

Graphs and Directed Graphs (Digraphs)

Instructor: Meng-Fen Chiang

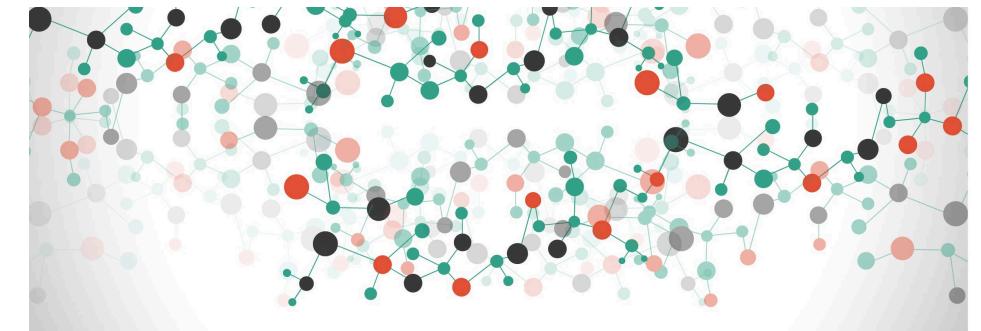
COMPCSI220: WEEK 9





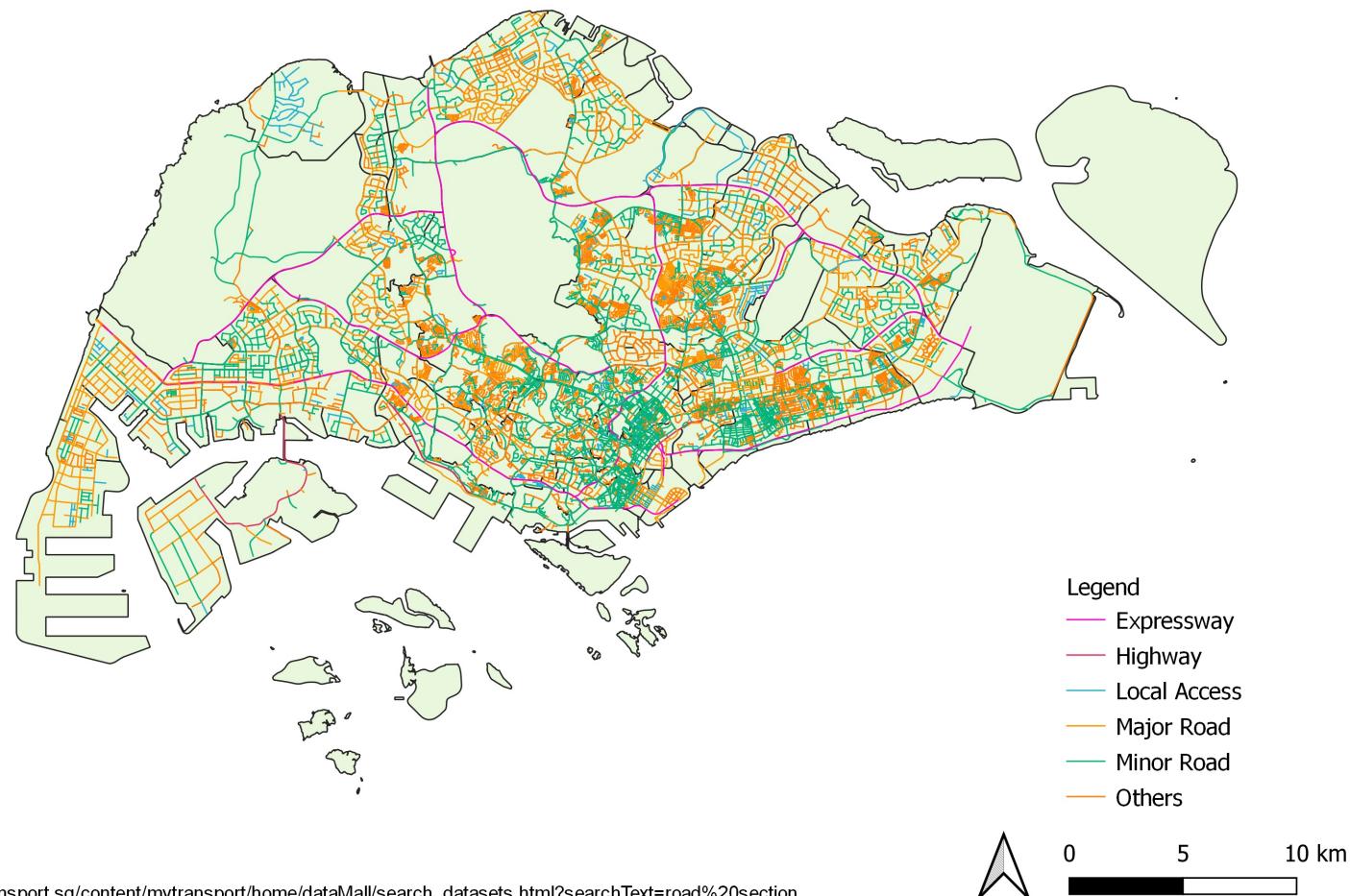
OUTLINE

- Graphs and Di-Graphs
- Preliminaries
- Basic operations



Graphs are Everywhere ...

Hierarchy Of Road Network System Of Singapore



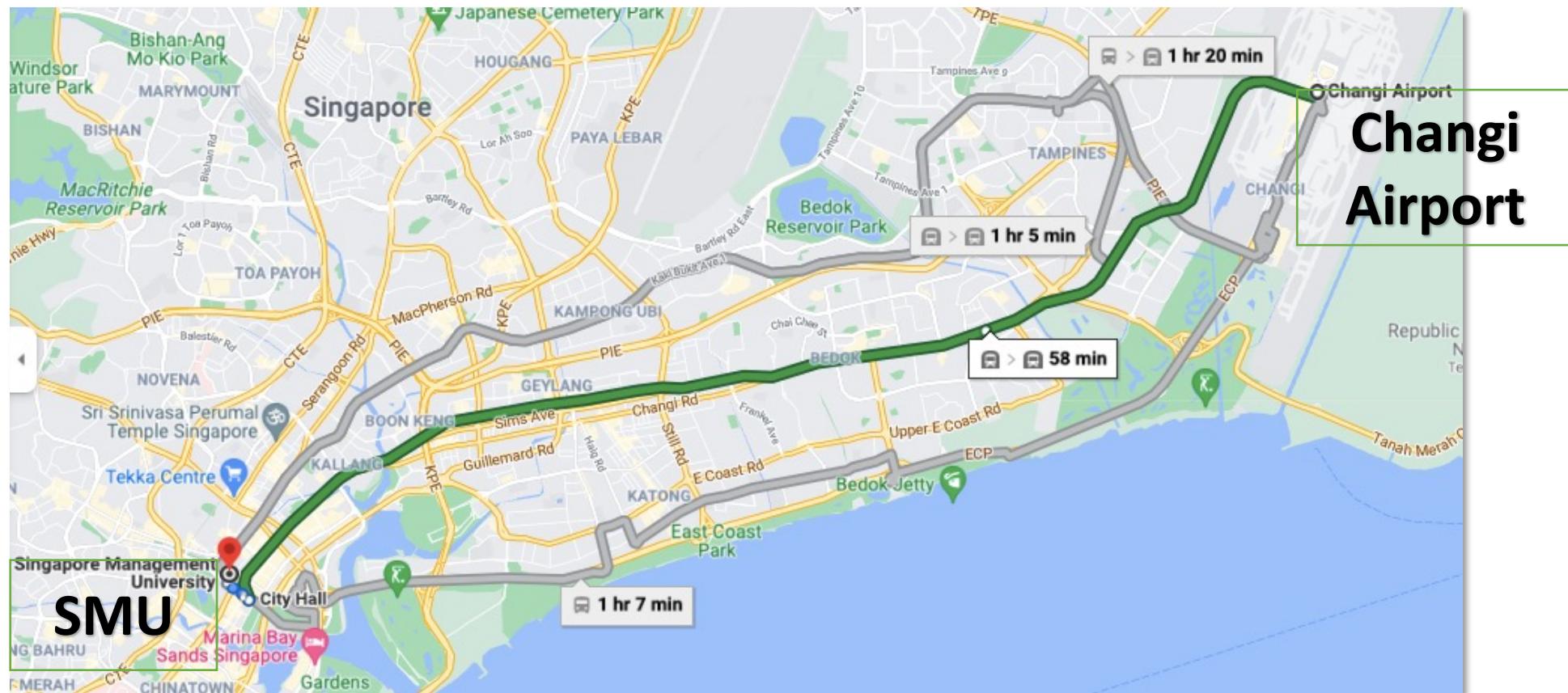
Source:
https://www.mytransport.sg/content/mytransport/home/dataMall/search_datasets.html?searchText=road%20section

