Advanced Algorithms

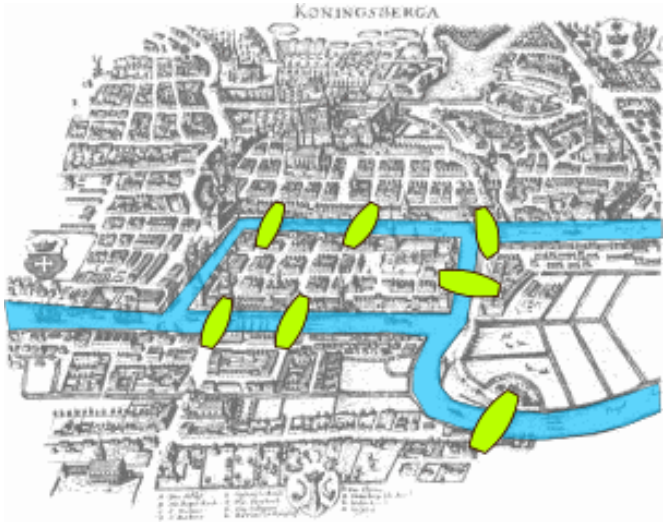
Graph Theory I

Valeria Fionda

KRDB Research Group
Faculty of Computer Science
Free University of Bozen-Bolzano
(Partially based on slides by Y. Peng)

Introduction

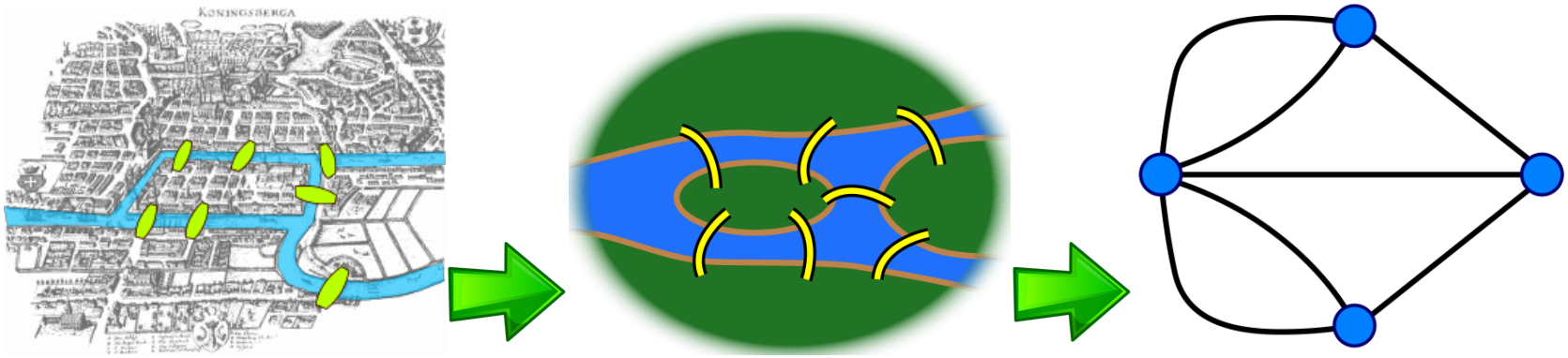
- Graph Theory was born in 1736 with Euler's paper on konigsberg bridge problem.



- **Problem**. The Seven Bridges of Knigsberg is a historically notable problem in mathematics. The city of Knigsberg included two large islands which were connected to each other and to the mainland by seven bridges. The problem was to find a walk through the city that would cross each bridge once and only once. The islands could not be reached by any route other than the bridges, and every bridge must have been crossed completely every time.

Introduction

- Graph Theory was born in 1736 with Euler's paper on konigsberg bridge problem.



- Euler proved that the problem has no solution.
- The difficulty was the development of a technique of analysis and of subsequent tests that established this assertion with mathematical rigour. Indeed for the next 100 years nothing more was done!

Introduction

- After more than 100 years, a particular class of graphs, *trees*, was introduced by Kirchhoff that developed a theory for electrical networks and by Caylay in theoretical chemistry.
- The term "graph" was introduced by Sylvester in a paper published in 1878.
- The first book on graph theory was published in 1936 by Konig.
- One of the most famous and productive problems of graph theory is the four color problem, proposed in 1852:

  Is it true that any map may have its regions coloured with four colors, in such a way that any two regions having a common border have different colors?

  Many incorrect proofs have been proposed. It remained unsolved for more than a century, when in 1969 Heinrich Heesch published a method for solving the problem using computers.

