Data structures and Algorithms

GRAPHS: BASIC CONCEPTS

Nguyễn Ngọc Thảo
nnthao@fit.hcmus.edu.vn

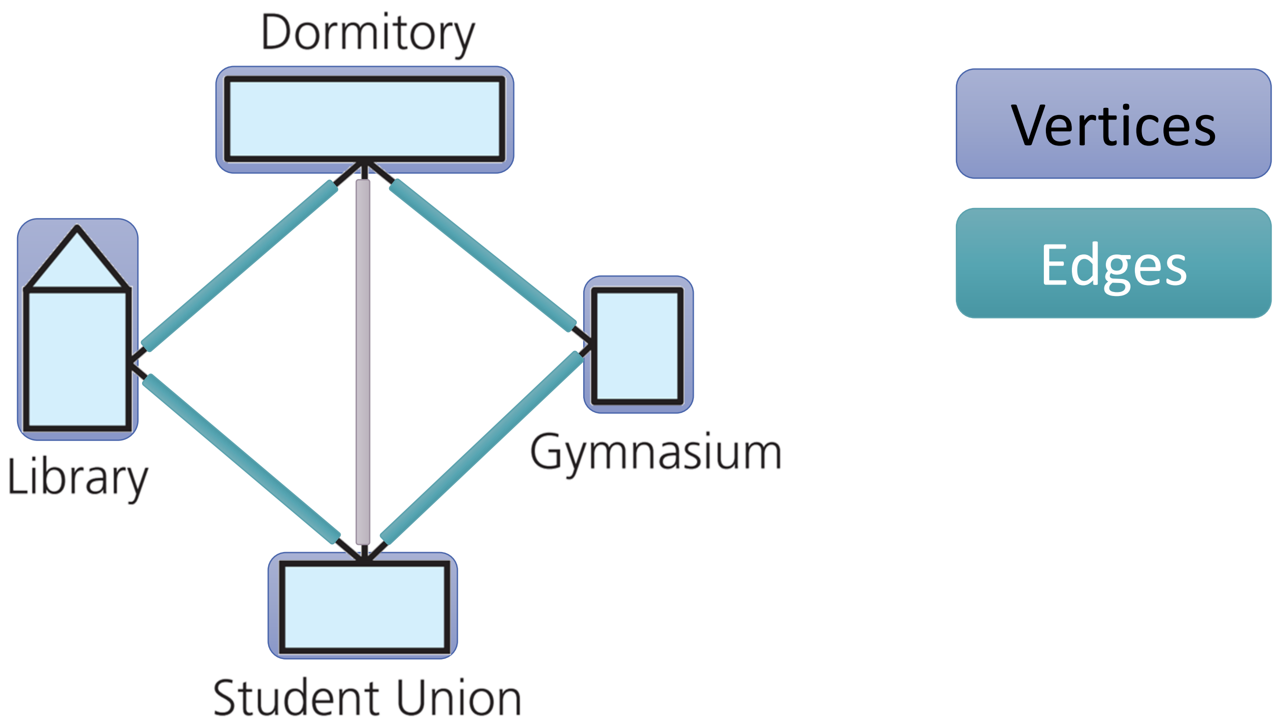
Outline

- Graph terminology
- Graphs as an ADT

Graph terminology

Graphs: A definition

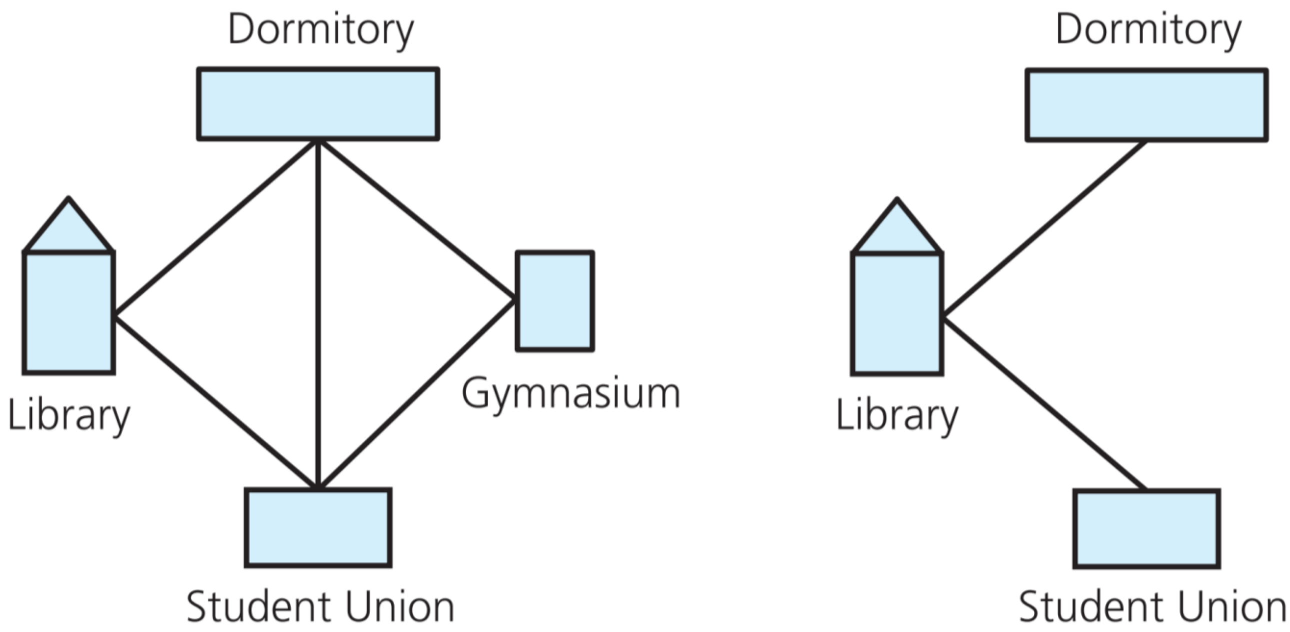
- A graph $\boldsymbol{G}$ consists of two sets: a set $\boldsymbol{V}$ of vertices, or nodes, and a set $\boldsymbol{E}$ of edges that connect the vertices.



- Graphs represent the relationships among data items

Graphs: Subgraph

- A subgraph consists of a subset of a graph's vertices and a subset of its edges.



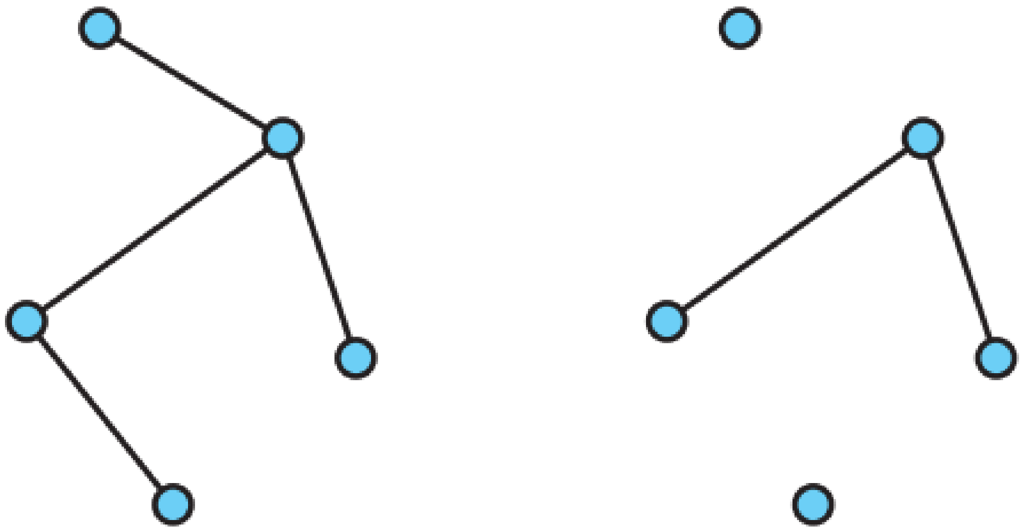
Left: A campus map as a graph. Right: A subgraph.

Graphs: Paths and Cycles

- Two adjacent vertices are joined by an edge.
  - E.g., the Library and the Student Union, etc.
- A path between two vertices is a sequence of edges that begins at one vertex and ends at another vertex.
  - E.g., **Dormitory** $\rightarrow$ Library $\rightarrow$ Student Union $\rightarrow$ **Library**
  - A simple path passes through a vertex only once.
- A cycle is a path that begins and ends at the same vertex.
  - E.g., **Library** $\rightarrow$ Student Union $\rightarrow$ Gymnasium $\rightarrow$ Dormitory $\rightarrow$ **Library**
  - A simple cycle passes through other vertices only once.

Graphs: Connected graphs

- A connected graph has a path between each pair of distinct vertices.
    - You can go from any vertex to any other vertex by following a path.
- Disconnected graphs are those unqualified for the above condition.

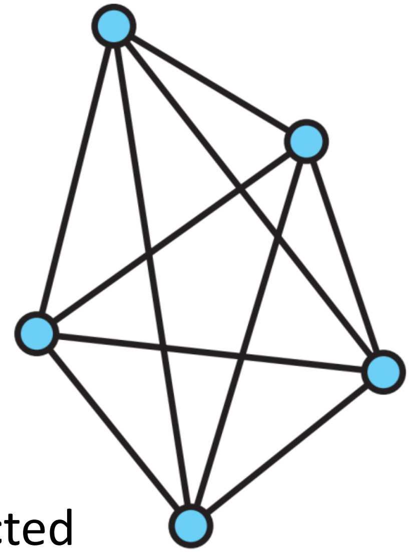


Left: A connected graph.

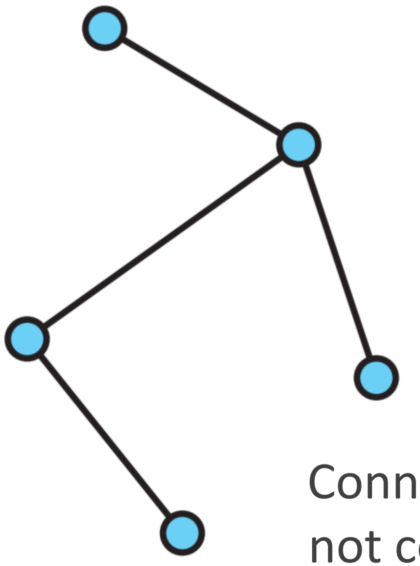
Right: A disconnected graph.

Graphs: Complete graphs

- A complete graph has an edge between each pair of distinct vertices.
  - A complete graph is also connected, but the converse is not true



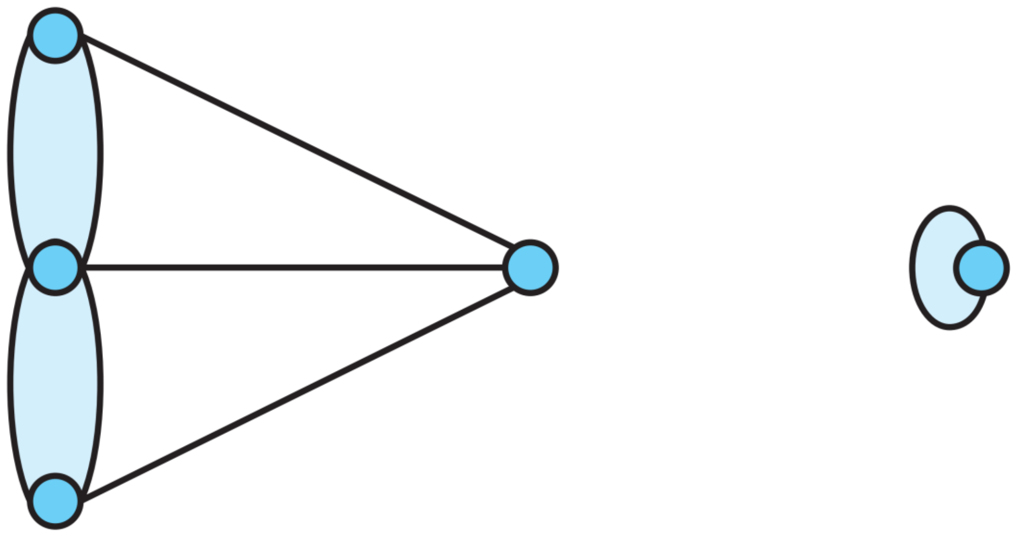
Completed
and hence connected



Connected but
not completed

Exceptions: Multigraphs and Loops

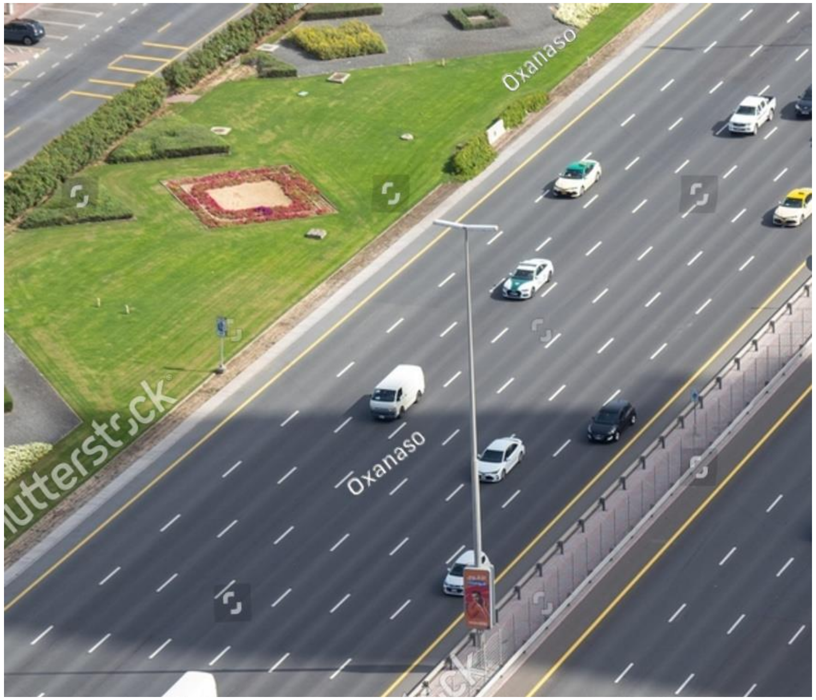
- A multigraph does allow multiple edges.
  - It is not considered as a formal graph since it violates the definition of “set of edges”.
- A self edge or loop begins and ends at the same vertex.

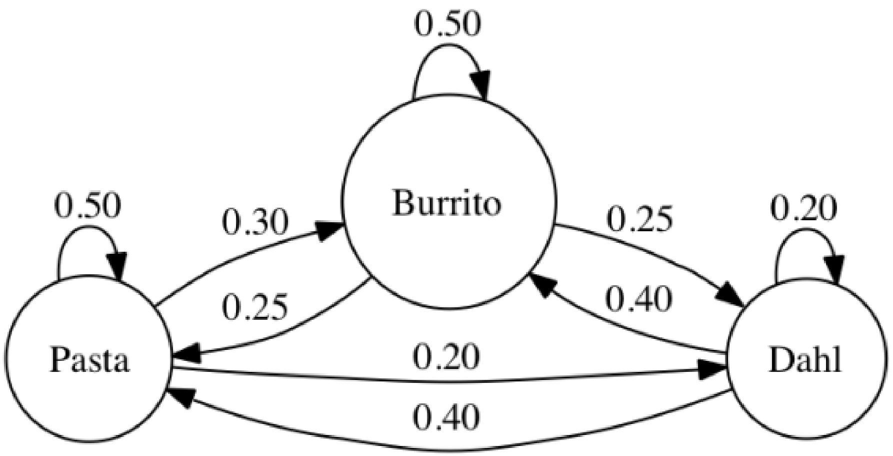


Left: A multigraph is not a graph. Right: a self edge is not allowed in a graph.

