POLITECNICO DI TORINO

e-Lite

Introduction to Graphs

Tecniche di Programmazione – A.A. 2015/2016

Summary

- Definition: Graph
- Related Definitions
- Applications
- Graph representation
- Graph visits

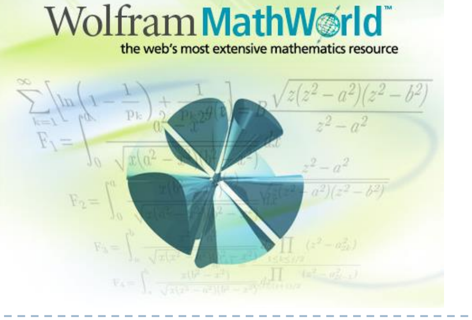
Definition: Graph

Introduction to Graphs

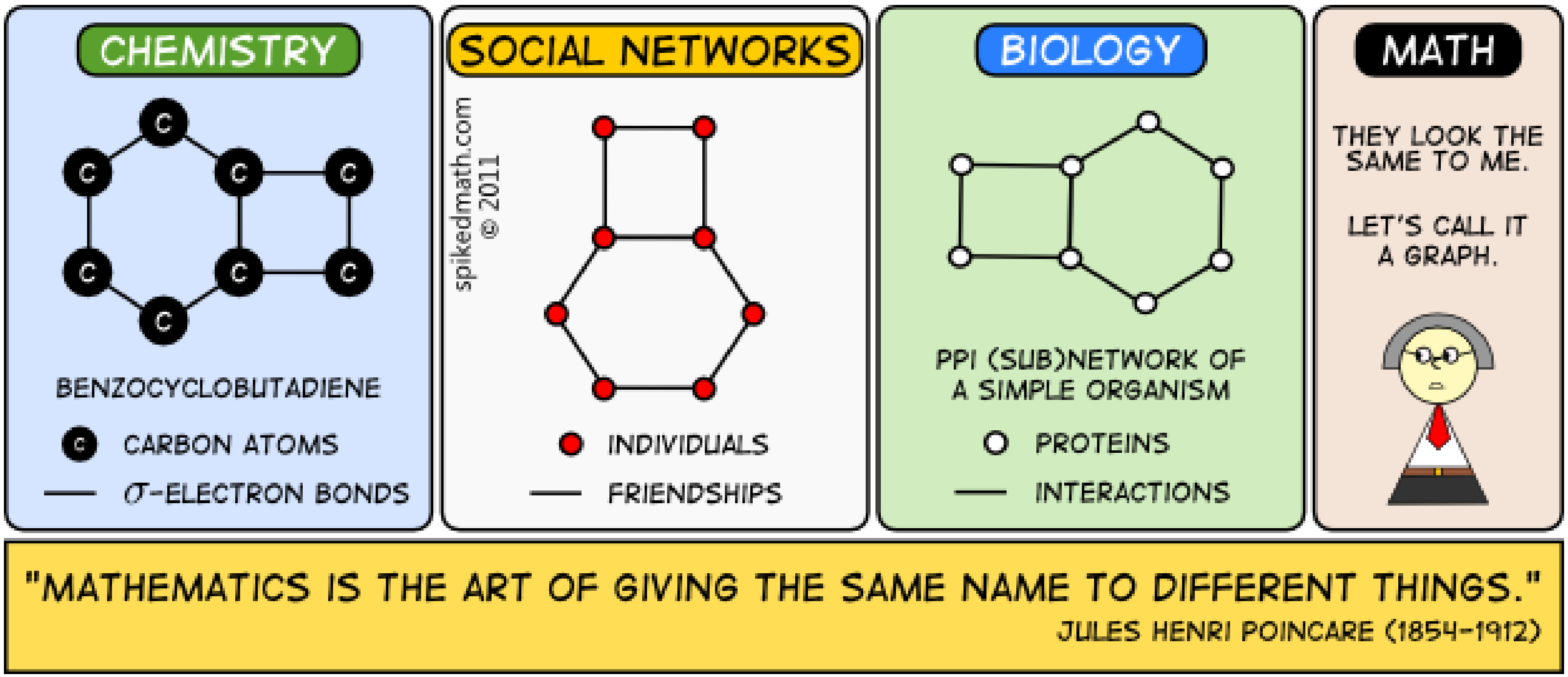
Definition: **Graph**

- A **graph** is a collection of **points** and **lines** connecting some (possibly empty) subset of them.
- The points of a graph are most commonly known as **graph vertices**, but may also be called "nodes" or simply "points."
- The lines connecting the vertices of a graph are most commonly known as **graph edges**, but may also be called "arcs" or "lines."

http://mathworld.wolfram.com/



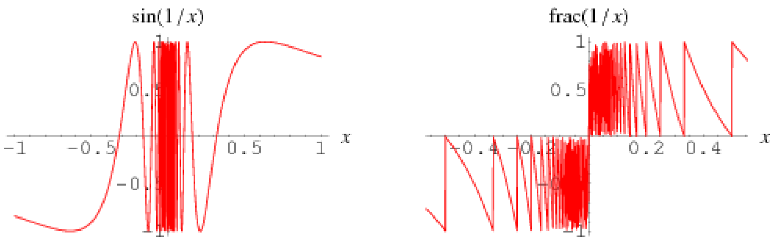
What's in a name?



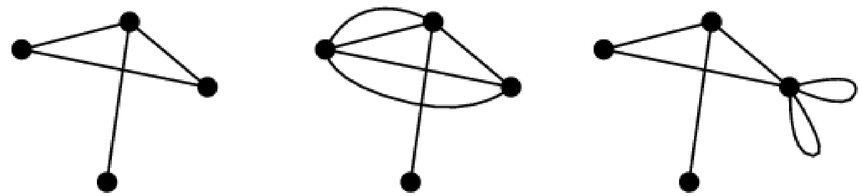
http://spikedmath.com/382.html

Big warning: Graph ≠ Graph ≠ Graph

Graph (plot)
(italiano: grafico)



Graph (maths)
(italiano: grafo)



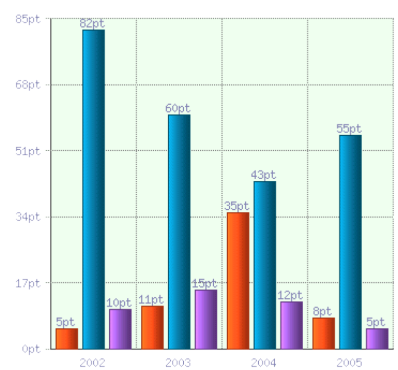
simple graph

nonsimple graph with multiple edges

nonsimple graph with loops

≠

Graph (chart)
(italiano: grafico)



History

- The study of graphs is known as **graph theory**, and was first systematically investigated by D. König in the 1930s
- Euler's proof about the *walk across all seven bridges of Königsberg* (1736), now known as the *Königsberg bridge* problem, is a famous precursor to graph theory.
- In fact, the study of various sorts of paths in graphs has many applications in real-world problems.

Königsberg Bridge Problem

- Can the 7 bridges the of the city of Königsberg over the river Preger all be traversed in a single trip without doubling back, with the additional requirement that the trip ends in the same place it began?

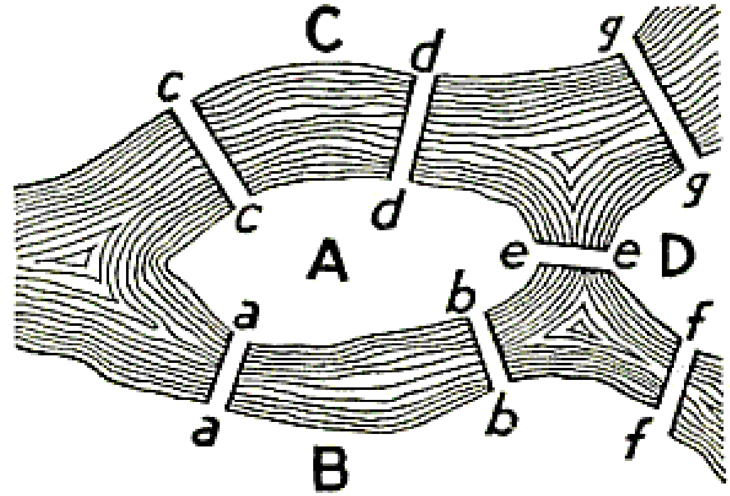
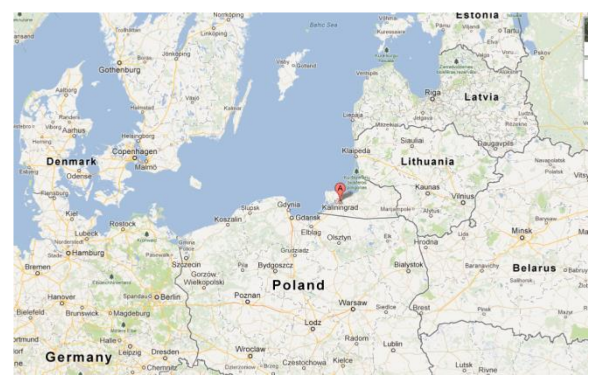


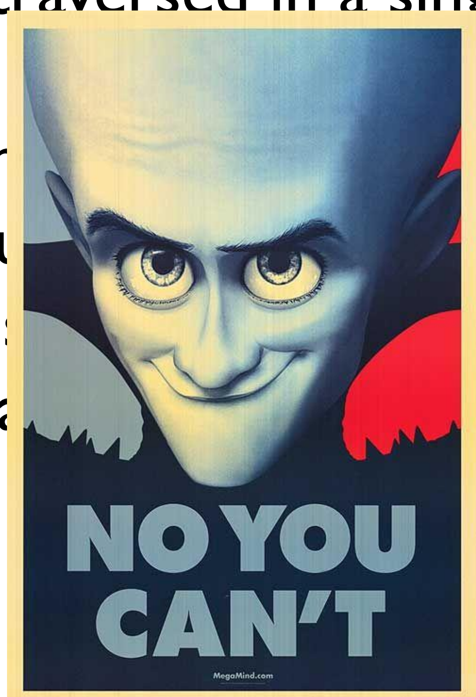
FIGURE 98. *Geographic Map: The Königsberg Bridges.*



Today: Kaliningrad, Russia

Königsberg Bridge Problem

- Can the 7 bridges the of the city of Königsberg over the river Preger all be traversed in a single
  trip … g back,
  with …
  requ … e trip
  ends … ce it
  beg…



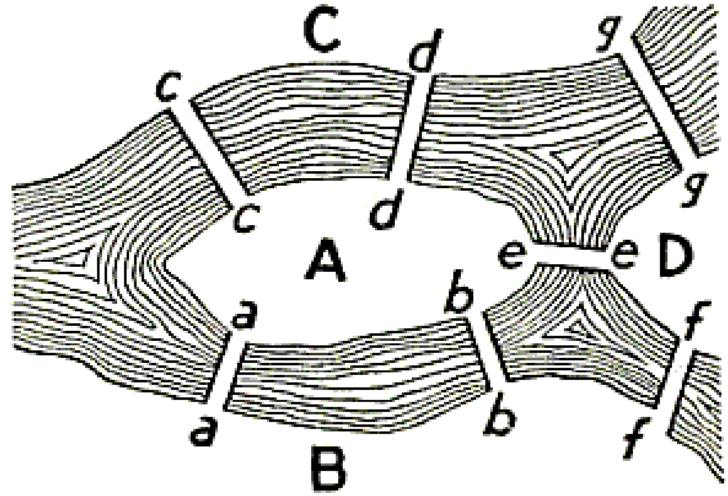
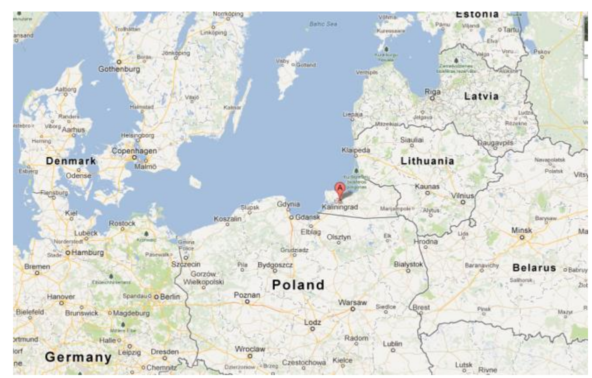
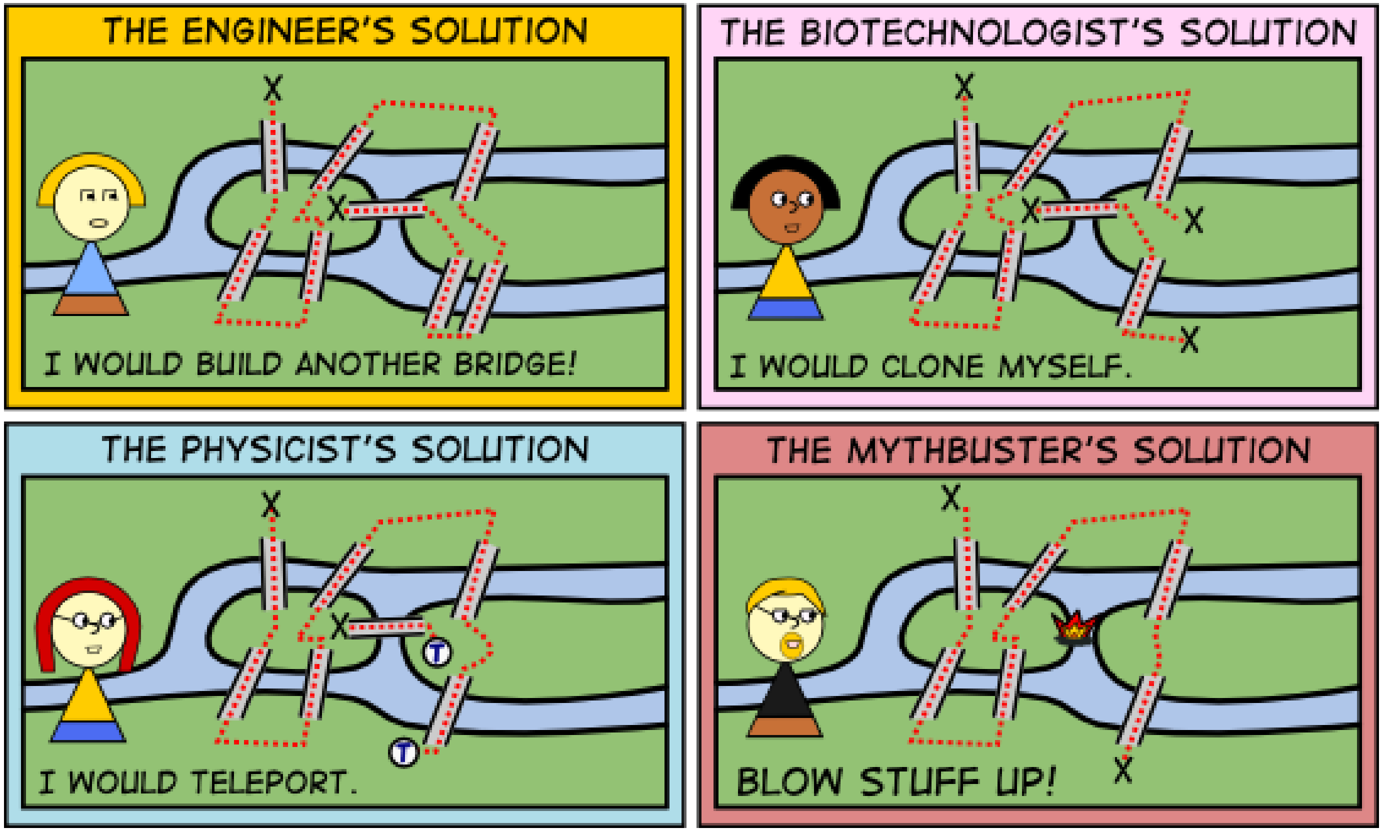


FIGURE 98. *Geographic Map: The Königsberg Bridges.*



Today: Kaliningrad, Russia

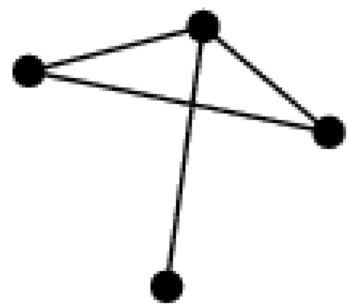
Unless…



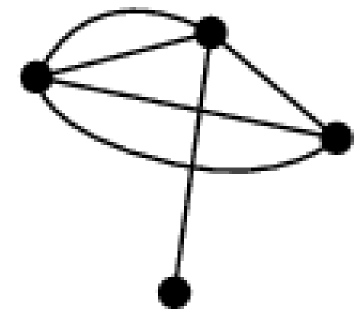
http://spikedmath.com/541.html

Types of graphs: edge cardinality

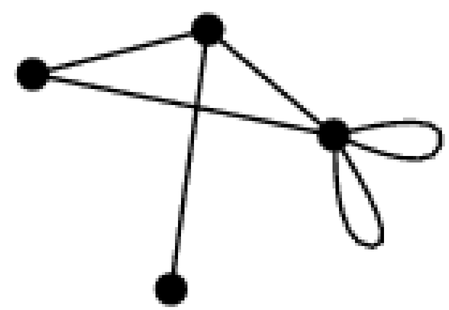
- Simple graph:
  - At most one edge (i.e., either one edge or no edges) may connect any two vertices
- Multigraph:
  - Multiple edges are allowed between vertices
- Loops:
  - Edge between a vertex and itself
- Pseudograph:
  - Multigraph with loops



simple graph



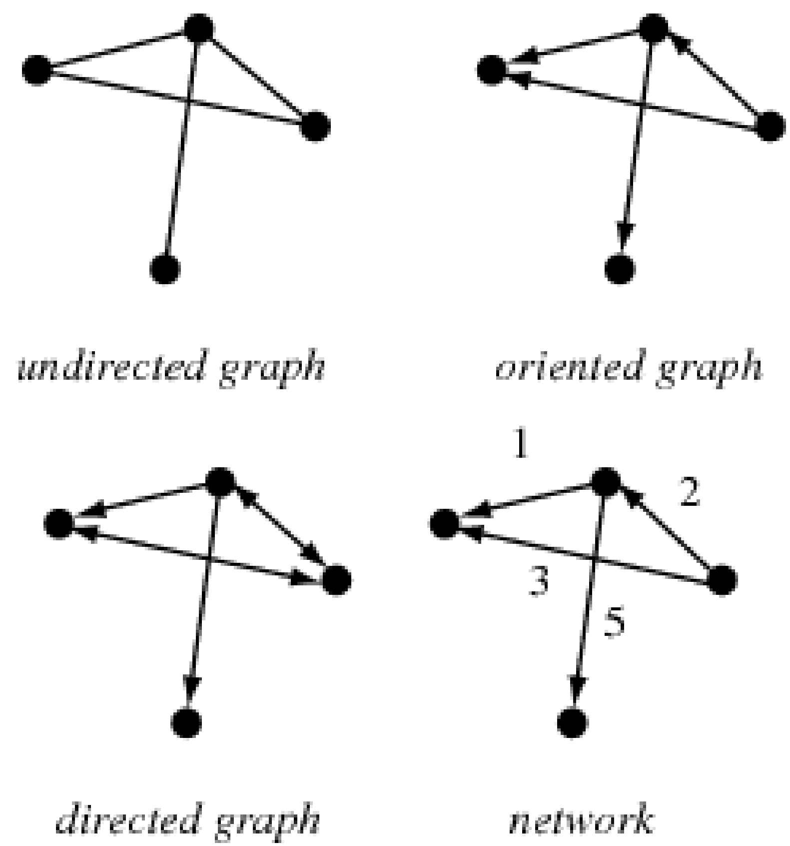
multigraph



pseudograph

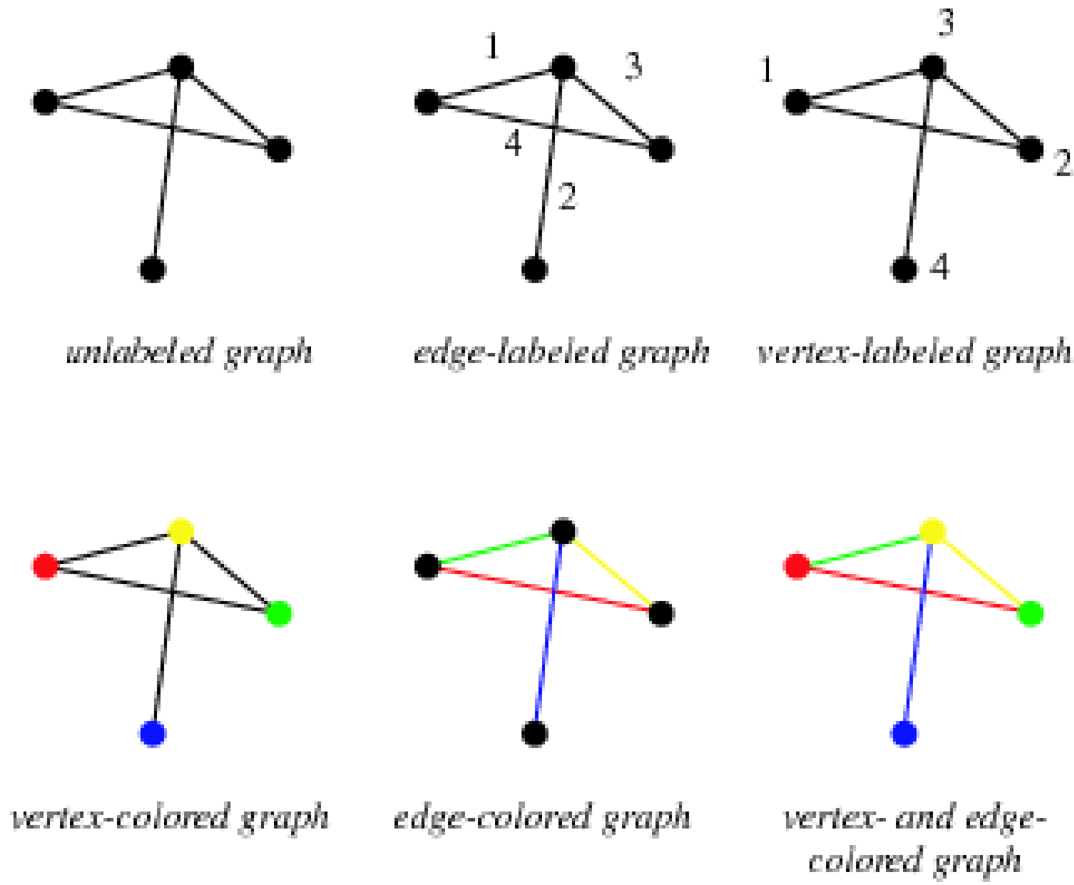
Types of graphs: edge direction

- Undirected
- Oriented
  - Edges have **one** direction (indicated by arrow)
- Directed
  - Edges may have **one or two** directions
- Network
  - Oriented graph with weighted edges



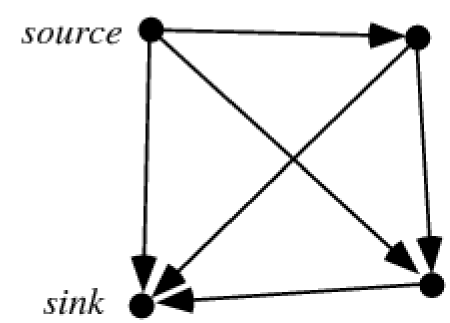
Types of graphs: labeling

- Labels
  - None
  - On Vertices
  - On Edges
- Groups (=colors)
  - Of Vertices
    - no edge connects two identically colored vertices
  - Of Edges
    - adjacent edges must receive different colors
  - Of both