Introduction

- The two main way to represent graphs are:
  - adjacency matrix;
  - adjacency list;

Adjacency matrix

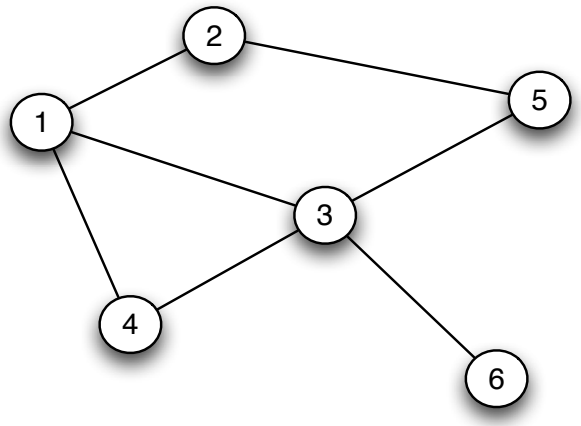
- Let $G = (V, E)$ such that $V = \{v_1, ..., v_n\}$ and $|V| = n$, the adjacency matrix is a nxn matrix each element of which is:

$$a_{ij} = \begin{array}{c} 1 \text{ if there is an edge from } v_i \text{ to } v_j \\ 0 \text{ otherwise} \end{array}$$

- For undirected graphs, the matrix is symmetric;
- **Pros**:
  - The presence of a particular edge can be checked in constant time, with just one memory access.
- **Cons**:
  - The matrix requires $O(n^2)$ space, which is wasteful if the graph does not have very many edges.

Adjacency matrix - Example

- Let $G = (V, E)$ be the graph represented in the following figure:



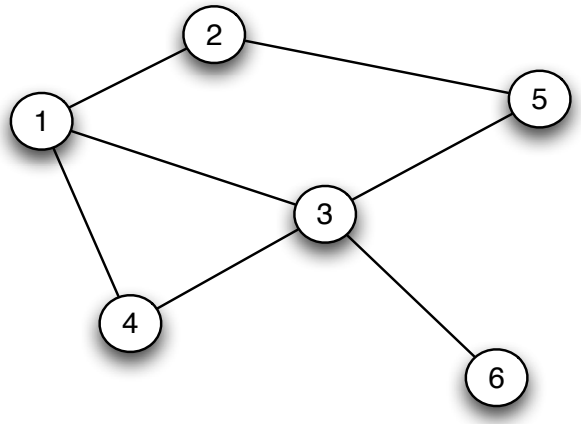
	1	2	3	4	5	6
1	0	1	1	1	0	0
2	1	0	0	0	1	0
3	1	0	0	1	1	1
4	1	0	1	0	0	0
5	0	1	1	0	0	0
6	0	0	1	0	0	0

Adjacency list

- Let $G = (V, E)$ such that $V = \{v_1, ..., v_n\}$ and $|V| = n$, the adjacency list consists of n linked lists, one per vertex.
- The linked list for vertex v holds the names of vertices to which v has an outgoing edge.
- For undirected graphs, the same edge is stored in two different linked lists;
- **Pros**:
  - The total size of the data structure is $O(|E|)$.
- **Cons**:
  - Checking for a particular edge (u, v) is no longer constant time, because it requires sifting through us adjacency list.

Adjacency list - Example

- Let $G = (V, E)$ be the graph represented in the following figure:



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

How to choose the best representation?

- The best representation depends on the graph and in particular on the relationship between $|V|$, the number of nodes in the graph, and $|E|$, the number of edges.
- Indeed, $|E|$ can vary from $|V|$ for sparse graphs to $|V|^2$ for dense graphs.
- In the former case (i.e., sparse graph) the adjacency list is better.
- In the latter case (i.e., dense graph) the adjacency matrix should be preferred.
- Dealing with sparse or dense graphs can influence also the choose of the better algorithm for solving a problem.

Connectivity

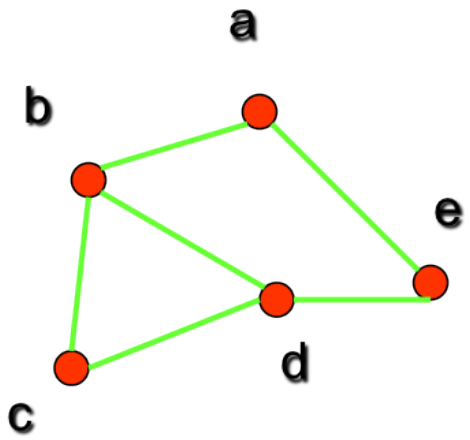
- **Definition**: A path of length n from u to v, where n is a positive integer is a sequence of edges e_1, e_2, , e_n of the graph such that $e_1 = (x_0, x_1), e_2 = (x_1, x_2), , e_n = (x_{n-1}, x_n)$, where $x_0 = u$ and $x_n = v$.
- A path is a *circuit* if it begins and ends at the same vertex, that is, if $u = v$.
- A path or circuit is simple if it does not contain the same edge more than once.

Connectivity

- **Definition**: A graph is called connected if there is a path between every pair of distinct vertices in the graph.
- *Note*: A graph consisting of only one vertex is always connected, because it does not contain any pair of distinct vertices.