Exceptions: Multigraphs and Loops

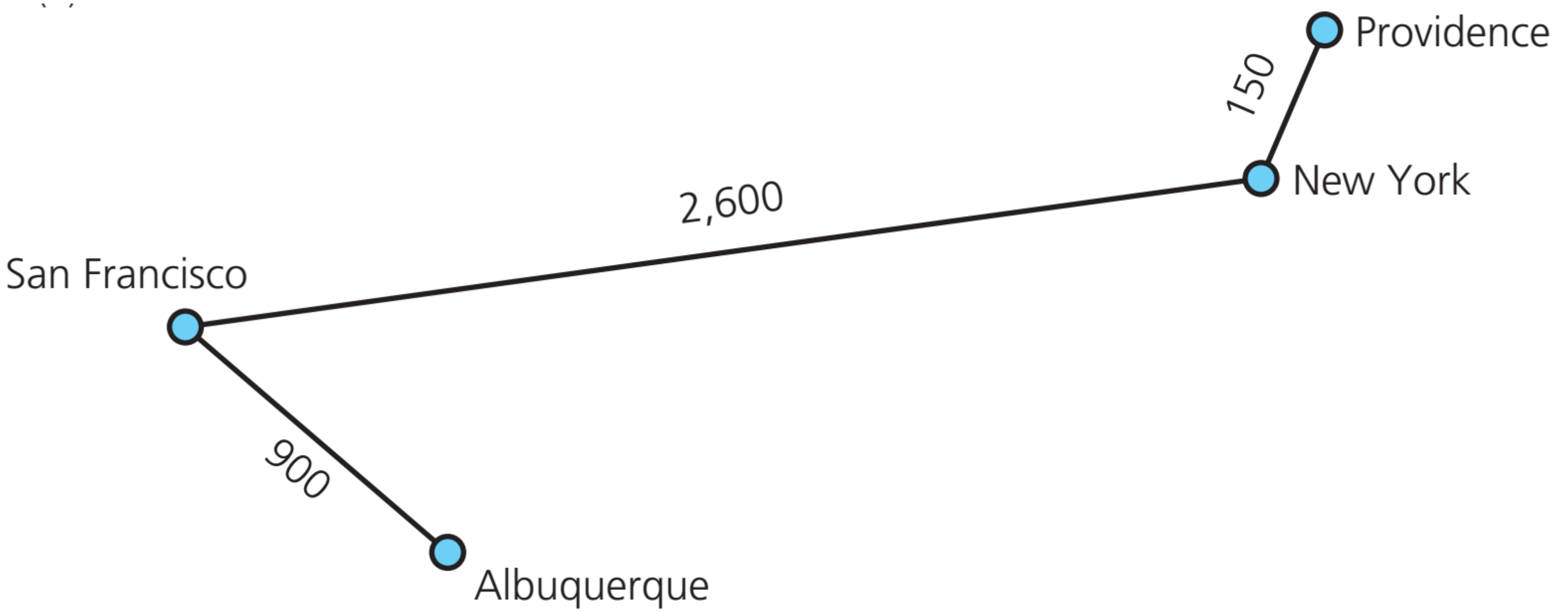
- However, these exceptions are common in practice.
- Multigraph: multi-lane highways, parallel networks, etc.
- Loops: repeated states in Markov models, etc.





Graphs: Weighted graphs

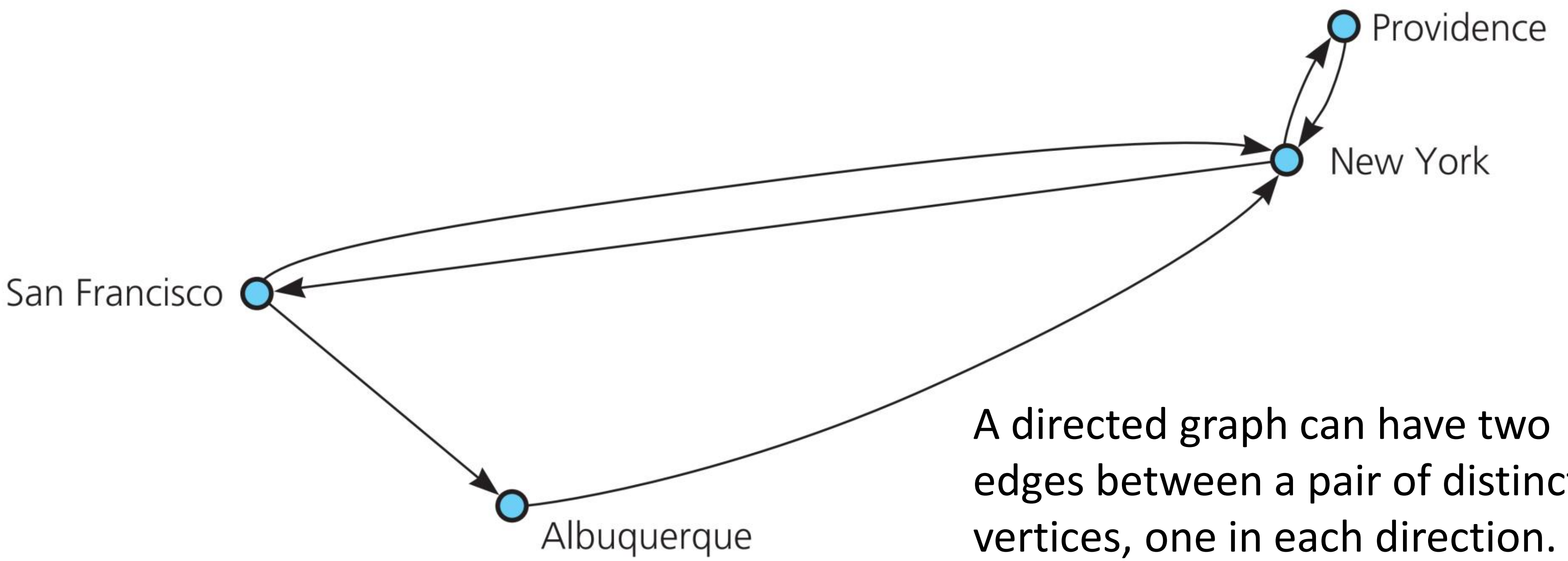
- A weighted graph has its edges labeled with numeric values.



A weighted graph whose edges are labeled with the distances between cities.

Graphs: Directed graphs

- A directed edge is an edge that has a direction.
- A directed graph (or digraph) has every edge being directed, while an undirected graph contains only undirected edges.

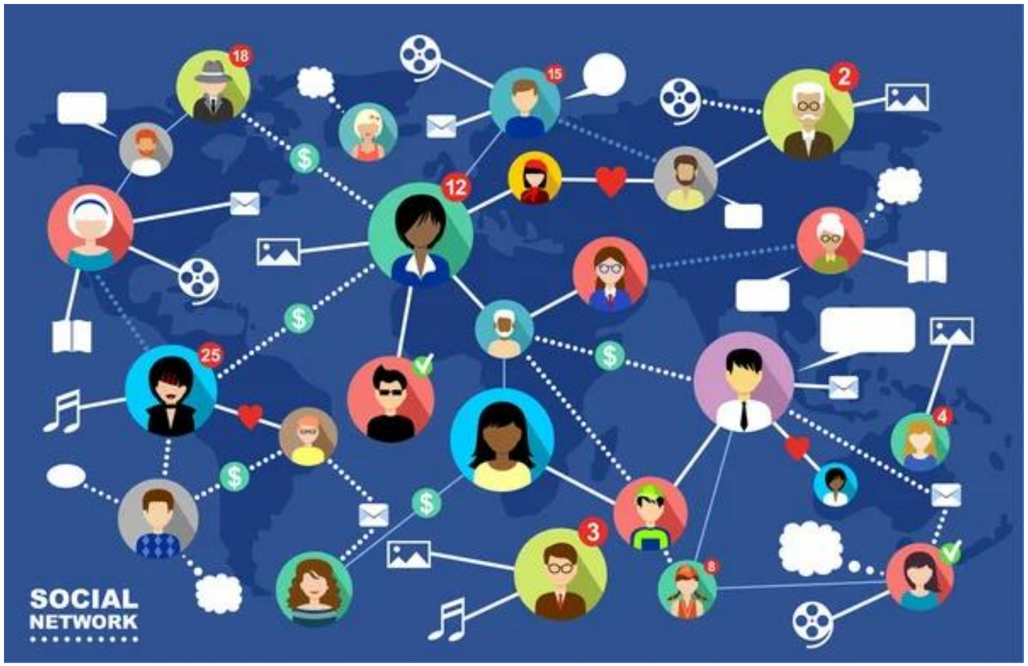


A directed graph can have two edges between a pair of distinct vertices, one in each direction.

