

1 Graph Theory Vocabulary

- Variations in the definition of a graph
- Degrees
- Subgraphs
- Graph operations
- Graph classes
- Paths and cycles

2 Exercises for the 3rd seminar (october 13-17 week)

Variations in the definition of a graph

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms
* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph
Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru -

Multigraph: $G = (V, E)$, where V is a non-empty set (of vertices), and E is the **multiset** (of edges) on V , i. e., there exists a map $m : \binom{V}{2} \rightarrow \mathbb{N}$.

Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru
Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru

$e \in \binom{V}{2}$, with $m(e) > 0$ is an edge of the multigraph G ; if $m(e) = 1$, then e is a **simple edge**, otherwise is a **multiple edge** of **multiplicity** $m(e)$.

- Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C.

The **support graph** of a multigraph, G , is the graph obtained from G by replacing each multiple edge by a simple one.

Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru -
Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru
- Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms *

Variations in the definition of a graph

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms
* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph
Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru -

Pseudograph (general graph): $G = (V, E)$, where V is a non-empty set (of vertices), and E is the **multiset** (of edges) on $V \cup \binom{V}{2}$, i. e., there exists a map $m : V \cup \binom{V}{2} \rightarrow \mathbb{N}$.

- Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C.
Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms *

$e \in E \cap V$ (i. e., $|e| = 1$) is called a **loop**.

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph
Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru -

The **support graph** of a pseudograph G is the graph obtained from G by replacing each multiple edge by a simple one and by removing the loops.

Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru -
Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru
- Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms *

Variations in the definition of a graph

Digraph (directed graph): $D = (V(D), E(D))$, where $V(D)$ is a non-empty set (of vertices), and $E(D) \subseteq V(D) \times V(D)$ is the set of **arcs** (or **directed edges**).

If $e \in E$ then $e = (u, v)$ (or simply $e = uv$) is an arc directed from u to v , and we say:

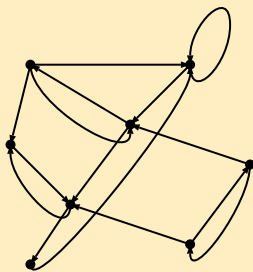
- u is the **initial extremity (tail)** of e , v is the **final extremity (head)** of e ;
- u and v are **adjacent**;
- e is **incident from** u and **into** v ;
- v is a **successor** of u , and u is a **predecessor** of v etc.

Variations in the definition of a graph

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms
* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph
Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru -

Example

A digraph:



Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru -
Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru
- Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms *

Variations in the definition of a graph

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms
* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph
Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru -

- **Symmetric pair of arcs:** (uv, vu) . uv is called the converse of vu .
- The **converse** of a digraph D : replace each arc in D with its converse.
- The **support graph** of a digraph D without loops, denoted by $M(D)$ is obtained by replacing each arc with the corresponding set of two vertices (an edge). $M(D)$ is a multigraph.
- When $M(D)$ is a (simple) graph, then D is called an **oriented graph**.
- **Complete symmetric digraph:** every two (distinct) vertices are joined by a symmetric pair of arcs.
- **Tournament:** an oriented complete graph (every two (distinct) vertices are joined by exactly one arc).

Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru -
Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru
- Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms *

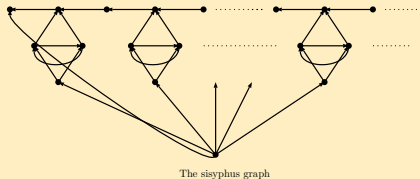
Variations in the definition of a graph

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms
* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph

Infinite (di)graphs: the set of vertices and/or the set of edges (arcs) is countable infinite.

An infinite graph is **locally finite** if $N(v)$ is a finite set, for any vertex v .

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph
Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru -



Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru -
- Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms *

Variations in the definition of a graph

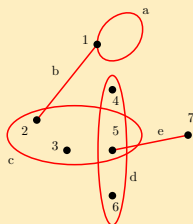
C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms
* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph

Hypergraphs (Finite Set Systems)

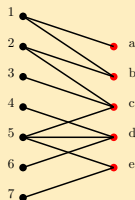
- Edges, now called **hyperedges**, are not restricted to be 2-subsets of the vertex set. A hyperedge is a non-empty subset of the vertex set.
- **k -uniform hypergraph**: every edge has cardinality k .

Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru

Every hypergraph can be represented as a bipartite graph:



Hypergraph H



Bipartite graph associated with H

Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms

Let $G = (V, E)$ be a graph and $v \in V$.