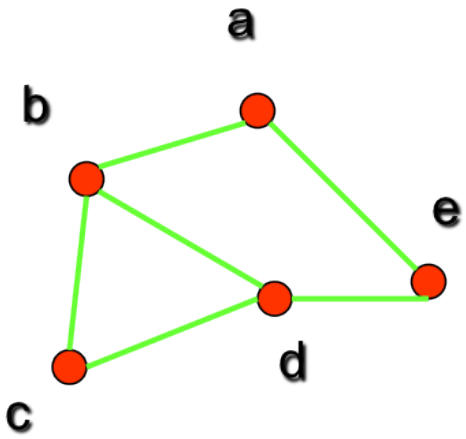
Connectivity

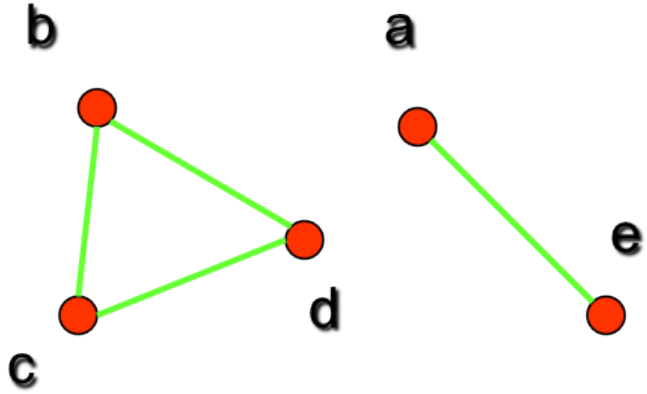
- **Example**: Are the following graphs connected?.



Connectivity

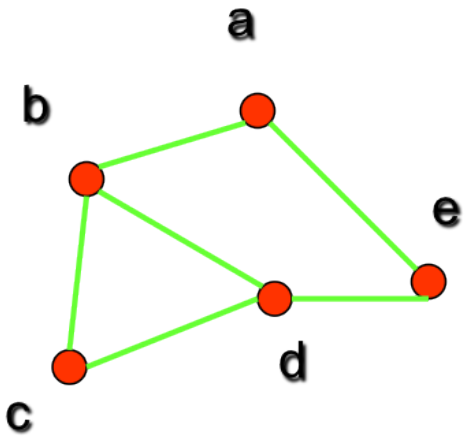
- **Example**: Are the following graphs connected?.

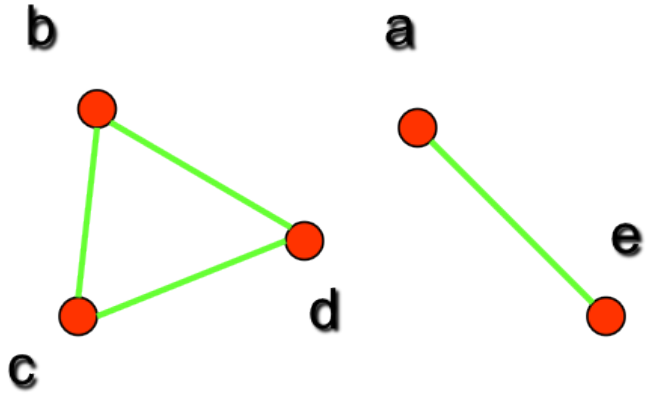


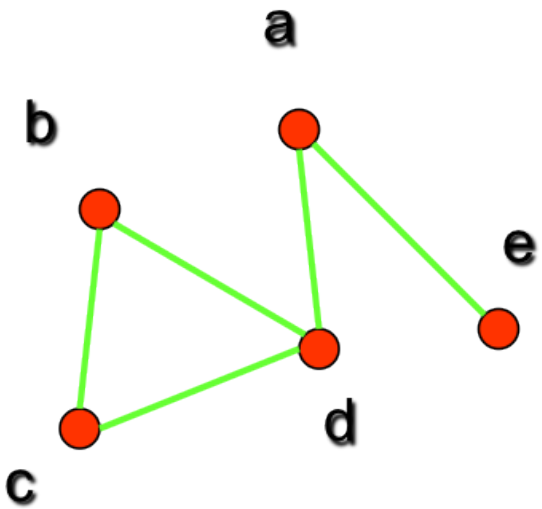


Connectivity

- **Example**: Are the following graphs connected?.

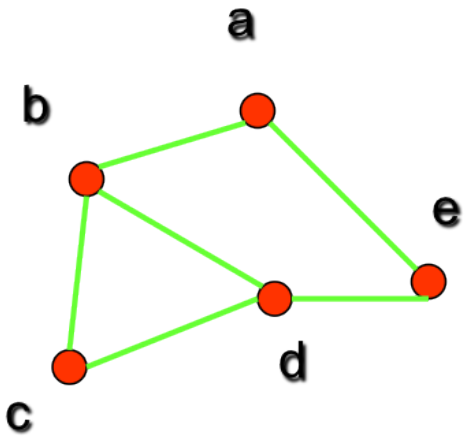


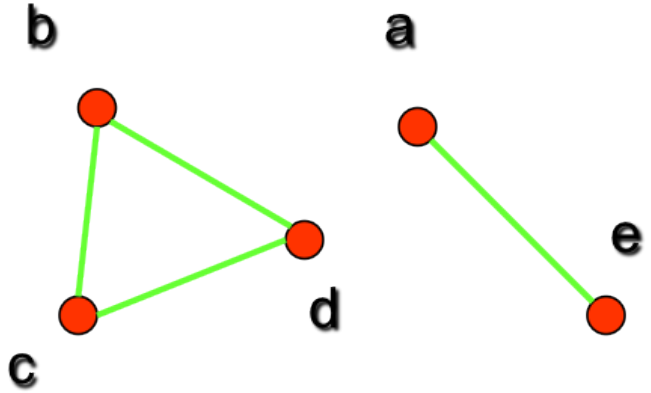


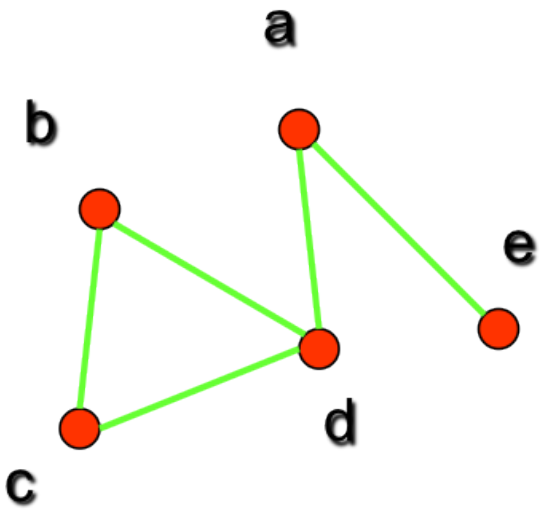


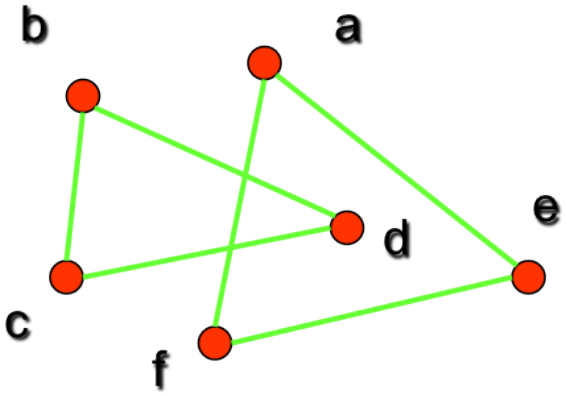
Connectivity

- **Example**: Are the following graphs connected?.







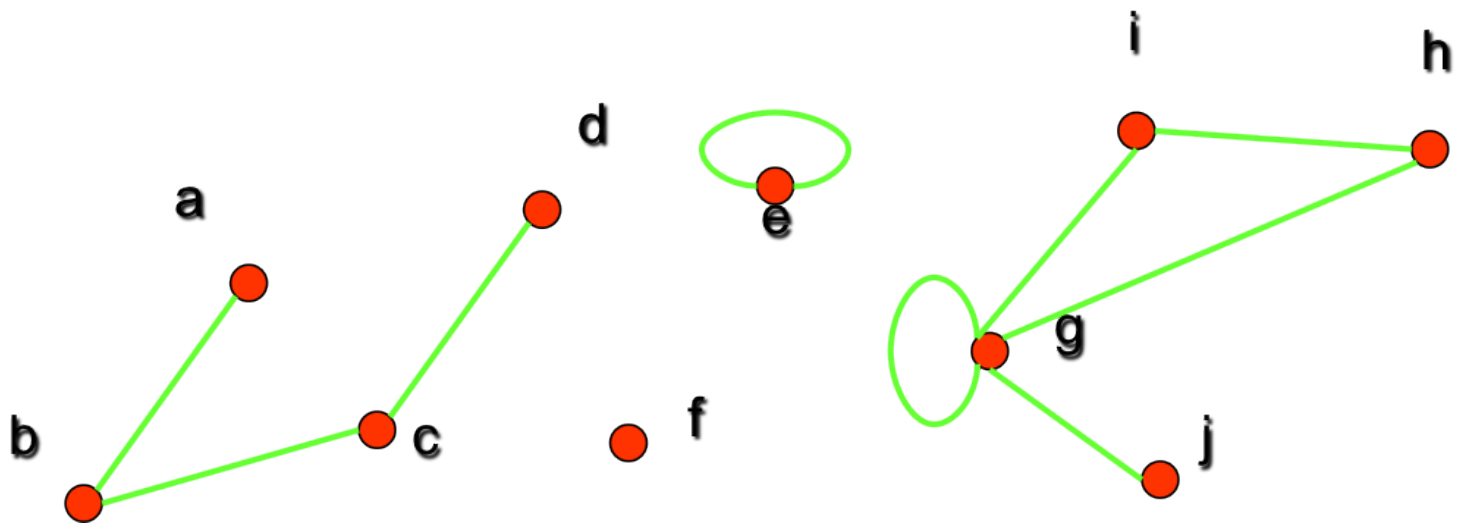


Connectivity

- **Definition**: A graph that is not connected is the union of two or more connected subgraphs, each pair of which has no vertex in common. These disjoint connected subgraphs are called the connected components of the graph.
- **Definition**: A connected component of a graph G is a maximal connected subgraph of G. (i.e., if vertex v in G belongs to a connected component, then all other vertices in G that is connected to v must also belong to that component).

Connectivity

- **Example**: What are the connected components in the following graph?.