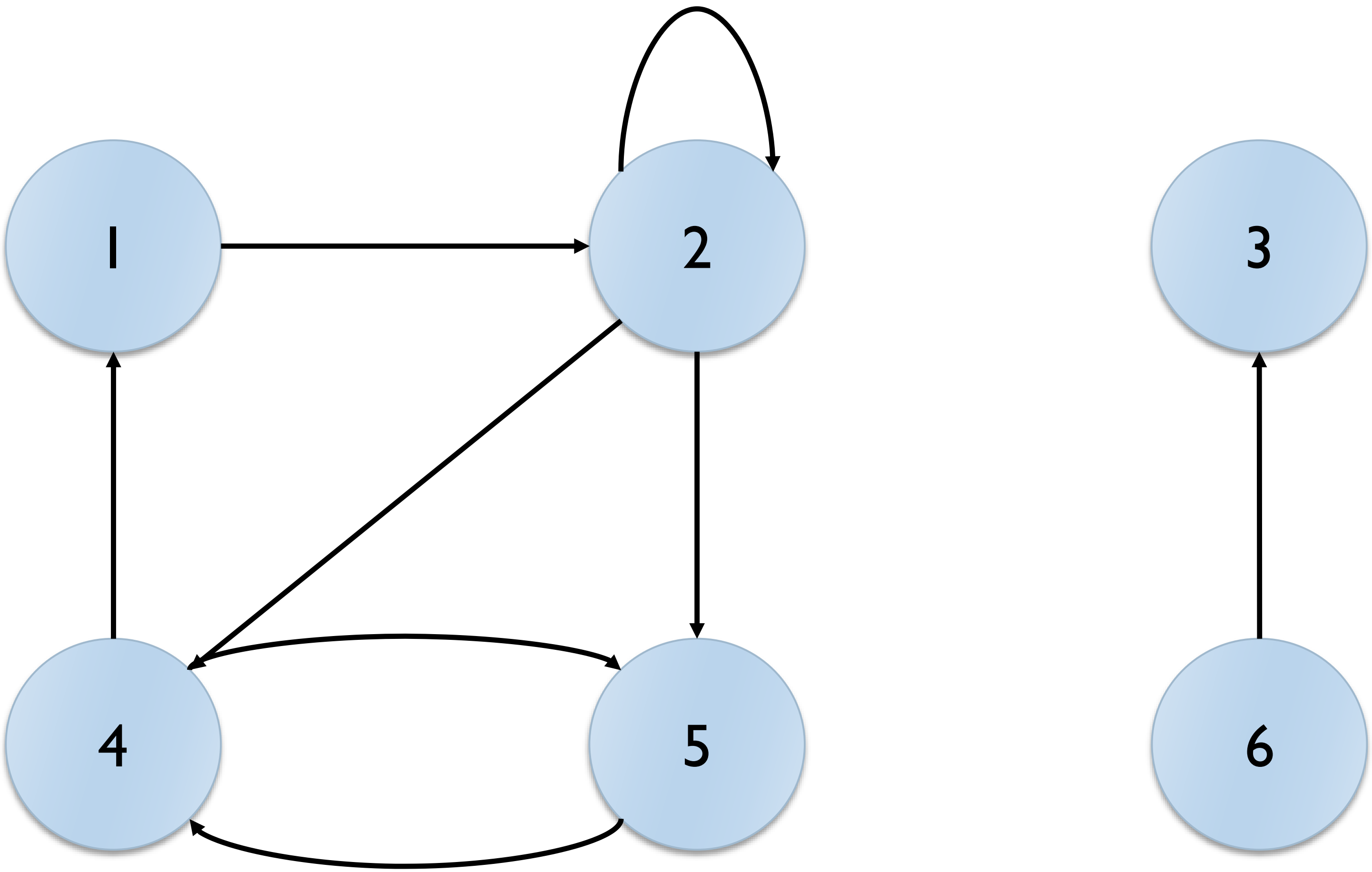


Directed and Oriented graphs

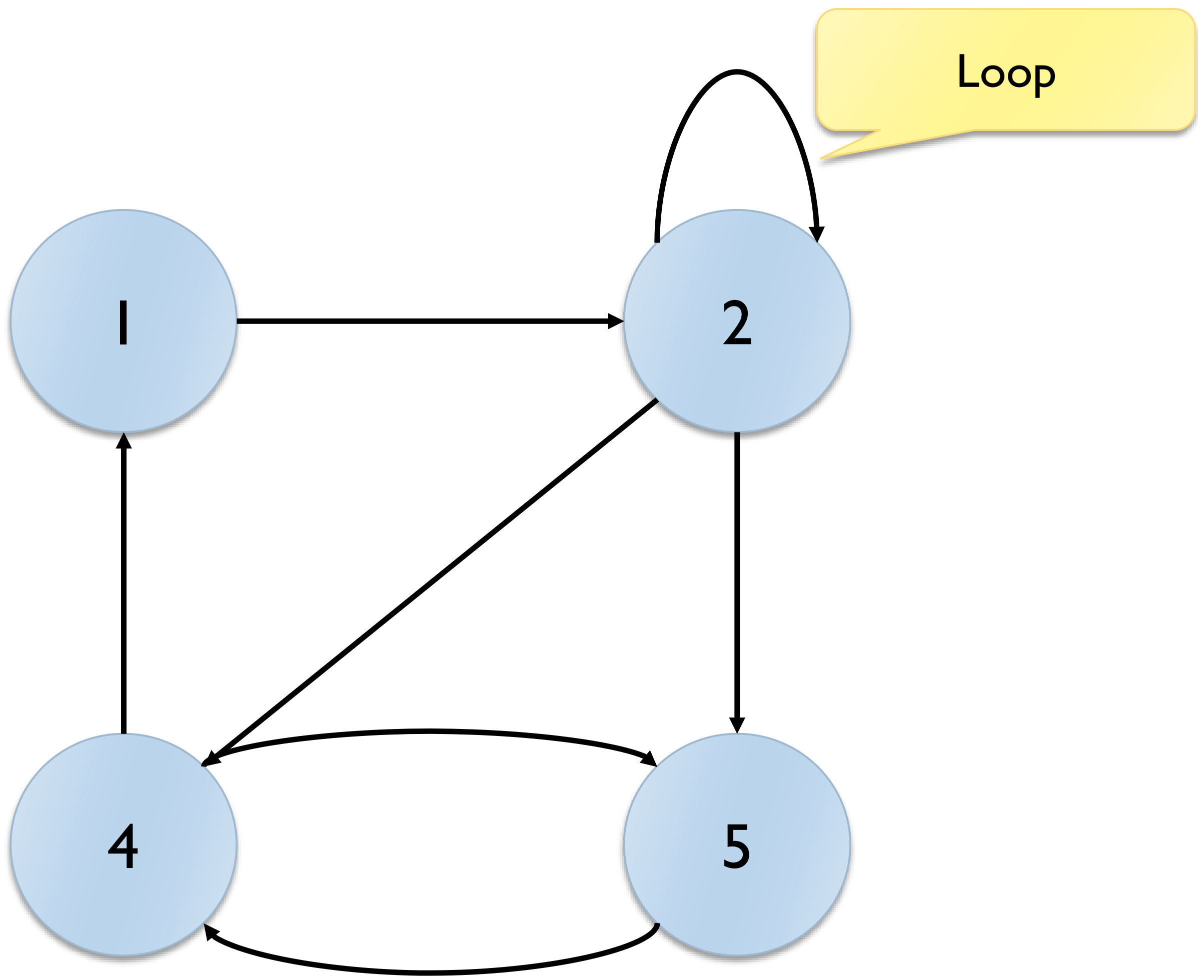
- A Directed Graph (*di-graph*) G is a pair (V,E), where
  - V is a (finite) set of *vertices*
  - E is a (finite) set of *edges*, that identify a binary relationship over V
    - $E \subseteq V \times V$



Example



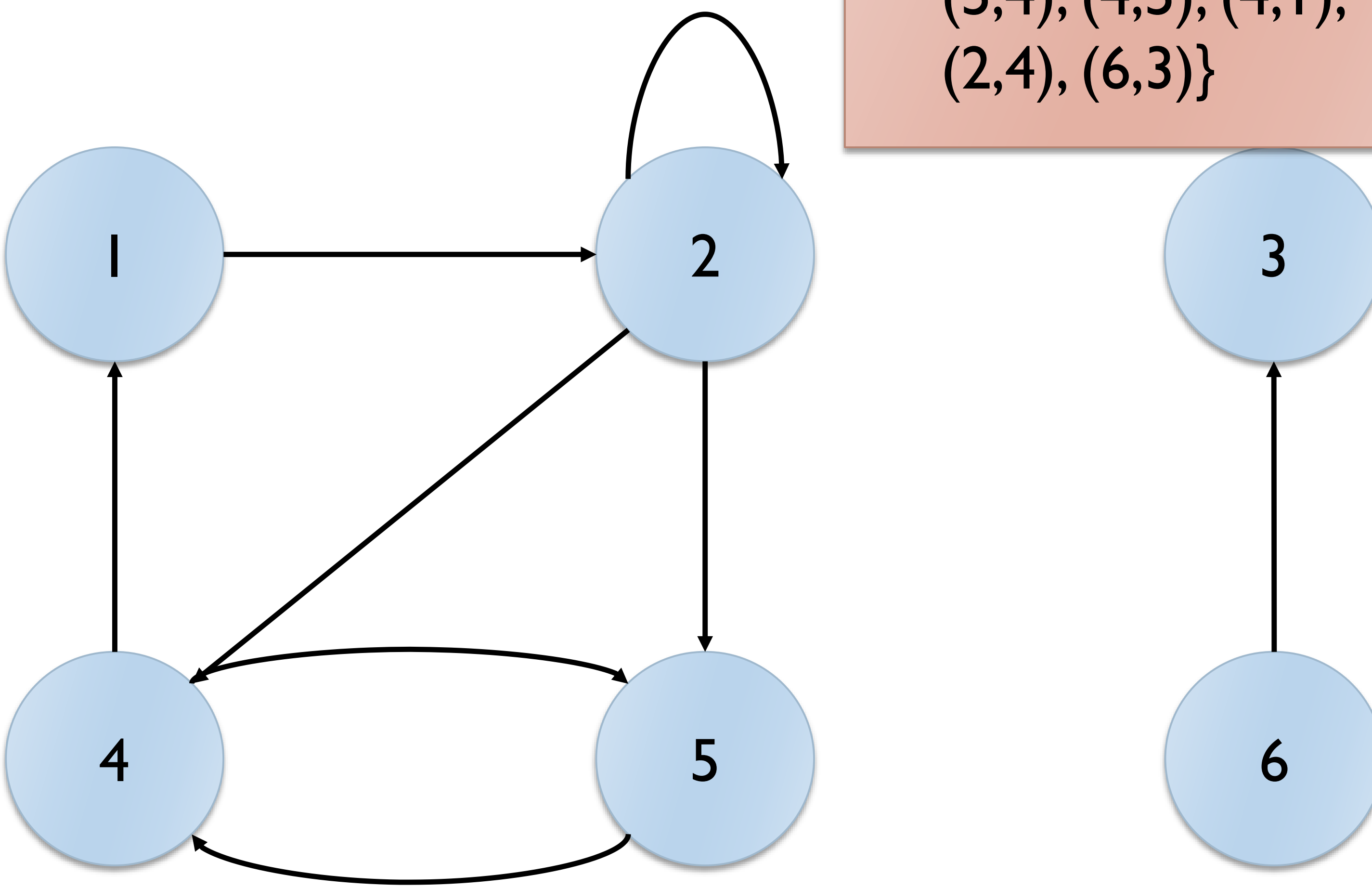
Example



Example

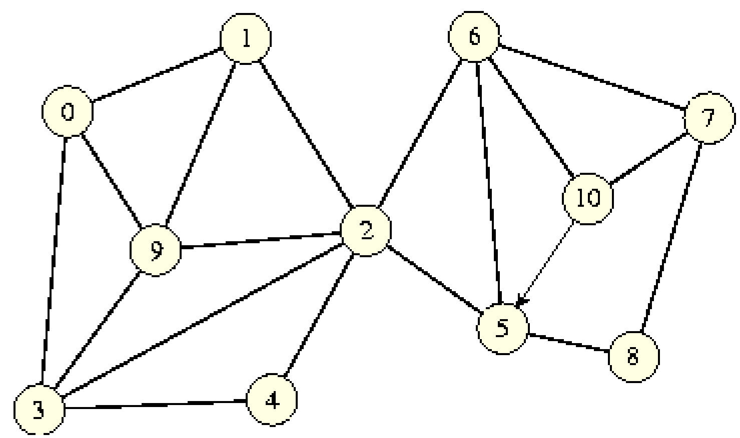
V={1,2,3,4,5,6}

E={(1,2), (2,2), (2,5), (5,4), (4,5), (4,1), (2,4), (6,3)}



Undirected graph

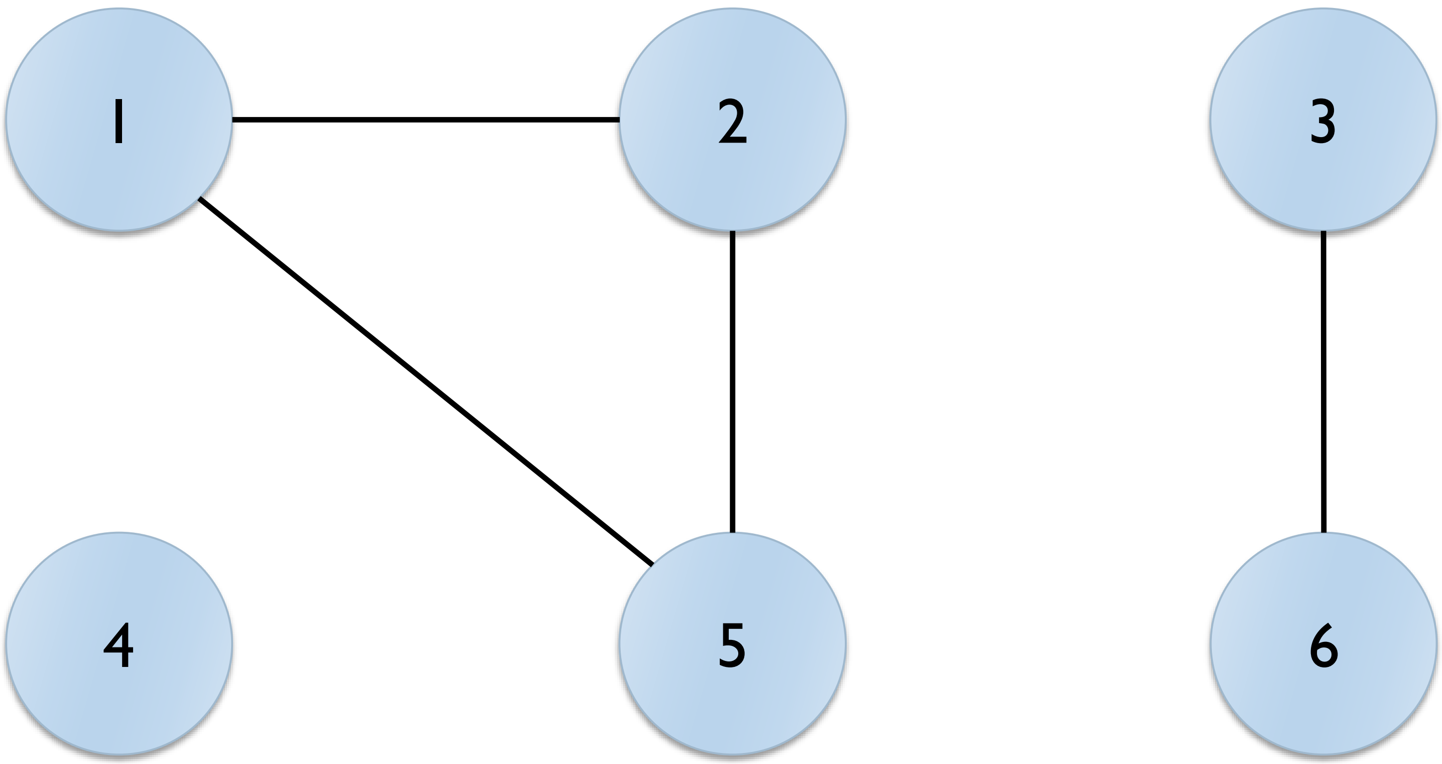
- Ad **Undirected** Graph is still represented as a couple G=(V,E), but the set E is made of **non-ordered pairs** of vertices



Example

V={1,2,3,4,5,6}

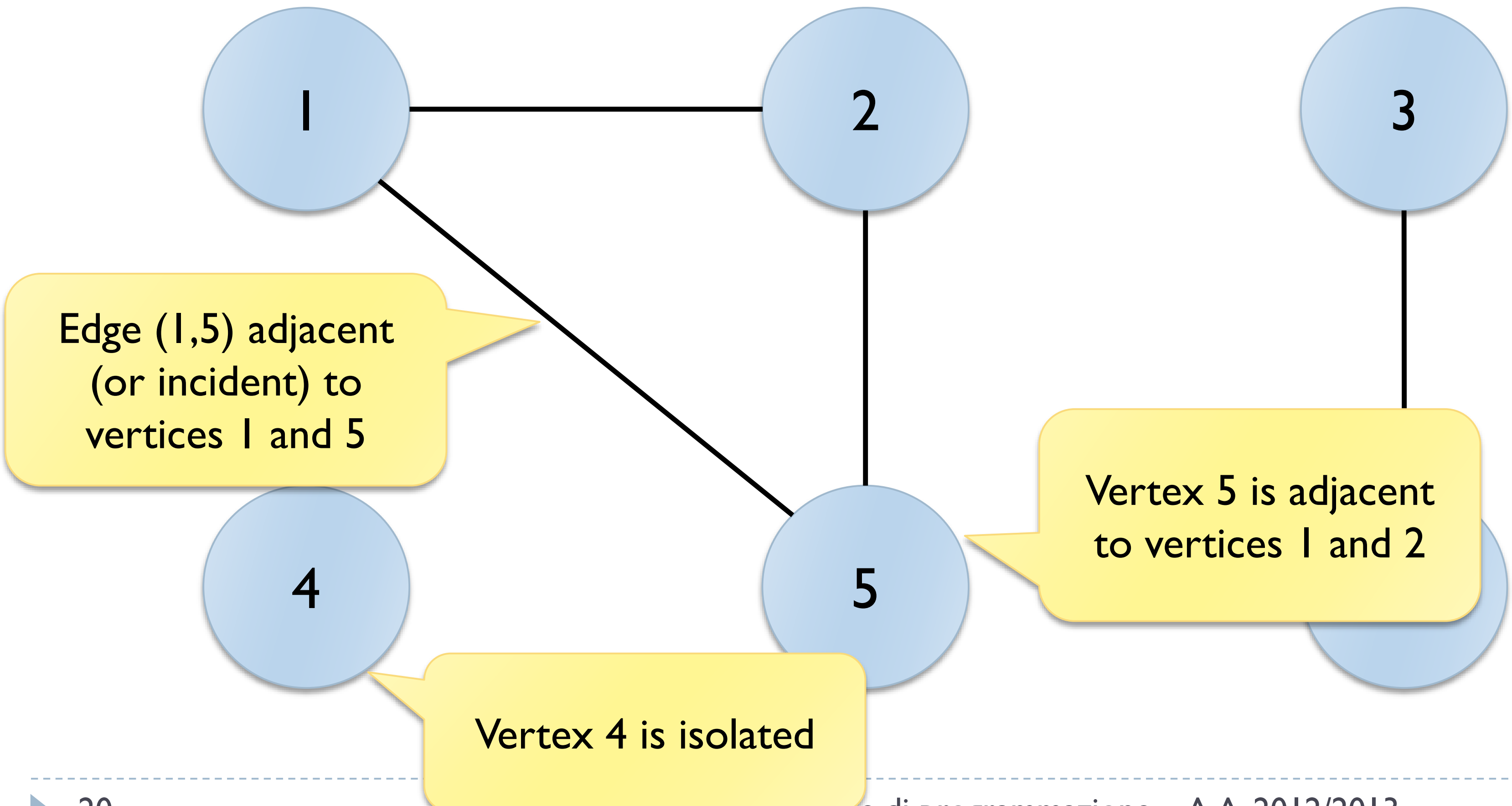
E={(1,2), (2,5), (5,1), (6,3)}



Example

$V=\{1,2,3,4,5,6\}$

$E=\{(1,2), (2,5), (5,1), (6,3)\}$

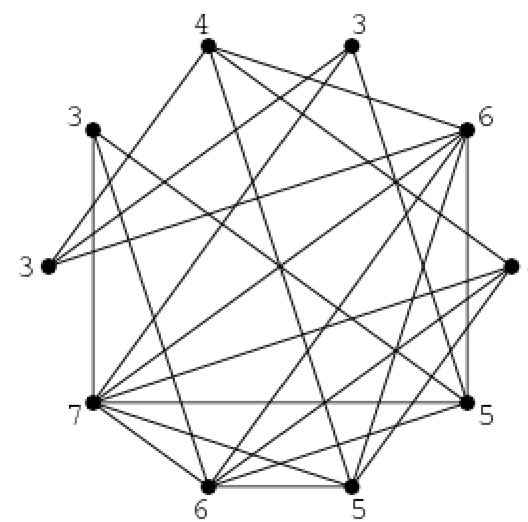


Related Definitions

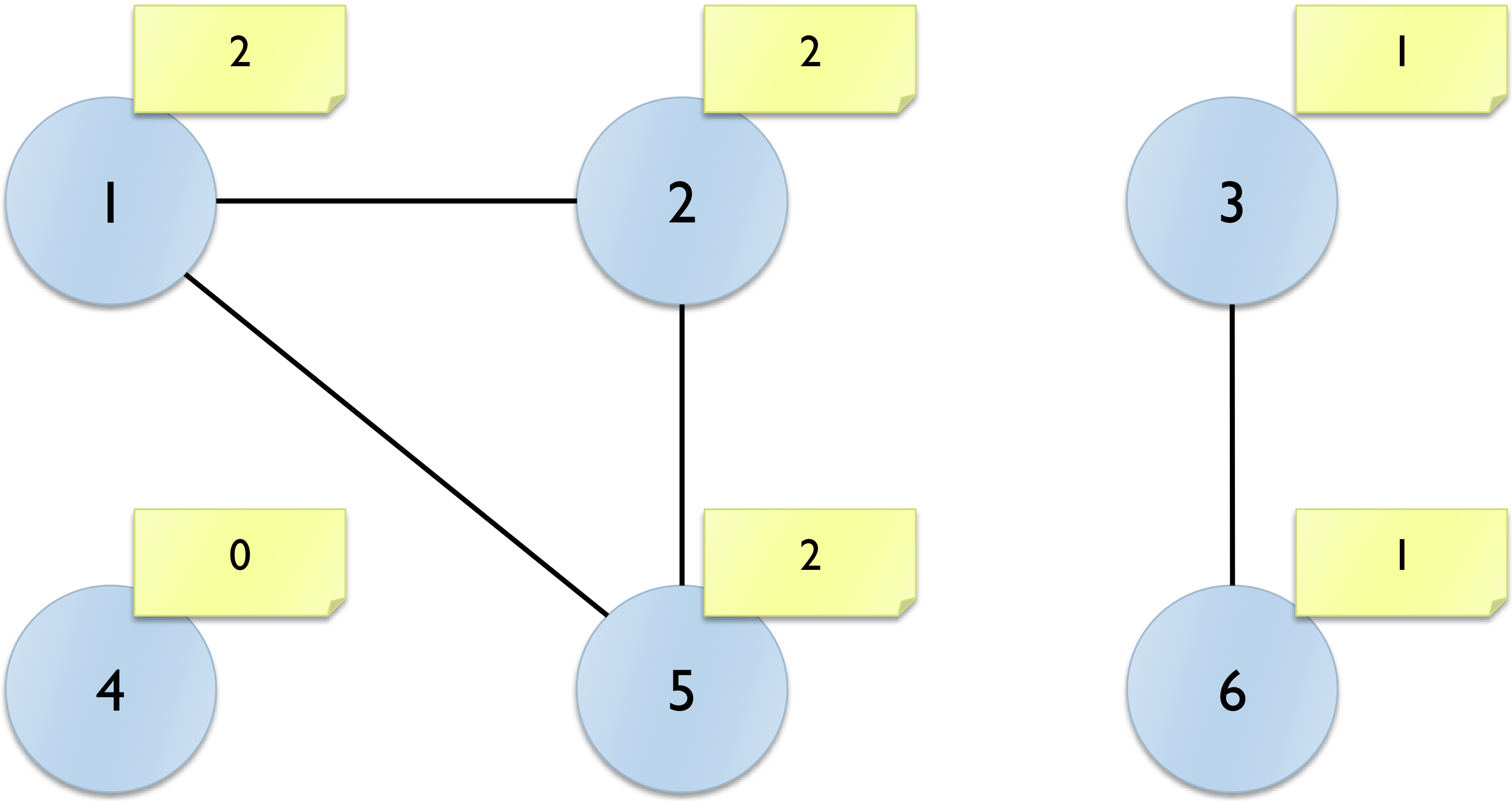
Introduction to Graphs

Degree

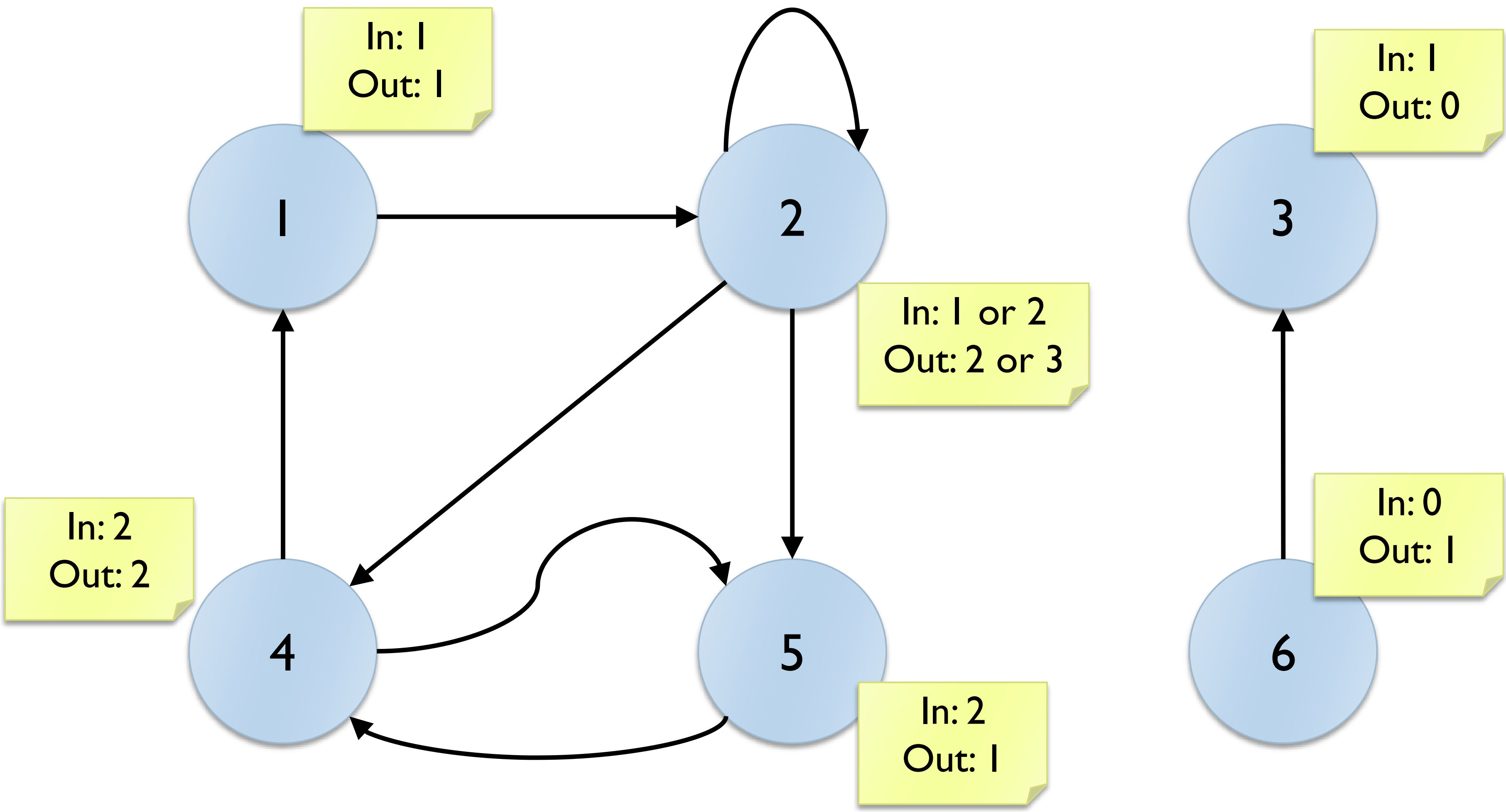
- In an *undirected* graph,
  - the **degree** of a vertex is the number of incident edges
- In a *directed* graph
  - The **in-degree** is the number of incoming edges
  - The **out-degree** is the number of departing edges
  - The **degree** is the sum of in-degree and out-degree
- A vertex with degree 0 is **isolated**



