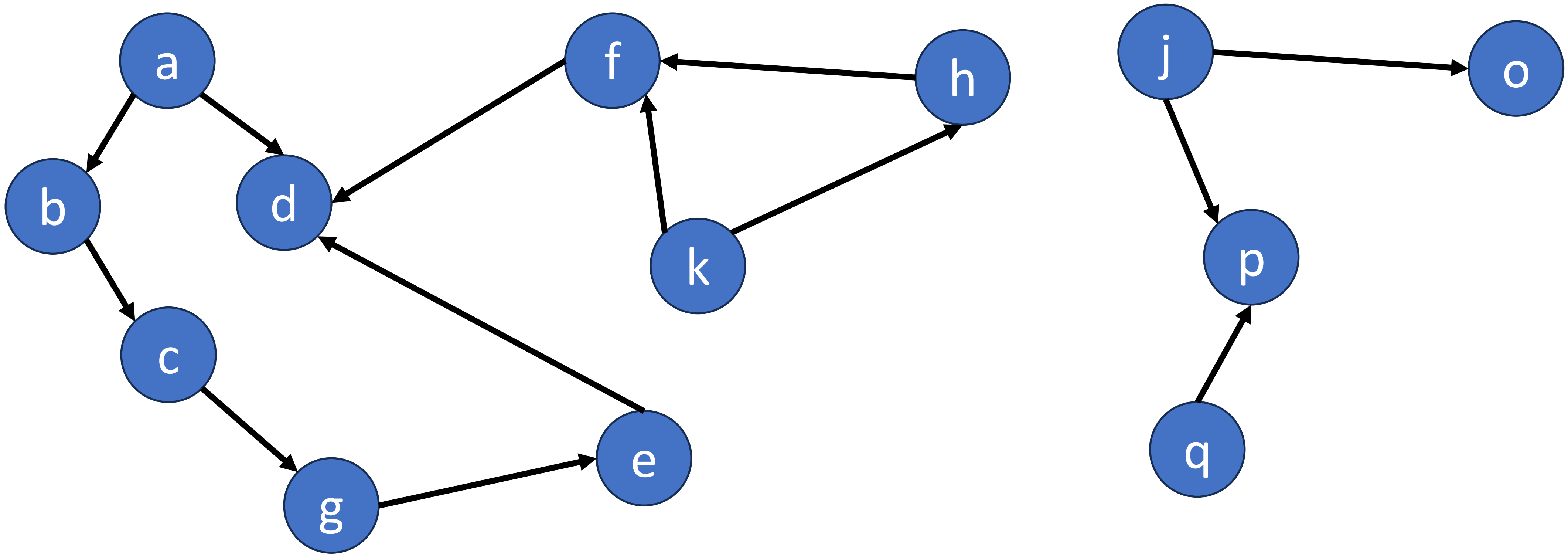


Example:

in-degree(d) = 3, in-degree(f) = 2, in-degree(h) = 1 , in-degree(q) = 0

Property: Out-degree of a vertex

- Number of outbound edges.



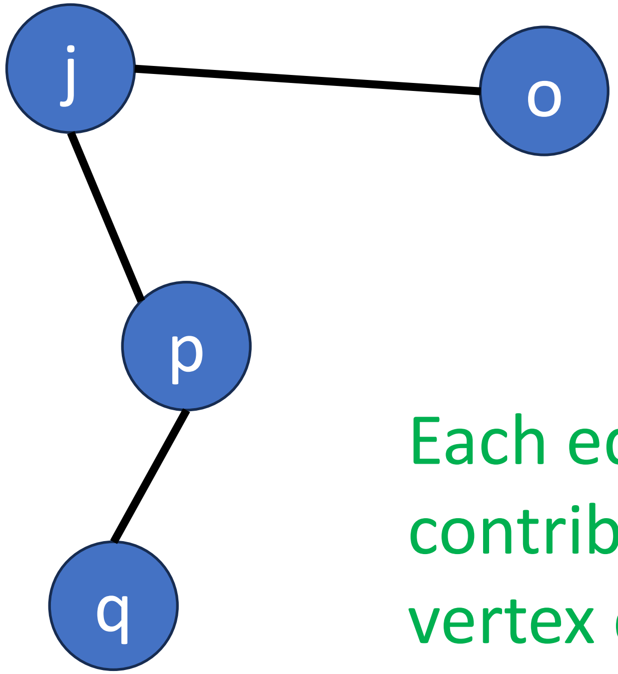
Example:

out-degree(d) = 0, out-degree(f) = 1, out-degree(h) = 1 , out-degree(q) = 1

Sum of degree

- e: number of edges

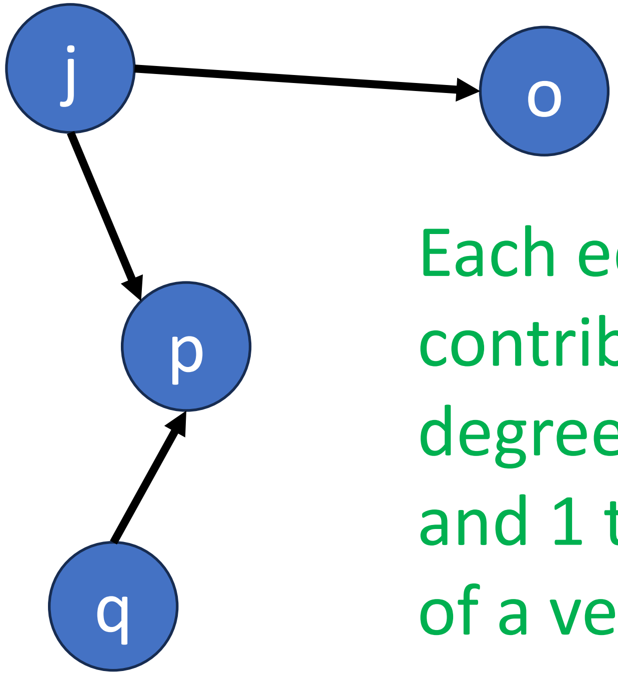
Undirected graph



Each edge contributes 2 to vertex degree.

Sum of vertex degrees = 2 e

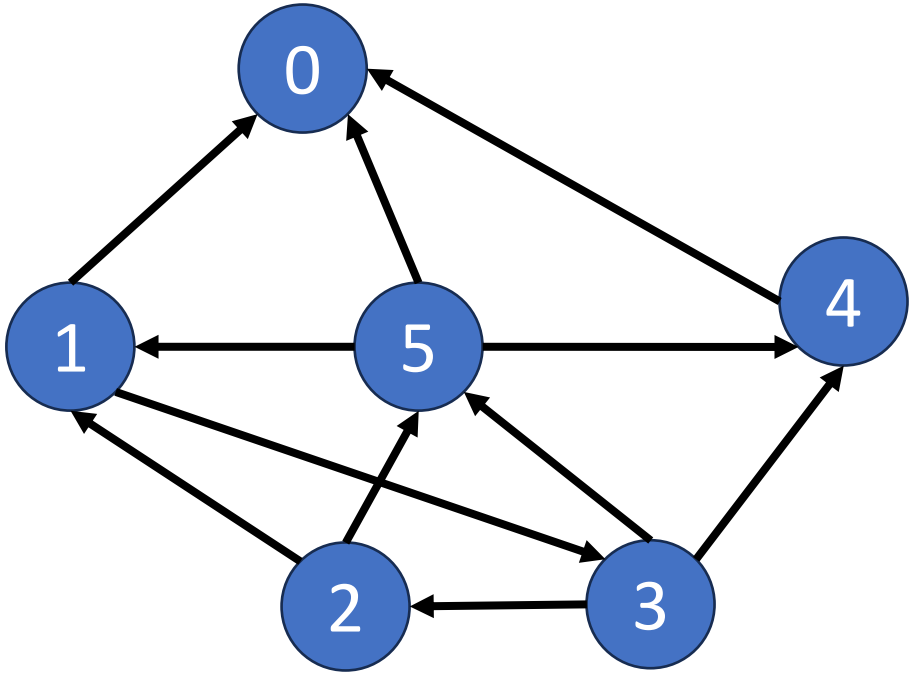
Digraph



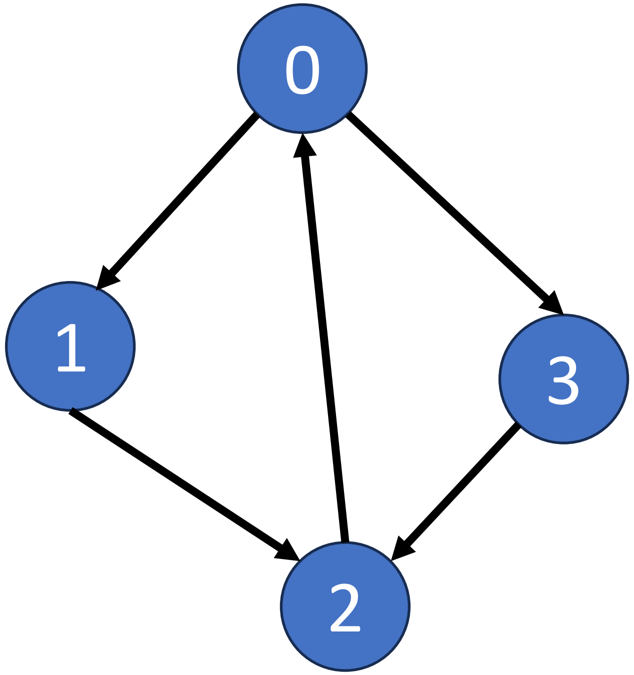
Each edge contributes 1 to in-degree of a vertex and 1 to out-degree of a vertex.

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

Exercise



G1



G2

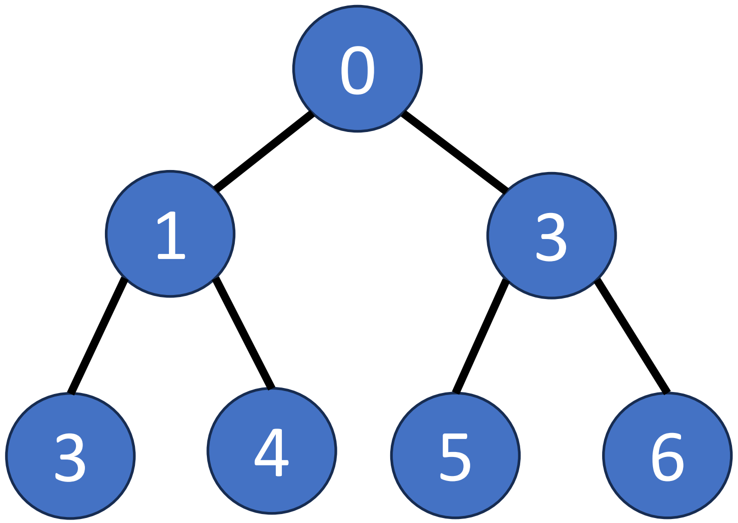
Please reply your answers of Q6 – Q8 via the following link:

Group members: 1~3 people

- Q6: Write out the in-degree of vertices 0, 1, ..., 5 in G1.
- Q7: Write out the out-degree of vertices 0, 1, ..., 5 in G1.
- Q8: Is the directed graph G2 strongly connected? Why?

Tree

- Acyclic graph: a tree has no cycles.
- Connected graph:
  - all pairs of nodes are connected.
  - n vertices connected graph with n-1 edges.

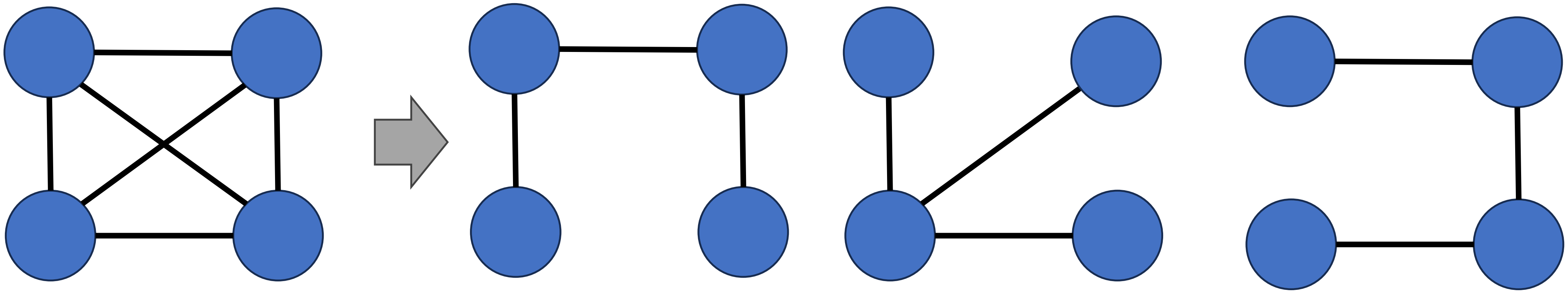


$\boldsymbol{G}_2$

Spanning tree

- A subgraph that includes all vertices of the original graph.
- A tree.

If the original graph has *n* vertices, the spanning tree has *n* vertices and *n*-1 edges.