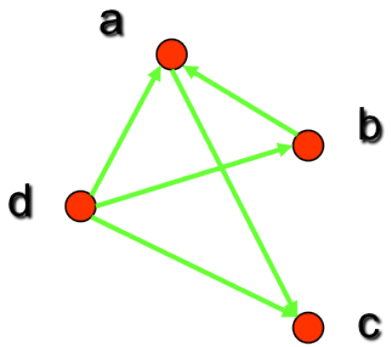


Connectivity

- **Definition**: A directed graph is strongly connected if there is a path from a to b and from b to a whenever a and b are vertices in the graph.
- **Definition**: A directed graph is weakly connected if there is a path between any two vertices in the underlying undirected graph.

Connectivity

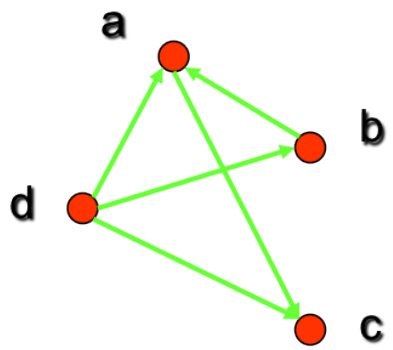
- **Example**: Are the following directed graphs strongly or weakly connected?



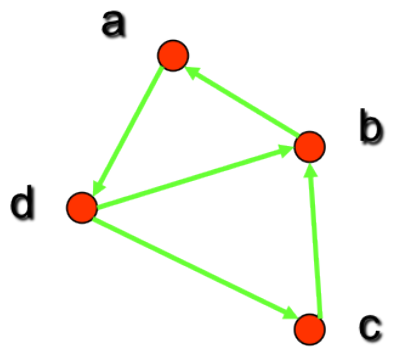
- Weakly connected, because, for example, there is no path from b to d.

Connectivity

- **Example**: Are the following directed graphs strongly or weakly connected?



- Weakly connected, because, for example, there is no path from b to d.



- Strongly connected, because there are paths between all possible pairs of vertices.

Connectivity

- Some graphs are more connected than others.
- The connectivity is measured by two parameters:
  - edge connectivity: $\lambda(G) = \{k \mid k = |S|, G - S \text{ disconnected }, S \subseteq E_G\}$, $\lambda(G)$ is the smallest subset of edges whose removal disconnects G.
  - vertex connectivity: $k(G) = \{c \mid c = |S|, G - S \text{ disconnected }, S \subseteq V_G\}$, $k(G)$ is the minimum number of vertices whose removal disconnects G.

Cut set

- A cut set of a connected graph G is a set of edges with the following properties:
  - The removal of all edges in S disconnects G.
  - The removal of some (but not all) of edges in S does not disconnect G.
- Cut set is also known as edge-cut.

Vertex Cut set

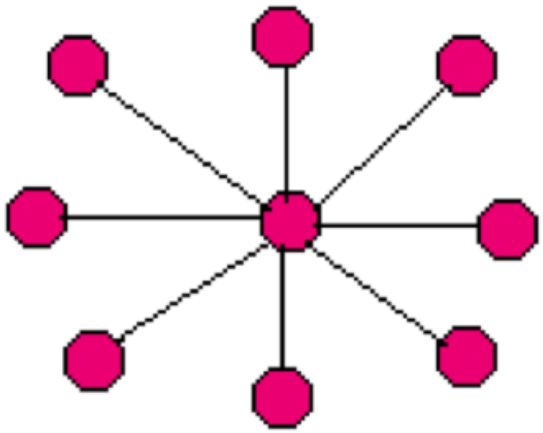
- A vertex-cut set or separating set of a connected graph G is a set of vertices with the following properties:
  - The removal of all the vertices in S disconnects G.
  - The removal of some (but not all) vertices in S does not disconnect G.

Connectivity

- A cut-edge (no edge-cut) or bridge is an edge-cut consisting of a single edge.
- Similarly, a cut-vertex (no vertex-cut) is a vertex-cut consisting of a single vertex.

Connectivity

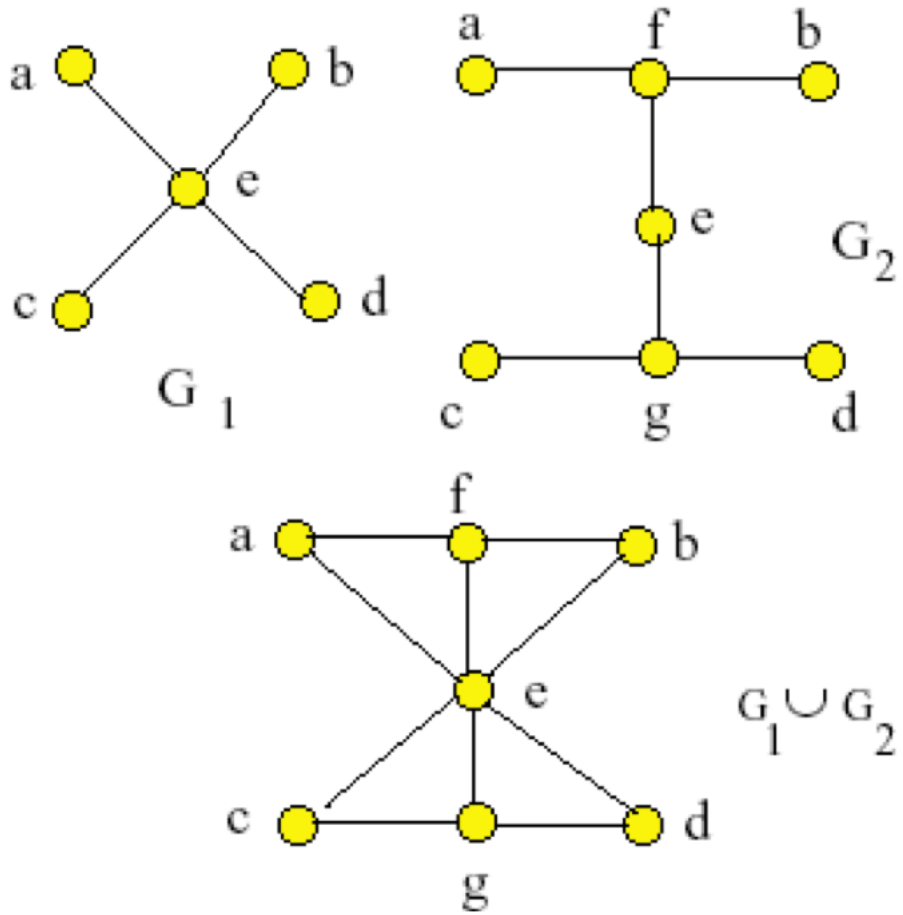
- **Example**: In the star network the center vertex is a cut vertex. All edges are cut edges.



$K_{1,8}$

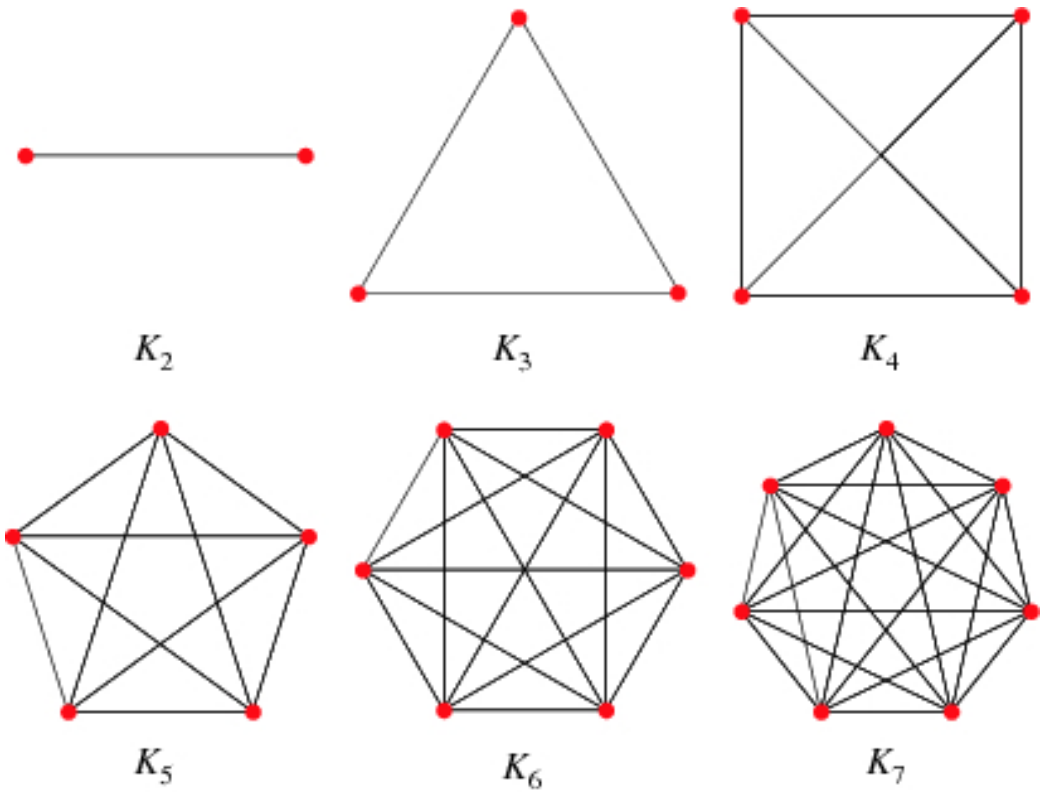
Connectivity

- **Example**: In the graphs G_1 and G_2 every edge is a cut edge. In the union, no edge is a cut edge. Vertex e is a cut vertex in all graphs.



Complete graphs

- The graphs for which $k(G) = |V_G| - 1$ are said to be complete.
- A complete graph is a graph in which every pair of distinct vertices is connected by a unique edge.
- The complete graph on n vertices has $n(n-1)/2$ edges and is denoted by K_n



Connectivity

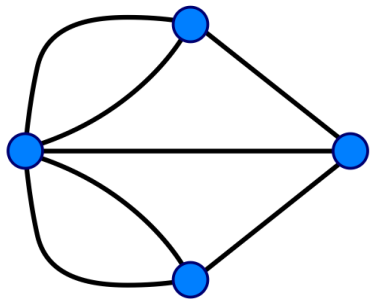
- For every connected graph:

$$k(G) \leq \lambda(G) \leq \delta_{min}(G)$$

where $\delta_{min}(G)$ is the minimum degree of the nodes of G.