Degree



Degree

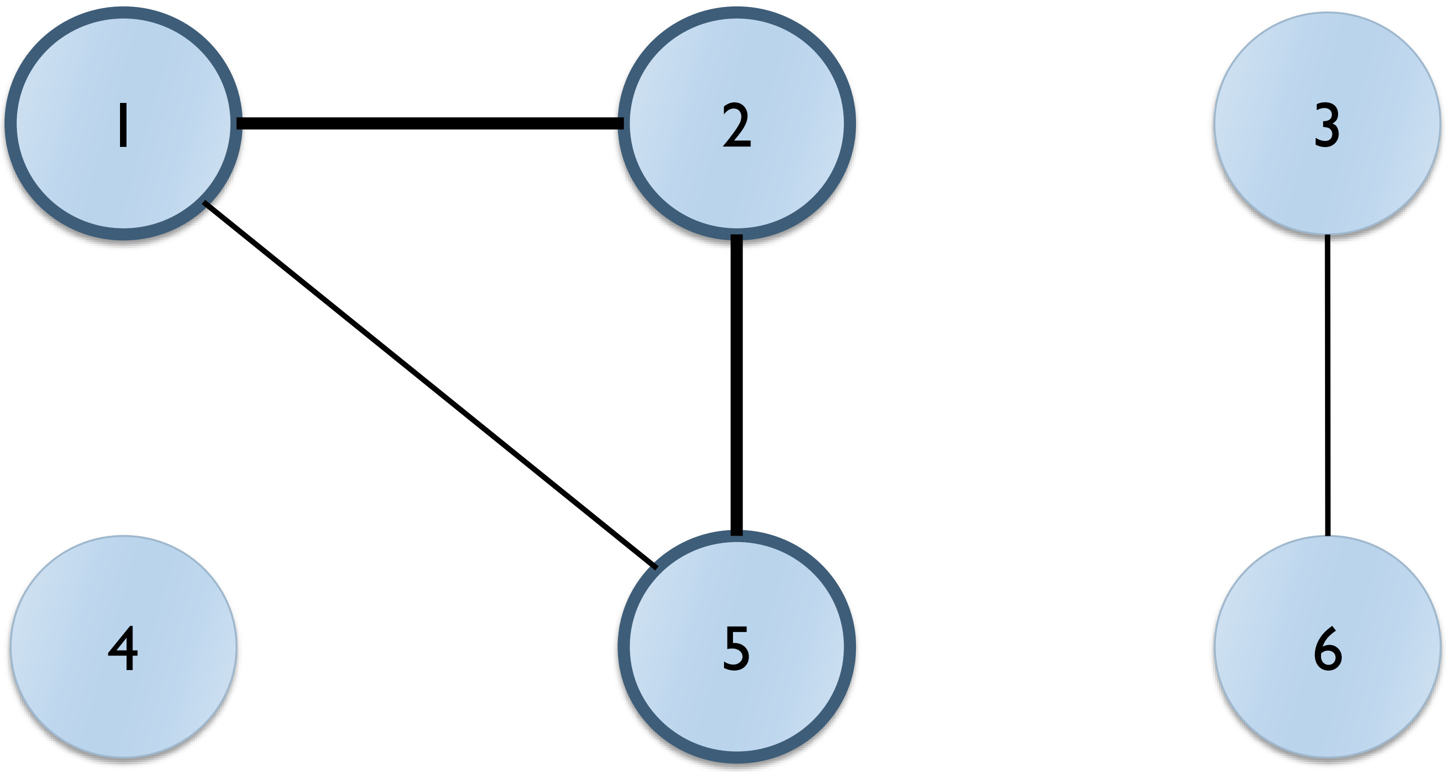

In: 1
Out: 1
1
2
In: 1 or 2
Out: 2 or 3
3
In: 1
Out: 0
In: 2
Out: 2
4
5
In: 2
Out: 1
6
In: 0
Out: 1

Paths

- A **path** on a graph G=(V,E) also called a trail, is a sequence $\{v_1, v_2, \ldots, v_n\}$ such that:
  - $v_1, \ldots, v_n$ are vertices: $v_i \in V$
  - $(v_1, v_2), (v_2, v_3), \ldots, (v_{n-1}, v_n)$ are graph edges: $(v_{i-1}, v_i) \in E$
  - v_i are distinct (for "simple" paths).
- The **length** of a path is the number of edges (n-1)
- If there exist a path between v_A and v_B we say that v_B is **reachable** from v_A

Example

Path = { 1, 2, 5 }
Length = 2

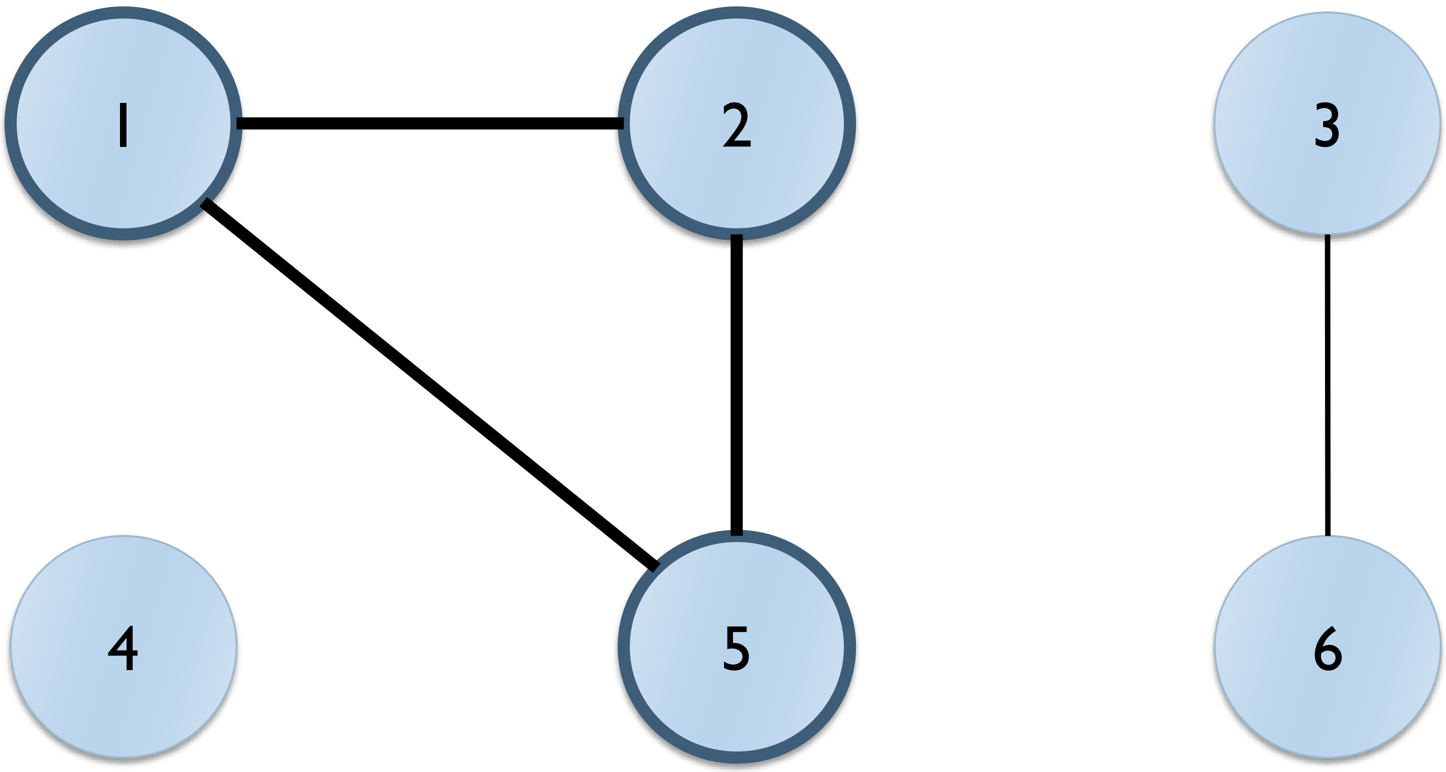


Cycles

- A cycle is a path where $v_1 = v_n$
- A graph with no cycles is said acyclic

Example

Path = { 1, 2, 5, 1 }
Length = 3

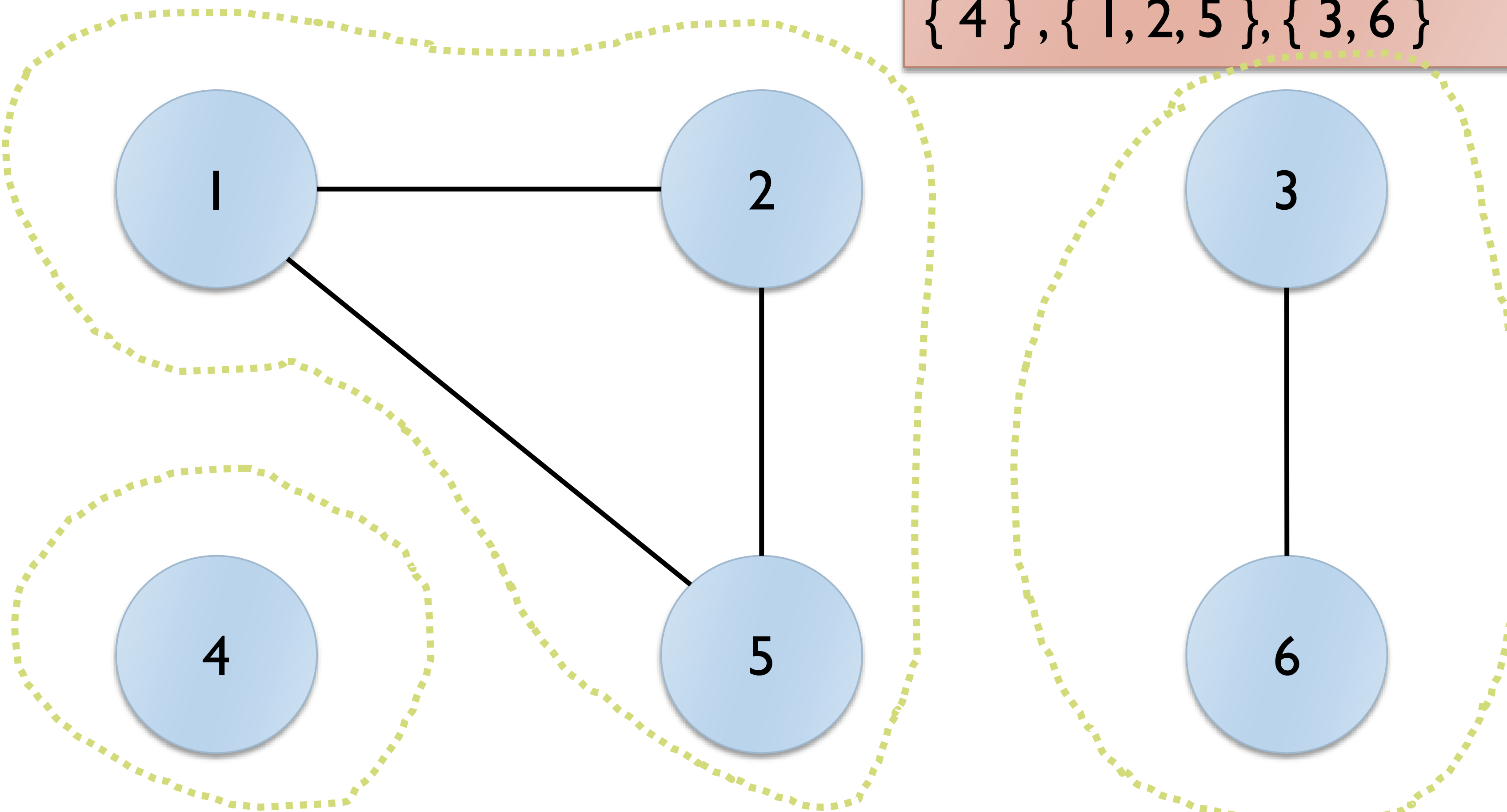


Reachability (Undirected)

- An undirected graph is **connected** if, for every couple of vertices, there is a path connecting them
- The connected sub-graph of maximum size are called **connected components**
- A connected graph has exactly one connected component

Connected components

The graph is **not** connected.
Connected components = 3
{ 4 } , { 1, 2, 5 }, { 3, 6 }

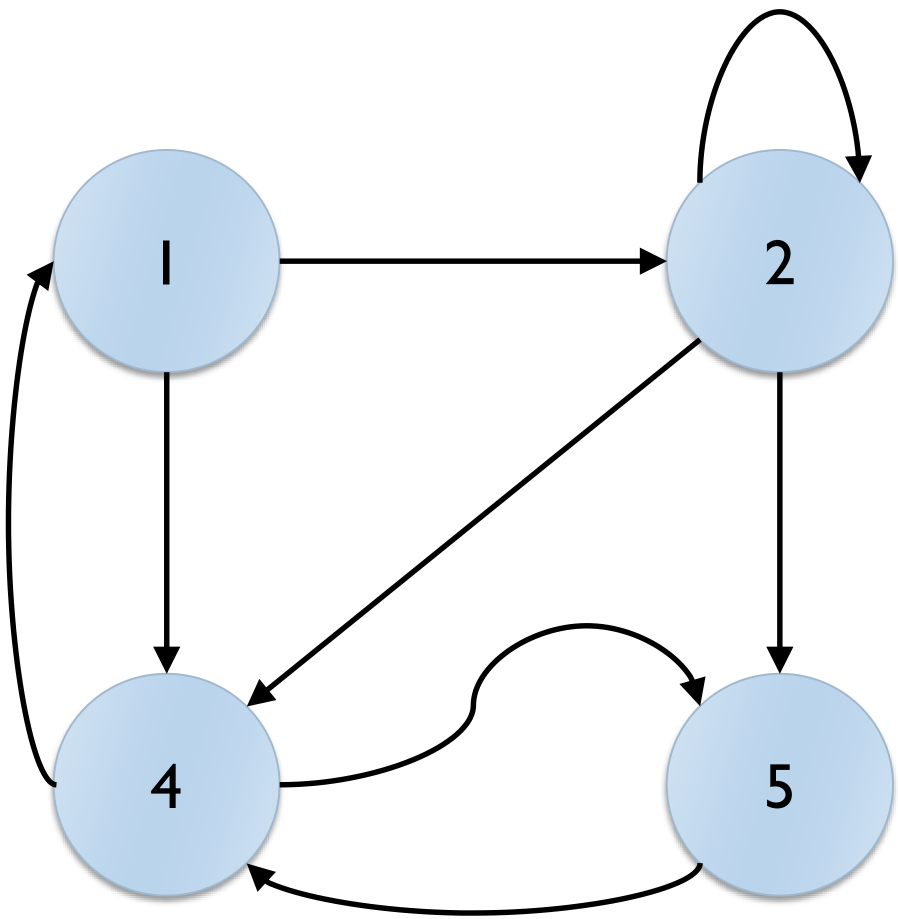


Reachability (Directed)

- A directed graph is **strongly connected** if, for <u>every</u> ordered pair of vertices (v, v'), there exists at least one path connecting v to v'

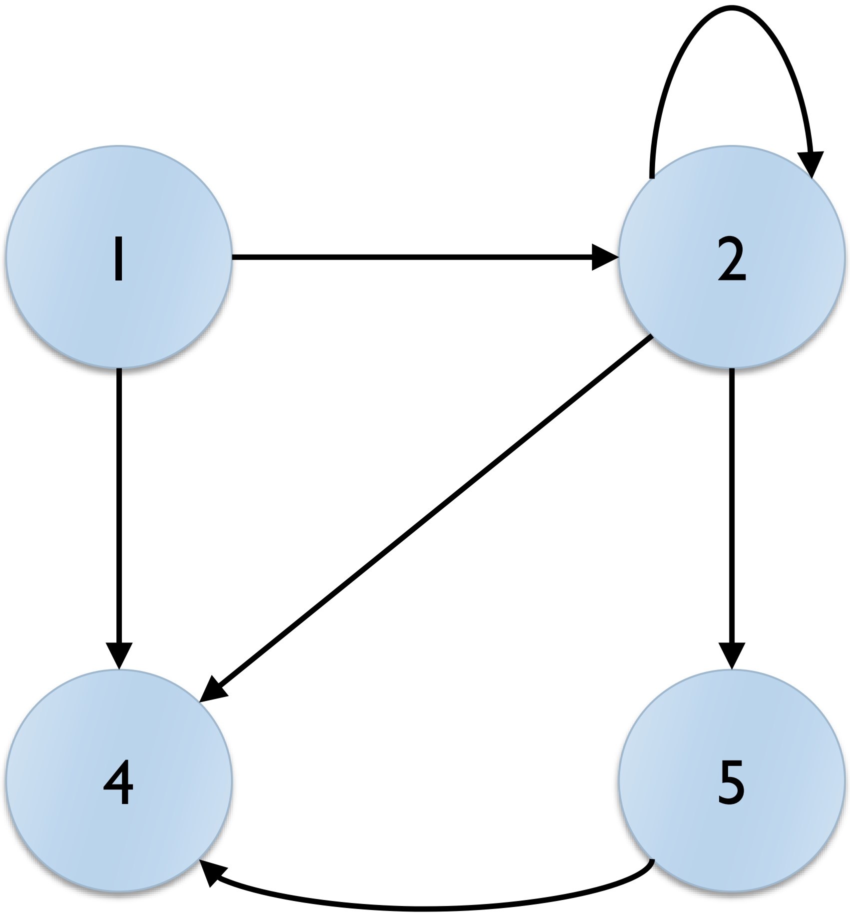
Example

The graph is **strongly connected**



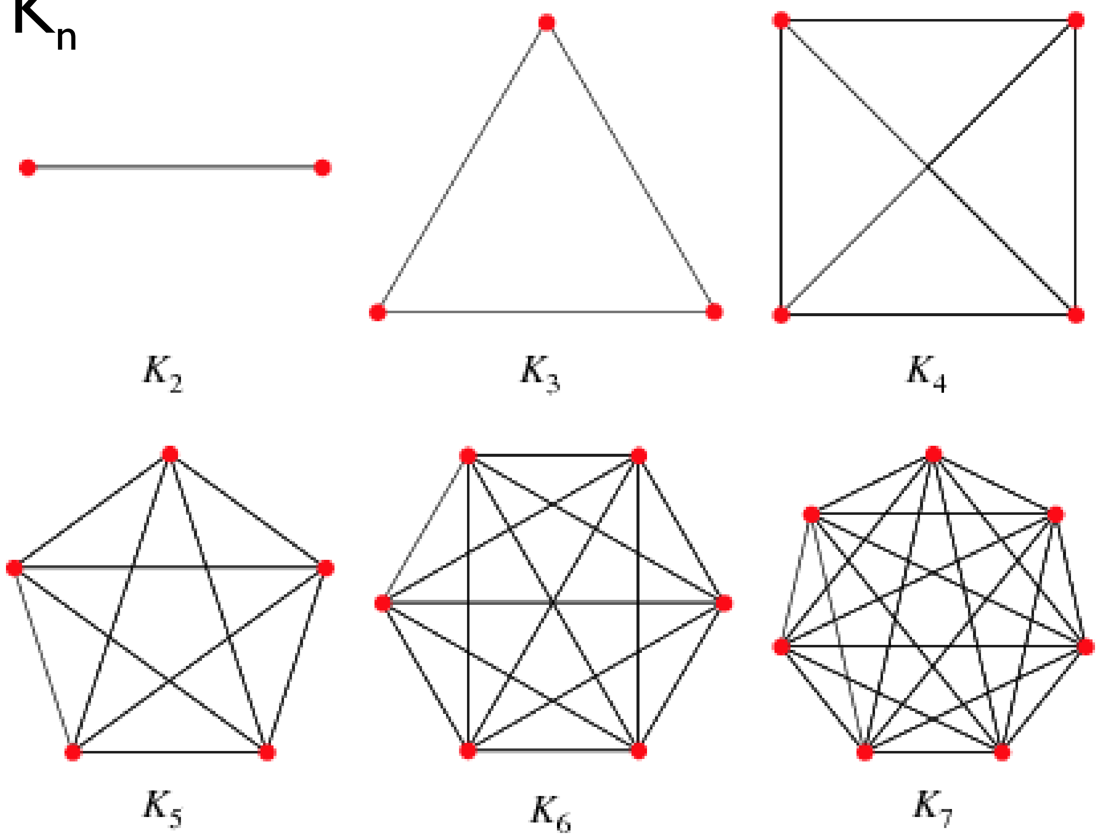
Example

The graph is **not** strongly connected



Complete graph

- A graph is complete if, for every pair of vertices, there is an edge connecting them (they are adjacent)
- Symbol: K_n



Complete graph: edges

- In a **complete** graph with n vertices, the number of **edges** is
  - n(n-1), if the graph is directed
  - n(n-1)/2, if the graph is undirected
  - If self-loops are allowed, then
    - n^2 for directed graphs
    - n(n-1) for undirected graphs

Density

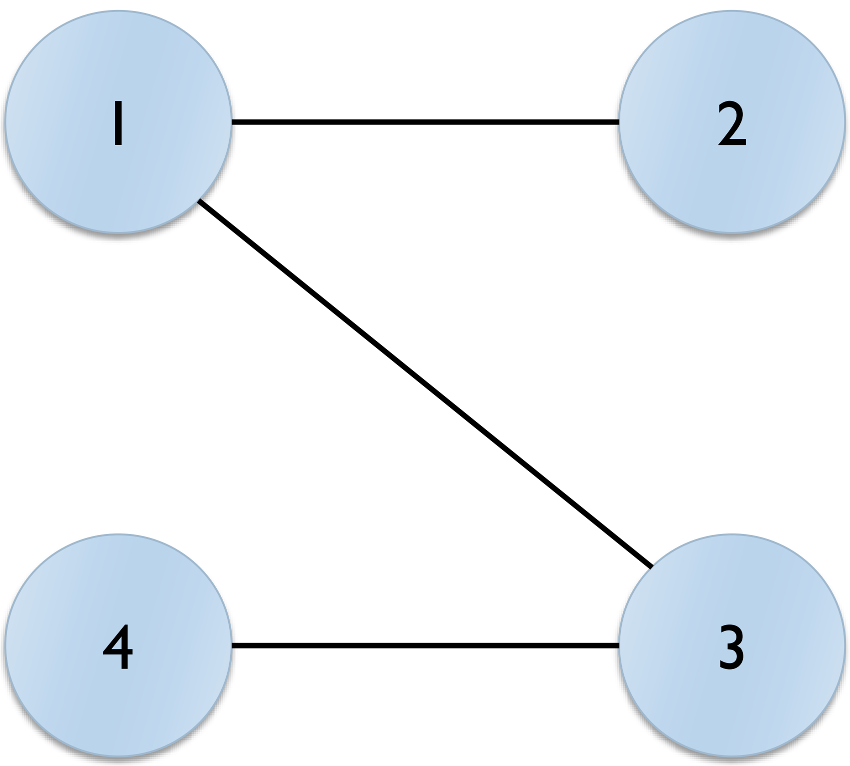
- The density of a graph G=(V,E) is the ratio of the number of edges to the total number of edges

$$d = \frac{|E(G)|}{\left|E\left(K_{|V(G)|}\right)\right|}$$

Esempio

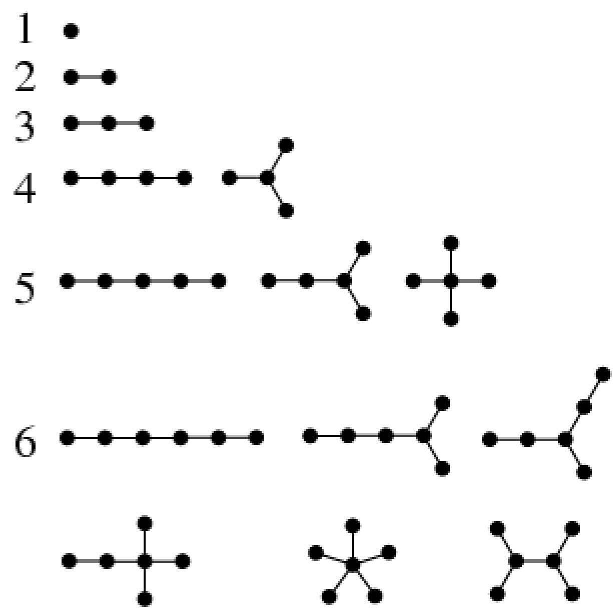
Density = 0.5

Existing: 3 edges
Total: 6 possible edges



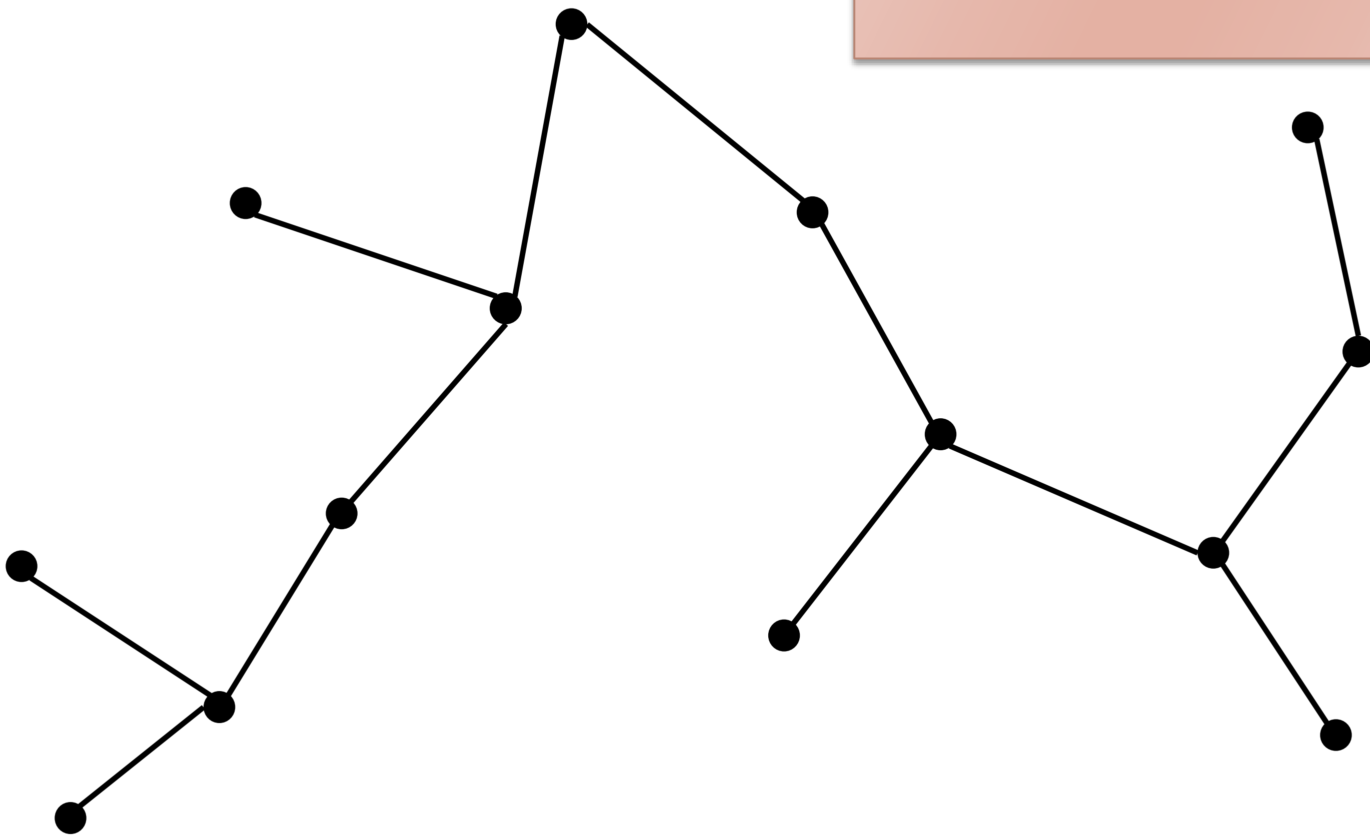
Trees and Forests

- An undirected acyclic graph is called **forest**
- An undirected acyclic connected graph is called **tree**



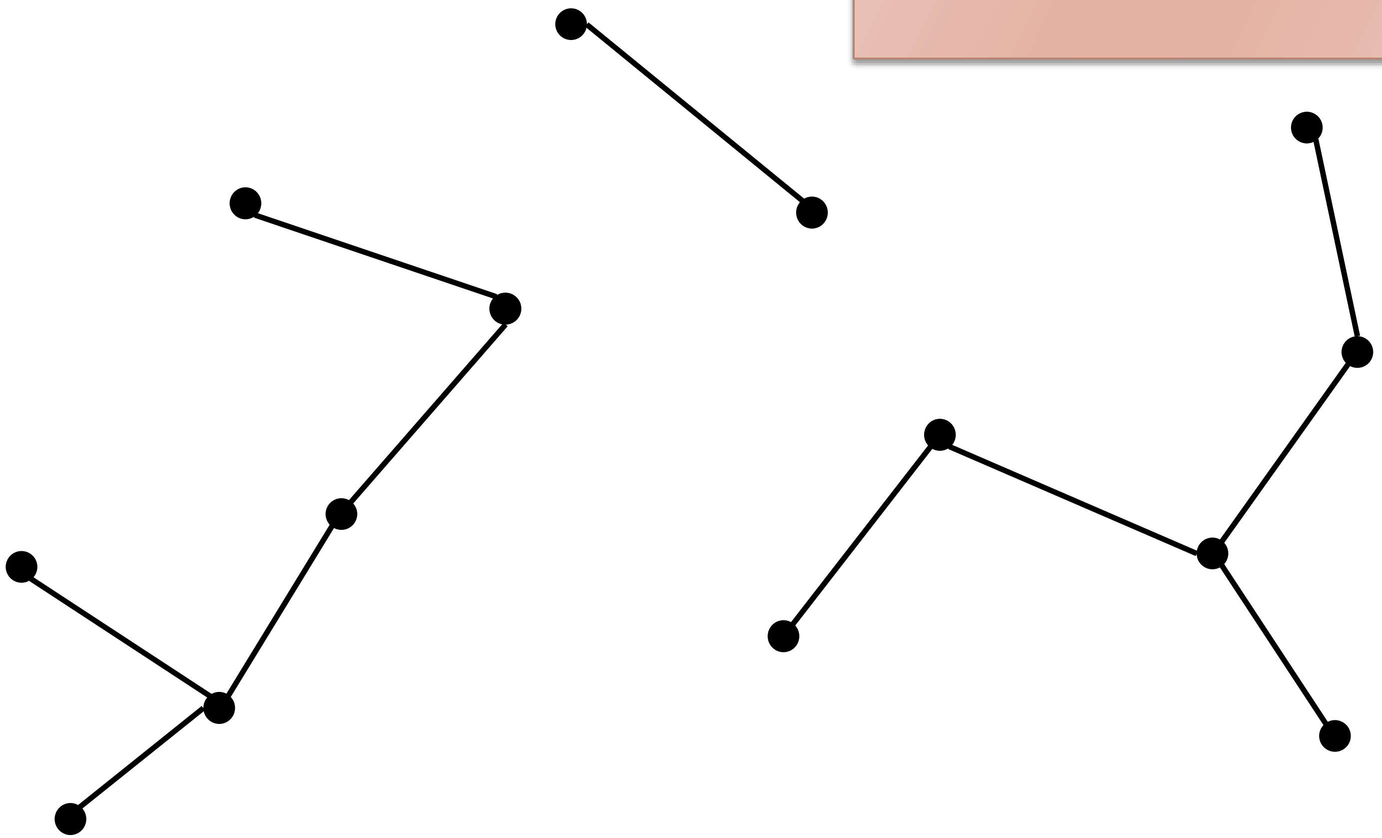
Example

Tree



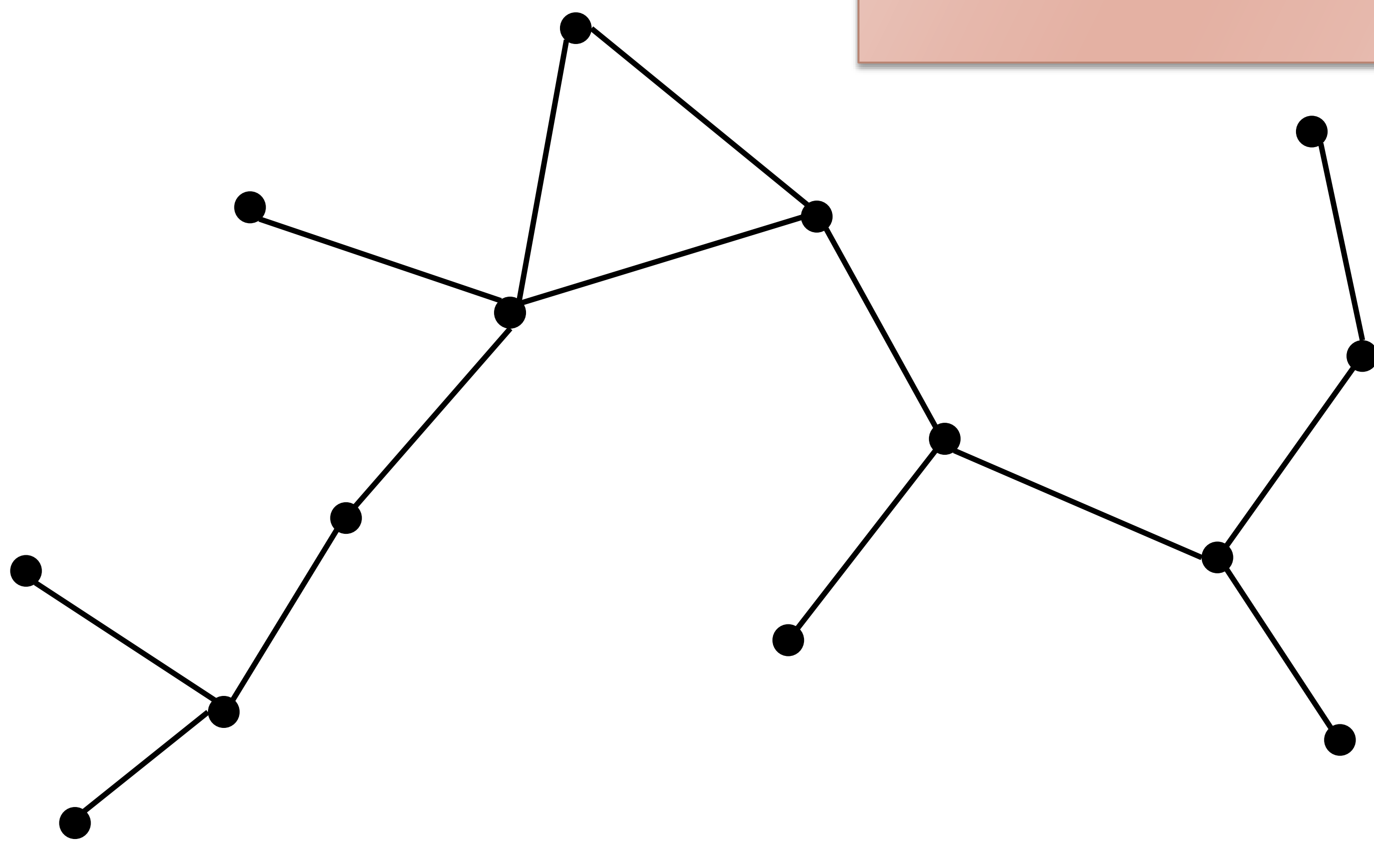
Example

Forest



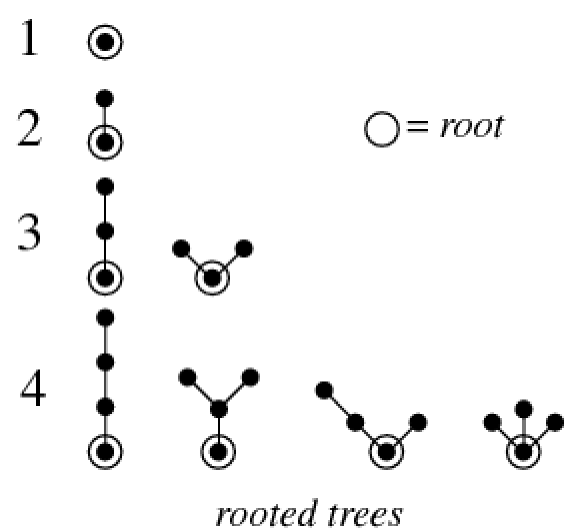
Example

This is not a tree nor a forest
(it contains a cycle)



Rooted trees

- In a tree, a special node may be singled out
- This node is called the "**root**" of the tree
- Any node of a tree can be the root

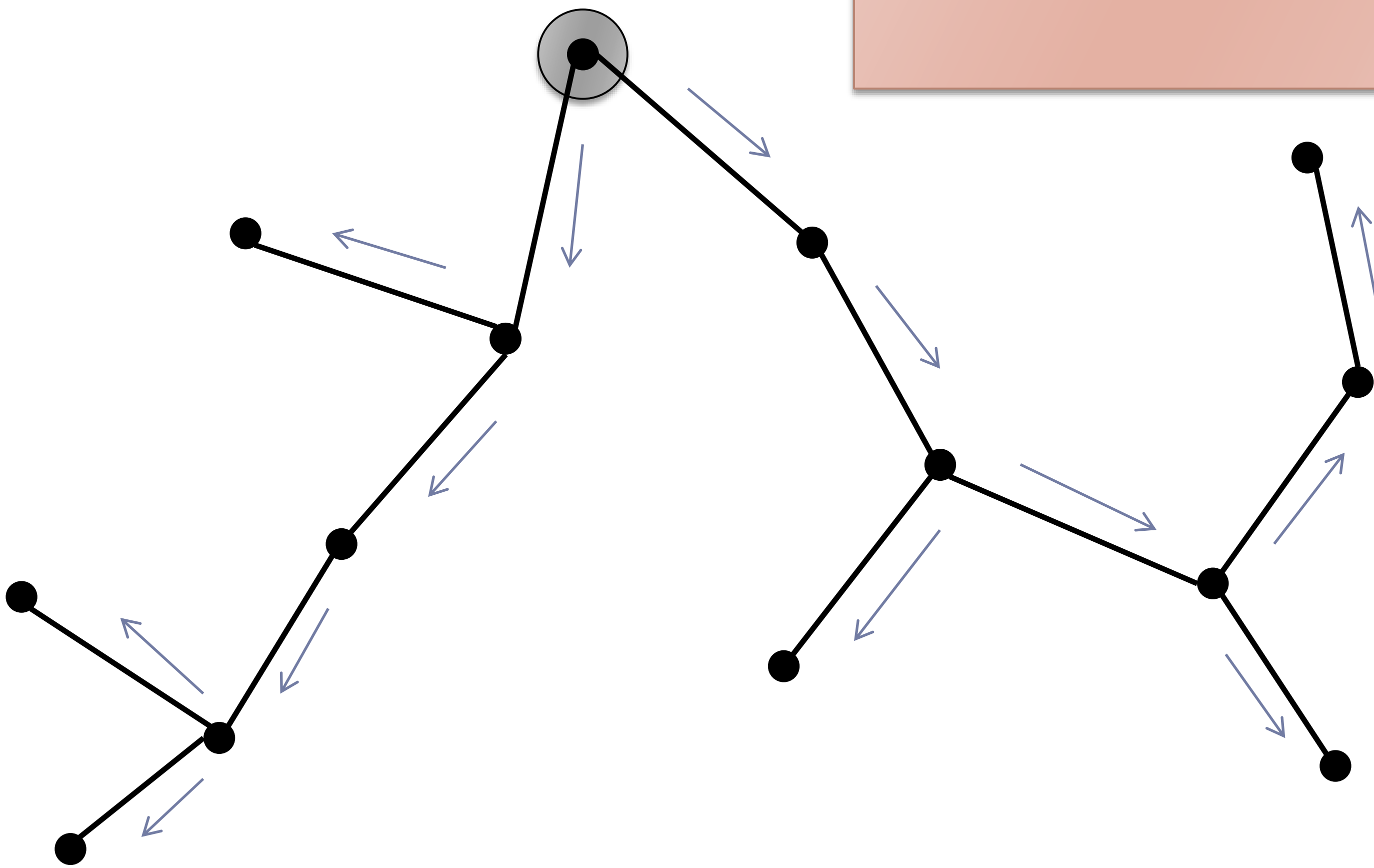


Tree (implicit) ordering

- The root node of a tree **induces an ordering** of the nodes
- The root is the "ancestor" of all other nodes/vertices
  - "children" are "away from the root"
  - "parents" are "towards the root"
- The root is the only node without parents
- All other nodes have exactly one parent
- The furthermost (children-of-children-of-children…) nodes are "leaves"

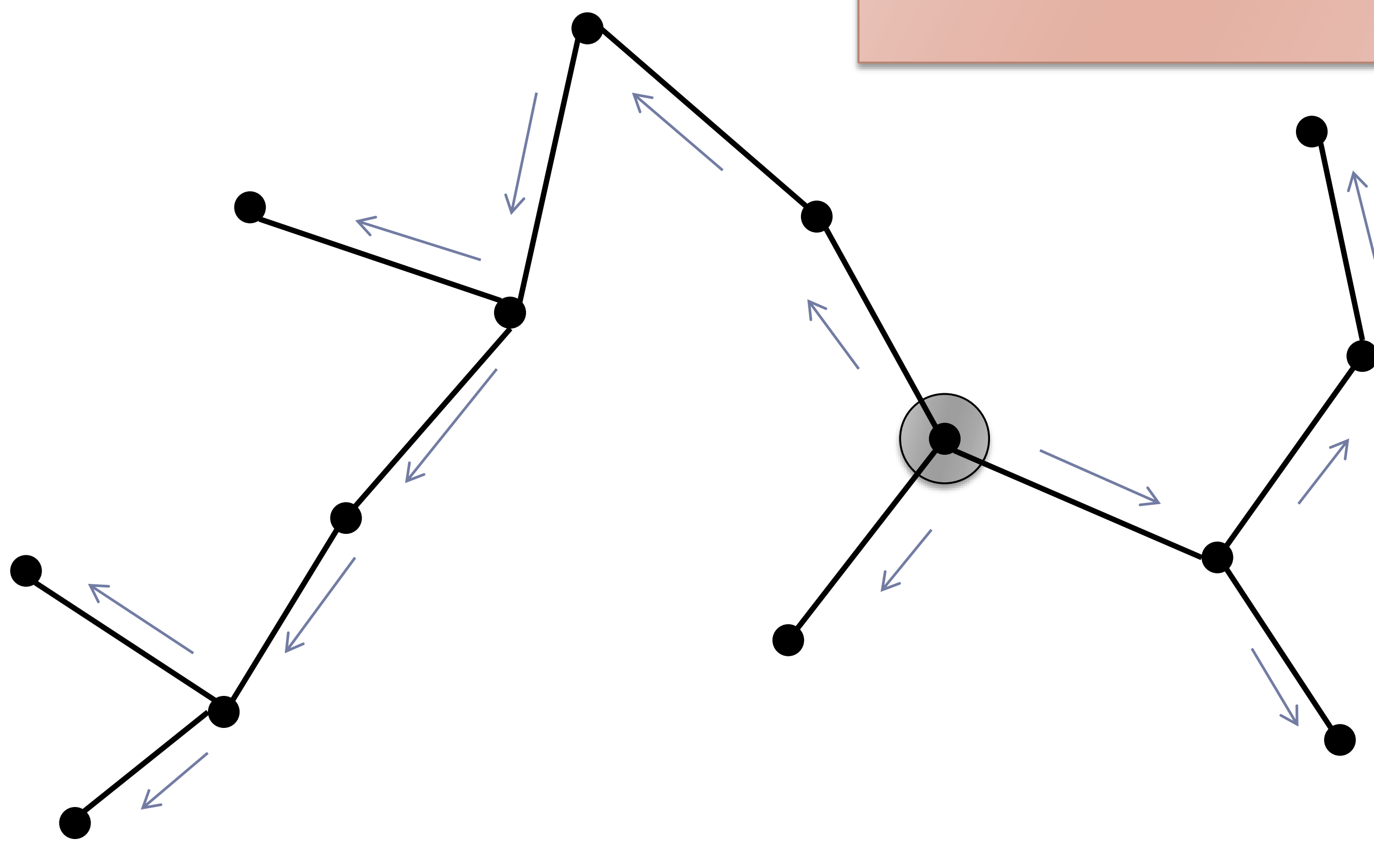
Example

Rooted Tree



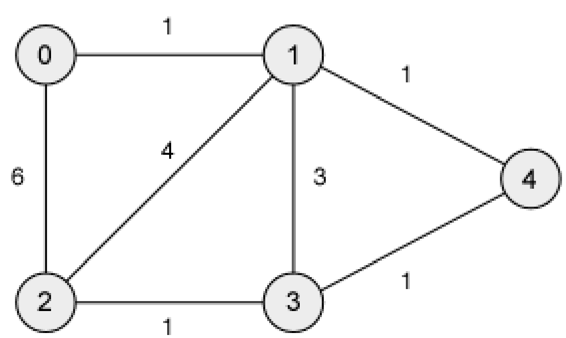
Example

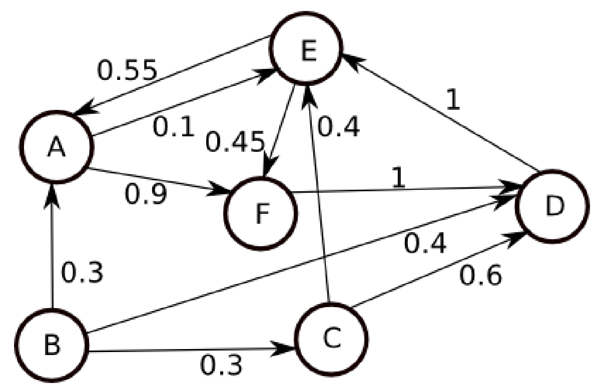
Rooted Tree



Weighted graphs

- A weighted graph is a graph in which each branch (edge) is given a numerical weight.
- A weighted graph is therefore a special type of labeled graph in which the labels are numbers (which are usually taken to be positive).





Applications

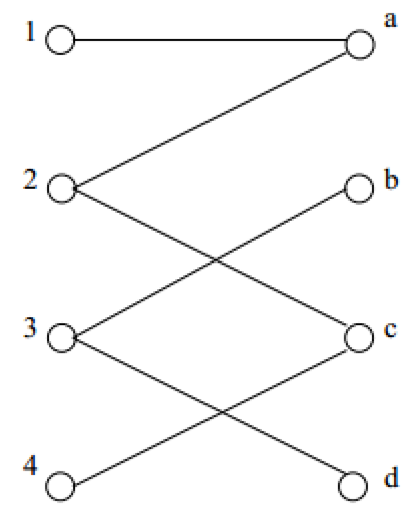
Introduction to Graphs

Graph applications

- Graphs are everywhere
  - Facebook friends (and posts, and 'likes')
  - Football tournaments (complete subgraphs + binary tree)
  - Google search index (V=page, E=link, w=pagerank)
  - Web analytics (site structure, visitor paths)
  - Car navigation (GPS)
  - Market Matching

Market matching

- H = Houses (1, 2, 3, 4)
- B = Buyers (a, b, c, d)
- V = H ∪ B
- Edges: (*h*, *b*) ∈ E if *b* would like to buy *h*
- Problem: can all houses be sold and all buyers be satisfied?
- Variant: if the graph is weighted with a purchase offer, what is the most convenient solution?
- Variant: consider a 'penalty' for unsold items

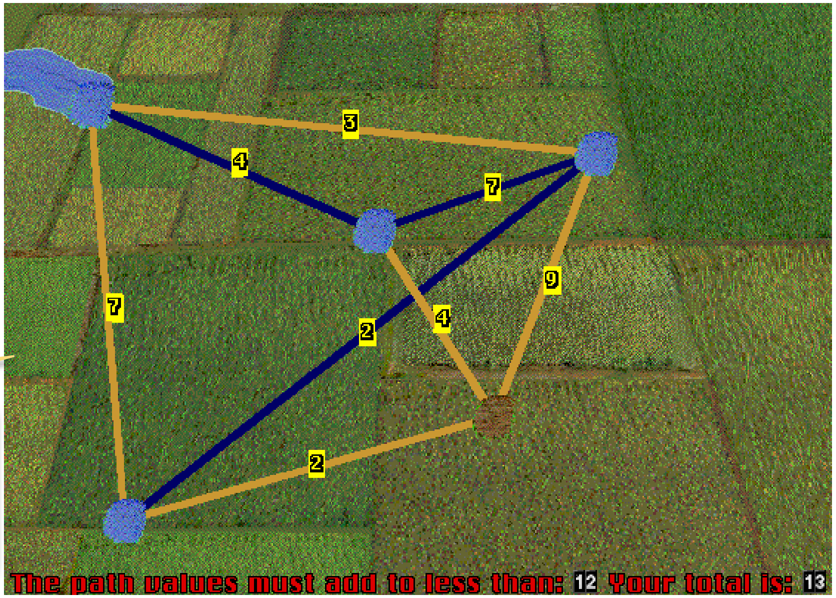


This graph is called "bipartite":
H ∩ B = ∅

Connecting cities

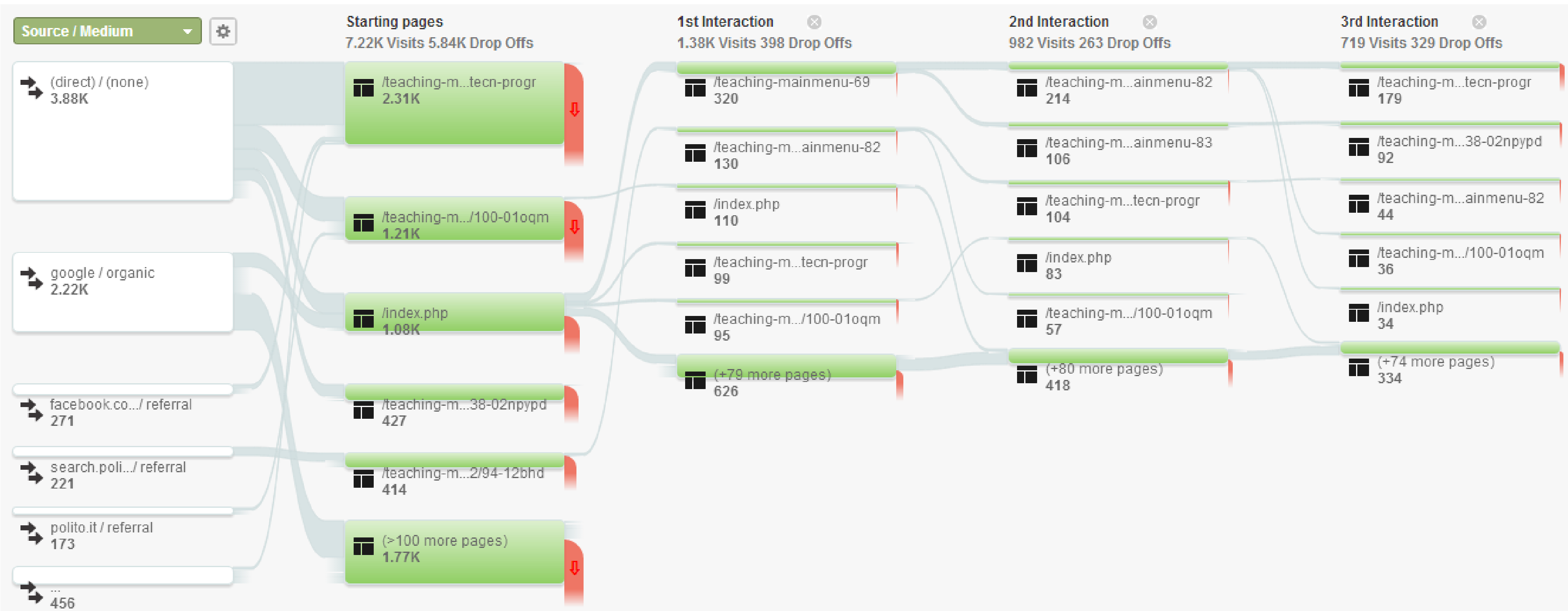
- We have a water reservoir
- We need to serve many cities
  - Directly or indirectly
- What is the most efficient set of inter-city water connections?