Graphs: Directed graphs

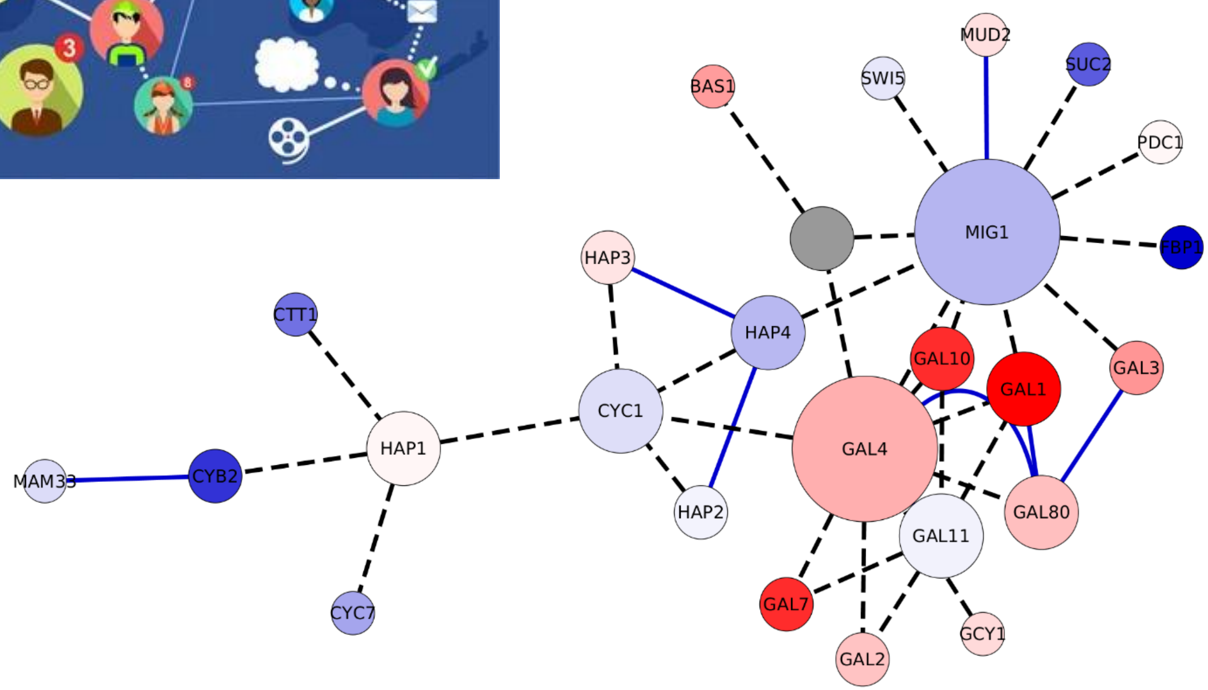
- The definitions given for undirected graphs apply also to directed graphs, with changes that account for direction.
- The vertex y is adjacent to vertex x if there is a directed edge from the predecessor x to the successor y.
  - E.g., Albuquerque is adjacent to San Francisco, but San Francisco is not adjacent to Albuquerque.
- A directed path is a sequence of directed edges between two vertices.
  - E.g**., Providence** $\rightarrow$ New York $\rightarrow$ **San Francisco**

Graphs: Applications



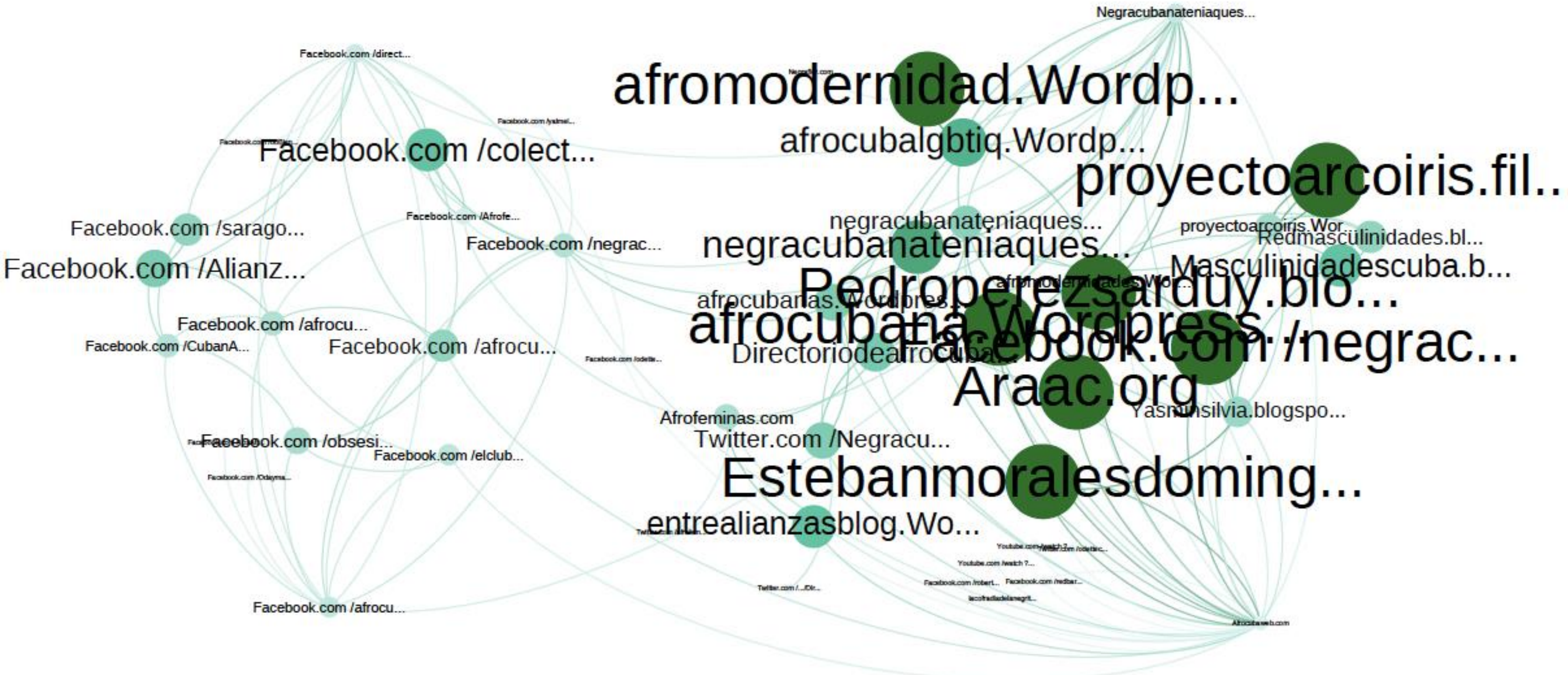
- Vertices: users
- Edges: connections

- Vertices: proteins
- Edges: interactions



Graphs: Applications

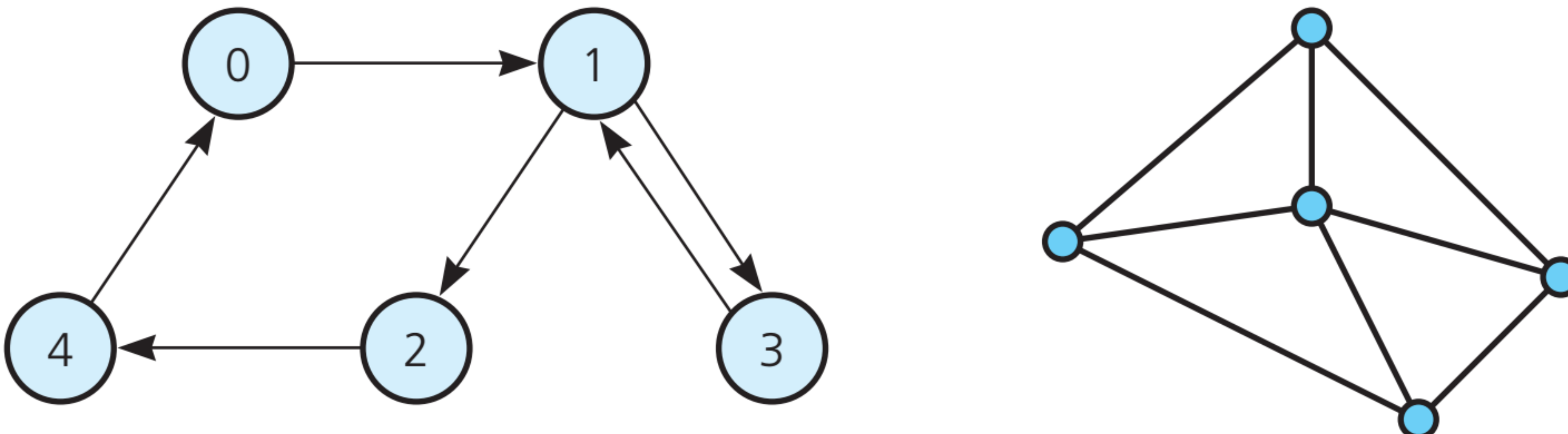
- Vertices: web pages
- Edges: hyperlinks



WWW is the biggest ever network in the virtual world.

Checkpoint 01: Describe the graphs

- Describe the below graphs.
- For example, are they directed? Connected? Complete? Weighted?



(a) (b)

Graphs as an ADT

Graphs as an ADT

- Insertion and removal are applied to both vertices and edges.
  - It is somewhat different from trees, in which these operations affects nodes only.
- The vertices in a graph may or may not contain values.
  - A graph whose vertices do not contain values represents only the relationships among vertices.
  - Many problems have no need for vertices' values
- However, the following graph operations do assume that the vertices contain values.

ADT graphs: Basic operations

- **Check** whether a graph is empty
- **Get** the number of vertices in a graph
- **Get** the number of edges in a graph
- **Get** from a graph the vertex that contains a given value
- **Check** whether an edge exists between two given vertices
- **Insert** a vertex in a graph whose vertices have distinct values that differ from the new vertex's value
- **Insert** an edge between two given vertices in a graph
- **Remove** a particular vertex from a graph and any edges between the vertex and other vertices
- **Remove t**he edge between two given vertices in a graph

Adjacency matrix

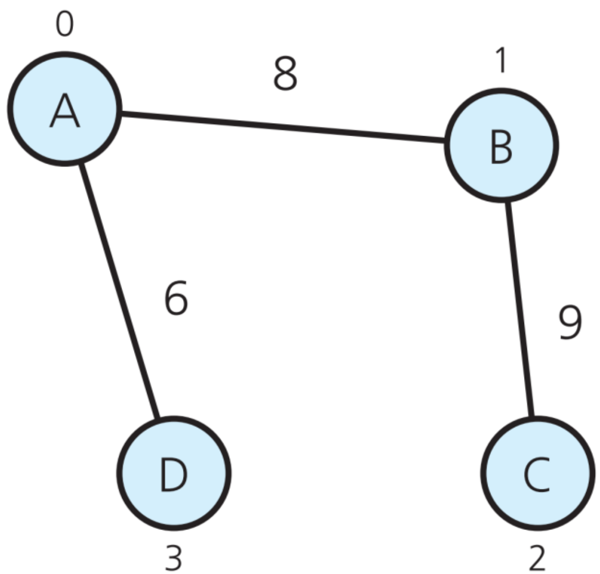
- Consider a graph with n vertices numbered $0, 1, \ldots, n-1$.
- An adjacency matrix is a $n \times n$ array such that

$$matrix[i][j] = \begin{cases} 1 & if\ there\ is\ an\ edge\ from\ i\ to\ j \\ 0 & otherwise \end{cases}$$

  for any pair of vertices i and j.

- In weighted graphs, $matrix[i][j]$ is the weight that labels the edge from vertex i to vertex j.
  - $matrix[i][j] = \infty$ when there is no edge from i to j
- The adjacency matrix for undirected graph is symmetrical.

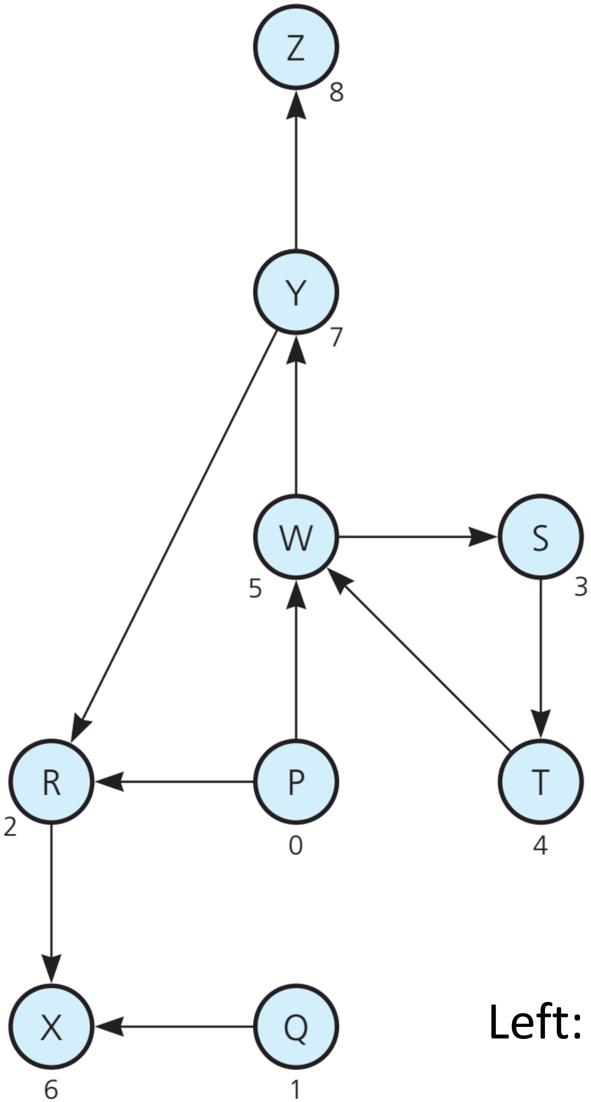
Adjacency matrix: An example



		0	1	2	3
		A	B	C	D
0	A	∞	8	∞	6
1	B	8	∞	9	∞
2	C	∞	9	∞	∞
3	D	6	∞	∞	∞

Left: A weighted undirected graph. Right: Its adjacency matrix.

Adjacency matrix: An example

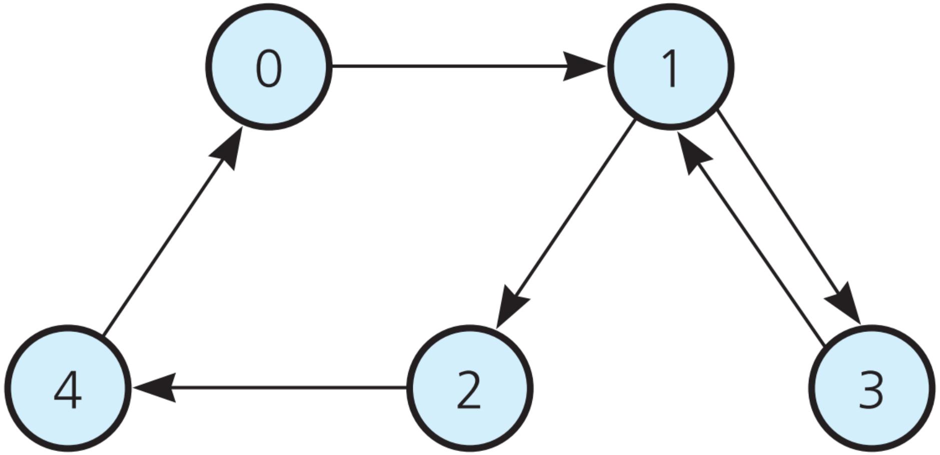


		0	1	2	3	4	5	6	7	8
		P	Q	R	S	T	W	X	Y	Z
0	P	0	0	1	0	0	1	0	0	0
1	Q	0	0	0	0	0	0	1	0	0
2	R	0	0	0	0	0	0	1	0	0
3	S	0	0	0	0	1	0	0	0	0
4	T	0	0	0	0	0	1	0	0	0
5	W	0	0	0	1	0	0	0	1	0
6	X	0	0	0	0	0	0	0	0	0
7	Y	0	0	1	0	0	0	0	0	1
8	Z	0	0	0	0	0	0	0	0	0

Left: A directed graph. Right: Its adjacency matrix

Checkpoint 02a: Adjacency matrix of a graph

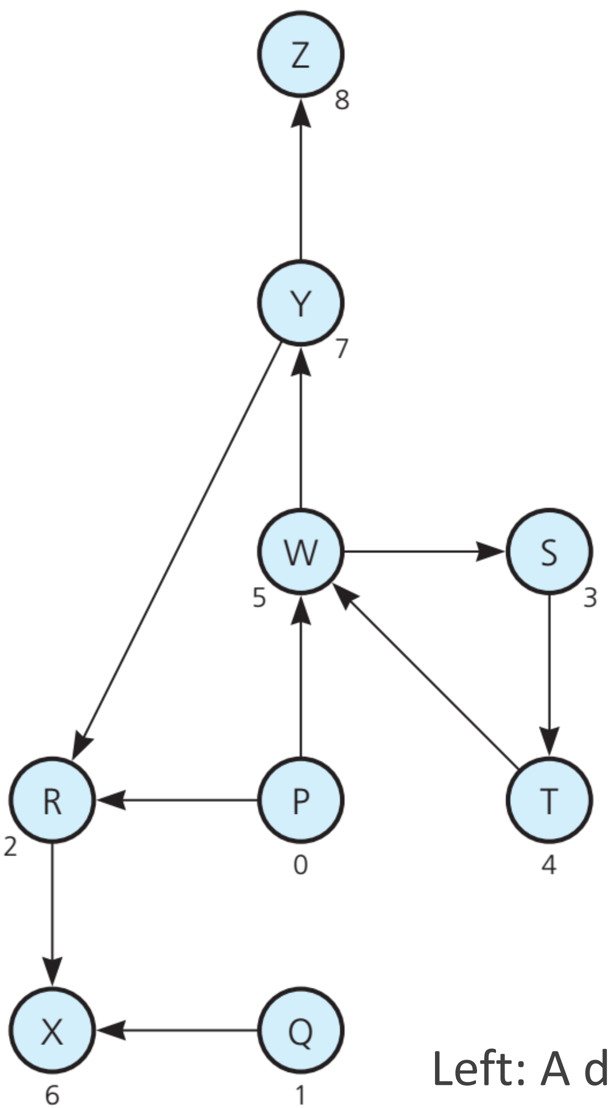
- Write the adjacency matrix for the following graph.

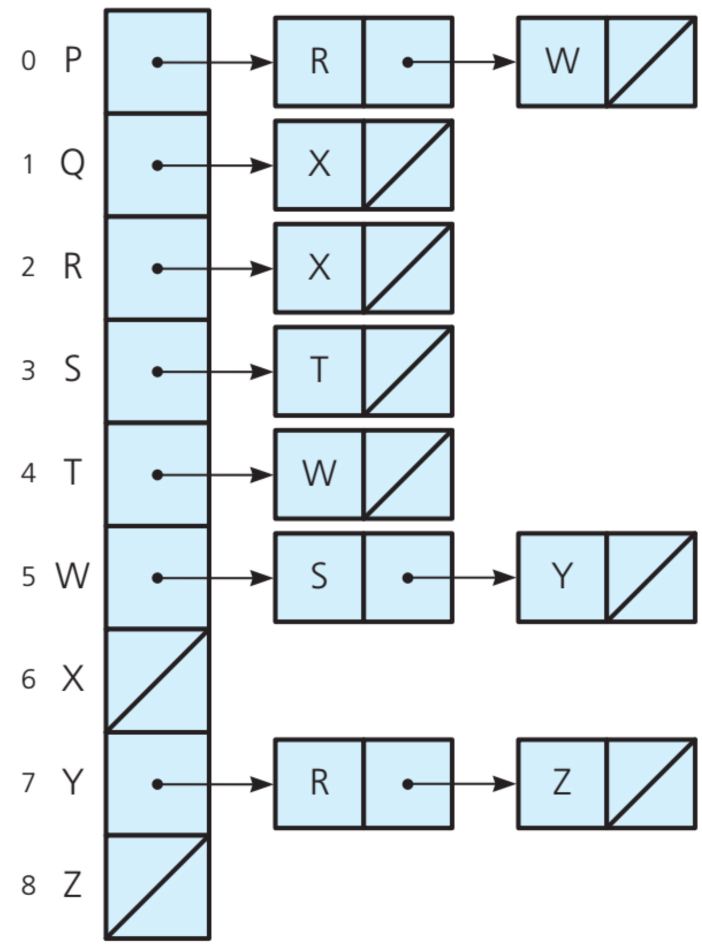


Adjacency list

- Consider a graph with n vertices numbered $0, 1, \ldots, n-1$.
- An adjacency list contains n linked chains, each per vertex.
  - The i^{th} linked chain has a node for vertex j if and only if the graph contains an edge from vertex i to vertex j.
- In undirected graphs, each edge is treated as if it were two directed edges in opposite directions.

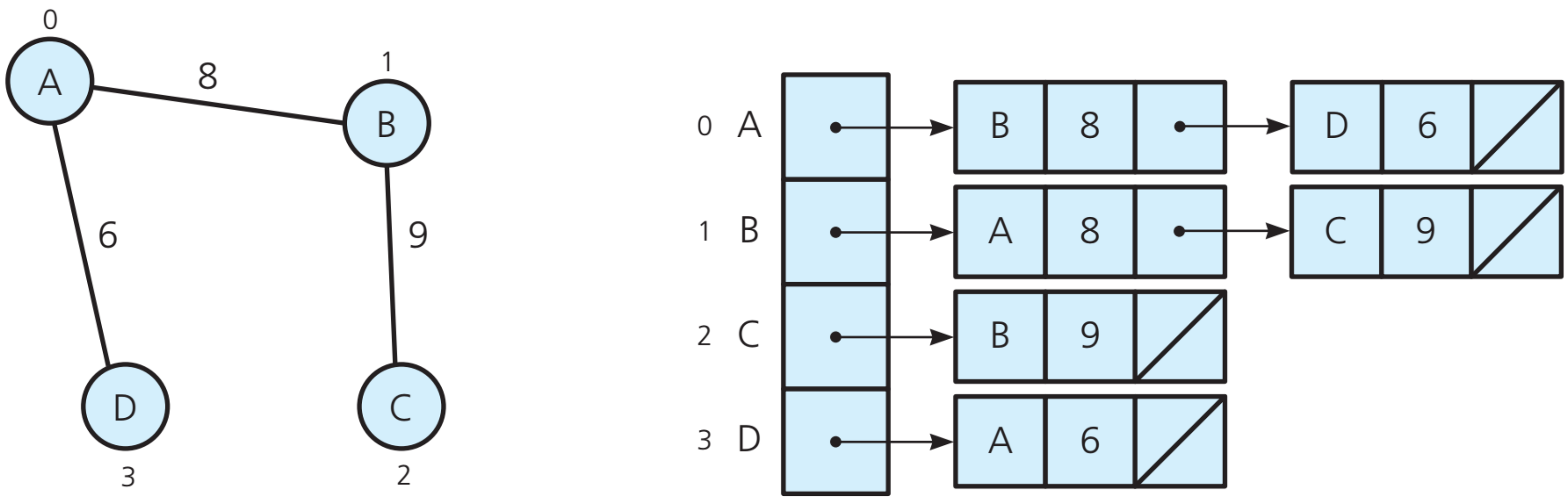
Adjacency list: An example





Left: A directed graph. Right: Its adjacency list

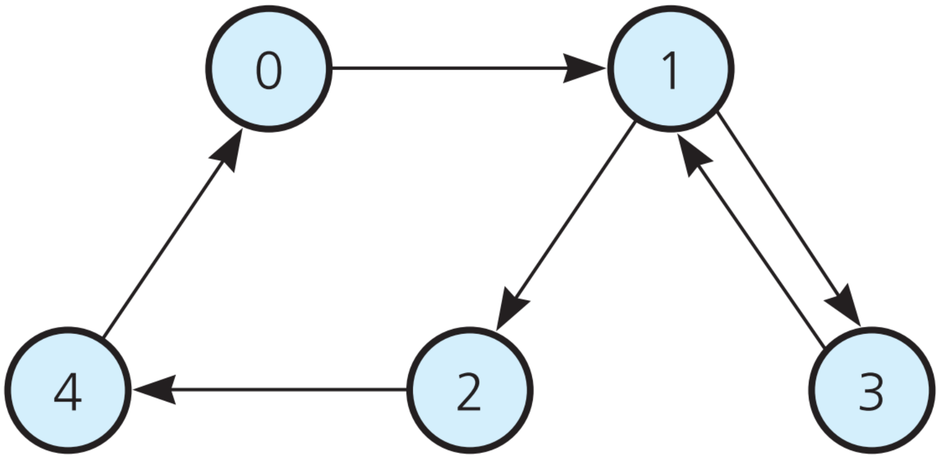
Adjacency list: An example



Left: A weighted undirected graph. Right: Its adjacency list

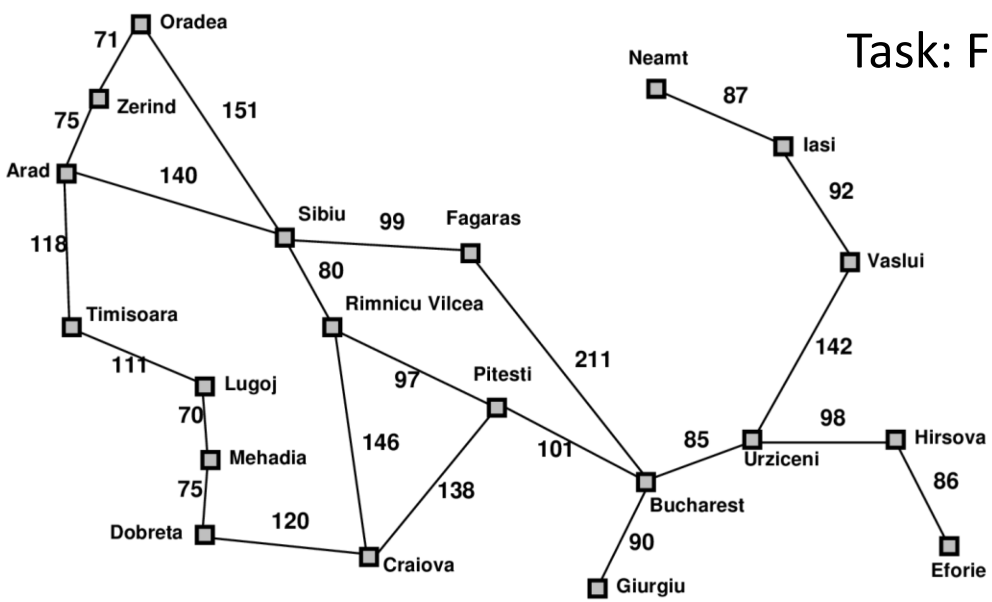
Checkpoint 02b: Adjacency list of a graph

- Write the adjacency list for the following graph.



Adjacency matrix vs. Adjacency list

- The choice of graph implementations depends on how your application uses the graph.
  - What operations performed most frequently on the graph
  - The number of edges that the graph is likely to contain



Task: Find all cities adjacent to a given city.

Which one, an adjacency list or an adjacency matrix, better facilitate the task?

Adjacency matrix vs. Adjacency list

- The two most commonly performed graph operations are

Determine whether there is an edge from vertex i to vertex j

- **Adjacency matrix:** check the value of the entry $matrix[i][j]$
- **Adjacency list:** traverse the i^{th} linked chain to determine whether a node corresponding to vertex j is present

Find all vertices adjacent to a given vertex i

- **Adjacency matrix:** traverse the i^{th} row to find all vertices adjacent to a given vertex i
- **Adjacency list:** traverse the i^{th} linked chain with fewer nodes

Adjacency matrix vs. Adjacency list

- Even though the adjacency list also has n head pointers, it often requires less storage than an adjacency matrix.

Adjacency matrix	Adjacency list
Always n^2 entries	Maximum $n(n-1)$ entries
Each entry is simply an integer	Each entry contains both a value and a pointer

Acknowledgements

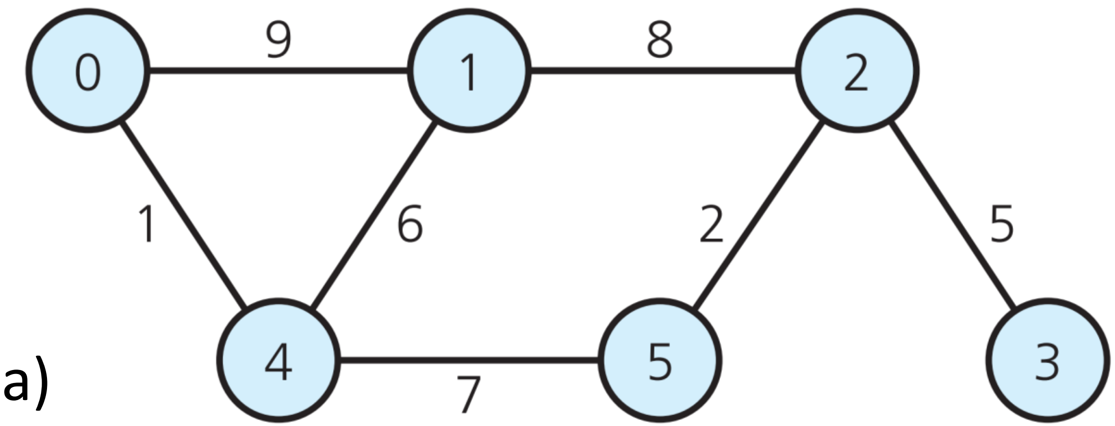
- The content of this lecture is adapted from

  [1] Frank M. Carrano, Robert Veroff, Paul Helman (2014) "*Data Abstraction and Problem Solving with C++: Walls and Mirrors*" Sixth Edition, Addion-Wesley. **Chapter 20.**