Graph contains **nodes** with **edges** connecting those them.

Example: Road Network
Nodes: Intersections
Edges: Road

Graphs are Everywhere ...

- Application: finding shortest travel route on road networks



Graphs are Everywhere ...

- Application: cycling route recommendation



Graphs are Everywhere ...

- Social networks: community detection, advertisement



Monthly Active Users (MAU)

Twitter

300 millions

Facebook

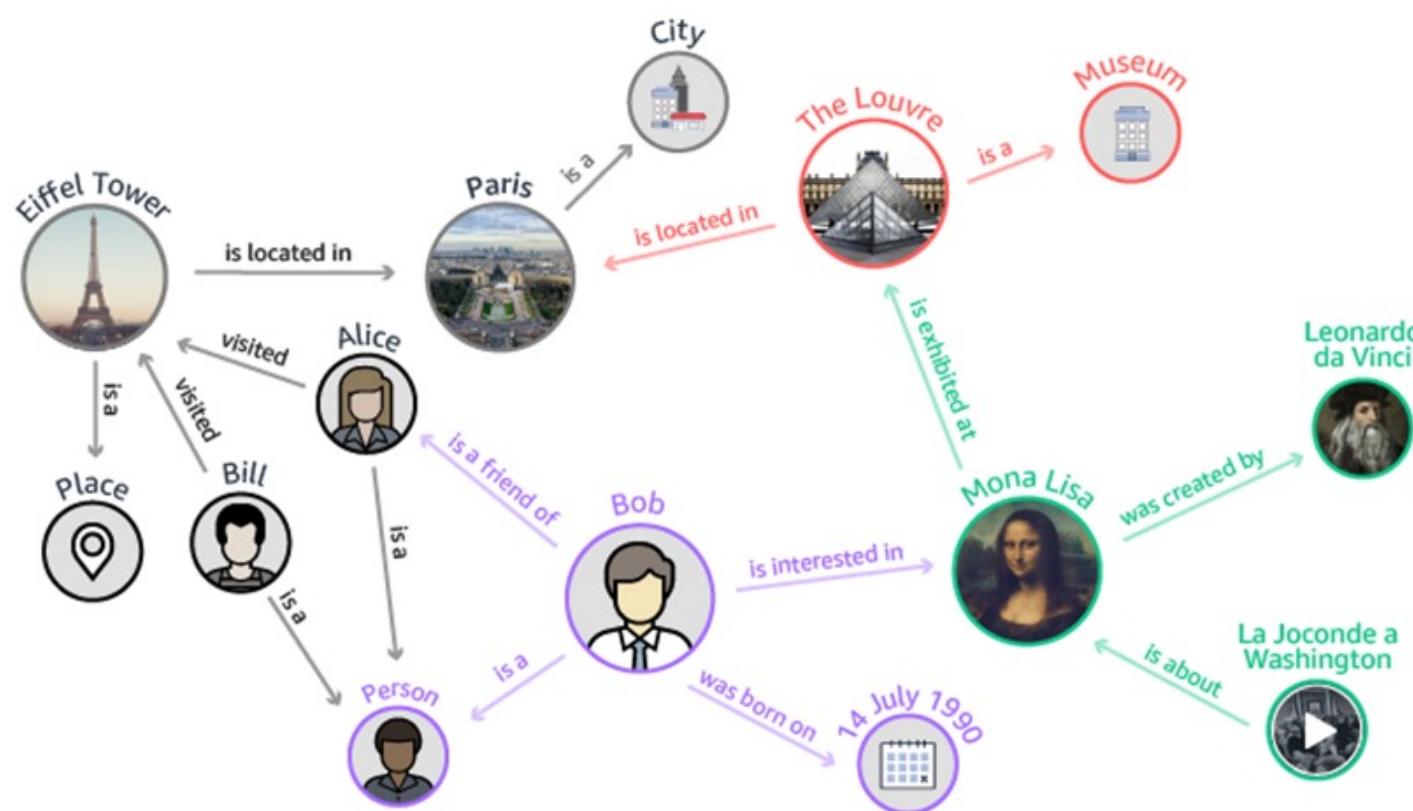
2.6 billions

Tiktok

800 millions

Graphs are Everywhere ...