- Also for telephony, gas, electricity, …

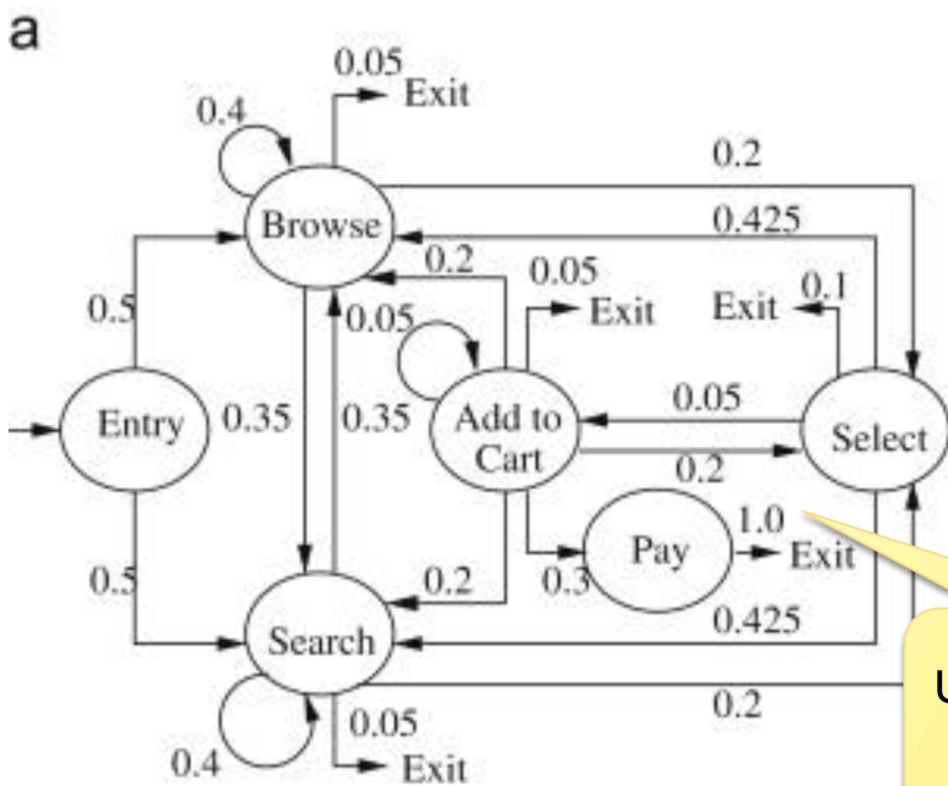


We are searching for the "minimum spanning tree"

Google Analytics (Visitors Flow)



Customer behavior



User actions encoded as frequencies

Street navigation

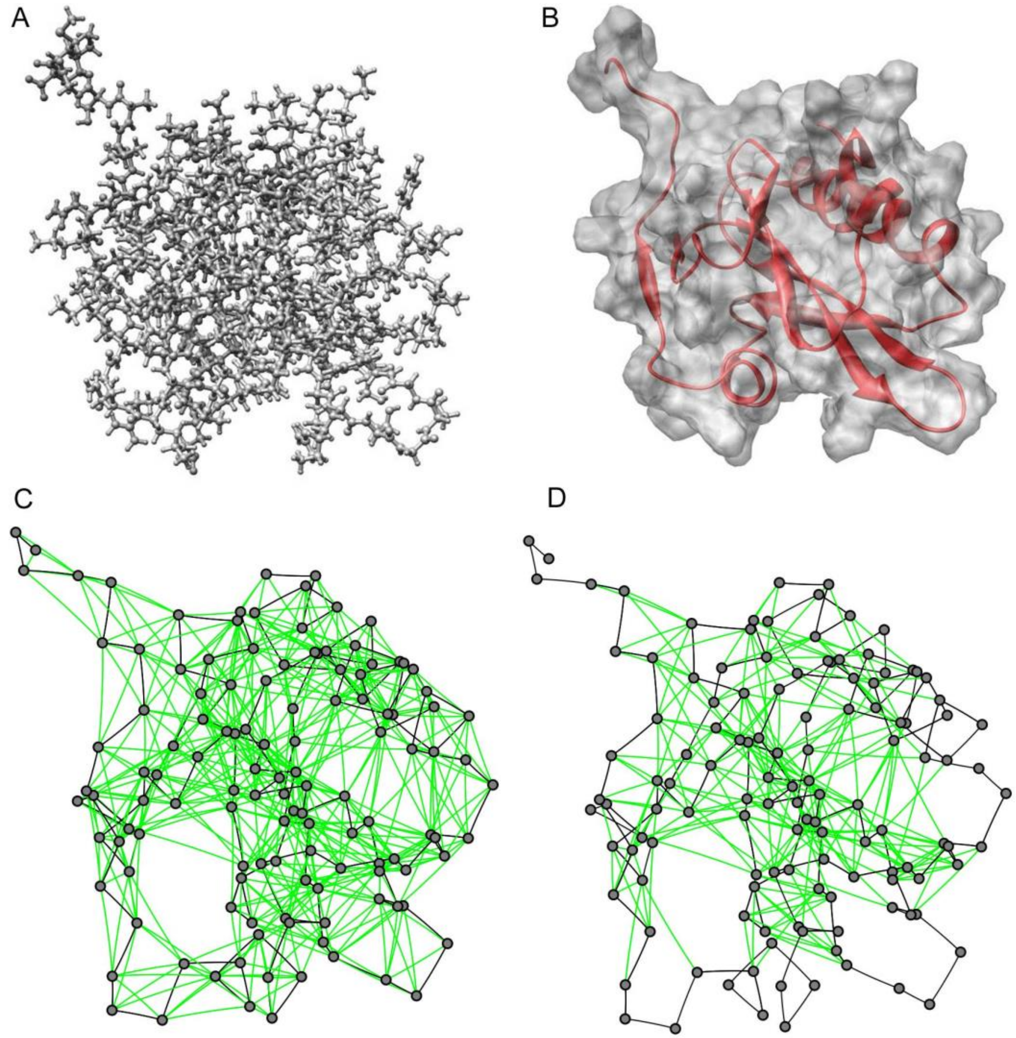
TSP: The traveling salesman problem

We must find a “Hamiltonian cycle”

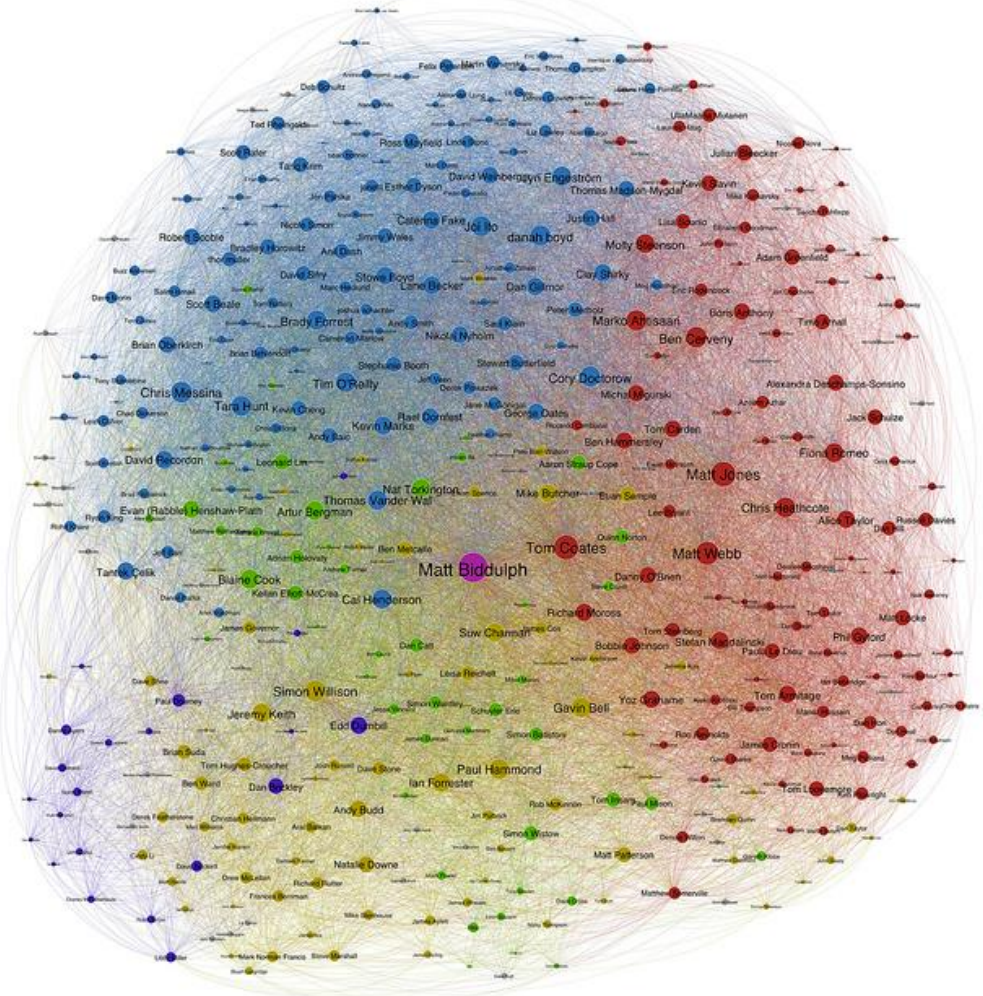
Train maps

Kensal Rise
Brondesbury Park
Harlesden
West Hampstead
Finchley Road
Swiss Cottage
St. John's Wood
Chalk Farm
Kentish Town
Caledonian Road
Holloway Road
Drayton Park
Camden Road
Camden Town
Brondesbury
Kensal Green
Queen's Park
Kilburn Park
Maida Vale
Warwick Avenue
Royal Oak
Westbourne Park
Ladbroke Grove
Latimer Road
White City
Holland Park
East Acton
Shepherd's Bush
Goldhawk Road
Barons Court
Ravenscourt Park
West Kensington
Kensington (Olympia)
Edgware Road
Marylebone
Baker Street
Great Portland Street
Euston
Mornington Crescent
King's Cross St. Pancras
Caledonian Road & Barnsbury
Highbury & Islington
Canonbury
Hackney Central
Dalston Kingsland
Angel
Essex Road
Farringdon
Barbican
Old Street
Liverpool Street
Bethnal Green
Paddington
Bayswater
Queensway
Marble Arch
Notting Hill Gate
Lancaster Gate
Bond Street
Oxford Circus
Regent's Park
Warren Street
Euston Square
Goodge Street
Russell Square
Holborn
Tottenham Court Road
Chancery Lane
Moorgate
St. Paul's
Bank
Shoreditch
Aldgate East
Aldgate
Whitechapel
Shadwell
Wapping
Covent Garden
Leicester Square
City Thameslink
Cannon Street
Monument
Tower Hill
Tower Gateway
High Street Kensington
Hyde Park Corner
Knightsbridge
Green Park
Piccadilly Circus
Gloucester Road
Sloane Square
Westminster
Charing Cross
Blackfriars
Temple
Embankment
Mansion House
Fenchurch Street
River Thames
Rotherhithe
Earl's Court
South Kensington
Victoria
St. James's Park
West Brompton
Fulham Broadway
Parsons Green
Pimlico
Waterloo
Southwark
Lambeth North
Borough
London Bridge
Bermondsey
Canada Water
Surrey Quays

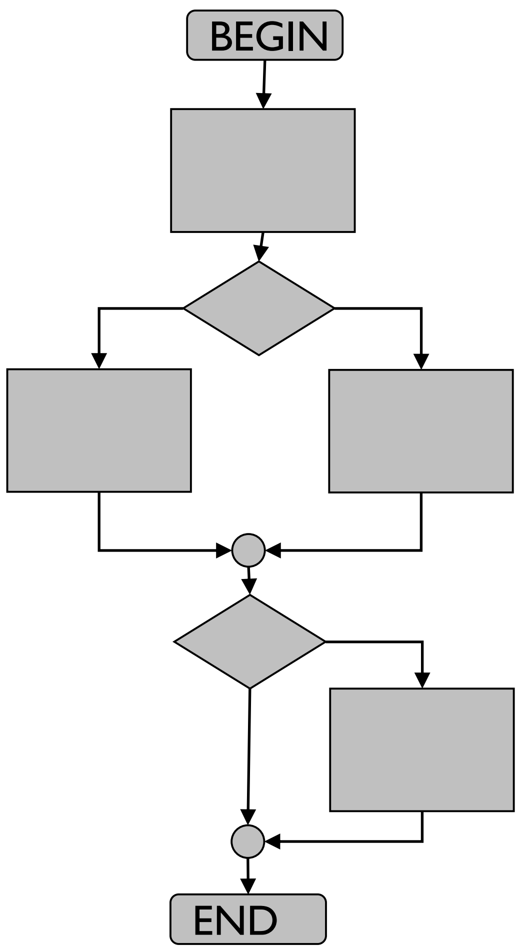
Chemistry (Protein folding)

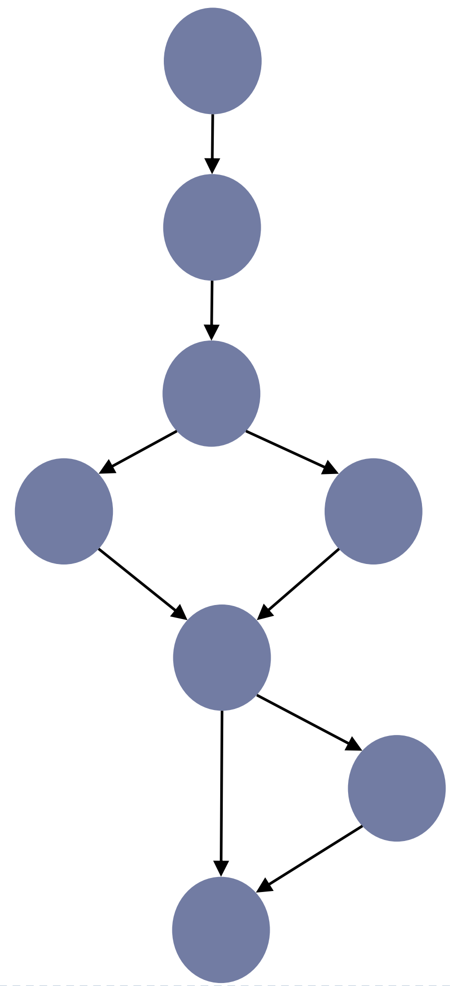


Facebook friends



Flow chart





Graph representation

Representing and visiting graphs

Representing graphs

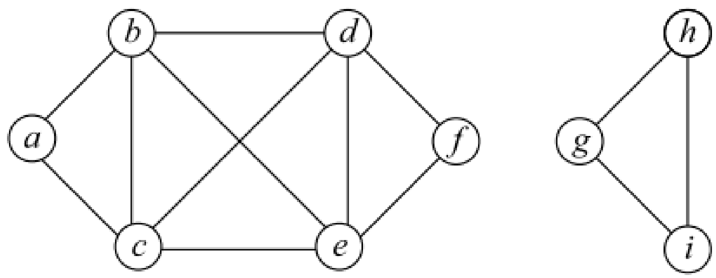
List structures

- **Adjacency list**
  - Each vertex has a list of which vertices it is adjacent to.
  - For undirected graphs, information is duplicated
- Incidence list
  - Each vertex has a list of 'edge' objects
  - Edges are represented by a pair (a tuple if directed) of vertices (that the edge connects) and possibly weight and other data.

Matrix structures

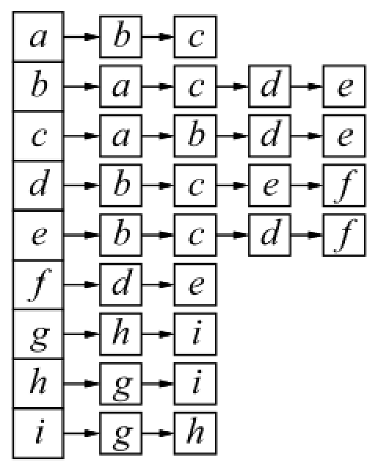
- **Adjacency matrix**
  - A = |V| x |V| matrix of Booleans or integers
  - If there is an edge from a vertex v to a vertex v', then the element A[v,v'] is 1, otherwise 0.
- Incidence matrix
  - IM = |V| x |E| matrix of integers
  - IM[v,e] = 1 (incident), 0 (not incident)
  - For directed graphs, may be -1 (out) and +1 (in)

Example



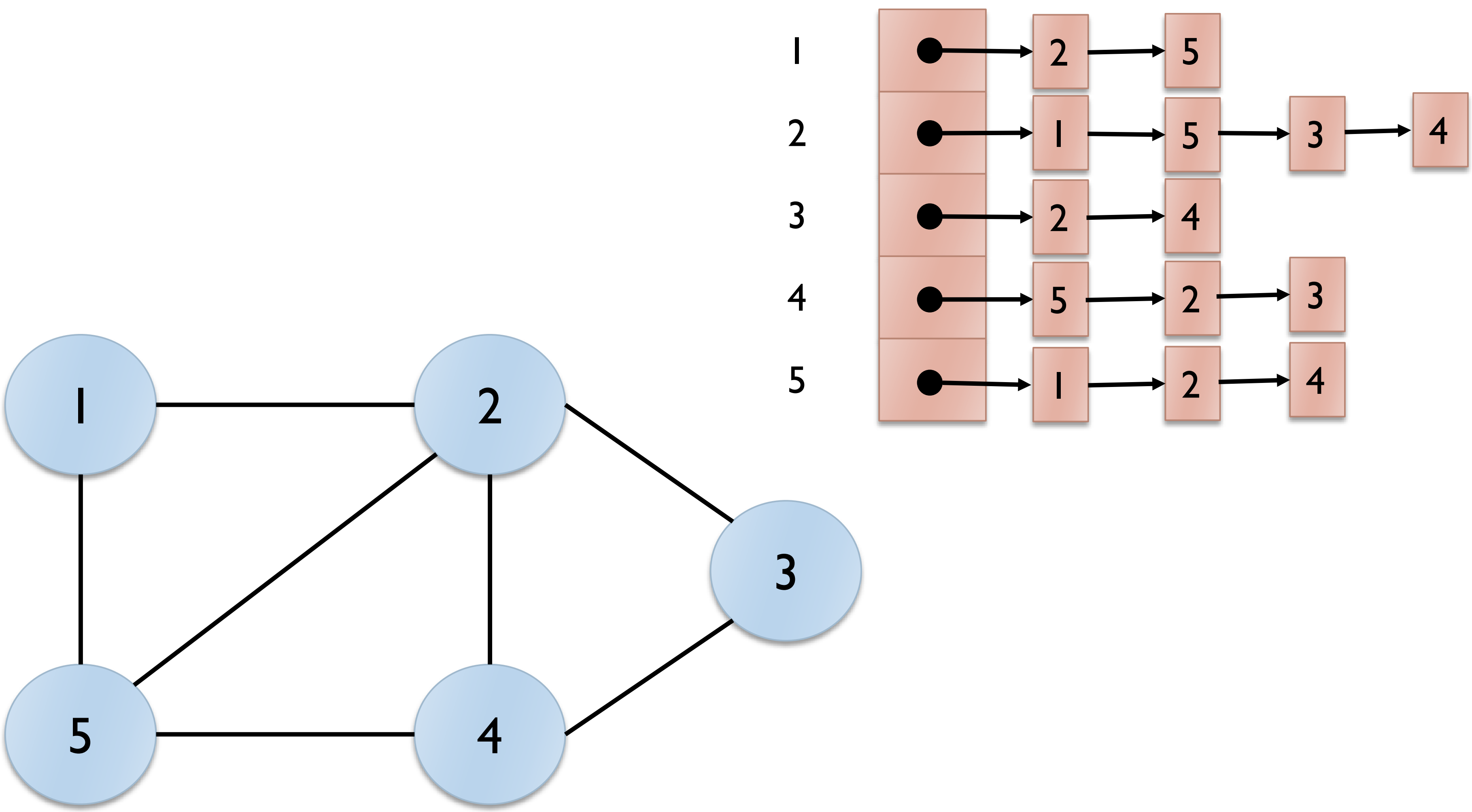
Adjacency matrix

	a	b	c	d	e	f	g	h	i
a	0	1	1	0	0	0	0	0	0
b	1	0	1	1	1	0	0	0	0
c	1	1	0	1	1	0	0	0	0
d	0	1	1	0	1	1	0	0	0
e	0	1	1	1	0	1	0	0	0
f	0	0	0	1	1	0	0	0	0
g	0	0	0	0	0	0	0	1	0
h	0	0	0	0	0	0	1	0	1
i	0	0	0	0	0	0	1	1	0

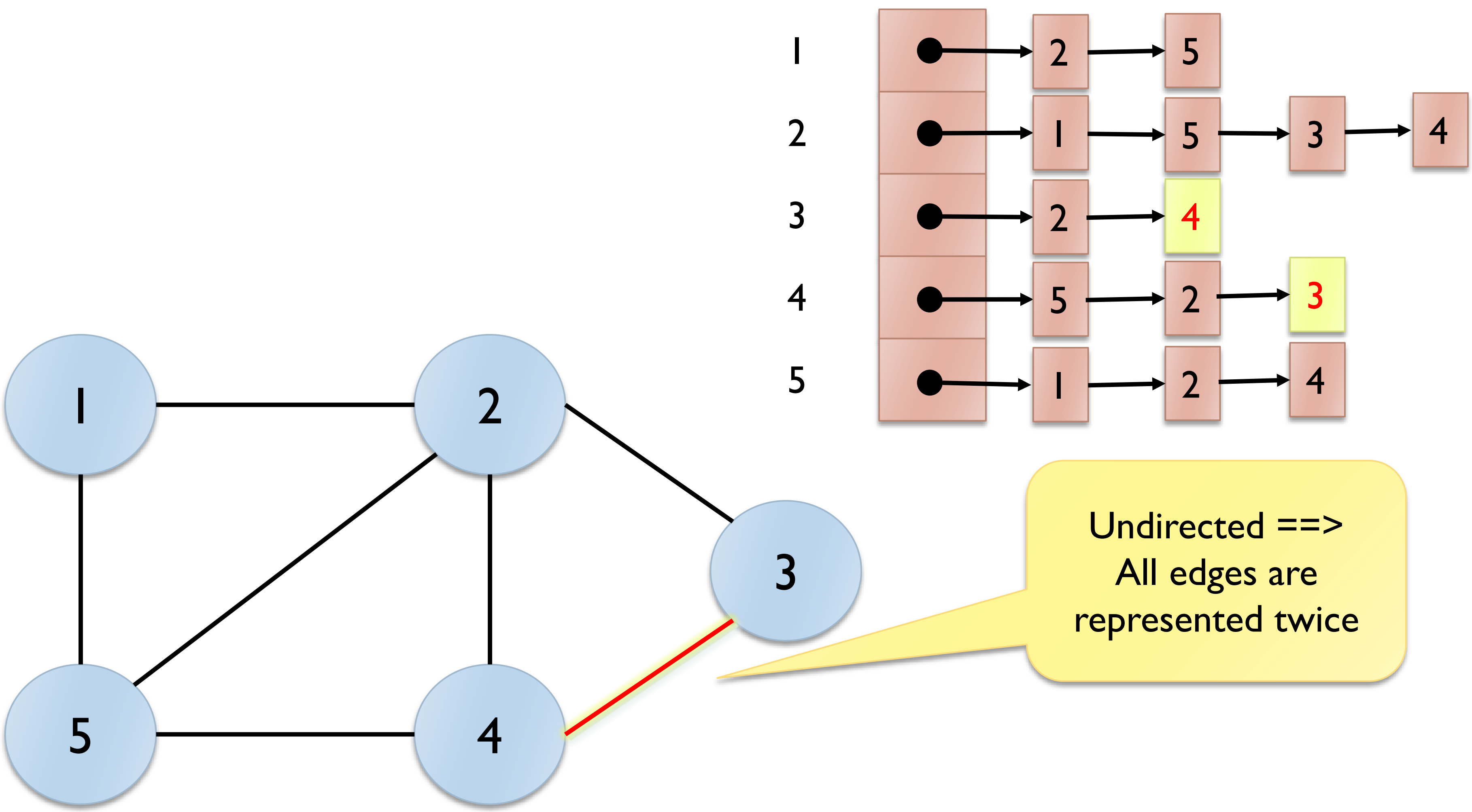


Adjacency list

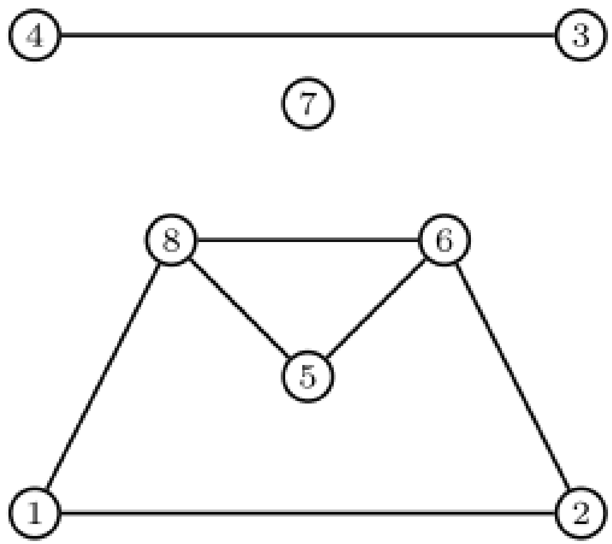
Adjacency list (undirected graph)



Adjacency list (undirected graph)



Adjacency list (un-connected graph)