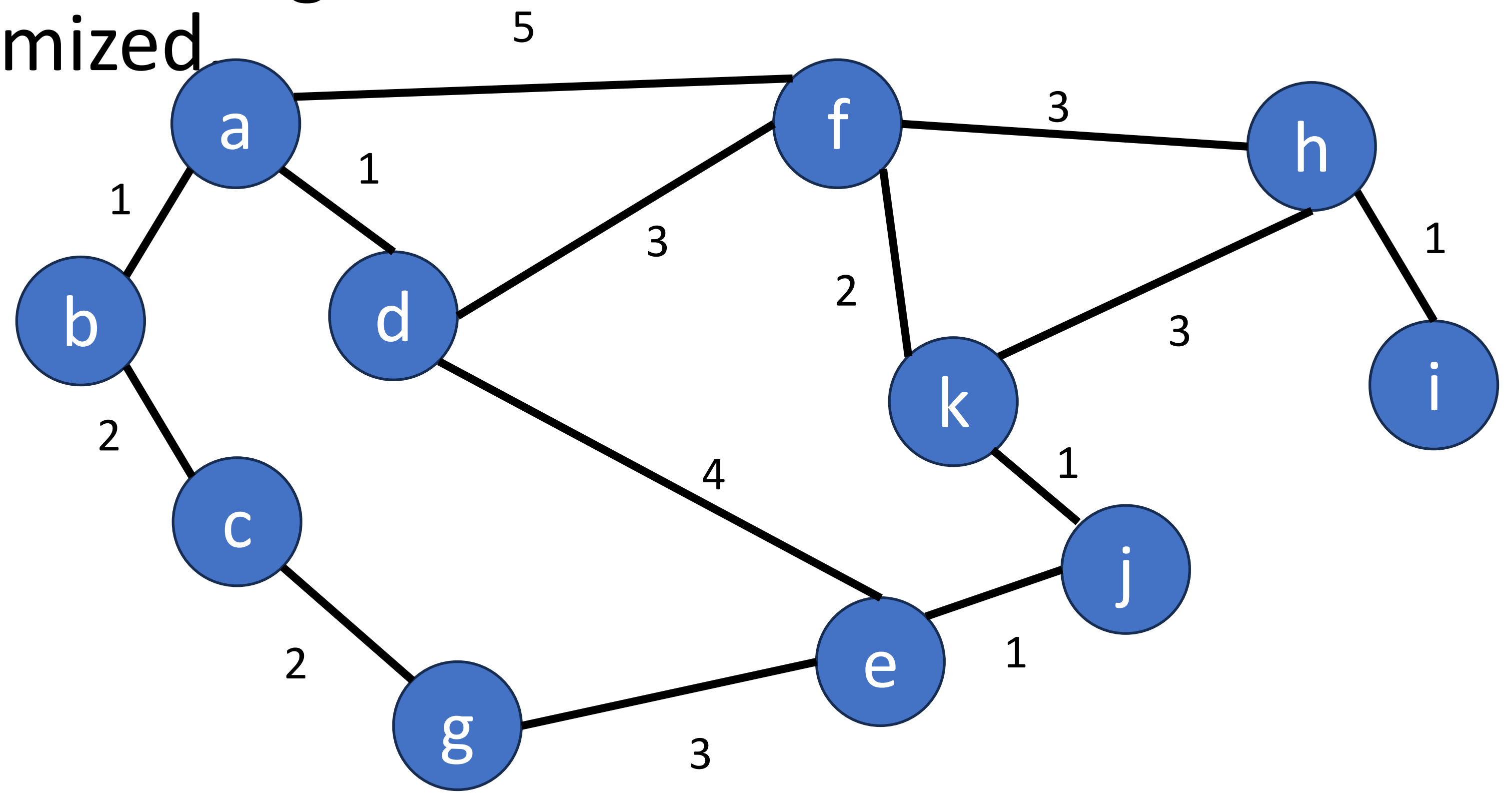


Original graph Three examples of spanning trees

Minimum cost spanning tree

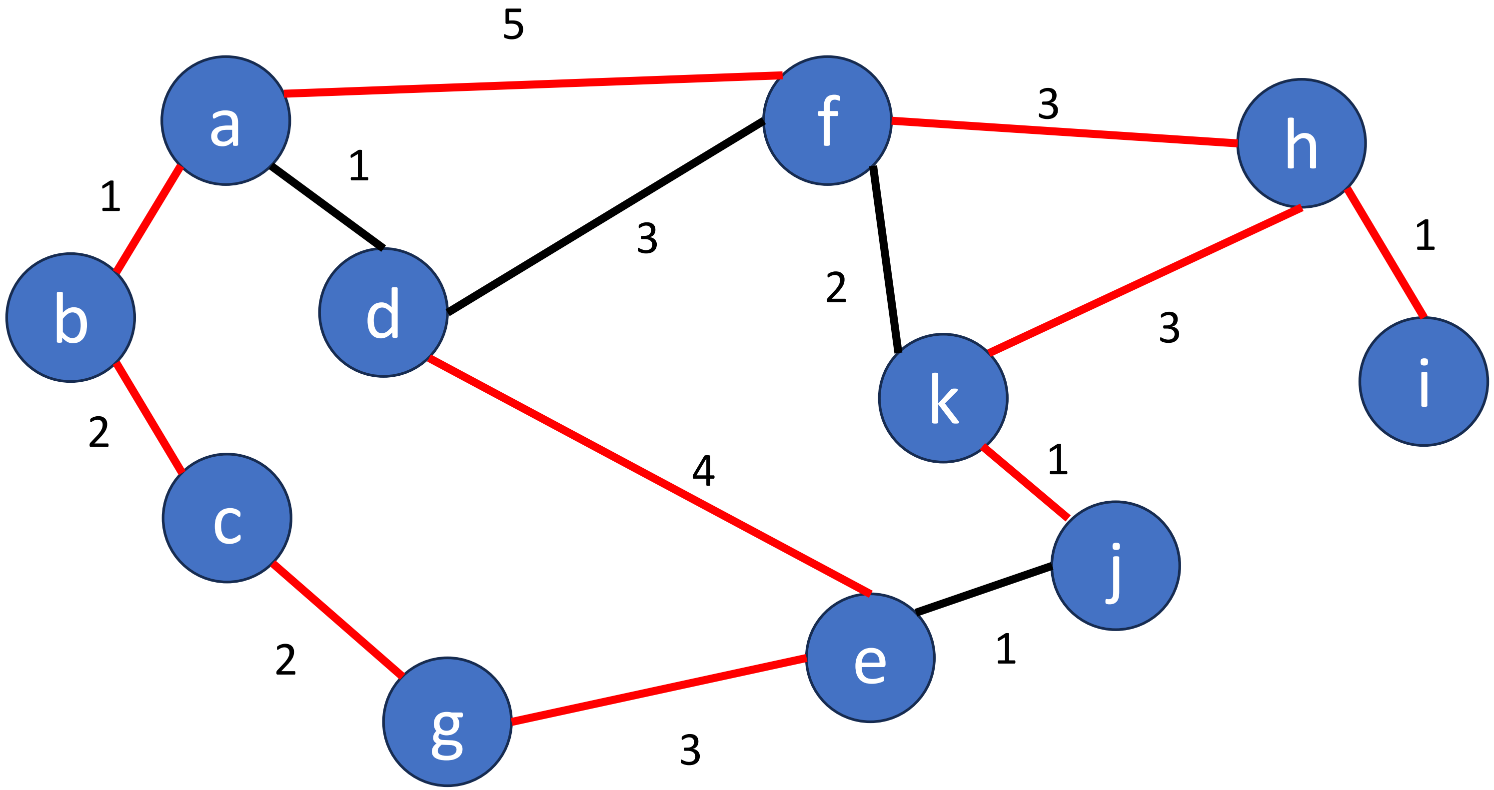
- A spanning tree with least cost
  - Tree cost: sum of edge weights/costs.

- Application: Building communication links for all cities while the cost is minimized.



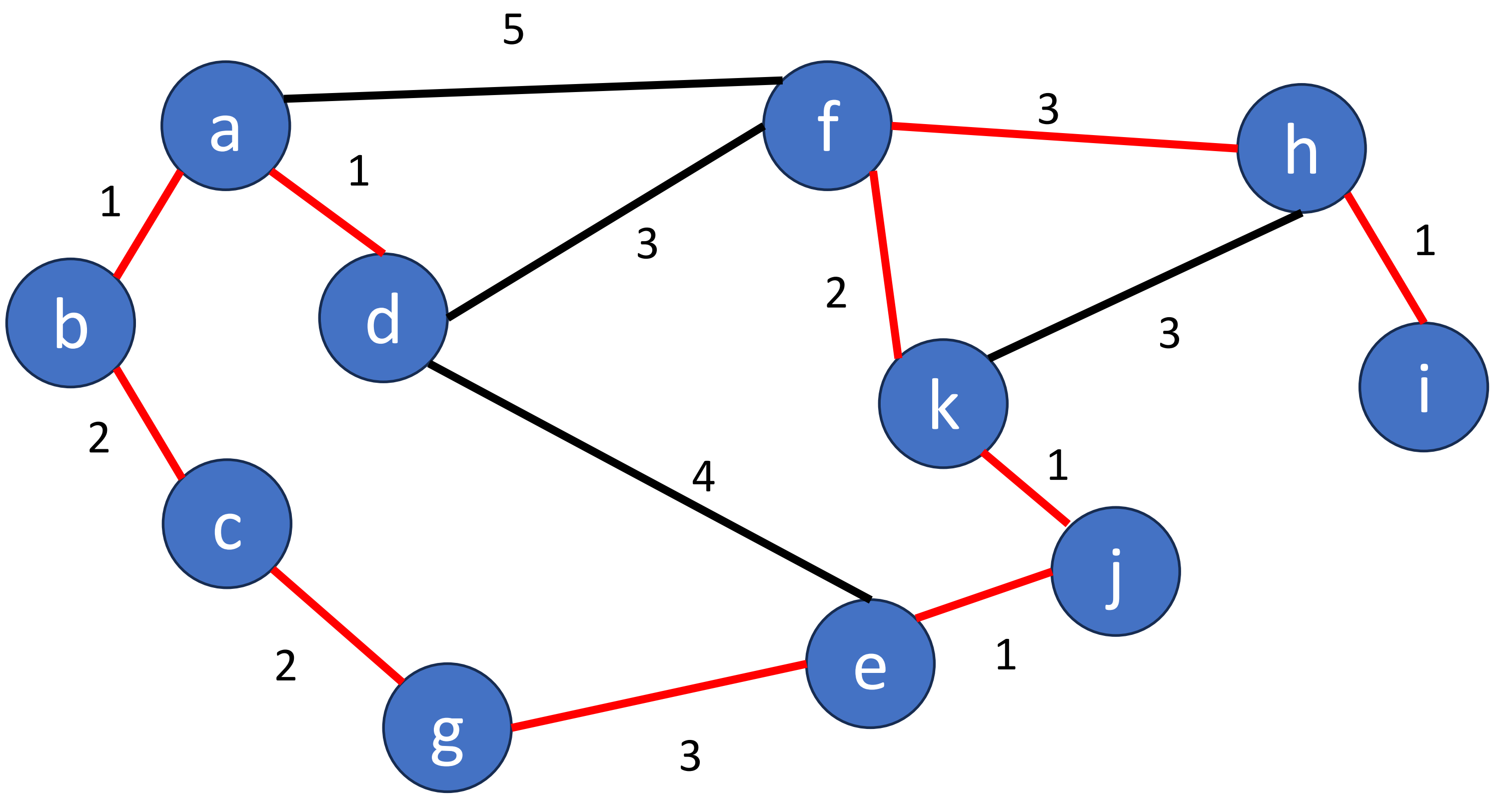
Example of spanning tree

- Spanning tree cost = 1 + 2 + 2 + 3 + 4 + 5 + 3 + 1 + 3 + 1 = 25



Example of spanning tree

- Spanning tree cost = 1 + 1 + 2 + 2 + 3 + 1 + 1 + 2 + 3 + 1 = 17



Graph representations

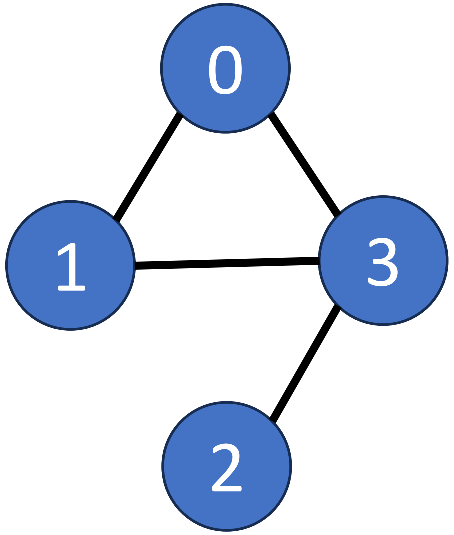
- Adjacency matrix

- Adjacency lists
  - Linked adjacency lists
  - Array adjacency lists

Adjacency matrix

- Using 2D *n*-by-*n* matrix *A*
  - *n*: number of vertices
  - *A*[*i*][*j*] = 1: an edge between vertices *i* and *j*
  - Diagonal entries are zeros.