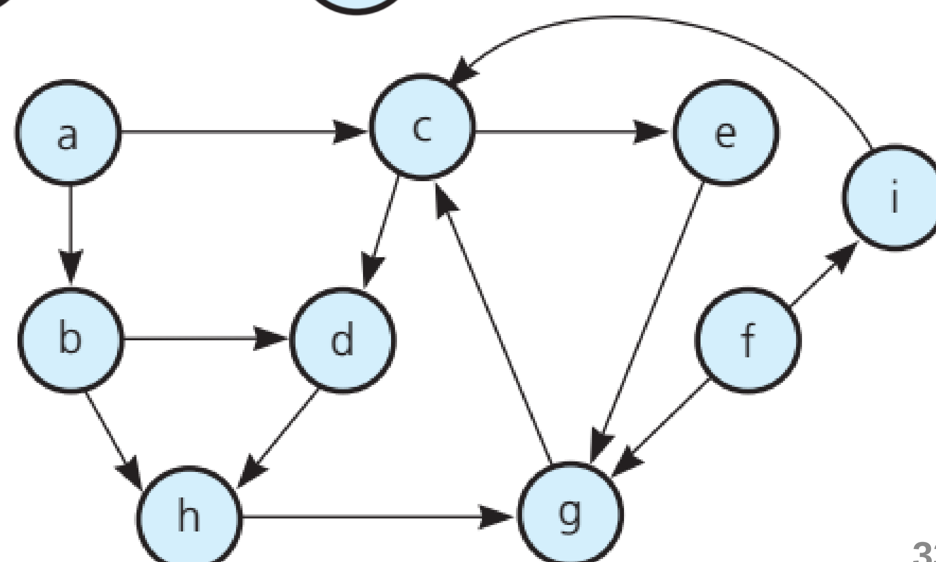
Exercises

01. Adjacency matrix / list

- Give the adjacency matrix and adjacency list for the following graphs



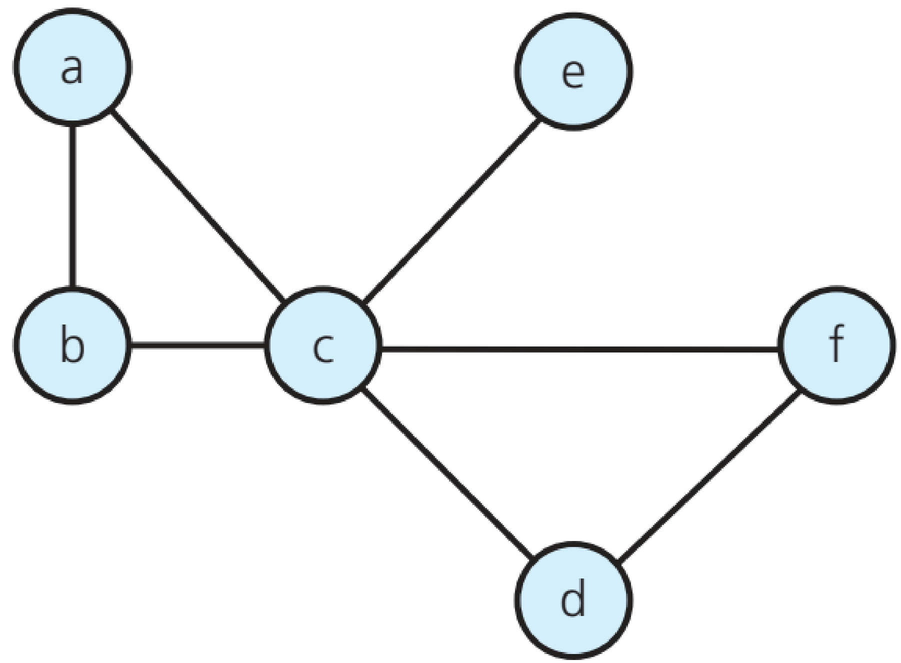
(a)



(b)

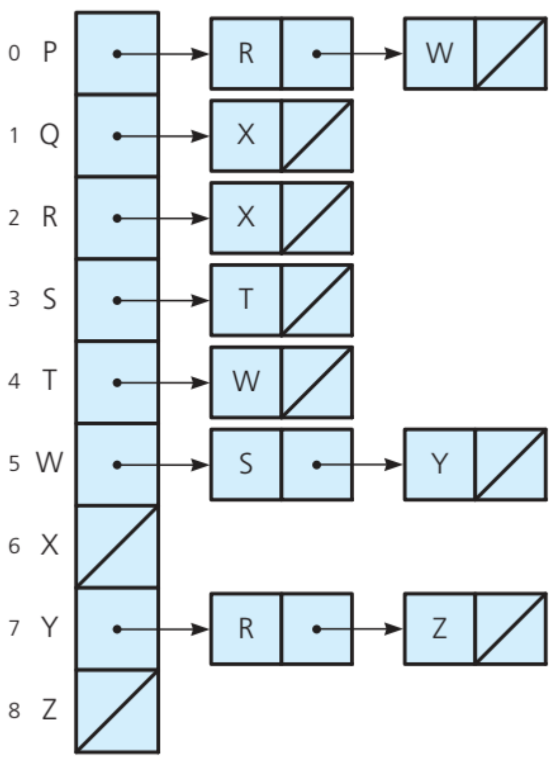
02. Adjacency matrix / list

- Consider the given graph and answer the following questions
- Will the adjacency matrix be symmetrical?
- Provide the adjacency matrix.
- Provide the adjacency list.



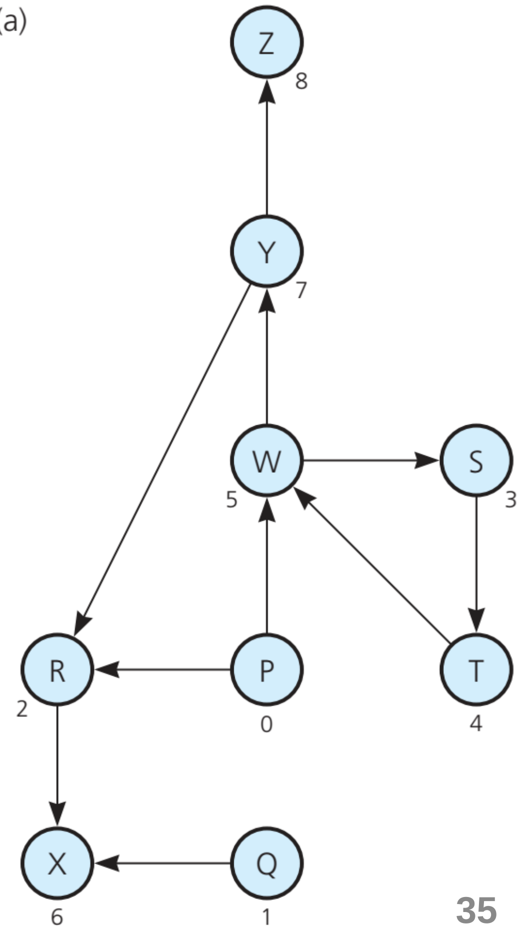
03. Adj. matrix vs. Adj. list

- Consider the following graph and its associated adjacency list and adjacency matrix. Show that the adjacency list requires less memory than the adjacency matrix.



		0	1	2	3	4	5	6	7	8
		P	Q	R	S	T	W	X	Y	Z
0	P	0	0	1	0	0	1	0	0	0
1	Q	0	0	0	0	0	0	1	0	0
2	R	0	0	0	0	0	0	1	0	0
3	S	0	0	0	0	1	0	0	0	0
4	T	0	0	0	0	0	1	0	0	0
5	W	0	0	0	1	0	0	0	1	0
6	X	0	0	0	0	0	0	0	0	0
7	Y	0	0	1	0	0	0	0	0	1
8	Z	0	0	0	0	0	0	0	0	0

(a)




...the end.