$L_1 = [2, 8]$

$L_2 = [1, 6]$

$L_3 = [4]$

$L_4 = [3]$

$L_5 = [6, 8]$

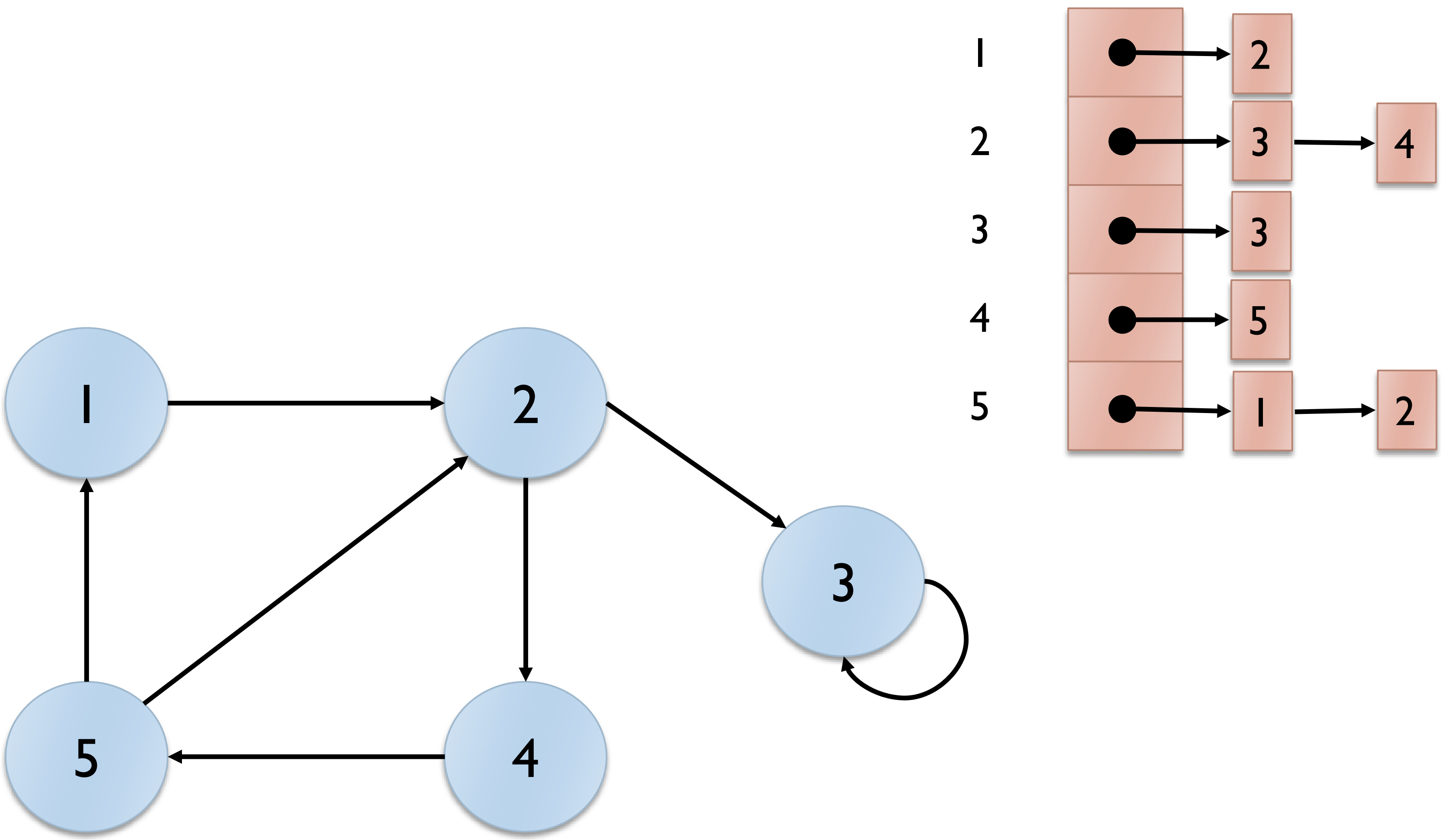
$L_6 = [2, 5, 8]$

$L_7 = []$

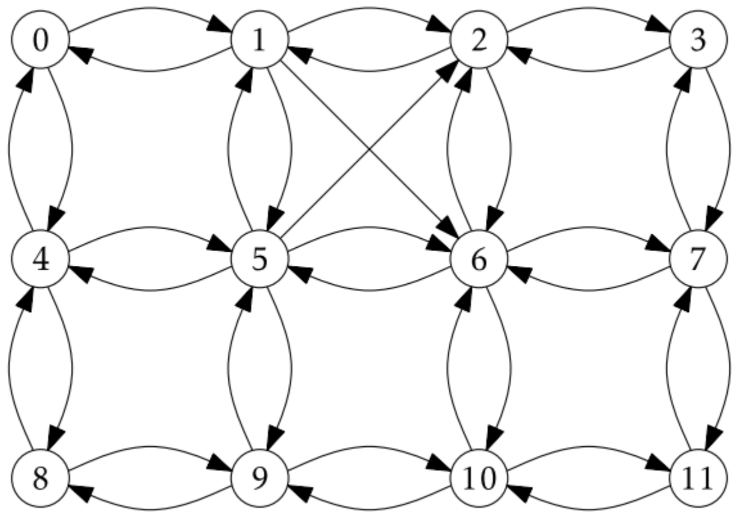
$L_8 = [1, 5, 6]$

Un-connected graph, same rules

Adjacency list (directed graph)



Adjacency list

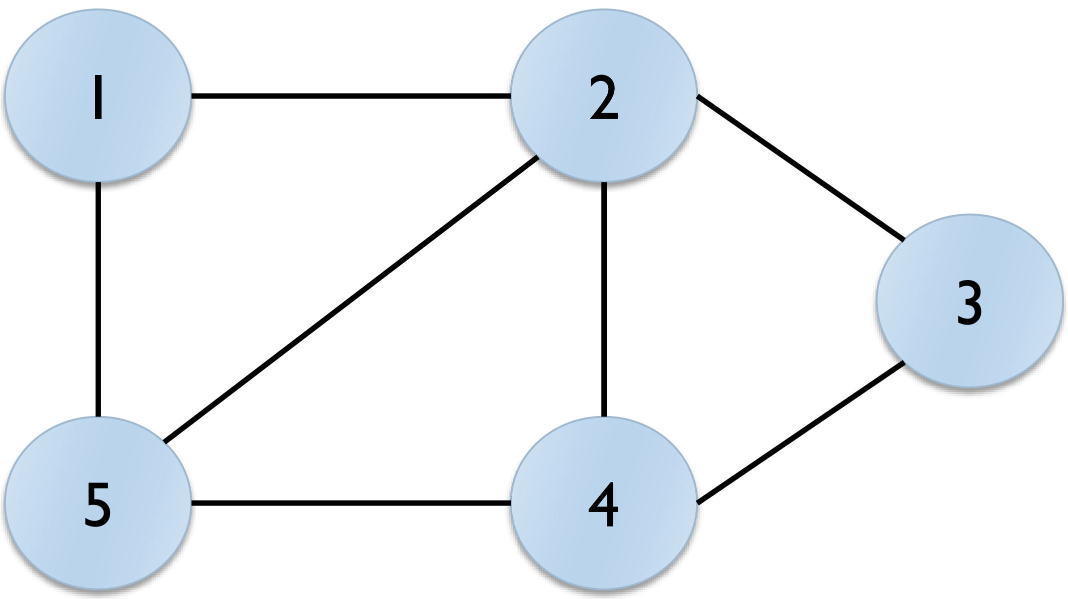


0	1	2	3	4	5	6	7	8	9	10	11
1	0	1	2	0	1	5	6	4	8	9	10
4	2	3	7	5	2	2	3	9	5	6	7
	6	6		8	6	7	11		10	11	
	5				9	10					
					4						

Adjacency matrix (undirected graph)

Undirected => symmetric matrix

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



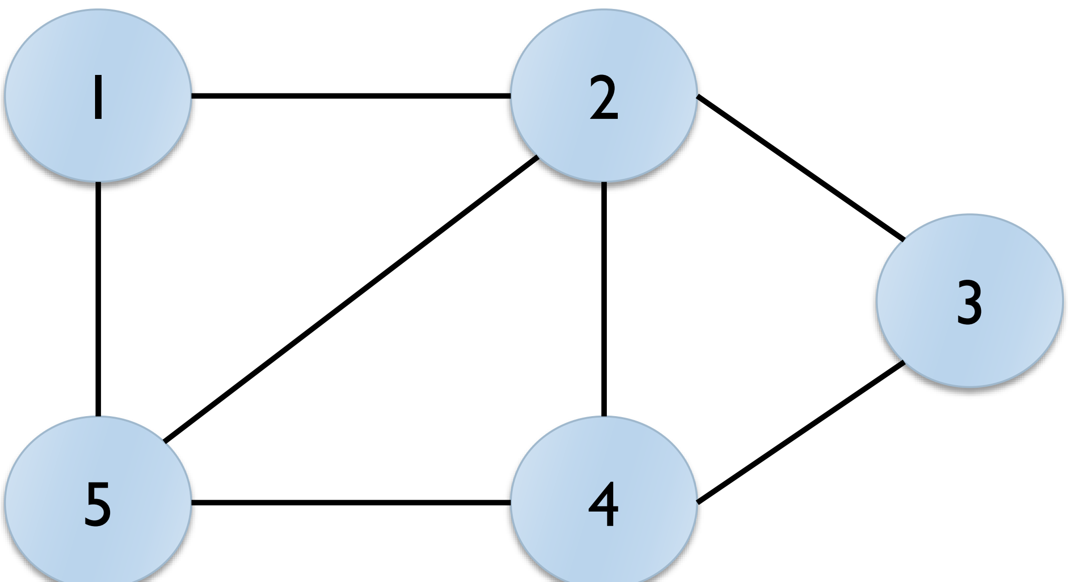
No self-loops: zero diagonal

Adjacency matrix (undirected graph)

Undirected => symmetric matrix

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

50% of memory can be saved



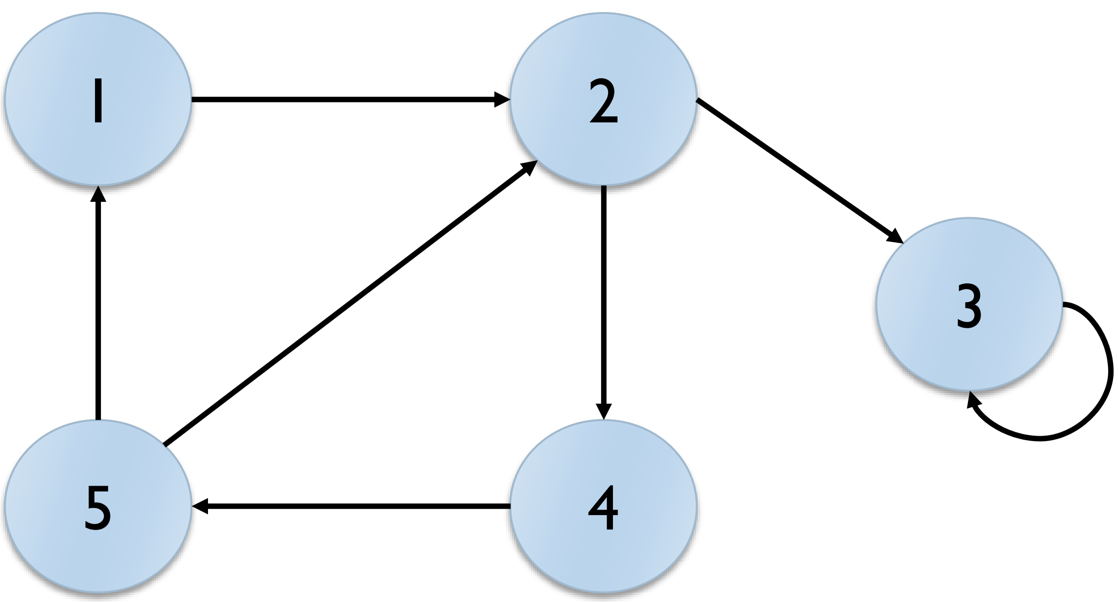
Adjacency matrix (directed graph)

To-vertex (in-edges)

From-vertex (out-edges)

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

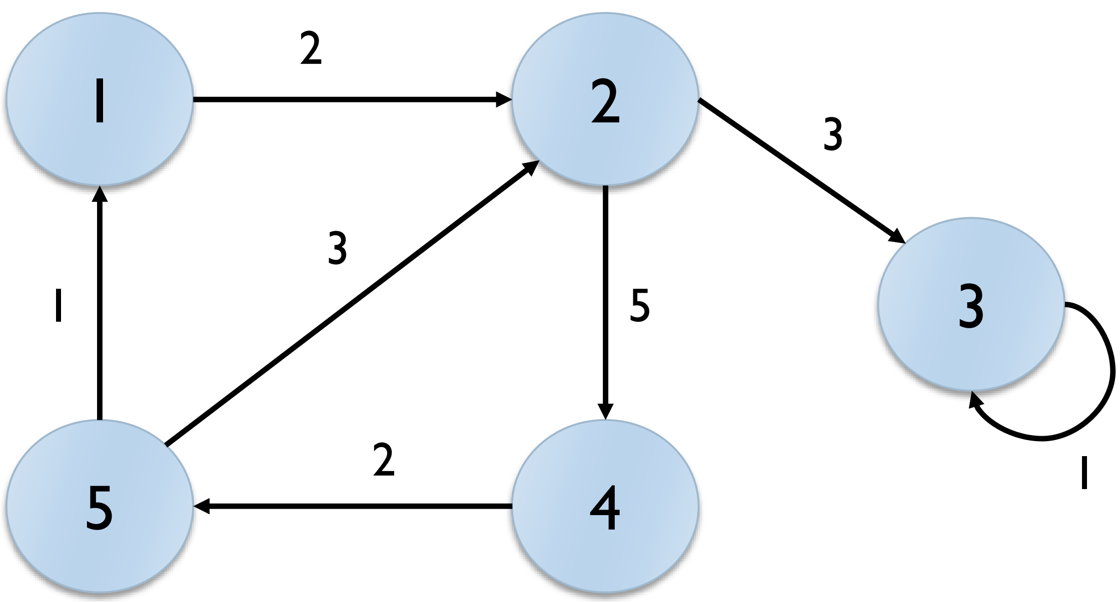
Self-loops in the diagonal



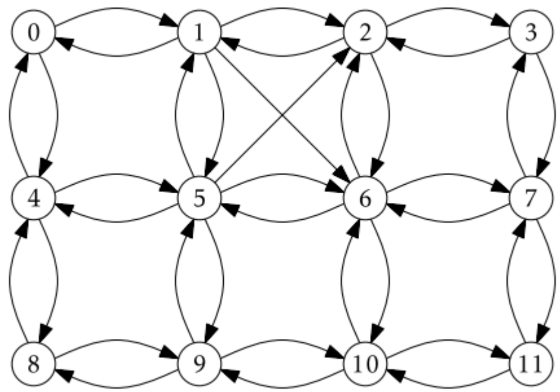
Adjacency matrix (weighted graph)

Values = edge weight

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



Adjacency matrix



	0	1	2	3	4	5	6	7	8	9	10	11
0	0	1	0	0	1	0	0	0	0	0	0	0
1	1	0	1	0	0	1	1	0	0	0	0	0
2	1	0	0	1	0	0	1	0	0	0	0	0
3	0	0	1	0	0	0	0	1	0	0	0	0
4	1	0	0	0	0	1	0	0	1	0	0	0
5	0	1	1	0	1	0	1	0	0	1	0	0
6	0	0	1	0	0	1	0	1	0	0	1	0
7	0	0	0	1	0	0	1	0	0	0	0	1
8	0	0	0	0	1	0	0	0	0	1	0	0
9	0	0	0	0	0	1	0	0	1	0	1	0
10	0	0	0	0	0	0	1	0	0	1	0	1
11	0	0	0	0	0	0	0	1	0	0	1	0

Incidence matrix (undirected graph)

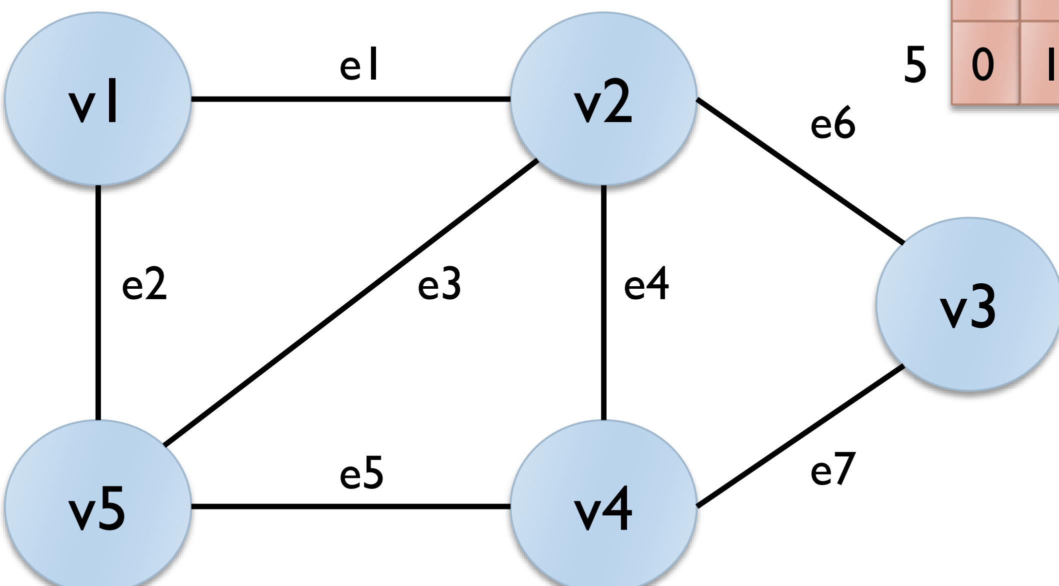
Vertices v1…v5

Edges e1…e7

	1	2	3	4	5	6	7
1	1	1	0	0	0	0	0
2	1	0	1	1	0	1	0
3	0	0	0	0	0	1	1
4	0	0	0	1	1	0	1
5	0	1	1	0	1	0	0

of "ones" in a row = vertex degree

Exactly 2 "ones" in every column



Incidence matrix (directed graph)

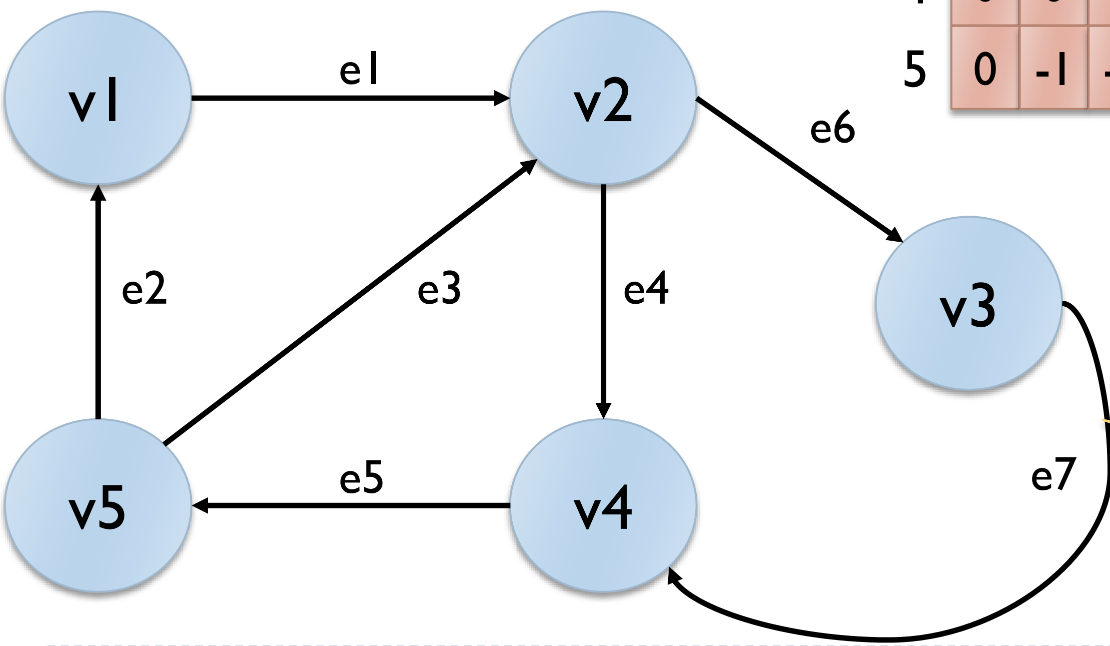
Vertices v1…v5

-1: exit
+1: enter

	1	2	3	4	5	6	7
1	-1	+1	0	0	0	0	0
2	+1	0	+1	-1	0	-1	0
3	0	0	0	0	0	+1	-1
4	0	0	0	+1	-1	0	+1
5	0	-1	-1	0	+1	0	0

Edges e1…e7

indegree = count(+1)
outdegree = count(-1)



Self loops can't be represented

Complexity & trade-offs

	Adjacency List	Adjacency Matrix	Incidence Matrix
Space	O(|V| + |E|)	O(|V| x |V|)	O(|V| x |E|)
Space	For sparse graphs $\|E\| << \|V\|^2$	For dense graphs $\|E\| \sim \|V\|^2$	
Check edge	O(1+deg(v))	O(1)	O(|E|) if *v,v'* are known, or O(1) if e is known
Find all adjacent	O(1+deg(v))	O(|V|)	O(|V|+|E|)

Graph visits

Representing and visiting graphs

Visit Algorithms

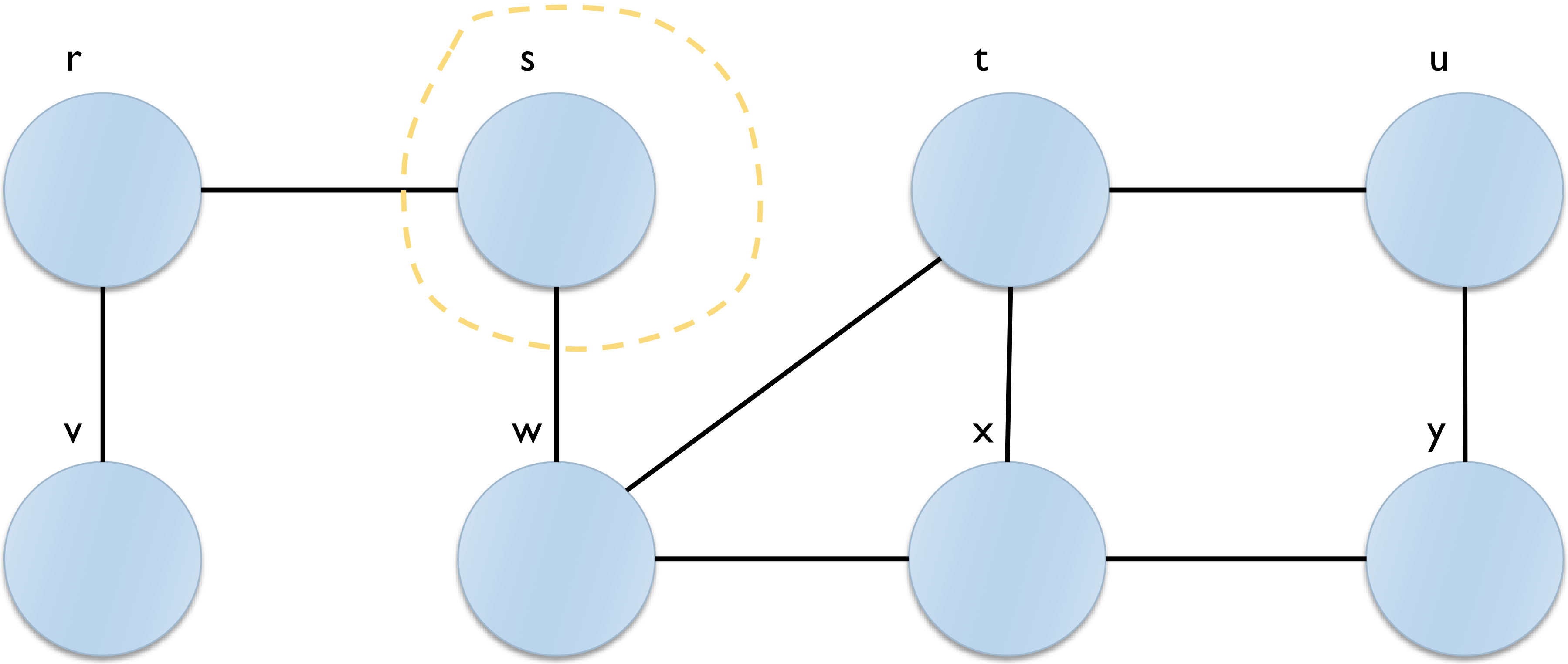
- Visit =
  - Systematic exploration of a graph
  - Starting from a ‘source’ vertex
  - Reaching all reachable vertices
- Main strategies
  - Breadth-first visit (“in ampiezza”)
  - Depth-first visit (“in profondità”)

Breadth-First Visit

- Also called Breadth-first search (BFV or BFS)
- All reachable vertices are visited "by levels"
  - L – level of the visit
  - S_L – set of vertices in level L
  - $L=0$, $S_0=\{ v_{source} \}$
  - Repeat while S_L is not empty:
    - S_{L+1} = set of all vertices:
      - not visited yet, and
      - adjacent to at least one vertex in S_L
    - $L=L+1$