- Knowledge graphs: commonsense reasoning, question answering



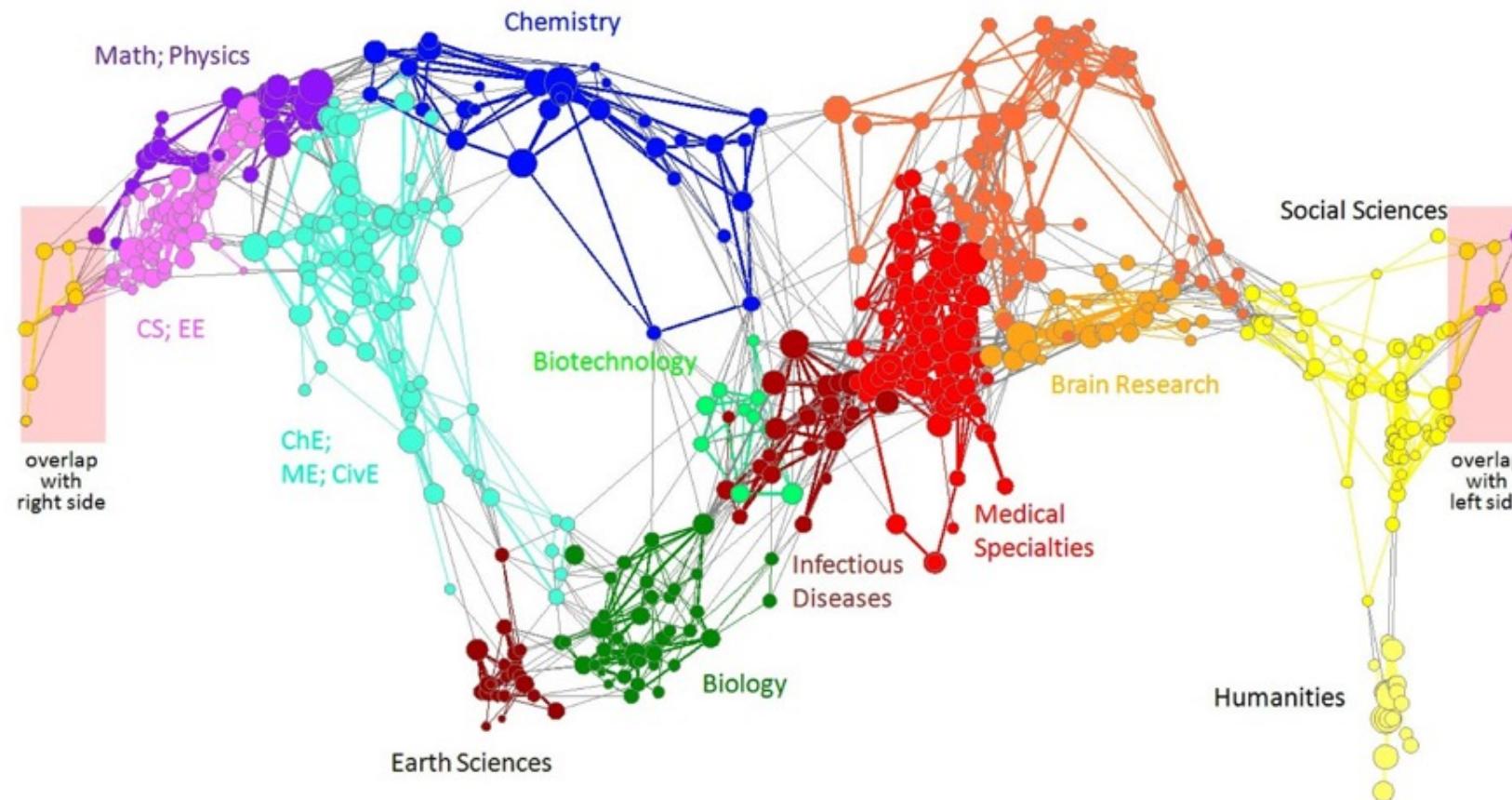
Yago
10M nodes
120M facts

 DBpedia
4.58M nodes
3B facts

 ConceptNet
300K nodes
1.6M facts

Graphs are Everywhere ...

- Citation network (Börner et al., 2012) – grouping articles with topics



Graphs are Everywhere ...

- Web as a graph (4.2 billion webpages) – information retrieval
- Nodes: Webpages, Edges: Hyperlinks



About the school >

Contact our people >

Computer Science study options >

Meet our students and graduates >

Computer Science research >

Data Science: Working with industry >

Our courses and programmes >

Auckland ICT Graduate School >



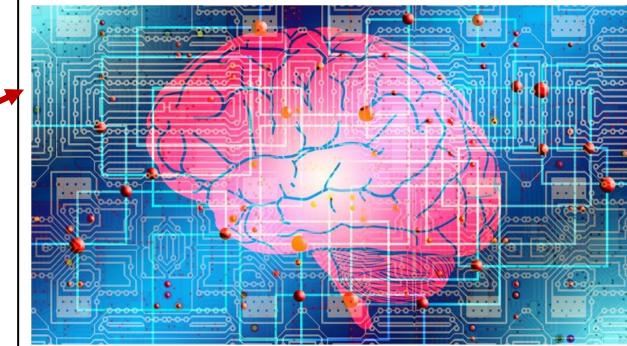
Artificial intelligence and machine learning

Our research covers a broad range of topics, including the cognitive sciences, philosophy of mind, and biologically inspired AI.

Explore >

Artificial intelligence and machine learning

This broad research area covers interdisciplinary collaboration on topics such as the cognitive sciences, philosophy of mind, and biologically inspired AI.



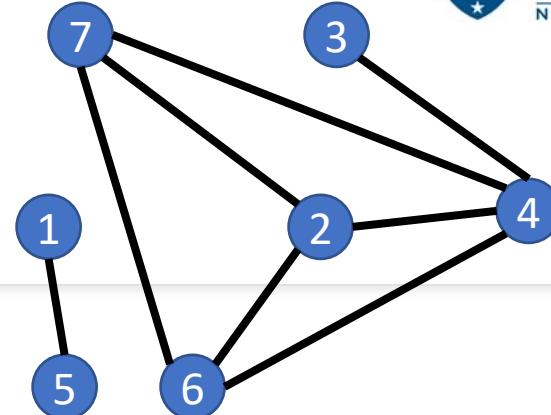
Related links

- School of Computer Science >
- Computer Science research >
- Take 10 with... Mark Wilson >
- High-powered computer sees red >

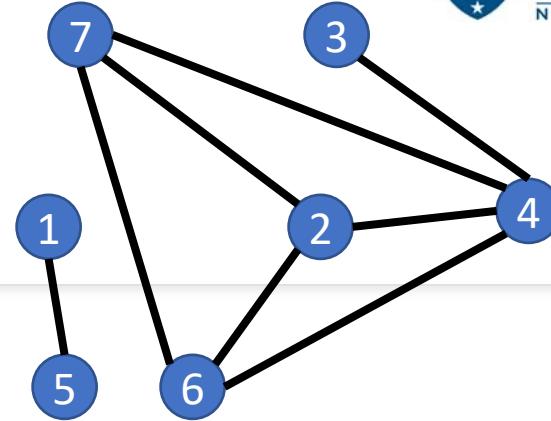
Visit the Machine Learning Group website >

Graphs

- A graph G consists of two sets V and E .
- V is the set of vertices (also called nodes). A vertex (not "vertice"!) is often graphically displayed as a point or junction point (like the seven little circles here).
- E is the set of edges. Each edge connects two vertices u and v in V . Edges have no direction – they equally go from u to v and from v to u . We denote an edge as (u, v) with the understanding that $(u, v) = (v, u)$. The graph here contains eight edges (the black lines).



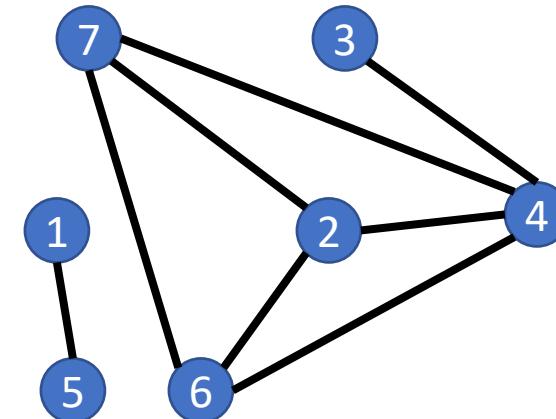
Graphs (Contd.)



- If two vertices u and v are connected by an edge, they are said to be adjacent. We also say that u is a neighbour of v .
- It is not uncommon to number the vertices or otherwise label them uniquely. This is then called a labelled graph.

Graphs (Contd.)