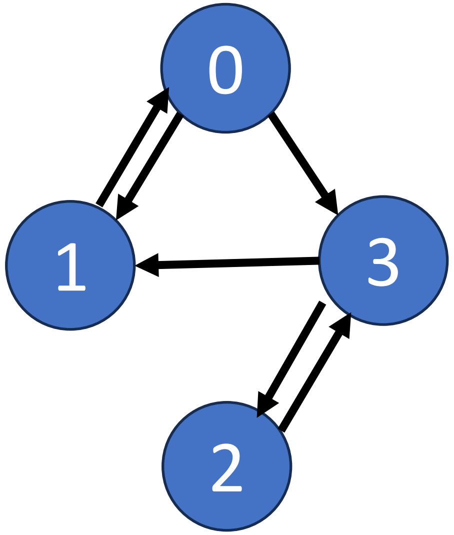
Undirected graph



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

Adjacency matrix of an undirected graph is symmetric.

Digraph



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

Adjacency matrix of a directed graph need **not** be symmetric.

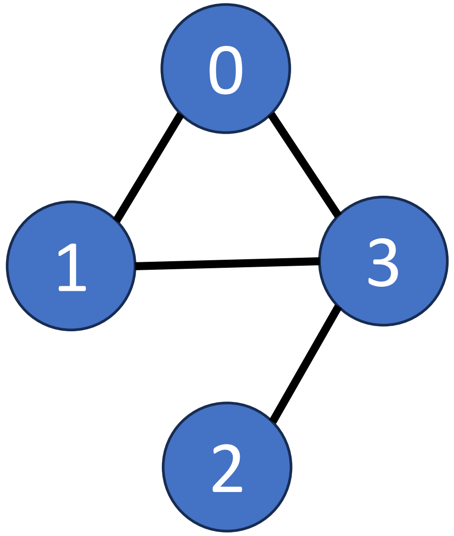
Count number of edges in a graph

- Searching the matrix time complexity: O(n^2)

- What if the graph is sparse?

$$e << n^2$$

Most elements in the adjacency matrix are zeros.

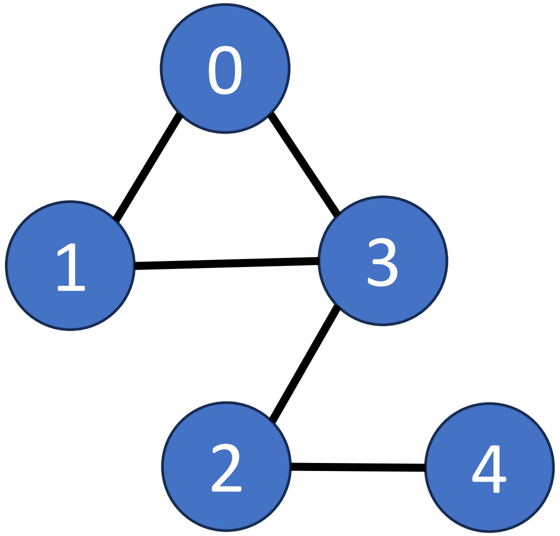


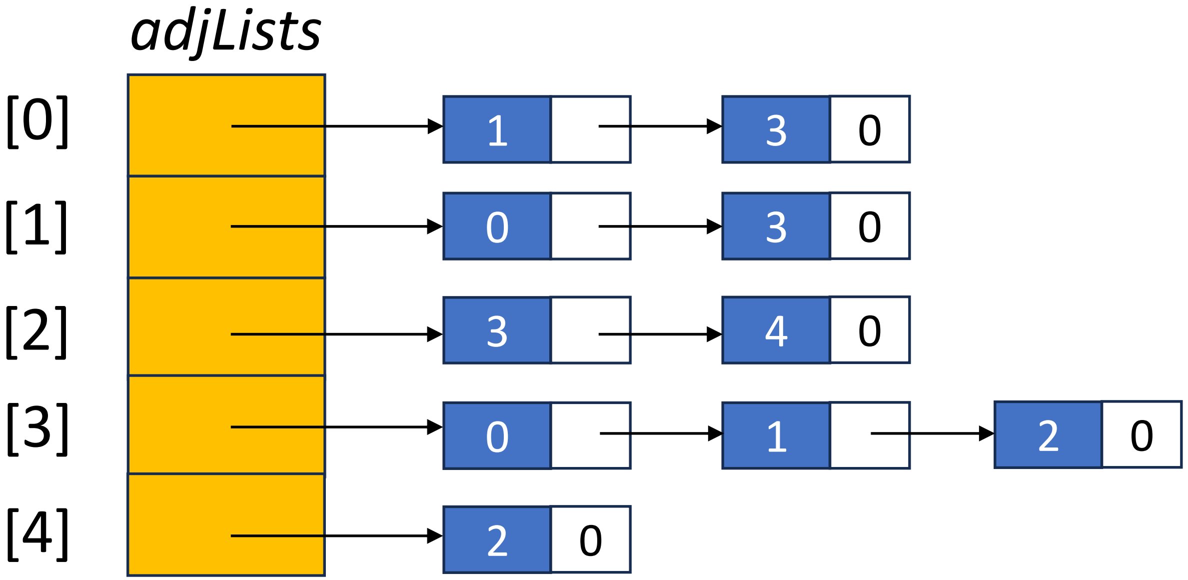
	0	**1**	**2**	**3**
0	0	1	0	1
1	1	0	0	1
2	0	0	0	1
3	1	1	1	0