- **Degree** of vertex v : $d_G(v)$ = number of edges incident to v .
- v is an **isolated vertex** if $d_G(v) = 0$ and **pendant** (or **leaf**) if $d_G(v) = 1$.

$$\sum_{v \in V} d_G(v) = 2|E|.$$

- **Maximum degree** $\Delta(G)$ and **minimum degree** $\delta(G)$:

$$\Delta(G) = \max_{v \in V} d_G(v), \quad \delta(G) = \min_{v \in V} d_G(v).$$

- If $\Delta(G) = \delta(G) = k$, then G is **k -regular**.
- **Null graph**: a 0-regular graph.

Subgraphs

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms
* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph
Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru -

Let $G = (V(G), E(G))$ be a graph.

- **Subgraph** of G : a graph $H = (V(H), E(H))$ such that $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$.
- **Spanning subgraph** of G : a subgraph H of G such that $V(H) = V(G)$.
- **Subgraph spanned by $B \subseteq E(G)$ in G** : a subgraph $H = (V(H), E(H))$ such that $E(H) = B$ and $V(H) = V(G)$; is denoted by $\langle B \rangle_G$.
- **Induced subgraph**: a subgraph H of G such that $E(H) = \binom{V(H)}{2} \cap E(G)$. If $A \subseteq V(G)$; the **subgraph induced by A in G** is denoted by $[A]_G$ or $G[A]$.

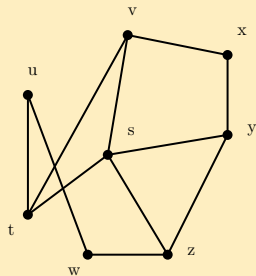
Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru -
- Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms *

Subgraphs

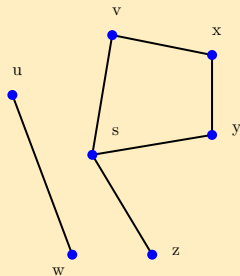
C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms
* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph
Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru -

Example

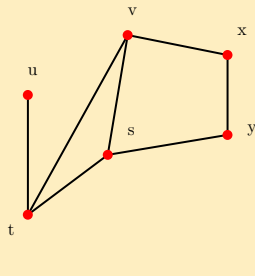
A graph G , a subgraph H' of G , and an induced subgraph of G : $H'' = G[\{u, v, x, y, s, t\}]$.



G



H'



H''

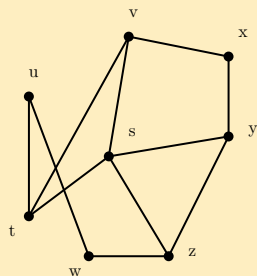
Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms *

Subgraphs

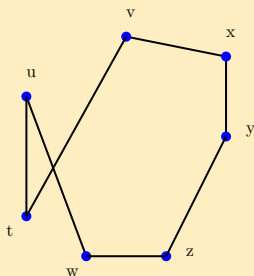
C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms
* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph
Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru -

Example

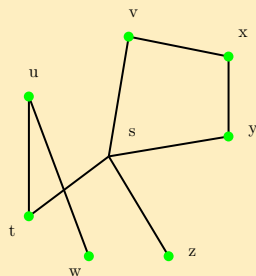
A graph G , $G - s$, and $G - \{vt, wz, zy\}$.



G



$G - s$



$G - \{vt, wz, zy\}$

Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru -
Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru
- Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms *

Graph operations

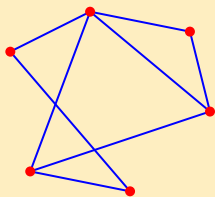
C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms
* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph
Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru -

Unary: $G = (V(G), E(G))$

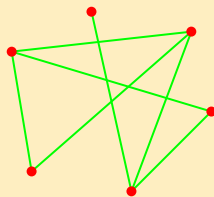
- The **complement** of G : the graph \overline{G} , with $V(\overline{G}) = V(G)$ and

$$E(\overline{G}) = \binom{V(G)}{2} \setminus E(G).$$

- Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C.



G



\overline{G}

Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru
- Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms *

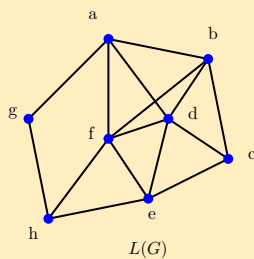
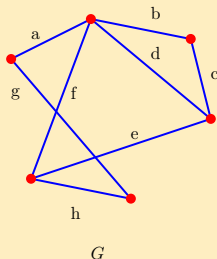
Graph operations

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms
* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph
Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru -

Unary: $G = (V(G), E(G))$

- The **line graph** of G : the graph $L(G)$, with $V(L(G)) = E(G)$ and $E(L(G)) = \{ef : e, f \in E(G), e \text{ and } f \text{ are adjacent in } G\}$.

Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru -
Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru -



Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru -
Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru -
Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms *

Graph operations

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms
* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph

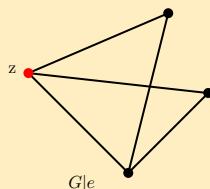
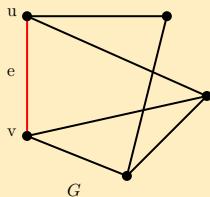
Unary: $G = (V(G), E(G))$

- The graph obtained from G by contracting the edge $e = uv \in E(G)$: the graph $G|e$ with

$$V(G|e) = V(G) \setminus \{u, v\} \cup \{z\},$$

$$E(G|e) = E([V(G) \setminus \{u, v\}]_G) \cup \{yz : yu \text{ or } yv \in E(G)\}.$$

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms
* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph



- Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms *

Graph operations

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms

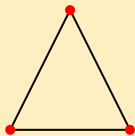
Binary: G, G' with $V(G) \cap V(G') = \emptyset$

- The **cartesian product** of graphs G and G' : the graph $G \times G'$ with

$$V(G \times G') = V(G) \times V(G').$$

$$E(G \times G') = \{(u, u')(v, v') : u, v \in V(G), u', v' \in V(G'), \\ u = v \text{ and } u'v' \in E(G') \text{ or } u' = v' \text{ and } uv \in E(G)\}.$$

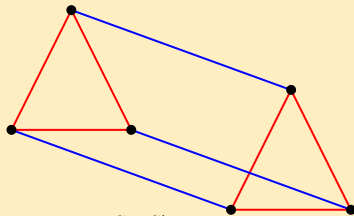
C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms
* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph



G



G'



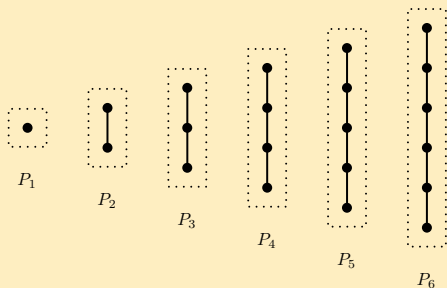
$G \times G'$

Graph classes - Paths P_n

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms
* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph
Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru -

The **path of order n** , P_n : $V(P_n) = \{1, 2, \dots, n\}$ and $E(P_n) = \{12, 23, \dots, n-1n\}$.

* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph



Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru -
Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru
- Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms *

Graph classes - Cliques

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms

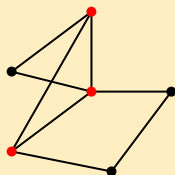
A k -subset of vertices of a graph G that induces a complete graph is called a k -clique.

$$\text{clique number of } G : \omega(G) = \max_{Q \text{ clique in } G} |Q|.$$

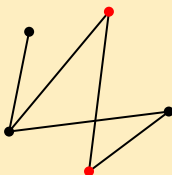
Remark. $\omega(G) = \alpha(\overline{G})$.

Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms *

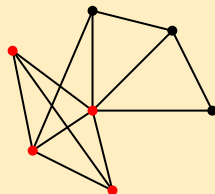
Example



$$\omega(G) = 3$$



$$\omega(G) = 2$$



$$\omega(G) = 4$$

Graph classes - Bipartite graphs

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms

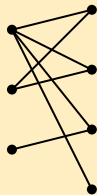
Bipartite graph: a graph G with the property that $V(G)$ can be partitioned in two stable sets.

If $V(G) = S \cup T$, $S \cap T = \emptyset$, $S, T \neq \emptyset$, S, T stable sets in G , then G is denoted $G = (S, T; E(G))$.

Complete bipartite graph: $G = (S, T; E(G))$, with $uv \in E(G)$, $\forall u \in S$ and $\forall v \in T$; denoted by $K_{s,t}$, where $s = |S|$, $t = |T|$.

Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms *

Example



A bipartite graph



$K_{1,1}$



$K_{1,2}$



$K_{2,3}$



$K_{3,3}$

Graph classes - Planar graphs

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms
* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph
Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru -

Planar graph: a graph that can be represented in a plane such that to each vertex corresponds a point of that plane and to each edge corresponds a simple curve joining the points corresponding to its extremities and **these curves intersects only at their endpoints.**

A graph which is not planar is a **non-planar graph.**

Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms *

Planar graphs: **Decision problem**

PLAN Instance: G graph.