Example

Source = s
L = 0
S_0 = {s}

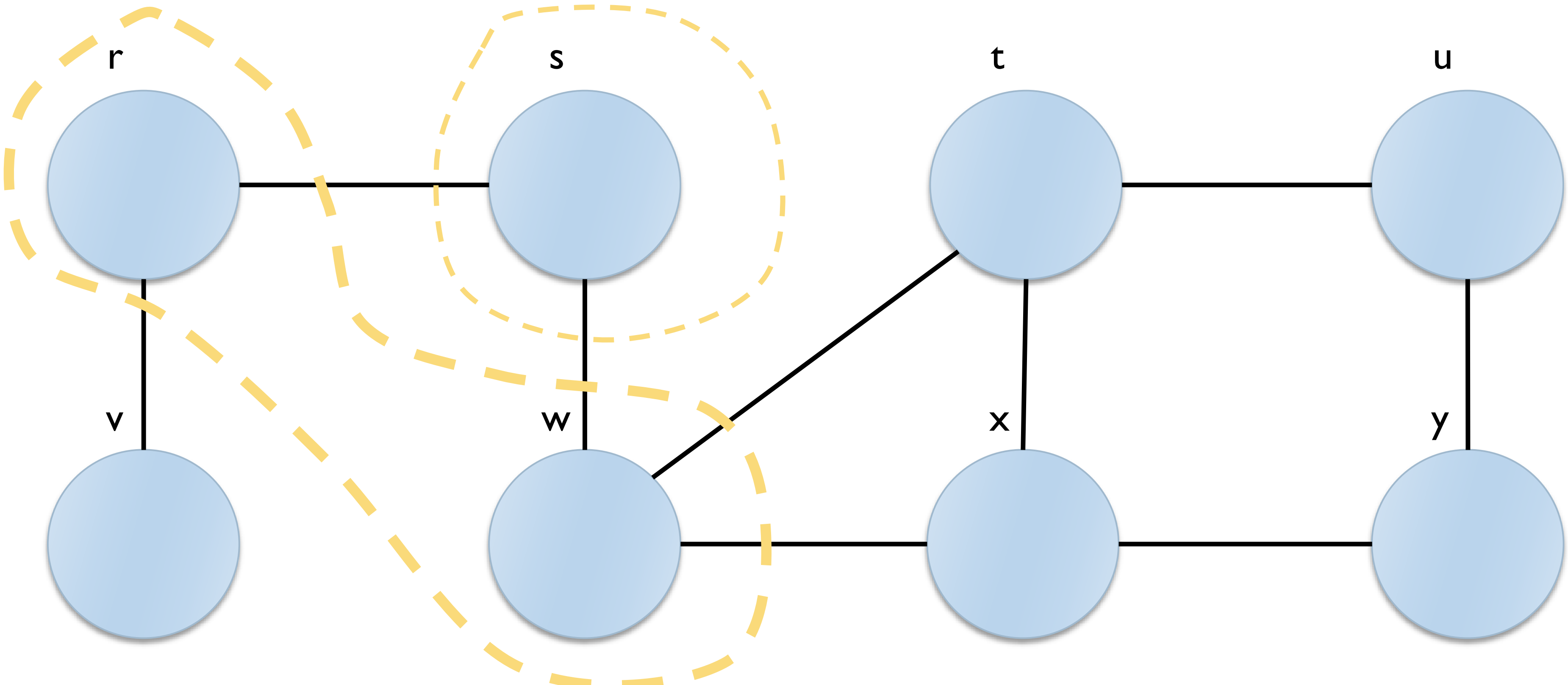


Example

L = 1

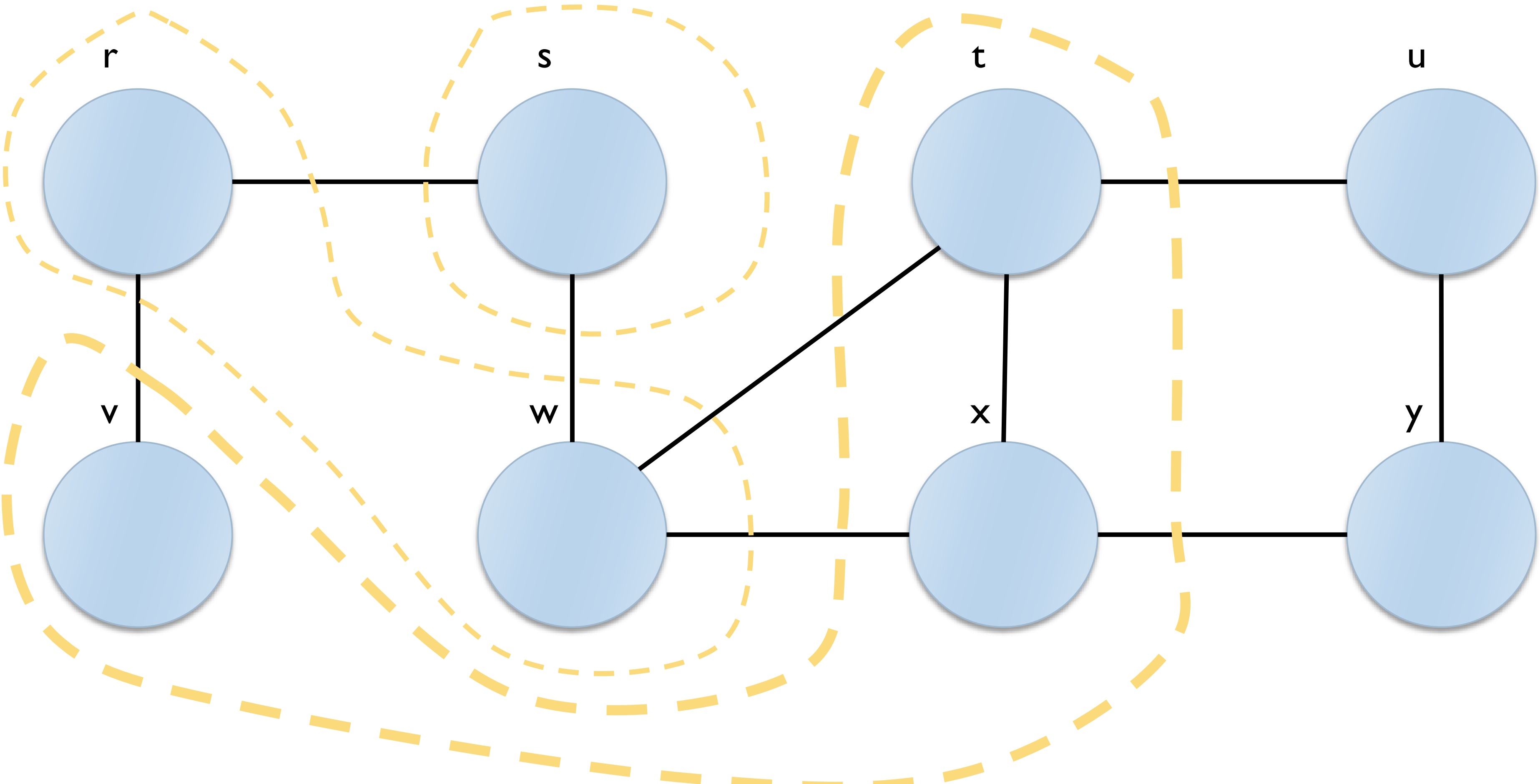
$S_0 = \{s\}$

$S_1 = \{r, w\}$



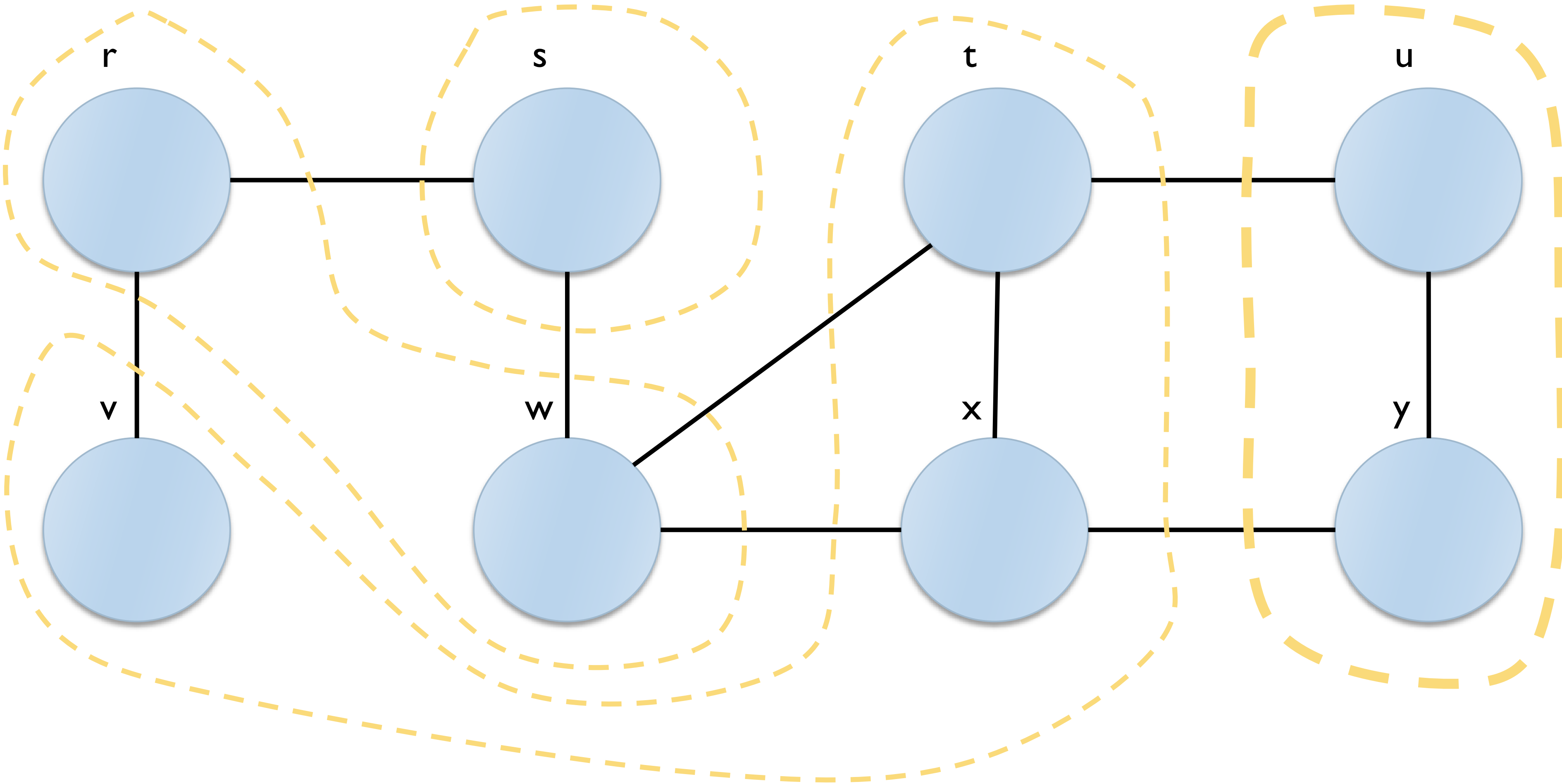
Example

L = 2
S_1 = {r, w}
S_2 = {v, t, x}



Example

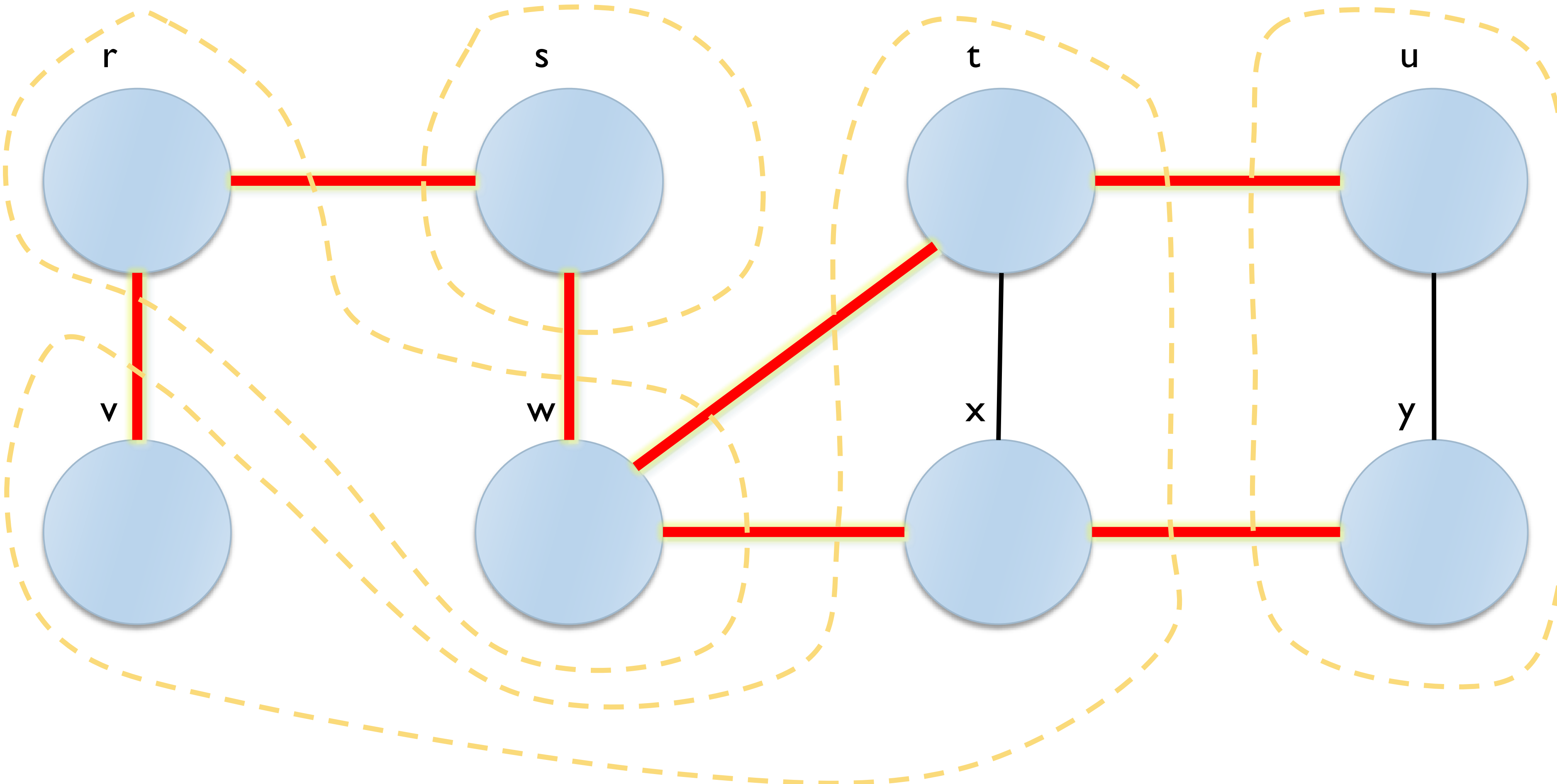
$L = 3$
$S_2 = \{v, t, x\}$
$S_3 = \{u, y\}$



BFS Tree

- The result of a BFV identifies a "visit tree" in the graph:
  - The tree root is the source vertex
  - Tree nodes are all graph vertices
    - (in the same connected component of the source)
  - Tree are a subset of graph edges
    - Those edges that have been used to "discover" new vertices.

BFS Tree



Minimum (shortest) paths

- Shortest path: the minumum number of edges on any path between two vertices
- The BFS procedure computes all minimum paths for all vertices, starting from the source vertex
- NB: unweighted graph : path length = number of edges

Depth First Visit

- Also called Depth-first search (DFV or DFS)
- Opposite approach to BFS
- At every step, visit one (yet unvisited) vertex, adjacent to the last visited one
- If no such vertex exist, go back one step to the previously visited vertex
- Lends itself to recursive implementation
  - Similar to tree visit procedures

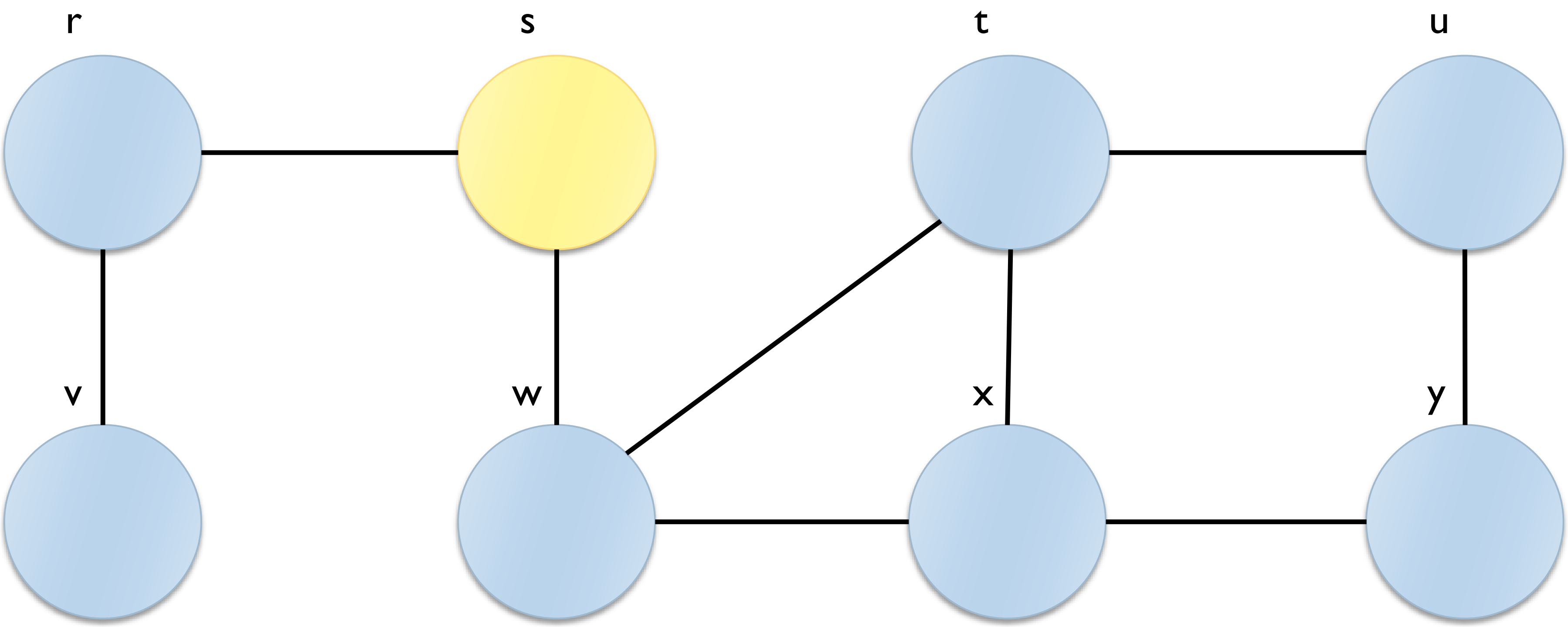
DFS Algorithm

- DFS(Vertex v)
  - For all (w : adjacent_to(v))
    - If(not visited (w))
      - Visit (w)
      - DFS(w)

- Start with: DFS(source)