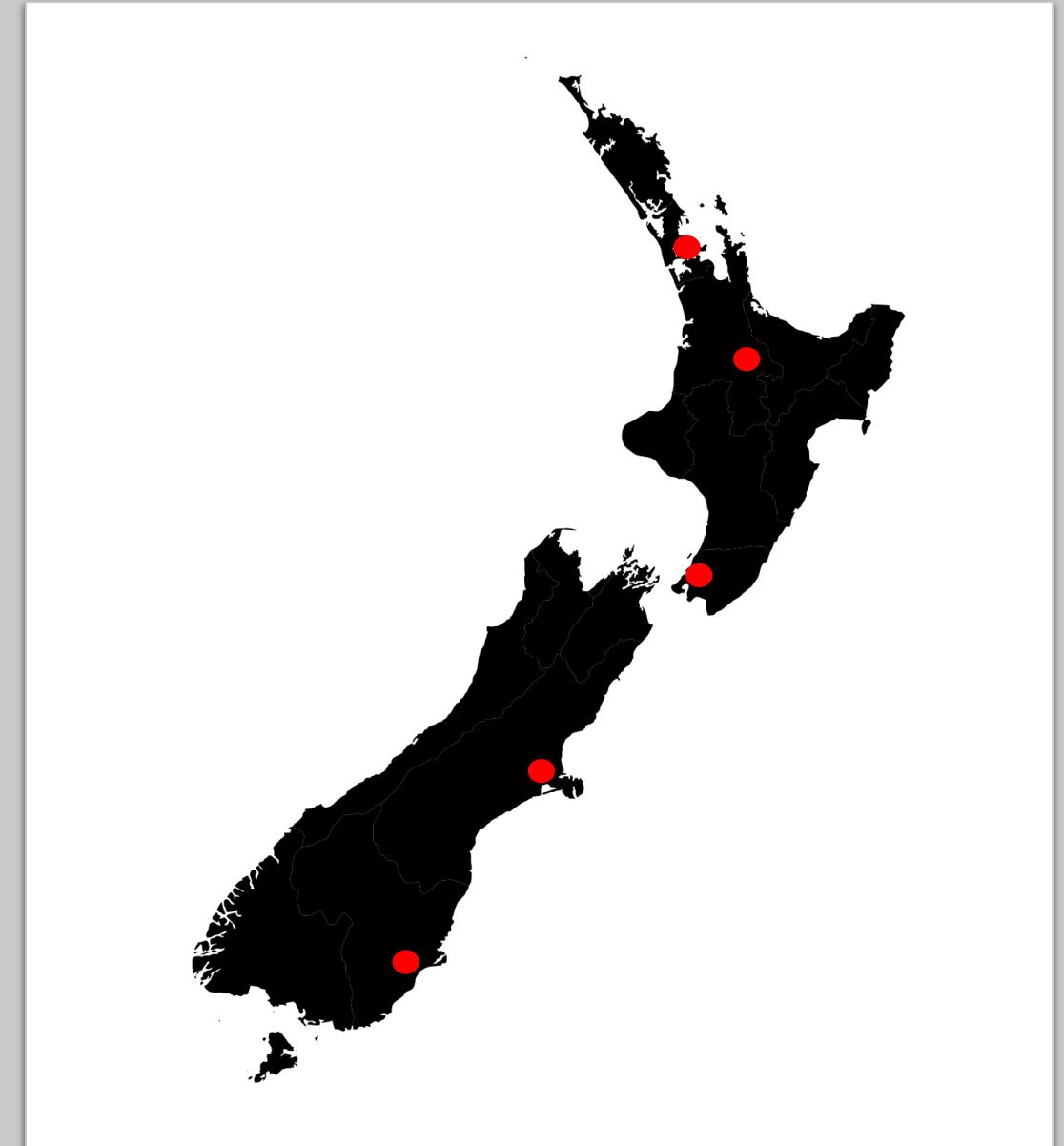
- We also write $G(V, E)$ for the graph itself, or $E(G)$ for the edges, or $V(G)$ for the vertices.
- Note
 - not every vertex of a graph needs to have an edge attached;
 - not every pair of vertices needs to be connected by an edge, and it's perfectly fine to have a graph with many vertices but no edges at all!



This is **one** graph, and it is a **disconnected** graph.

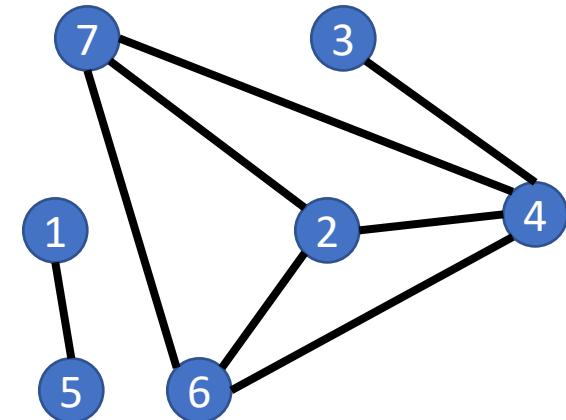
Example: Graph

- Road network does not connect North island cities with South island cities. We have a **disconnected graph**.



Graph: Order and Size

- The **order** of a graph G is the **number of its vertices**: $| V(G) |$
- The **size** of a graph G is the **number of its edges**: $| E(G) |$
- Remember that $| X |$ denotes the number of elements of a set X .



In this example:

$$\begin{aligned}|V(G)| &= ? \\ |E(G)| &= ?\end{aligned}$$

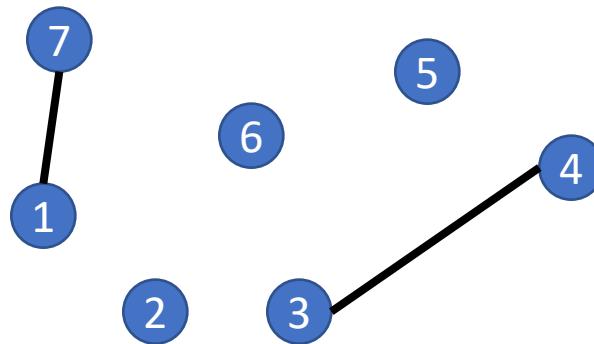
How many edges can a graph have?

- **Observation 1:** At most one edge per pair of different vertices.
- **Observation 2:** There are $|V(G)|(|V(G)| - 1)$ possible such vertex pairs (u, v) , but there can be only one such edge for (u, v) and (v, u) .
- So we need to divide this number by two:
- $0 \leq |E(G)| \leq |V(G)|(|V(G)| - 1)/2$.

Sparse and Dense Graphs

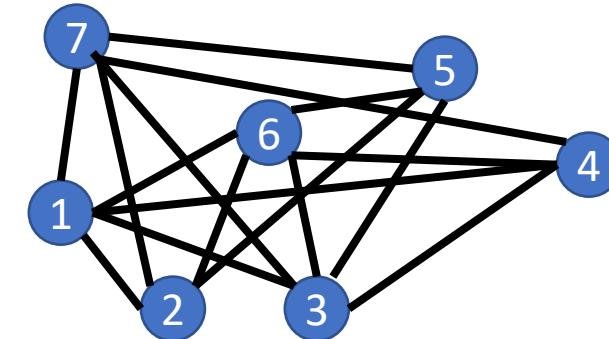
- If $|E(G)|$ approaches the **high** end of the scale $0 \leq |E(G)| \leq |V(G)|(|V(G)| - 1)/2$, we say that the graph is **dense**
- If $|E(G)|$ approaches the **low** end of the scale $0 \leq |E(G)| \leq |V(G)|(|V(G)| - 1)/2$, we say that the graph is **sparse**
- Knowing whether a graph that we are dealing with is dense or sparse makes a difference in how it is best stored when **space efficiency** is important.
- There is no defined boundary between when we would call a graph sparse or dense – but the criterion of storage could help us decide if we wanted such a boundary.