Euler Graph

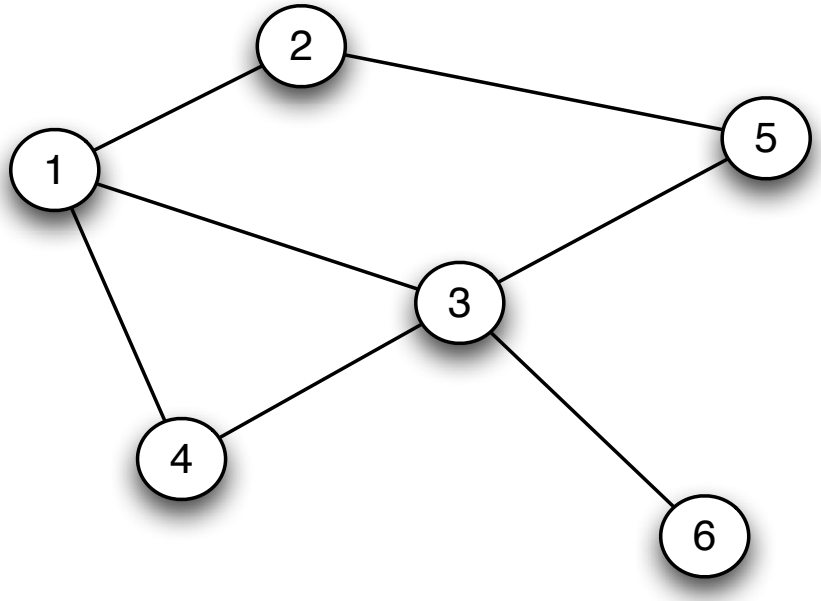
- **Euler Graph**: If we can find a path (resp., circuit) in a graph that involves all the edges of the graph, this path(resp., circuit) is called an Eulerian path (resp., Eulerian circuit).
- A graph having an eulerian circuit is called eulerian graph.
- A connected graph G is an Eulerian graph iff all vertices in G have even degree.
- For the existence of Eulerian path it is necessary that no more than two vertices have an odd degree.
- If there are no vertices of odd degree, all Eulerian paths are circuits.
- If there are exactly two vertices of odd degree, all Eulerian path starts at one of them and end at the other.



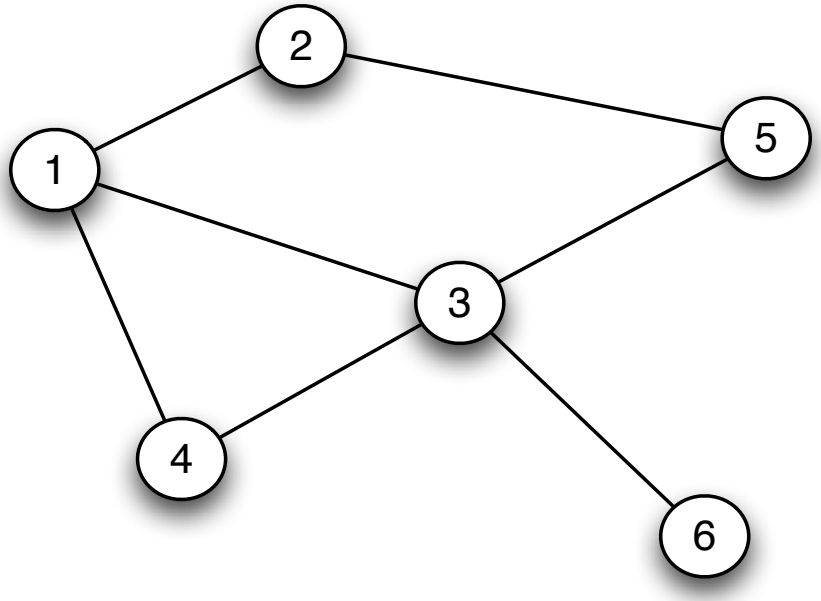
Constructing Eulerian paths and circuits

- **Fleury's algorithm**: Let G be a connected graph having at most two vertices of odd degree.
  1. start with a vertex of odd degree (if the graph has none, start with any vertex);
  2. at each step we move across an edge whose deletion would not disconnect the graph, unless we have no choice, then we delete that edge.
  3. at the end of the algorithm there are no edges left, and the sequence of edges we moved across forms a Eulerian cycle if the graph has no vertices of odd degree; or a Eulerian path if there are exactly two vertices of odd degree.
- **Complexity**: the graph traversal in Fleury's algorithm is linear in the number of edges i.e. $O(|E|)$; the complexity of a naive algorithm for detecting bridges is $O(|E|)$ and such an algorithm must be run every time an edge is deleted; thus, the complexity is $O(|E|^2)$.