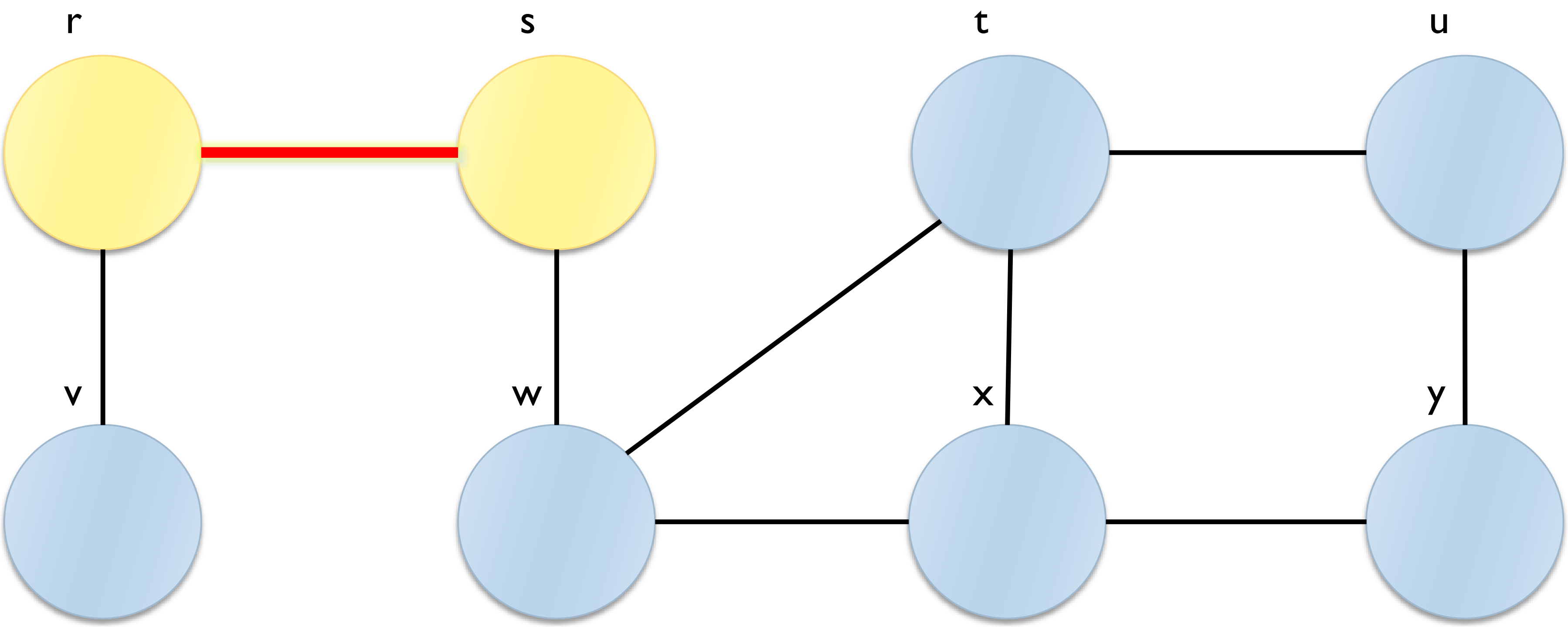
Example

Source = s



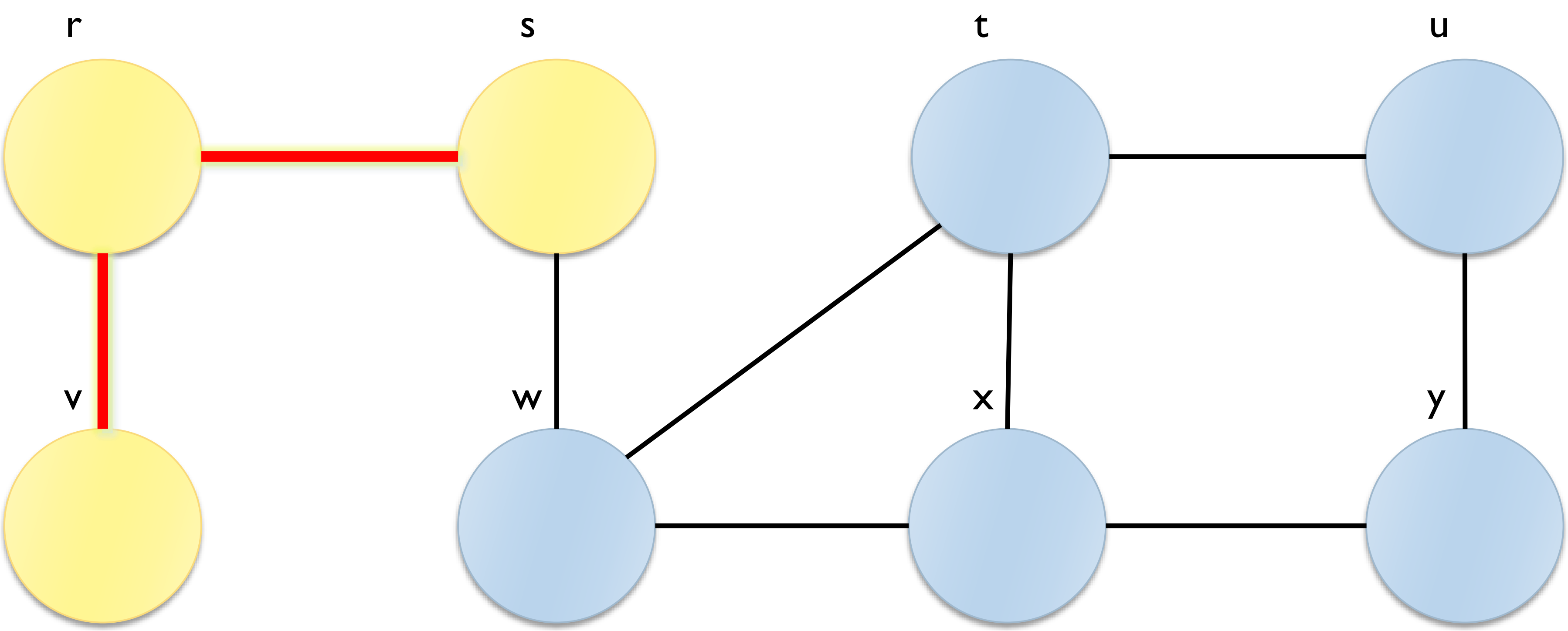
Example

Source = s
Visit r



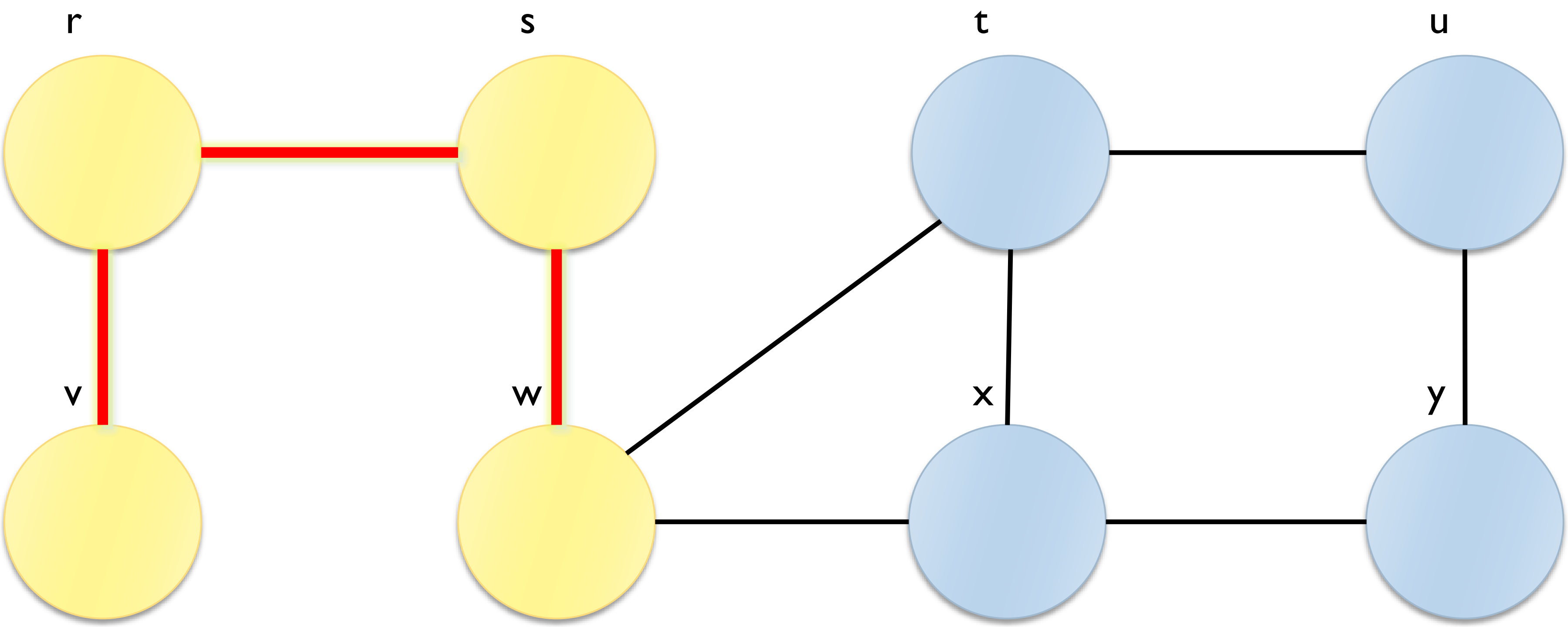
Example

Source = s
Visit r
Visit v



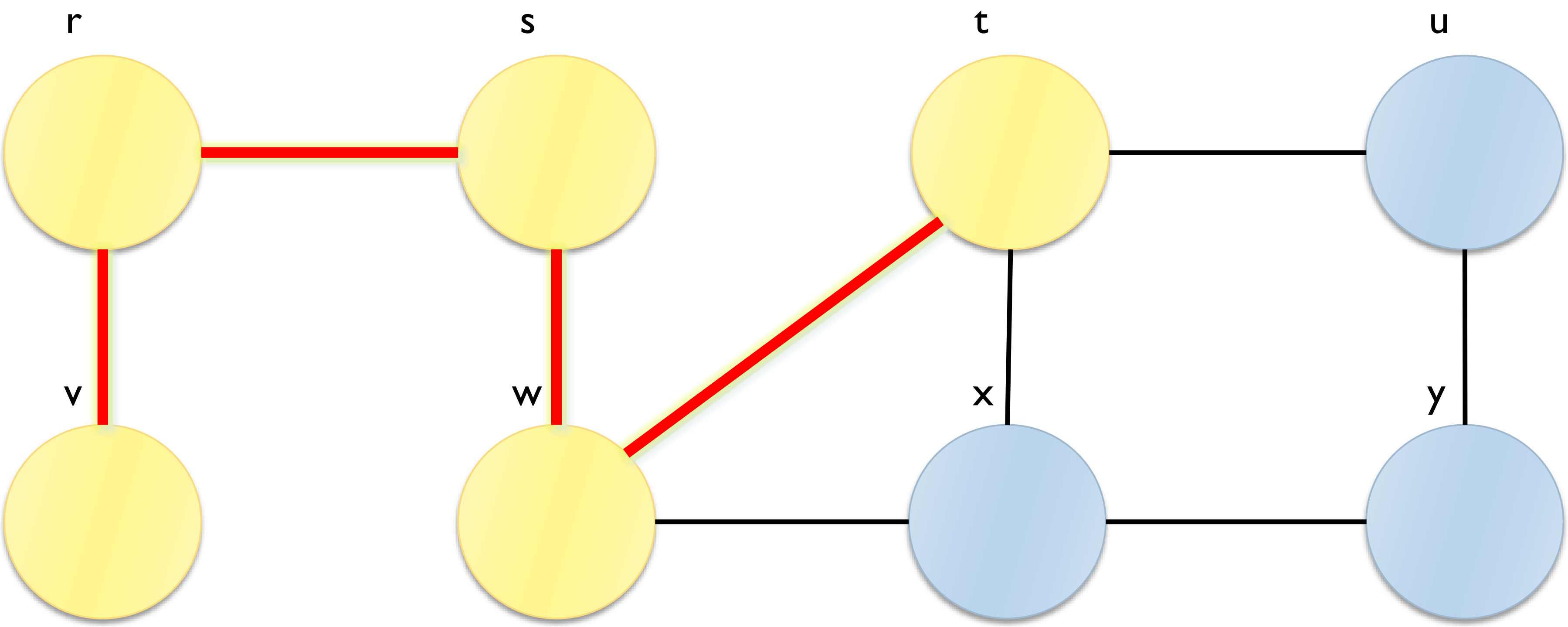
Example

Source = s
Back to r
Back to s
Visit w



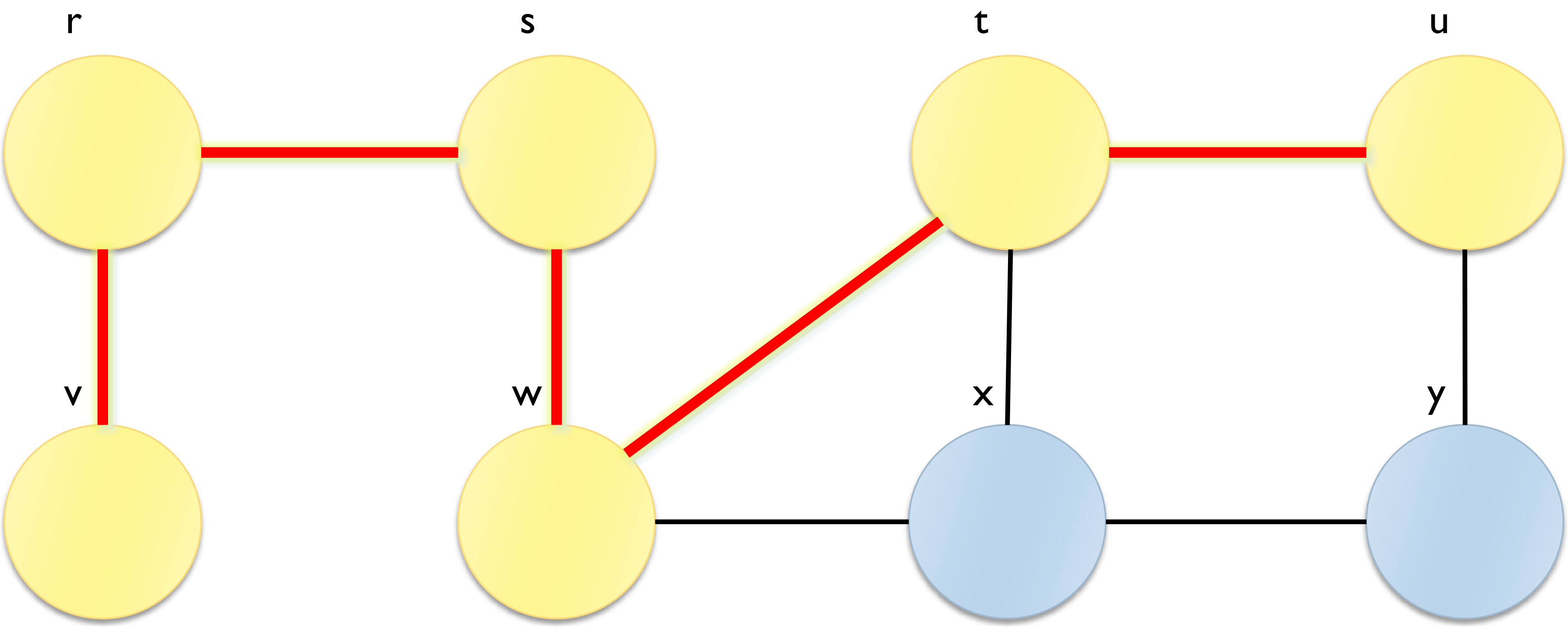
Example

Source = s
Visit w
Visit t



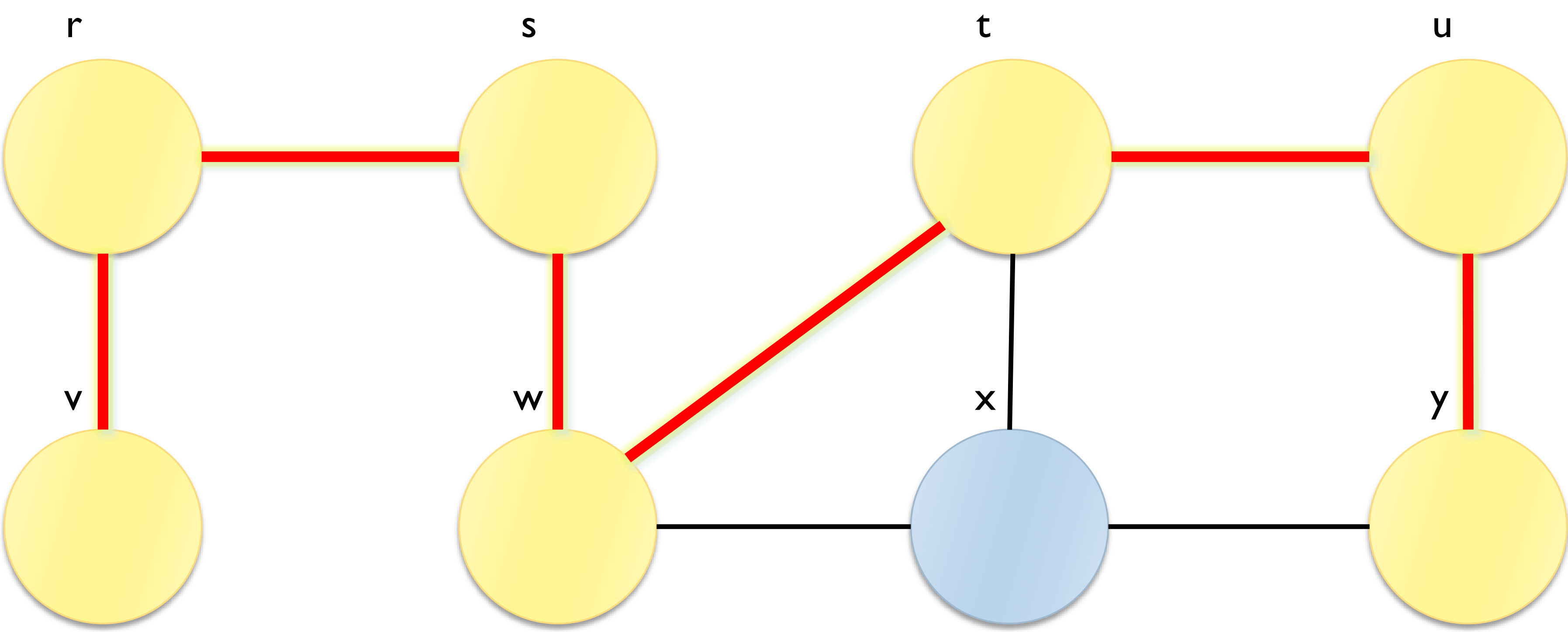
Example

Source = s
Visit w
Visit t
Visit u



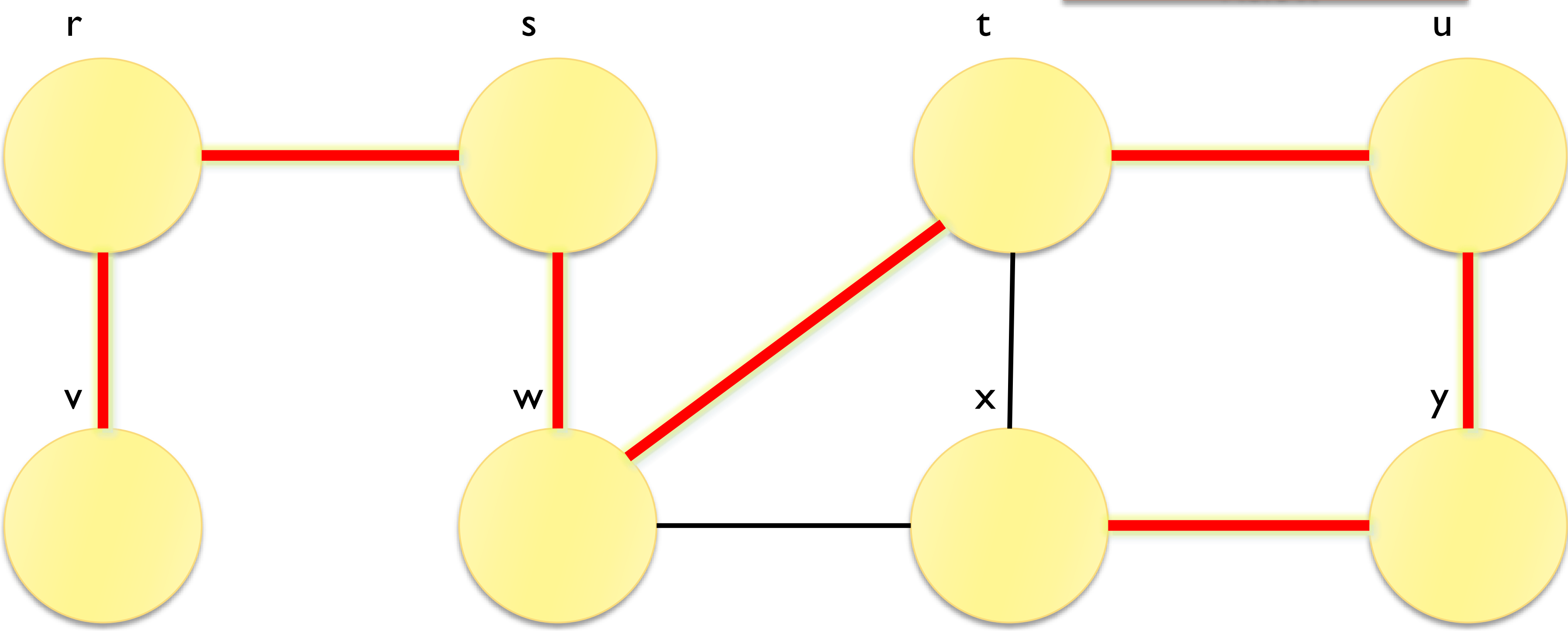
Example

Source = s
Visit w
Visit t
Visit u
Visit y

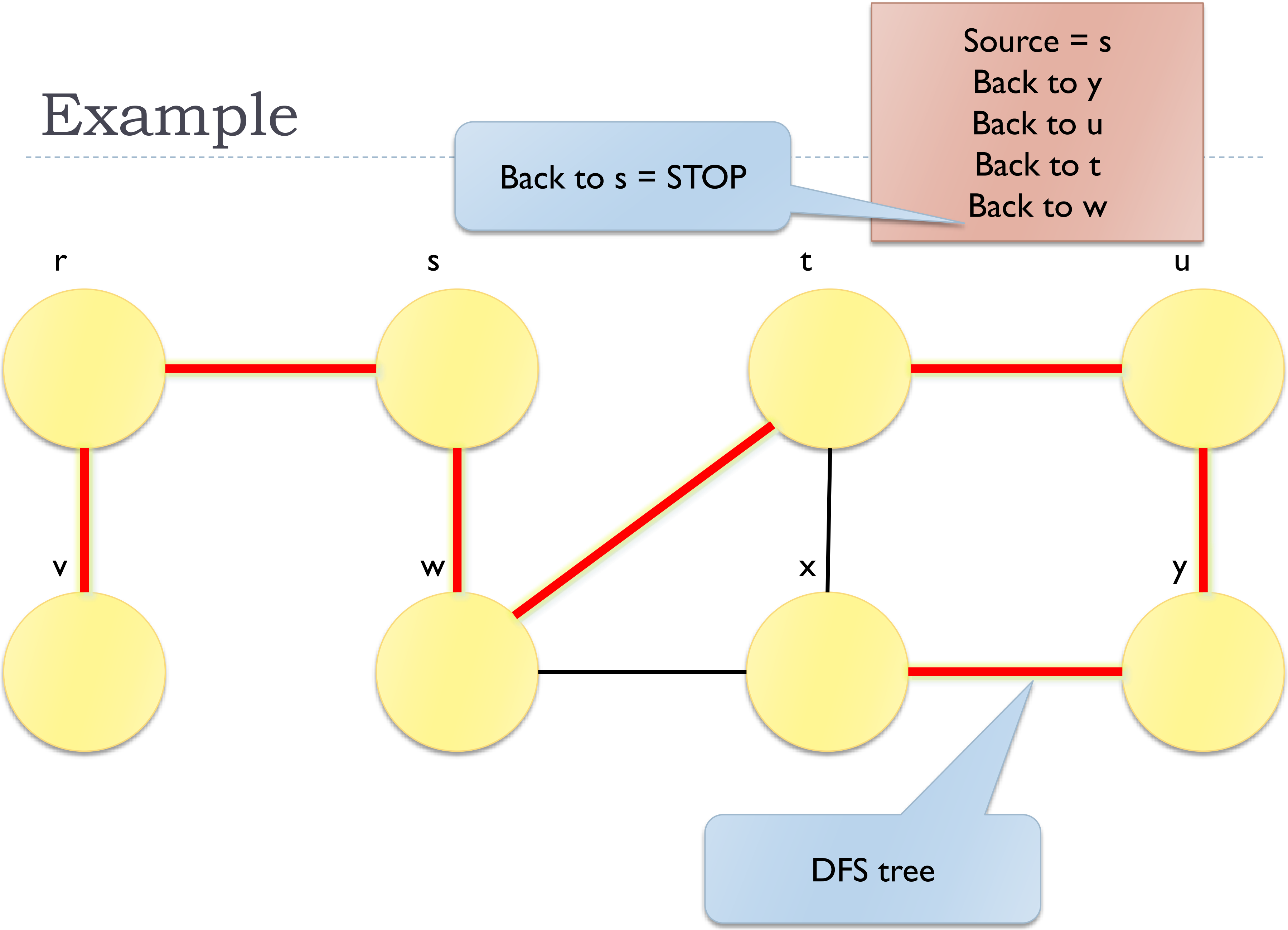


Example

Source = s
Visit w
Visit t
Visit u
Visit y
Visit x



Example



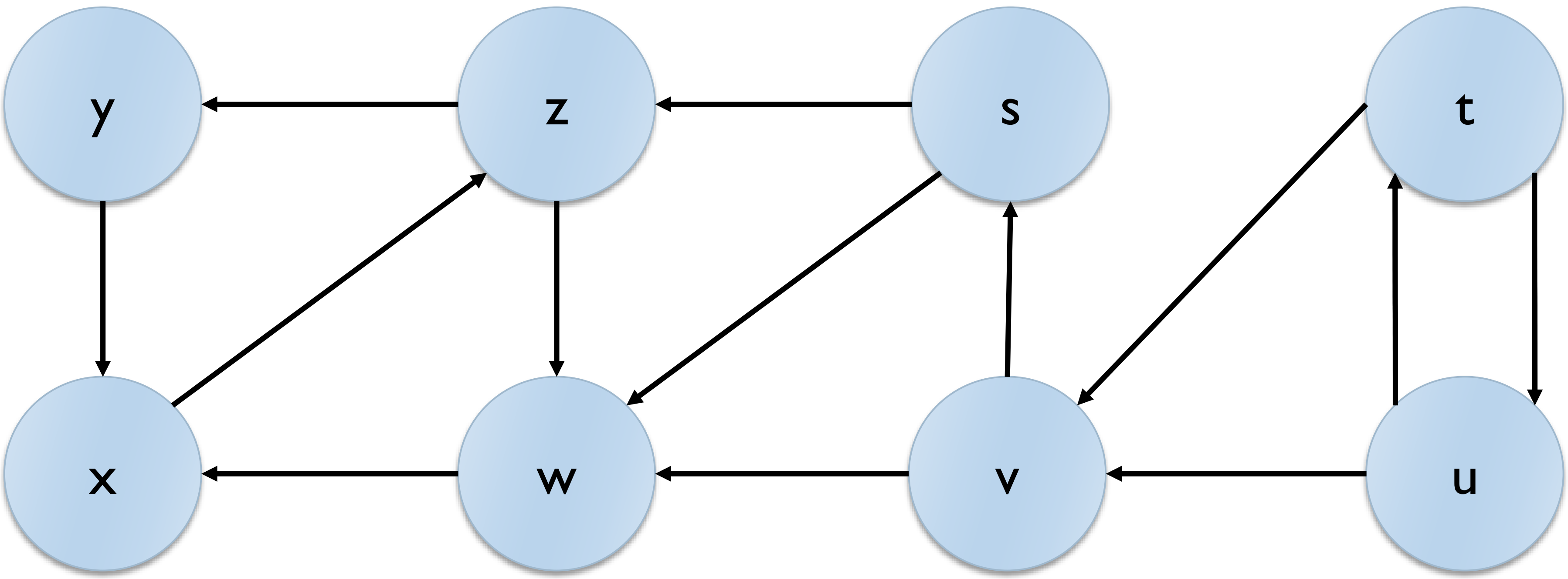
Source = s
Back to y
Back to u
Back to t
Back to w
Back to s = STOP
r
s
t
u
v
w
x
y
DFS tree

Edge classification

- In an directed graph, after a DFS visit, all edges fall in one of these 4 categories:
  - T: **Tree** edges (belonging to the DFS tree)
  - B: **Back** edges (not in T, and connect a vertex to one of its ancestors)
  - F: **Forward** edges (not in T and B, and connect a vertex to one of its descendants)
  - C: **Cross** edges (all remaining edges)

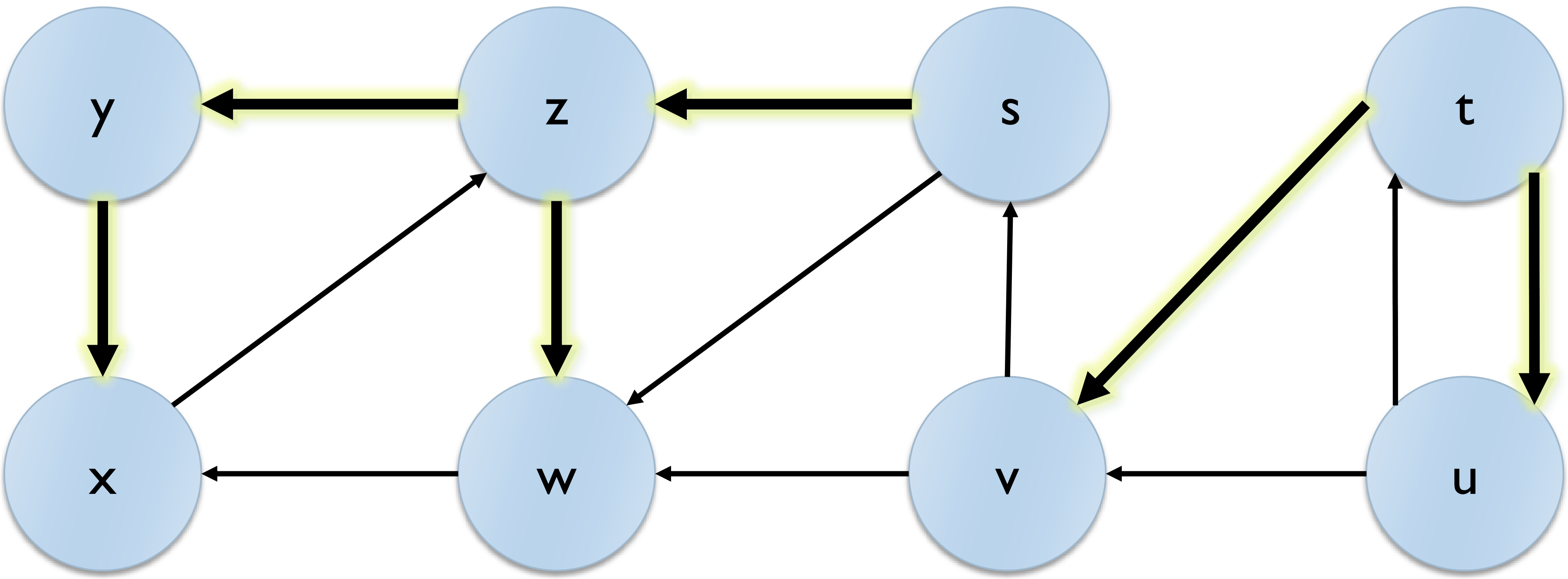
Example

Directed graph



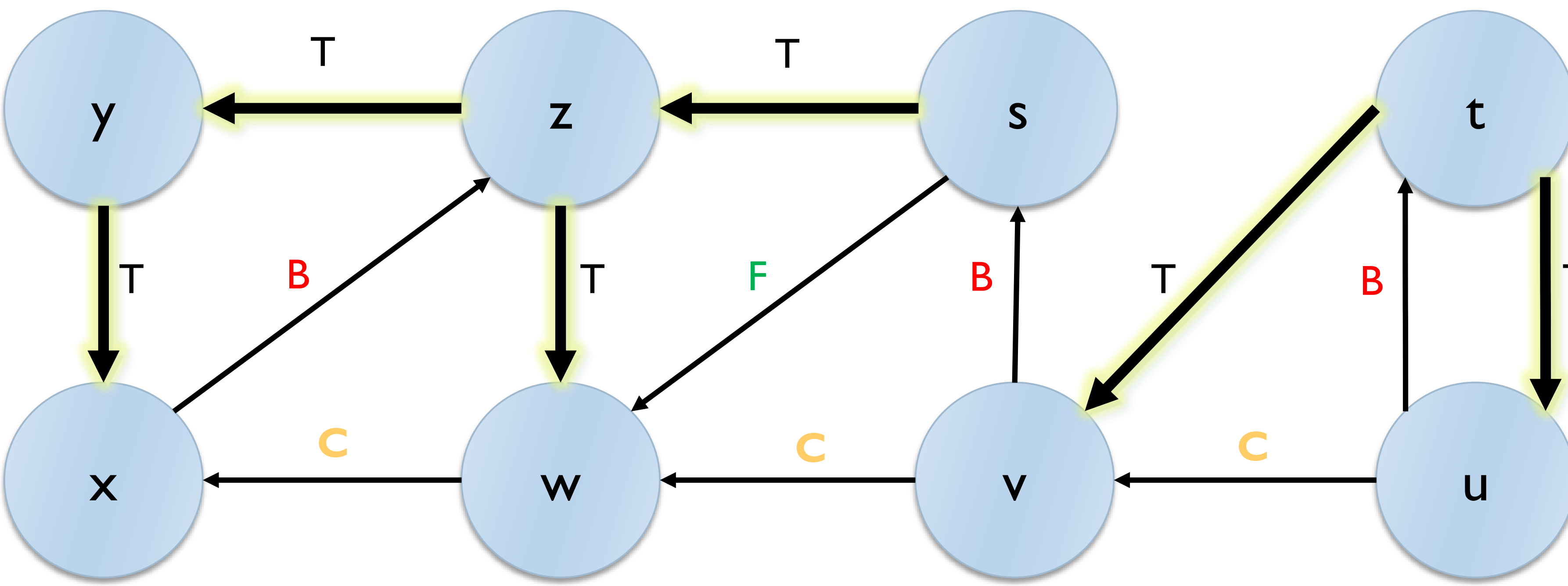
Example

DFS visit
(sources: s, t)



Example

Edge classification



Cycles

- Theorem:
- A directed graph is acyclic if and only if a depth-first visit does not produce any B edge

Complexity

- Visits have linear complexity in the graph size
  - BFS : O(V+E)
  - DFS : Θ(V+E)
- N.B. for dense graphs, $E = O(V^2)$

Resources

- Maths Encyclopedia: http://mathworld.wolfram.com/
- Basic Graph Theory with Applications to Economics
  http://www.isid.ac.in/~dmishra/mpdoc/lecgraph.pdf
- Application of Graph Theory in real world

http://prezi.com/tseh1wvpves-/application-of-graph-theory-in-real-world/

Resources

- Open Data Structures (in Java), Pat Morin, http://opendatastructures.org/
- Algorithms Course Materials, Jeff Erickson, http://www.cs.uiuc.edu/~jeffe/teaching/algorithms/
- Graphbook - A book on algorithmic graph theory, David Joyner, Minh Van Nguyen, and David Phillips, https://code.google.com/p/graphbook/

Licenza d'uso