Example: Sparse and Dense Graphs



A **sparse** graph
(2 out of 21 possible edges)

Generally more efficient to store edges by recording between which vertices they occur

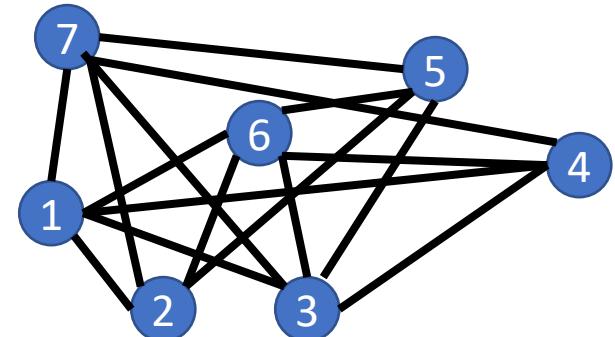


A **dense** graph
(16 out of 21 possible edges)

Generally more efficient to store edges by listing vertex pairs and recording between which pairs there are edges

Degree of a Node in a Graph

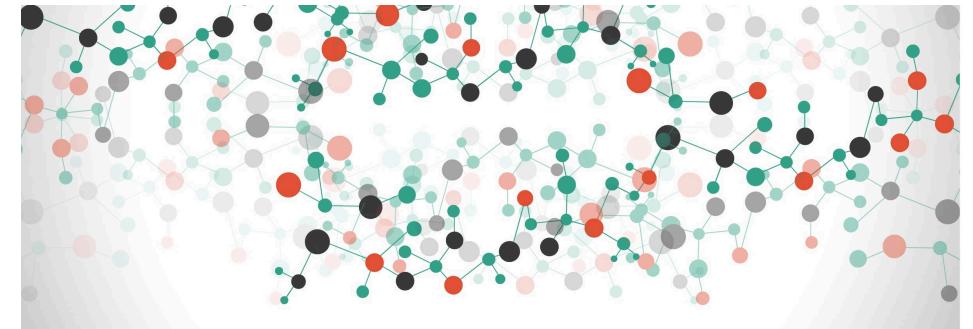
- The **degree** of a vertex v is the **number of edges** that terminate in this vertex
- **Observation 1:** The number of edges is the sum of the degrees of all vertices divide by 2
- **Observation 2:** In sparse graphs, the nodes tend to have a lower degree than in dense graphs.



Node 1: degree 5
Node 2: degree 4
Node 3: degree 5
Node 4: degree 4
Node 5: degree 4
Node 6: degree 5
Node 7: degree 5

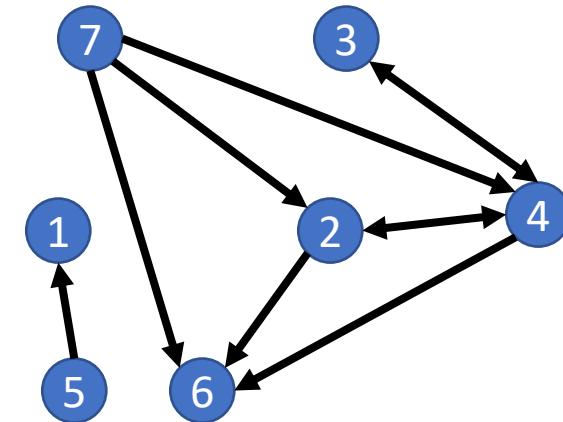
OUTLINE

- Graphs and Di-Graphs
- Preliminaries
- Basic operations



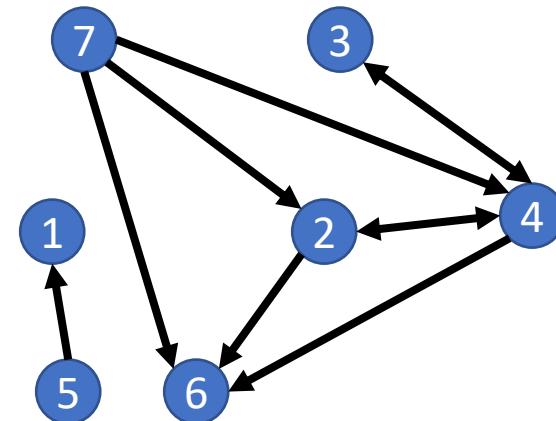
Directed Graphs

- The set E of edges is replaced by a set E of **arcs**.
- An arc has a **direction**. An edge runs between two vertices u and v , an arc runs from a vertex u to a vertex v
- Draws the arcs curved allows it to have them between two vertices in both directions. We use double-headed arrows here to indicate that two vertices are connected by arcs in both directions
- This also requires us to extend and amend our previous definitions and terminology (next slide).



Neighborhood

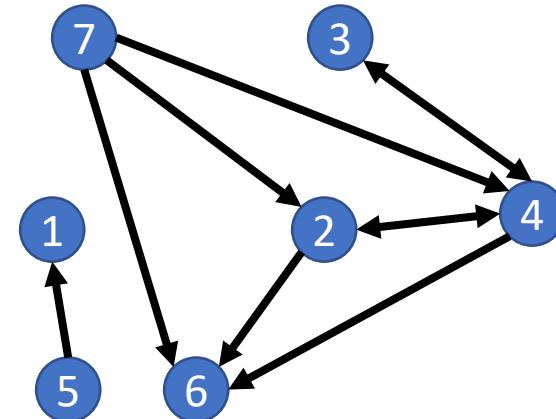
- There are **two types of neighbours**
 - A neighbour v of u is an **out-neighbour** of u if there is an arc from u to v .
 - A neighbour v of u is an **in-neighbour** of u if there is an arc from v to u .
- There are **two types of degrees**
 - The **out-degree** (number of out-neighbours) and
 - The **in-degree** (number of in-neighbours).
- Obviously, now $(u, v) \neq (v, u)$.
- Size of a digraph = number of arcs



- 1 is an out-neighbour of 5
- 2 is an in-neighbour of 6
- 2 is an in-neighbour and out-neighbour of 4 (and vice versa)
- 7 has in-degree 0 and out-degree 3
- 4 has in-degree 3 and out-degree 3
- Size is 10

Sources and Sinks

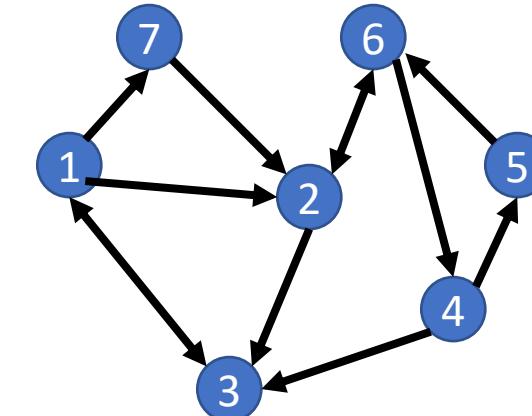
- A vertex with **in-degree 0** is called a **source** (all arcs "flow out" from the vertex).
- A vertex with **out-degree 0** is called a **sink** (all arcs "flow into" the vertex).
- Question: Is it possible to have a node that is both a source as well as sink?



- 1 and 6 are sinks
- 5 and 7 are sources
- The other nodes are neither sinks nor sources

Walk, Path, Cycle

- A **walk** is a sequence of vertices v_0, v_1, \dots, v_n , such that (v_i, v_{i+1}) is an arc in E for $0 \leq i < n$
- A walk **CAN** pass by the same vertex twice, i.e., $v_i = v_j$ is possible even for $i \neq j$
- The length of the walk is n . This is the **number of arcs** involved.



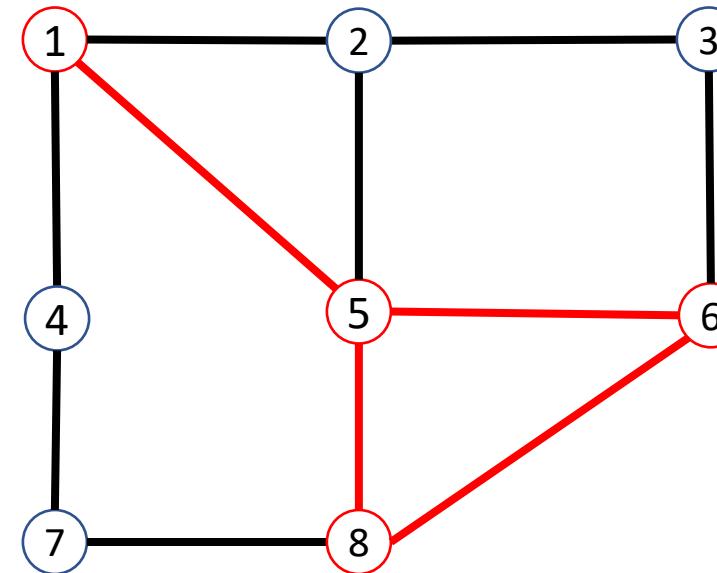
“4 5 6 4 3 1 2 3 1 7 2” is a walk

Walk, Path, Cycle (Contd.)

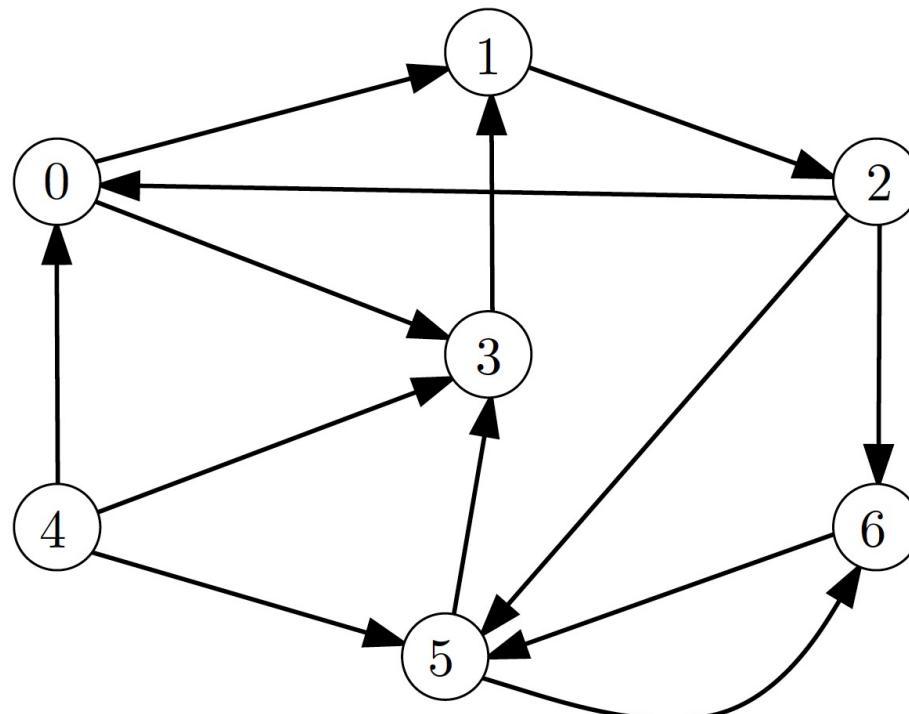
- A **path** is a walk in which **NO** vertex is repeated
- A **cycle** is a walk of length 3 or more on a graph or any walk on a digraph where $v_0 = v_n$,
 - A walk that ends in the same vertex that it started in.
 - No vertex is repeated other than the vertex at the start and end.

Exercise: This is NOT a Cycle

(1,5,6,8,5,1)

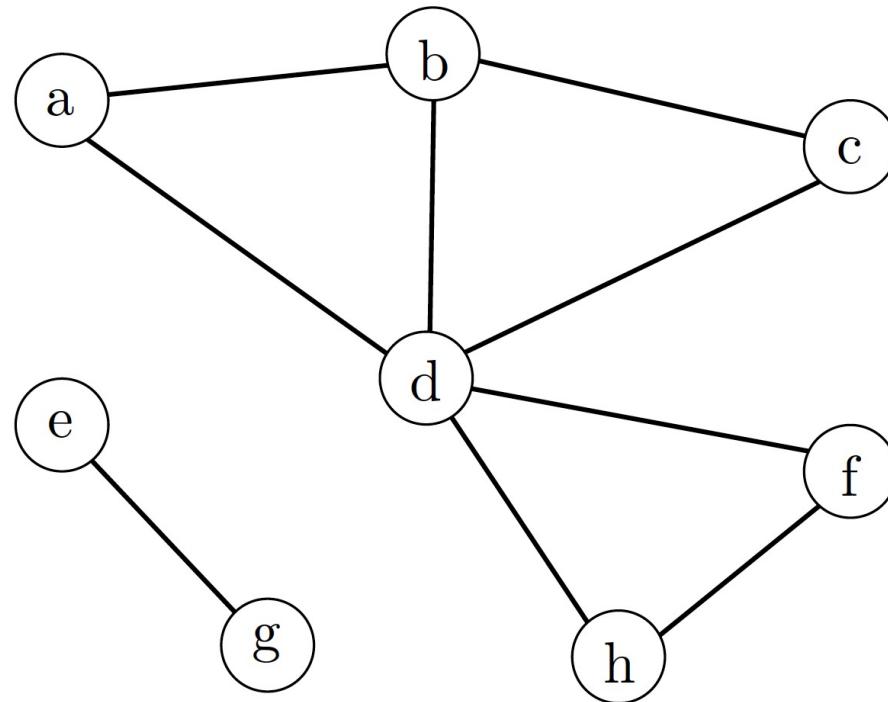


Exercise: Walks, Paths, and Cycles in a Digraph



| Sequence | Walk | Path? | Cycle? |
|-------------|------|-------|--------|
| 0 2 3 | no | no | no |
| 3 1 2 | yes | yes | no |
| 1 2 6 5 3 1 | yes | no | yes |
| 4 5 6 5 | yes | no | no |
| 4 3 5 | no | no | no |