Question: Is G planar?

belongs to P (Hopcroft, Tarjan, 1972, $\mathcal{O}(n + m)$).

Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru
- Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms *

Paths and cycles - Closed walks, closed trails, cycles

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms
* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph
Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru -

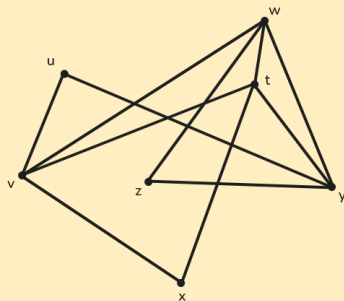
Let $G = (V, E)$ be a graph.

- **Closed walk**: a walk from u to u .
- **Closed trail**: a trail from u to u .
- **Cycle or closed path**: a walk with vertices that are distinct except the extremities which are equal.
- A **cycle** is **even** or **odd** depending on the parity of its length.
- The length of the shortest cycle (if any) is the **girth**, $g(G)$, of G .
- The maximum length of a cycle is the **circumference**, $c(G)$, of G .
- a **chord** in a path/cycle is an edge between two non-consecutive vertices from the path/cycle.

Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru
- Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms *

Example

Closed walks, closed trails, cycles.



- A closed walk: $u, uy, y, yt, t, tv, v, vw, w, wv, v, vu, u$.
- A closed trail: $v, vw, w, wz, z, zy, y, yw, w, wt, t, tx, x, xv, v$.
- A cycle: $u, uv, v, vx, x, xt, t, tw, w, wy, y, yu, u$ (in this cycle vt is a chord).

Paths and cycles - Distance, Diameter

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms

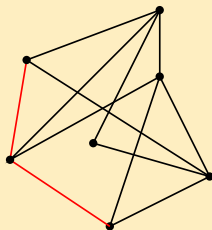
Let $G = (V, E)$ be a graph.

- The **distance in G from u to v** , $d_G(u, v)$, is the length of the shortest path in G from u to v (if any).
- The **diameter of the graph G** , $d(G)$, is:

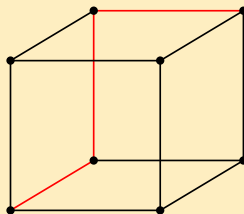
$$d(G) = \max_{u, v \in V} d_G(u, v).$$

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms



$d(G) = 2$



$d(G) = 3$

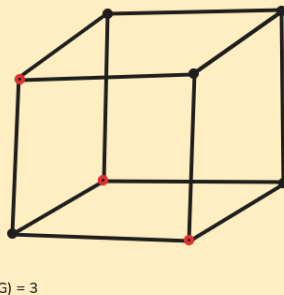
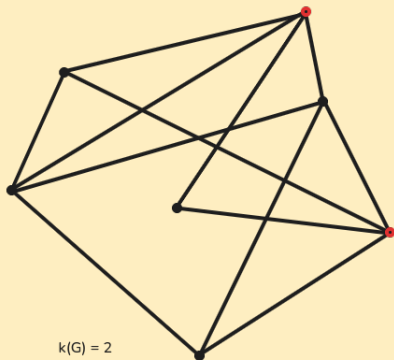
Let $G = (V, E)$ be a graph.

- **Connected graph**: there is a path between any pair of vertices. Otherwise the graph is **disconnected**.
- **Connected component** of a graph G : a maximal connected subgraph, H , of G (i. e., there is no connected subgraph H' of G , $H' \neq H$, H being a subgraph of H').
- Every graph can be expressed as a disjoint union of its connected components.
- The following binary relation is an equivalence relation: $\rho \subseteq V \times V$, given by $u\rho v$ (i. e., $(u, v) \in \rho$) if there is a path in G between u and v .
- The connected components of G are the subgraphs induced by the equivalence classes of ρ .

Paths and Cycles - Connectivity

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms
* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph
Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru -

Example



Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru -
Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru
- Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms *

Let $G = (V, E)$ be a connected graph.

- **Cut edge (bridge)**: an edge $e \in E$ such that $G - e$ is not connected.
- **Edge-cutset**: A subset of edges $S \subseteq E$ such that $G - S$ is not connected.
- A **cut generated by a bipartition** (V_1, V_2) of V is

$$E(V_1, V_2) = \{xy \in E : x \in V_1, y \in V_2\}.$$

- For $p \in \mathbb{N}^*$, G is a **p -edge-connected graph** if G has no edge-cutset of cardinality less than p .
- The **edge-connectivity number**, $\lambda(G)$, of the graph G is

$$\lambda(G) = \max \{p \in \mathbb{N}^* : G \text{ is } p\text{-edge-connected}\}.$$

More general: in a graph G that is not necessarily connected $e \in V(G)$ is a **cut edge** if $G - e$ has more connected components than G .