Linked adjacency list

- Each adjacency list is a chain. One chain for each vertex.
  - Array length = n
  - In undirected graph, total number of chain nodes = $2\ e$
  - In directed graph, total number of chain nodes = e

e: number of edges
n: number of vertices

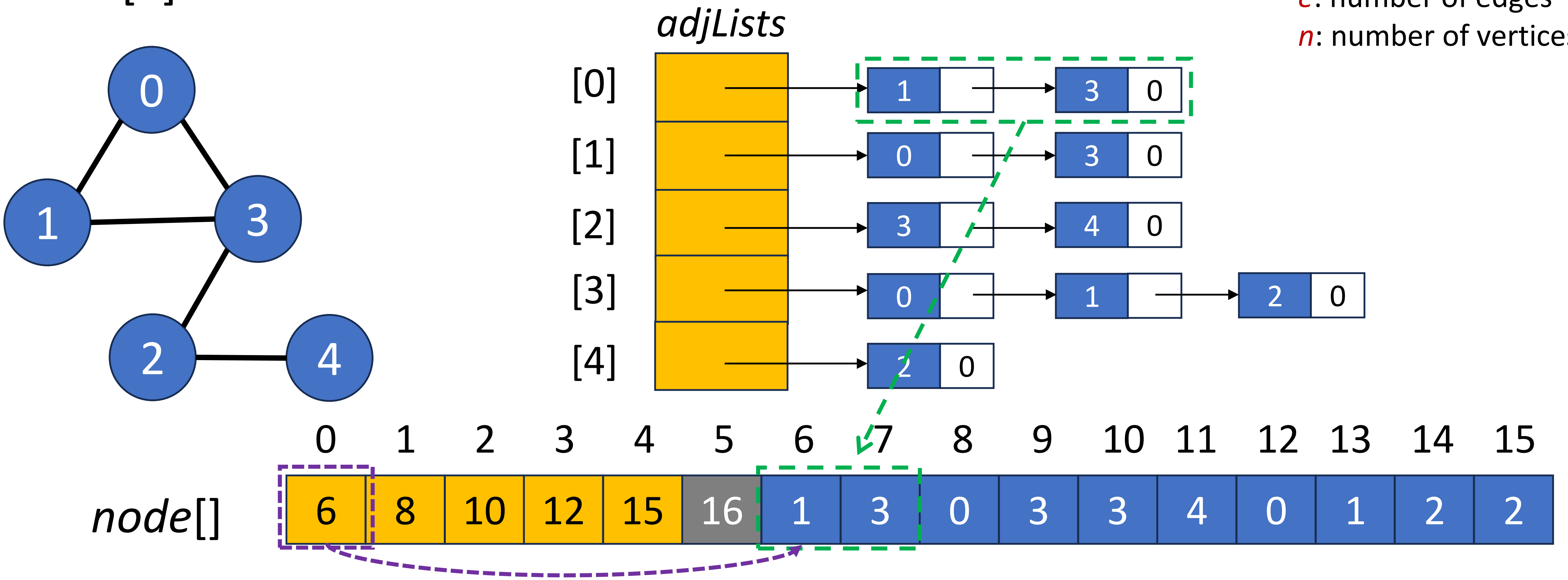




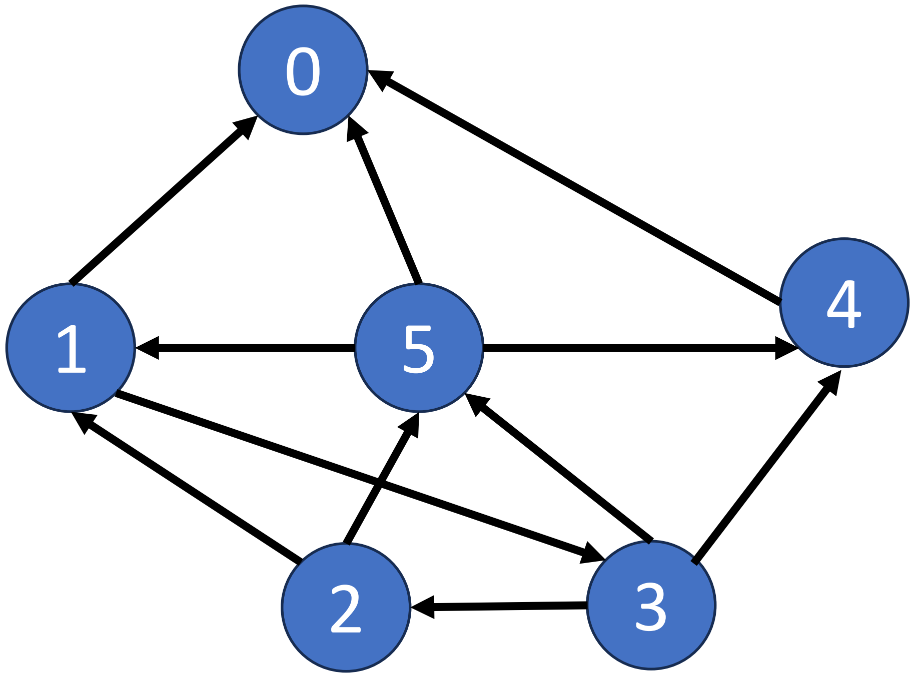
Array adjacency list

- Using an integer array *node*[] to store all adjacency lists.
  - array length = $n + 2e + 1$
  - i-th element in *node*[0, 1, …, n-1]: starting point of the list for vertex i.
  - *node*[n]: set to $n + 2e + 1$

e: number of edges
n: number of vertices



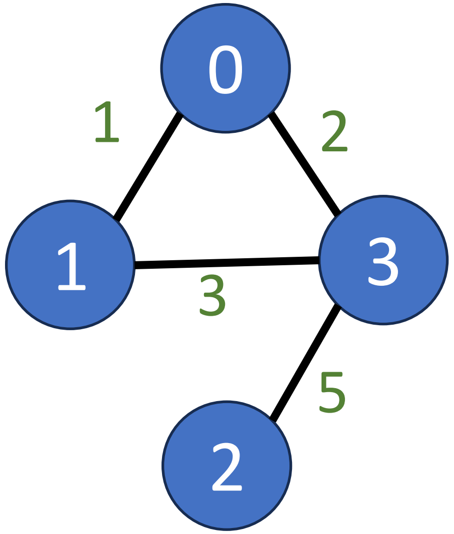