Fleury's algorithm - Example



Fleury's algorithm - Example



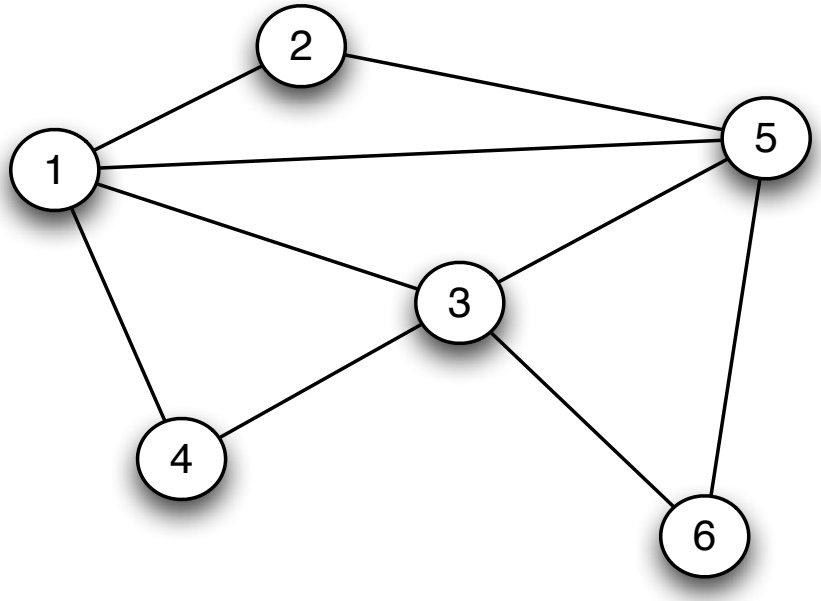
Solutions:

- $(1,2),(2,5),(5,3),(3,4),(4,1),(1,3),(3,6)$;
- $(1,3),(3,5),(5,2),(2,1),(1,4),(4,3),(3,6)$;
- $(1,3),(3,4),(4,1),(1,2),(2,5),(5,3),(3,6)$;
- $(6,3),(3,4),(4,1),(1,2),(2,5),(5,3),(3,1)$;
- ...;

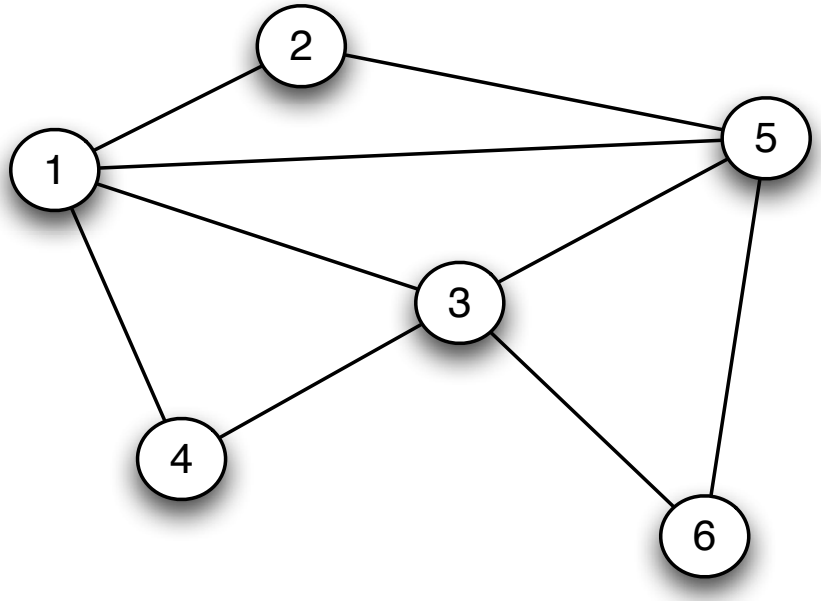
Constructing Eulerian paths and circuits

- **Hierholzer's algorithm**: Let G be a connected graph having all vertices of even degree.
  1. Choose any starting vertex v, and follow a path of edges from v until returning to v.
  2. Note that it is not possible to get stuck at any vertex other than v, because the even degree of all vertices ensures that, when the path enters another vertex w there must be an unused edge leaving w.
  3. The tour formed in this way is a closed tour, but may not cover all the vertices and edges of the initial graph.
  4. As long as there exists a vertex v that belongs to the current tour but that has adjacent edges not part of the tour, start another path from v, following unused edges until returning to v, and join the tour formed in this way to the previous tour.
- **Complexity**: using appropriate data structures the algorithm runs in linear time $O(|E|)$.

Hierholzer's algorithm - Example



Hierholzer's algorithm - Example



Solutions:

- $1 = (3,5),(5,2),(2,1),(1,4),(4,3)$; $2 = (3,1),(1,5),(5,6),(6,3)$;
- $1 = (1,3),(3,5),(5,2),(2,1)$; $2 = (1,4),(4,3),(3,6),(6,5),(5,1)$;
- $1 = (1,3),(3,5),(5,1)$; $2 = (1,4),(4,3),(3,6),(6,5),(5,1)$;
- $1 = (5,6),(6,3),(3,5)$; $2 = (5,2),(2,1),(1,5)$; $3 = (1,4),(4,3),(3,1)$;