Exercise: Walks, Paths, and Cycles in a Graph

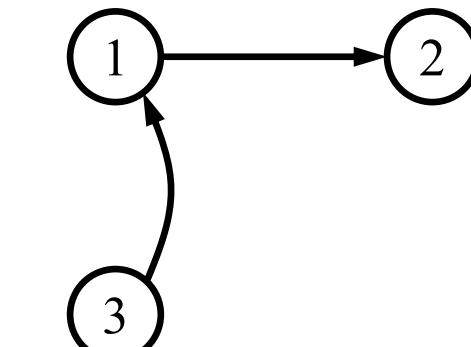
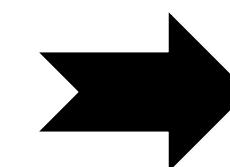
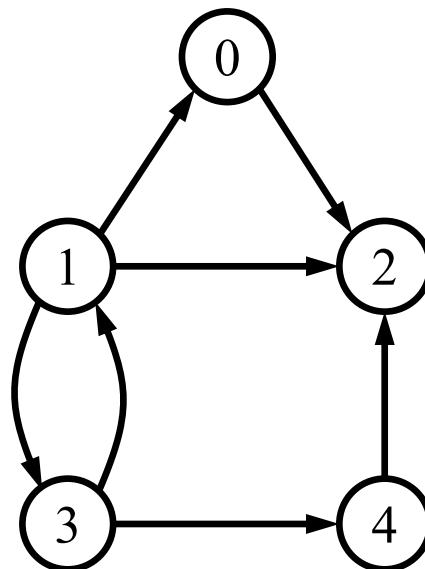


| Sequence | Walk | Path? | Cycle? |
|---------------------|------|-------|--------|
| $a \ b \ c$ | yes | yes | no |
| $e \ g \ e$ | yes | no | no |
| $d \ b \ c \ d$ | yes | no | yes |
| $d \ a \ d \ f$ | yes | no | no |
| $a \ b \ d \ f \ h$ | yes | yes | no |

Subgraphs and Subdigraphs

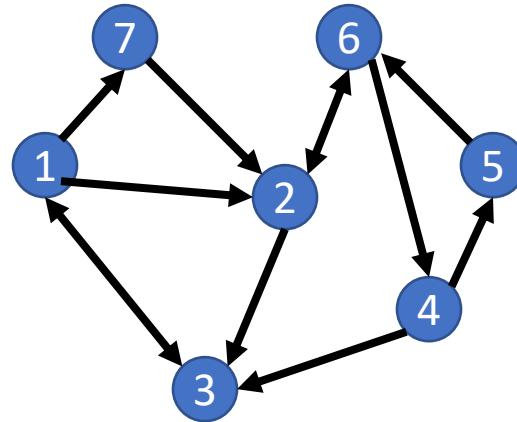
- A **sub(di)graph** $G' = (V', E')$ of a (di)graph $G = (V, E)$ is a (di)graph for which $V' \subseteq V$ and $E' \subseteq E$.

$$G = \left(\begin{array}{l} V = \{0,1,2,3,4\} \\ E = \left\{ \begin{array}{l} (0,2), (1,0), (1,2), \\ (1,3), (3,1), (4,2), \\ (3,4) \end{array} \right\} \end{array} \right)$$

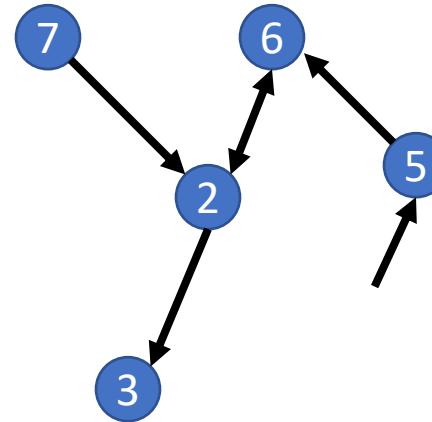


$$G' = \left(\begin{array}{l} V' = \{1,2,3\} \\ E' = \{(1,2), (3,1)\} \end{array} \right)$$

Exercise: Subdigraph or not?

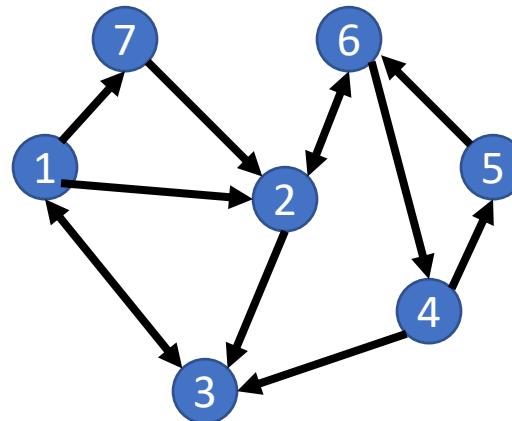


G

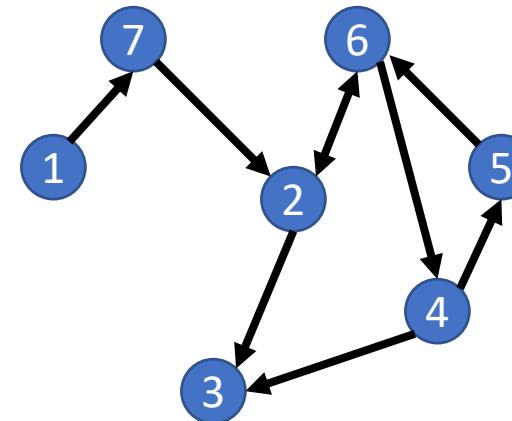


G'

Exercise: Subdigraph or not?



G

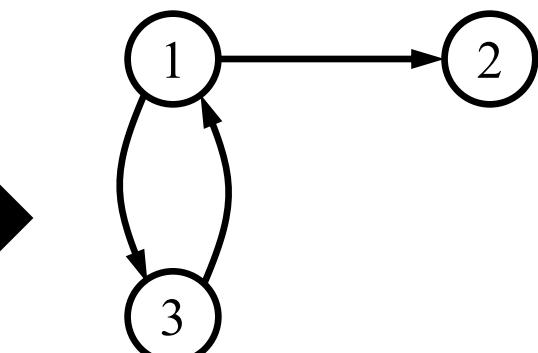
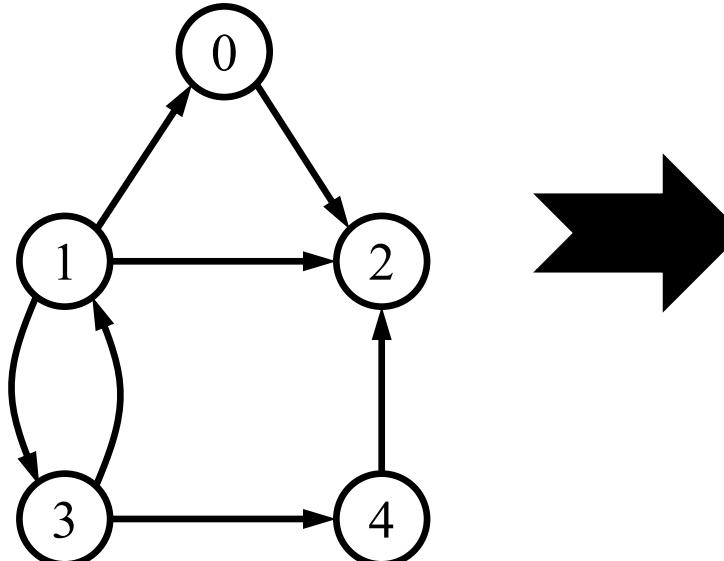


G'