Exercise



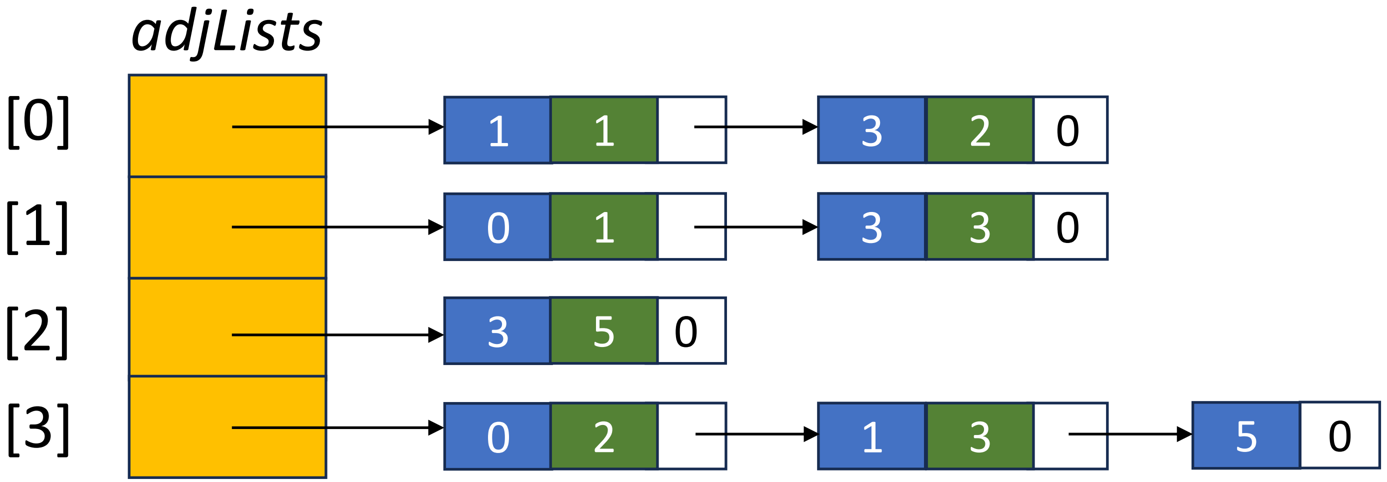
- Write out the adjacency matrix.
- Write out the linked adjacency-list representation.

Weighted graphs

- Cost adjacency matrix: *A*[*i*][*j*] = cost of edge(*i*, *j*)
- Adjacency lists: Each list element is a pair (adjacent vertex, edge weight)
- A graph with weighted edges is called a network.



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



Summary

- What is directed graph and indirected graph?
- Terminology of graph
  - graph, subgraph, path, simple path, cycle, connected components, degree
- Spanning tree
- Graph representation
  - Adjacency matrix
  - Adjacency lists