Paths and Cycles - Eulerian and Hamiltonian graphs

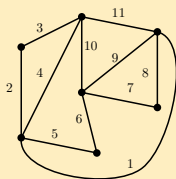
C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms

Let G be a (di)graph.

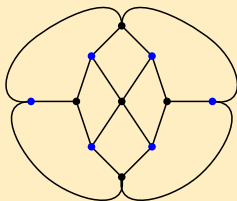
- G is **Eulerian** if there is a closed trail in G passing through each edge of G . (Such a trail is called an **Eulerian trail**.)
- G is **Hamiltonian** if there is a cycle in G passing through each vertex of G . (Such a cycle is called a **Hamiltonian cycle**.)

Polynomial time recognition of Eulerian (di)graphs (**Euler, 1736**).

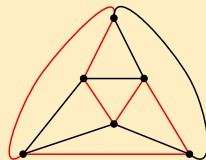
C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms



an Eulerian graph



a non Hamiltonian graph
(bipartite of odd order)



a Hamiltonian graph

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms
* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph
Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru -
Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru

Hamiltonian problems

HAM Instance: G a graph.
Question: Is G Hamiltonian?

NP-complete (Karp, 1972).

NH Instance: G a graph.
Question: Is G not Hamiltonian?

NH \in NP?

* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph
Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru -
Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru
- Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms *

Exercises for the 3rd seminar

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms
* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph
Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru -

Exercise 5. Let G be a graph with $n \geq 2$ vertices. Prove that:

- (a) If G is connected, then it contains at least one vertex that is not a cut vertex.
- (b) If $n \geq 3$ and G is connected, then it contains two vertices that are not cut vertices.
- (c) True or false: if G is connected and $x \in V(G)$ is not a cut vertex, then x is a leaf in a certain spanning tree of G ?

Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru

Exercise 6. Let G be a connected graph that doesn't contain two pendant nodes (leaves) having a common neighbor. Prove that there exist two adjacent vertices whose removal doesn't disconnect G .

Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru
- Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms *

Exercises for the 3rd seminar

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms
* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph

Exercise 7. Let G be a graph and H its line-graph ($H = L(G)$). Prove that H is $K_{1,3}$ -free.

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms
C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms

Exercise 8. Let G be a graph. Prove that:

- If G has exactly two odd degree vertices, then these two vertices are linked with a path in G .
- If G is connected with all vertices of even degree, then G has an edge which is not a bridge (cut-edge).
- If G is connected with all vertices of even degree, then G contains no bridge (cut-edge).
- If G is connected with all vertices of even degree and $|G| > 1$, then G is Eulerian.

- Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms *

Exercises for the 3rd seminar

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms
* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph
Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru -

Exercise 9. Let G be a graph. Prove that

- (a) The number of odd degree vertices is even.
- (b) If G is connected and has k odd degree vertices, then G is the union of $\lfloor k/2 \rfloor$ edge-disjoint trails.

- Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C.
Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms *

Exercise 10. Let G be a graph such that $N_G(u) \cup N_G(v) = V(G)$, $\forall u, v \in V(G)$, $u \neq v$. Prove that G is a complete graph.

Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru
Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C.

Exercise 11. Let G be a graph having the property that $d_G(u) + d_G(v) \geq |G| - 1$, $\forall u, v \in V(G)$, $u \neq v$. Prove that the diameter of $d(G) \leq 2$.

Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru -
Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru
- Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms *

C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms
* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph
Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru -
Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru -

Exercise 14. Let $G = (V, E)$ be a digraph. A strongly connected component of G is a maximal (related to inclusion) sub-digraph H , of G which is strongly connected (i. e., H is strongly connected and there is no strongly connected sub-digraph H' of G , $H \subsetneq H'$).

Define the following binary relation on V , $\rho \subseteq V \times V$, given by: $u\rho v$ (i. e., $(u, v) \in \rho$), if there is a directed path in G from u to v and a directed path from v to u .

- (a) Prove that ρ is an equivalence relation.
- (b) Show that the strongly connected components of G are the sub-digraphs induced by the equivalence classes of ρ .

Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru -
Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru
- Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms *