

1

Connectivity

- Menger's theorem
- p -connectivity
- König's theorem
- Hall's theorem
- Dirac's theorem

2

Trees

- Basics
- Counting spanning trees: Kirchhoff-Trent theorem

3

Exercises for the 6th seminar (november 3 - 7 week)

4

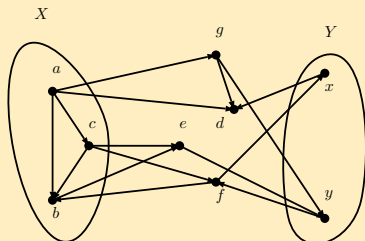
Annex - Generating all spanning trees

Definition 1

Let $G = (V, E)$ be a (di)graph and $X, Y \subseteq V$. A **XY -path** is any path P in G from a vertex $x \in X$ to a vertex $y \in Y$ such that $V(P) \cap X = \{x\}$ and $V(P) \cap Y = \{y\}$.

We denote by $\mathcal{P}(X, Y, G)$ the set of all **XY -paths** in G . Note that if $x \in X \cap Y$ then the path of length 0, $P = \{x\}$, is an XY -path.

Example



XY -paths: (b, e, y) , (c, f, x) , and (a, g, y) ; an YX -path: (y, f, b)

Definition 2

Let $G = (V, E)$ be a (di)graph and $X, Y \subseteq V$. A XY -separating set in G is any subset $Z \subseteq V$ such that

$$V(P) \cap Z \neq \emptyset, \text{ for each } P \in \mathcal{P}(X, Y; G).$$

We denote by

$$\mathcal{S}(X, Y; G) = \{Z : Z \text{ is } XY \text{ - separating set in } G\} \text{ and}$$

$$k(X, Y; G) = \min \{|Z| \mid Z \in \mathcal{S}(X, Y; G)\}$$

From the definition we easily get:

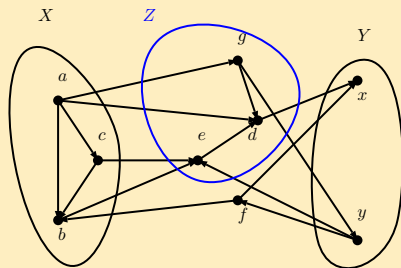
- If $Z \in \mathcal{S}(X, Y; G)$, then $\mathcal{P}(X, Y; G \setminus Z) = \emptyset$.
- $X, Y \in \mathcal{S}(X, Y; G)$.

Connectivity - Menger's theorem and applications

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

Example



A XY -separating set: $Z = \{g, e, d\}$

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

- If $Z \in \mathcal{S}(X, Y; G)$, then $A \in \mathcal{S}(X, Y; G)$, $\forall A$ such that $Z \subseteq A \subseteq V$.
- If $Z \in \mathcal{S}(X, Y; G)$ and $T \in \mathcal{S}(Z, Y; G)$, then $T \in \mathcal{S}(X, Y; G)$.

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

Theorem 1

Menger's theorem. Let $G = (V, E)$ be a (di)graph and $X, Y \subseteq V$. Then $p(X, Y; G) = k(X, Y; G)$.

(i. e., the maximum number of disjoint XY -paths = the minimum cardinality of an XY -separating set.)

Proof:

$k(X, Y; G) \geq p(X, Y; G) = r$. Let P_1, \dots, P_r disjoint XY -paths in G ; $Z \cap V(P_i) \neq \emptyset, \forall Z \in \mathcal{S}(X, Y; G)$. Since the paths P_i are vertex disjoint:

$$|Z| \geq \left| Z \cap \left(\bigcup_{i=1}^r V(P_i) \right) \right| = \sum_{i=1}^r |Z \cap V(P_i)| \geq \sum_{i=1}^r 1 = r.$$

Hence, $|Z| \geq r, \forall Z \in \mathcal{S}(X, Y; G)$; it follows that $k(X, Y; G) \geq r$.

Connectivity - Menger's theorem and applications

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

$k(X, Y; G) \leq p(X, Y; G)$. Omitted. (We will later show that $\forall G = (V, E)$ and $\forall X, Y \subseteq V$, $\exists k(X, Y; G)$ disjoint XY -paths in G using flows in a certain network.) \square

Menger (1927) equivalently enounced the above theorem, using internally-disjoint paths: $P_1, P_2 \in \mathcal{P}_{st}$ such that $V(P_1) \cap V(P_2) = \{s, t\}$:

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

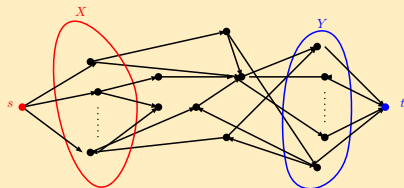
Theorem 2

Let $G = (V, E)$ be a (di)graph and $s, t \in V$, such that $s \neq t$, $st \notin E$. There are k internally-disjoint paths from s to t in G if and only if there exists at least one path from s to t in the (di)graph obtained from G by removing any set of $< k$ vertices different from s and t .

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

Proof of equivalence:

Theorem 1 \Rightarrow Theorem 2: take $X = N_G^+(s)$ ($N_G(s)$) and $Y = N_G^-(t)$ ($N_G(t)$). $p(X, Y; G)$ is the maximum number of internally-disjoint paths from s to t in G . $k(X, Y; G)$ is the minimum cardinality of a set of vertices whose removal disconnects s and t .



Theorem 2 \Rightarrow Theorem 1: add two new vertices s and t to the (di)graph G , and all (directed) edges from s to any vertex in X and from any vertex in Y to t . The maximum number of internally-disjoint paths from s to t in G is $p(X, Y; G)$; the minimum cardinality of a set of vertices whose removal disconnects s and t is $k(X, Y; G)$. \square

Applications: p -connectivity

- A graph G is p -connected ($p \in \mathbb{N}^*$) if either $G = K_p$, or $|G| > p$ and $G \setminus A$ is connected for any $A \subseteq V(G)$ with $|A| < p$.

- By Theorem 2, an equivalent characterization of the p -connectivity is:

A graph G is p -connected ($p \in \mathbb{N}^*$) if either $G = K_p$, or $\forall st \in E(\overline{G})$ there are p internally-disjoint paths from s to t in G .

- The **vertex connectivity number** of the graph G , $k(G)$, is the maximum p , for which G is p -connected.
- It follows that, in order to compute $k(G)$, we must find

$$\min_{st \notin E(G)} p(\{s\}, \{t\}; G)$$

which can be determined in polynomial time using network flows.

Applications: König's theorem

- A **vertex-cover** in a graph G is a set $Z \subseteq V(G)$ of vertices such that $G - Z$ is a null graph (each edge of G has at least one extremity in Z).
- A special case of the Theorem 1, is obtained when G is a bipartite graph and $X = S$, $Y = T$ are the two bipartite classes of G :

Theorem 3

(König, 1931) *Let $G = (S, T; E)$ be a bipartite graph. Then, the maximum cardinality of a matching in G is equal to the minimum cardinality of a vertex-cover.*

Proof: The maximum cardinality of a matching in G is $p(S, T; G) = k(S, T; G)$, by Theorem 1. Since a set of vertices is an ST -separating set if and only if is a vertex-cover, the Theorem 3 is proved. \square

Applications: Hall's theorem

- Let I and S be non-empty finite sets. A family of subsets of S (indexed by I) is a map $\mathcal{A} : I \rightarrow 2^S$, where $\mathcal{A}(i) = A_i$. We will denote $\mathcal{A} = (A_i)_{i \in I}$ and (using the functional notation) $\mathcal{A}(J) = \bigcup_{j \in J} A_j$ (for $J \subseteq I$).
- A representative function for the family $\mathcal{A} = (A_i)_{i \in I}$ is any function $r_{\mathcal{A}} : I \rightarrow S$ with the property $r_{\mathcal{A}}(i) \in A_i, \forall i \in I$; then, $(r_{\mathcal{A}}(i))_{i \in I}$ is called a system of representatives for \mathcal{A} .
- If the representative function, $r_{\mathcal{A}}$, is injective, then $r_{\mathcal{A}}(I)$ is a subset of S and is called a system of distinct representatives for \mathcal{A} , or a transversal of \mathcal{A} .
- The central problem in the Transversal Theory is to characterize the families that admit transversal (with some properties). Hall's Theorem (1935) is the first result of this type.

Theorem 4

(Hall, 1935) The family $\mathcal{A} = (A_i)_{i \in I}$ of subsets of S has a transversal if and only if

$$(H) \quad |\mathcal{A}(J)| \geq |J|, \forall J \subseteq I.$$

Proof: " \Rightarrow " If $r_{\mathcal{A}}$ is an injective representative function for \mathcal{A} , then $r_{\mathcal{A}}(J) \subseteq \mathcal{A}(J)$, $\forall J \subseteq I$. Hence, $r_{\mathcal{A}}$ being injective, $|\mathcal{A}(J)| \geq |r_{\mathcal{A}}(J)| = |J|$.

" \Leftarrow " Let $G_{\mathcal{A}} = (I, S; E)$ be the bipartite graph associated to \mathcal{A} (if $I \cap S \neq \emptyset$, we can consider disjoint isomorphic copies of these sets), where $E = \{is | i \in I, s \in A_i\}$. Note that $N_{G_{\mathcal{A}}}(i) = A_i$. Moreover, \mathcal{A} has a transversal if and only if $G_{\mathcal{A}}$ has a matching of cardinality $|I|$.

Connectivity - Menger's theorem and applications

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 -

Proof of Hall's Theorem (cont'd): We show that if the condition (H) holds, then any vertex-cover of $G_{\mathcal{A}}$ has at least $|I|$ vertices, and - by Konig's Theorem - $G_{\mathcal{A}}$ has a matching of cardinality $|I|$ (I itself is a vertex cover of $G_{\mathcal{A}}$).

Let $X = I' \cup S' \subseteq I \cup S$ be a vertex cover in $G_{\mathcal{A}}$: it follows that $N_{G_{\mathcal{A}}}(I \setminus I') \subseteq S'$, that is, $\mathcal{A}(I \setminus I') \subseteq S'$. Then,

$$|X| = |I'| + |S'| \geq |I'| + |\mathcal{A}(I \setminus I')|.$$

Since (H) holds, it follows that

$$|X| \geq |I'| + |\mathcal{A}(I \setminus I')| \geq |I'| + |I \setminus I'| = |I|. \quad \square$$

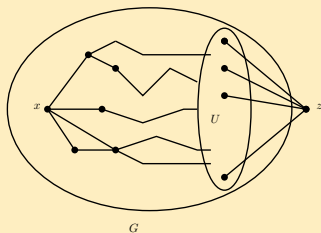
Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru
- Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms *

Applications: Dirac's theorem (p -connected graphs structure)

Lemma 1

Let $G = (V, E)$ be a p -connected graph of order $|G| \geq p + 1$, $U \subseteq V$, $|U| = p$ and $x \in V \setminus U$. Then there are p xU -paths such that any pair of them has x as the only common vertex.

Proof: Let $G' = (V \cup \{z\}, E')$, where $E' = E \cup \{zu : u \in U\}$.



Applications: Dirac's theorem (p -connected graphs structure)

Then, G' is a p -connected graph. Indeed, let $A \subseteq V(G')$ with $|A| \leq p - 1$. If $A \subseteq V(G)$, then $G' - A$ is connected (by the p -connectivity of G , $G - A$ is connected; since $|A| < p$, $\exists u \in U \setminus A$ and, hence, there exists the edge $zu \in E(G' - A)$. If $z \in A$, then $G' - A = G - A$ which is connected.

The lemma now follows by applying Theorem 2 to the graph G' and the pair x, z . \square

Proposition 1

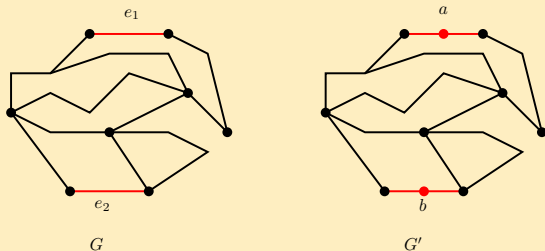
Let $G = (V, E)$ be a p -connected graph, $p \geq 2$. Then, for every two edges e_1 and e_2 of G and for every, x_1, \dots, x_{p-2} , $p - 2$ vertices of G , there exists a cycle in G containing all of them.

Connectivity - Menger's theorem and applications

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

Proof: Induction on p .

For $p = 2$, we must prove that in a 2-connected graph, G , every two edges e_1 and e_2 belong to a cycle. Let G' be the graph obtained from G by inserting one vertex a on e_1 and one vertex b on e_2 :



G' is 2-connected (any deleted subgraph $G' - v$ is connected). Hence, there are two internally-disjoint paths from a to b , giving the cycle in G containing e_1 and e_2 (after removing a and b).

In the inductive step, let $p \geq 3$, suppose that the proposition holds for every p' -connected graph with $2 \leq p' < p$, and consider a p -connected graph G , two of its edges, e_1 and e_2 and a set of $p - 2$ vertices $\{x_1, x_2, \dots, x_{p-2}\}$.

We can suppose that no extremity v of e_1, e_2 belongs to the set $\{x_1, x_2, \dots, x_{p-2}\}$ (otherwise, we can apply the induction hypothesis to infer that in G - which is also $(p - 1)$ -connected - there exists a cycle C containing e_1, e_2 and the set of vertices $\{x_1, x_2, \dots, x_{p-2}\} \setminus \{v\}$; but, clearly v is a vertex of C since e_1 and e_2 are edges of C).

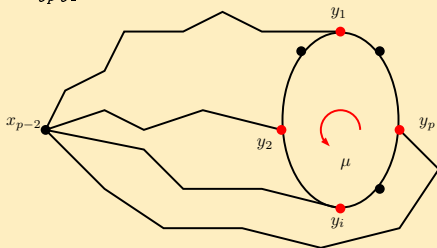
The graph $G - x_{p-2}$ is $(p - 1)$ -connected. By the induction hypothesis, there exists a cycle μ containing $x_1, x_2, \dots, x_{p-3}, e_1$ and e_2 . Let Y be the set of vertices of μ . Clearly, $|Y| \geq p$ (to the set of $p - 3$ vertices x_1, x_2, \dots, x_{p-3} , we add at least three extremities of the distinct edges e_1 and e_2). By Lemma 1, there are p x_{p-2} Y -paths such that any pair of them has x_{p-2} as the only common vertex.

Connectivity - Menger's theorem and applications

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

Let $P_{x_{p-2}y_1}, P_{x_{p-2}y_2}, \dots, P_{x_{p-2}y_p}$ be these paths, where the ordering y_1, \dots, y_p is obtained by performing a traversal of μ .

The vertices y_1, \dots, y_p split the cycle μ in the paths $P_{y_1y_2}, P_{y_2y_3}, \dots, P_{y_{p-1}y_p}, P_{y_p y_1}$:



At least one of the above paths doesn't contain any element from the set $x_1, x_2, \dots, x_{p-3}, e_1$ and e_2 (by the pigeon hole principle).

Let $P_{y_1y_2}$ be this path (otherwise, we appropriately change the ordering of y_i).

Connectivity - Menger's theorem and applications

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 -

Then,

$$P_{x_{p-2}y_2}, P_{y_2y_3}, \dots, P_{y_p y_1}, P_{y_1 x_{p-2}}$$

is the cycle in G containing $x_1, x_2, \dots, x_{p-2}, e_1$ and e_2 . \square

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

Theorem 5

(Dirac, 1953) *Through any $p \geq 2$ vertices of a p -connected graph passes a cycle.*

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

Proof: Let $G = (V, E)$ be a p -connected graph, $p \geq 2$. Let x_1, x_2, \dots, x_p be p vertices of G . Since G is connected, there exist the edges $e_1 = x_1 x_{p-1}$ and $e_2 = x_p x_1$. Then, the theorem follows from Proposition 1. \square

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

Connectivity - Menger's theorem and applications

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 -

A nice application of Theorem 5 and Proposition 1 is the next Hamiltonian sufficient condition given by Erdős and Chvatal.

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

Theorem 6

(Erdős-Chvatal, 1972) Let $G = (V, E)$ be a p -connected graph. If $\alpha(G) \leq p$ then G is a Hamiltonian graph.

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

Proof: Suppose, by contradiction, that G is not Hamiltonian. Let C be the set of vertices of a longest cycle in G .

By Dirac's Theorem, $|C| \geq p$ and, by our assumption, there exists a vertex $v \in V(G) \setminus V(C) \neq \emptyset$.

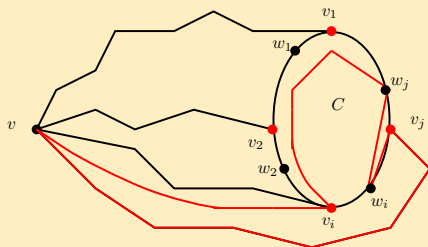
Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru
- Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms *

Connectivity - Menger's theorem and applications

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

Since $|C| \geq p$ we can repeat the argument in the proof of the Proposition from above in order to prove that there are $P_{vv_1}, P_{vv_2}, \dots, P_{vv_p}$, p vC -paths pairwise meeting just in v and with extremities v_i numbered in the order they are met during a traversal of the cycle C .

Let us denote by w_i the successor vertex of v_i on the cycle.



Note that $vw_i \notin E$ (otherwise, the cycle $vw_i, w_i, C \setminus \{w_i v_i\}, P_{v_i v}$ is longer than C , contradiction).

A **tree** is a connected graph without cycles.

Theorem 7

Let $G = (V, E)$ be a graph. Then the following statements are equivalent:

- (i) G is a tree (is connected and has no cycle).
- (ii) G is connected and it is minimal with this property.
- (iii) G has no cycle and is maximal with this property.

Proof: Is omitted. \square

The minimality and maximality in the above statements are given with respect to the inclusion relation on the family subsets of edges. More precisely, the above (ii) and (iii) statements means:

- (ii) G is **connected** and $\forall e \in E, G - e$ is not connected.
- (iii) G **has no cycle** and $\forall e \notin E, G + e$ has a cycle.

Definition

Let $G = (V, E)$ be a (multi)graph. A **spanning tree** of G is a spanning subgraph of $G, T = (V, E')$ ($E' \subseteq E$), which is a tree. We denote by \mathcal{T}_G the set of all spanning trees of G .

Remarks

1. $\mathcal{T}_G \neq \emptyset$ if and only if G is connected. Indeed, if $\mathcal{T}_G \neq \emptyset$, then there is a spanning tree $T = (V, E')$ of G . T is connected, hence between any two vertices of G there is a path P in T . Since $E' \subseteq E$, P is a path in G , therefore G is connected.

Conversely, if G is connected, then let us consider the following algorithm:

```

 $T \leftarrow G;$ 
while ( $\exists e \in E(T)$  such that  $T - e$  is connected) do
     $T \leftarrow T - e;$ 

```

By construction, T is a spanning subgraph of G , and the statement (ii) in the Theorem 7 is fulfilled, therefore it is a tree.

2. Another constructive proof is based on the remark that **there exists a crossing-edge between the two classes of any bipartition of V : $\exists e = v_1 v_2 \in E$ with $v_i \in V_i, i = \overline{1, 2}$.**

If $|V| = n > 0$, then the following algorithm constructs a spanning tree of the connected graph $G = (V, E)$:

```

 $k \leftarrow 1; T_1 \leftarrow (\{v\}, \emptyset); // v \in V$ 
while ( $k < n$ ) do
  let  $xy \in E$  with  $x \in V(T_k), y \in V \setminus V(T_k)$ ;
  // such an edge exists by the connectedness of  $G$ 
   $V(T_{k+1}) \leftarrow V(T_k) \cup \{y\}$ ;
   $E(T_{k+1}) \leftarrow E(T_k) \cup \{xy\}$ ;
   $k ++$ ;

```

Clearly, T_k is a tree $\forall k = \overline{1, n}$ (inductively, if T_k is a tree then, by construction, T_{k+1} is connected and has no cycle). Moreover, we have $|V(T_k)| = k$ and $|E(T_k)| = k - 1, \forall k = \overline{1, n}$.