Induced Sub(di)graphs

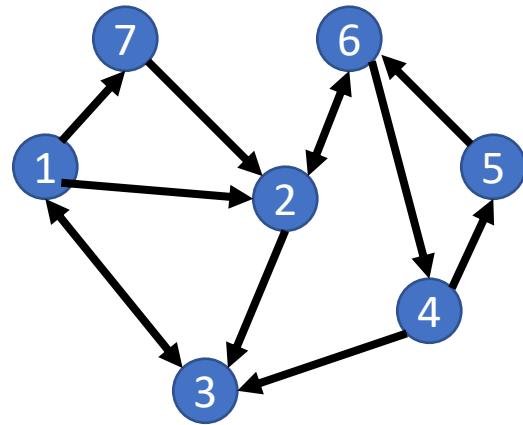
- A **subdigraph induced** by a subset V' of V is the digraph $G' = (V', E')$ where $E' = \{(u, v) \in E \mid u \in V' \text{ and } v \in V'\}$.

$$G = \left(\begin{array}{l} V = \{0, 1, 2, 3, 4\} \\ E = \left\{ \begin{array}{l} (0, 2), (1, 0), (1, 2), \\ (1, 3), (3, 1), (4, 2), \\ (3, 4) \end{array} \right\} \end{array} \right)$$

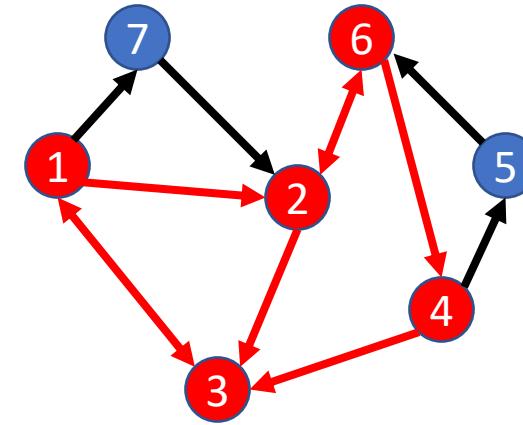


$$G' = \left(\begin{array}{l} V' = \{1, 2, 3\} \\ E' = \left\{ \begin{array}{l} (1, 2), (1, 3), \\ (3, 1) \end{array} \right\} \end{array} \right)$$

Example: An Induced Subdigraph



G



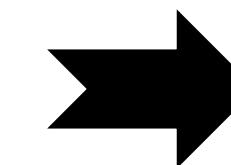
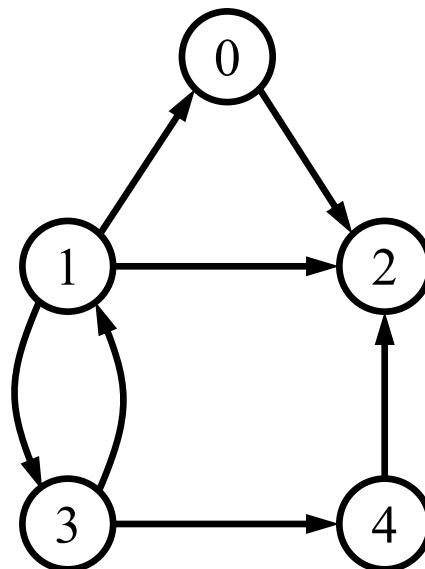
G'

E.g., $V' = \{1, 2, 3, 4, 6\}$

Spanning Sub(di)graphs

- A **spanning subdigraph** contains all nodes, that is, $V'=V$.

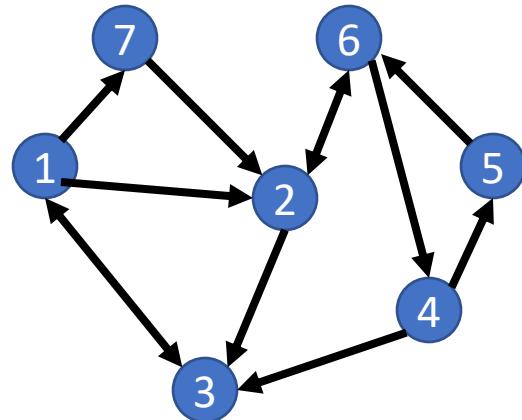
$$G = \left(\begin{array}{l} V = \{0,1,2,3,4\} \\ E = \left\{ \begin{array}{l} (0,2), (1,0), (1,2), \\ (1,3), (3,1), (4,2), \\ (3,4) \end{array} \right\} \end{array} \right)$$



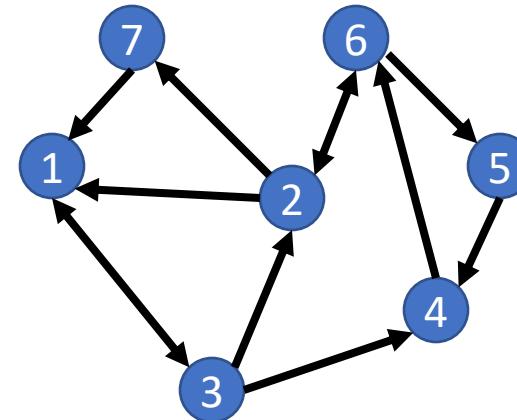
$$G' = \left(\begin{array}{l} V' = \{0,1,2,3,4\} \\ E' = \left\{ \begin{array}{l} (0,2), (1,2), \\ (3,4) \end{array} \right\} \end{array} \right)$$

Reverse Digraph

- The **reverse digraph** of the digraph $G=(V,E)$, is the digraph $Gr=(V,E')$ where $(u, v) \in E'$ if and only if $(v, u) \in E$.



G

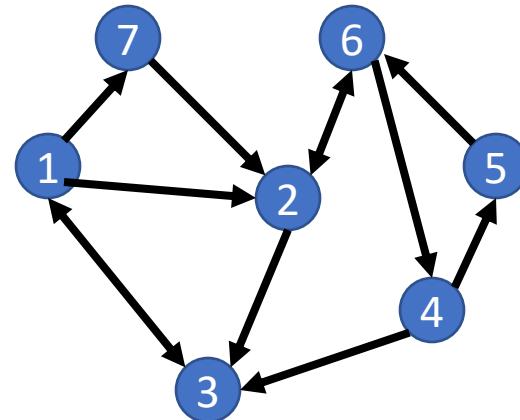


G_r

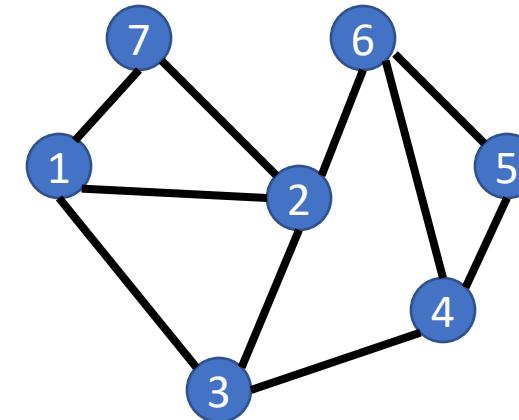
We simply reverse all the arrows

Underlying Graph of a Digraph

- The underlying graph of a digraph $G=(V,E)$ is the graph $G'=(V,E')$ where $E'=\{\{u, v\} \mid (u, v) \in E\}$.



G



G'

SUMMARY

- Definition of Graphs and Di-Graphs
- Preliminaries on terminology and properties (e.g., size, order, degree, walk, path, cycle)
- Basic operations
 - walking and sub-graphing