3. If this construction is applied to a tree G with n vertices, we obtain that G has $n - 1$ edges. This property can be used to extend the Theorem 7 with other characterizations of a tree:

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 -

Theorem 8

The following statements are equivalent for a graph $G = (V, E)$ with n vertices:

- (i) G is a tree.
- (ii) G is **connected** and has $n - 1$ edges.
- (iii) G has **no cycles** and has $n - 1$ edges.
- (iv) $G = K_n$ for $n \in \{1, 2\}$ and $G \neq K_n$ for $n \geq 3$ and $G + e$ has exactly one cycle, for every edge $e \notin E$.

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

Proof: Omitted. \square

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 *

Trees - Counting spanning trees

Let $G = (V, E)$ be a multi-graph with $V = \{1, 2, \dots, n\}$, and adjacency matrix $A = (a_{ij})_{n \times n}$ (a_{ij} = multiplicity of edge ij if $ij \in E$, 0 otherwise). Let

$$D = \text{diag}(d_G(1), d_G(2), \dots, d_G(n)) = \begin{pmatrix} d_G(1) & 0 & \dots & 0 \\ 0 & d_G(2) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & d_G(n) \end{pmatrix}.$$

The **Laplacian matrix** of G is defined as:

$$L[G] = D - A.$$

Note that the sum of the entries in $L[G]$ on every row and every column is 0. We denote by $L[G]_{ij}$ the minor of the matrix $L[G]$ obtained by removing the i th row and j th column.

Theorem 9

Matrix-tree Theorem (Kirchhoff-Trent). Let G be a (multi)graph with vertex set $\{1, \dots, n\}$ and Laplacian matrix $L[G]$. Then, the number of spanning trees of G is: $|\mathcal{T}_G| = \det(L[G]_{ii}), \forall 1 \leq i \leq n$.

Proof: Omitted. \square

Corollary

(Cayley's formula). $|\mathcal{T}_{K_n}| = n^{n-2}$.

Proof:

$$L[K_n] = \begin{pmatrix} n-1 & -1 & \dots & -1 \\ -1 & n-1 & \dots & -1 \\ \vdots & \vdots & \ddots & \vdots \\ -1 & -1 & \dots & n-1 \end{pmatrix}.$$

Trees - Counting spanning trees

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

Hence:

$$\det(L[K_n]_{11}) = \begin{vmatrix} n-1 & -1 & \dots & -1 \\ -1 & n-1 & \dots & -1 \\ \vdots & \vdots & \ddots & \vdots \\ -1 & -1 & \dots & n-1 \end{vmatrix}$$

If we add all the lines to the first one we get

$$\begin{vmatrix} 1 & 1 & \dots & 1 \\ -1 & n-1 & \dots & -1 \\ \vdots & \vdots & \ddots & \vdots \\ -1 & -1 & \dots & n-1 \end{vmatrix} = n^{n-2}. \text{ (Why?)}$$

□

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

Exercise 1. Let $G = (V, E)$ be a connected graph and $v \in V$ such that $N_G(v) \neq V \setminus \{v\}$. For $X \subseteq V$ we denote $N_G(X) = \left(\bigcup_{v \in X} N_G(v) \right) \setminus X$.

Clearly, the set $A = \{v\}$ satisfies the following conditions

- (i) $v \in A$ and $[A]_G$ is connected.
 - (ii) $N = N_G(A) \neq \emptyset$.
 - (iii) $R = V \setminus (A \cup N) \neq \emptyset$.
- (a) Show that, if $A \subseteq V$ is any maximal (w.r.t " \subseteq ") set of vertices satisfying (i) - (iii), then $\forall x \in R$ and $\forall y \in N$ we have $xy \in E$.
- (b) Prove that if A is as in (a) and G is $\{C_k\}_{k \geq 4}$ -free, then N is a clique in G .
- (c) Deduce that K_n ($n \in \mathbb{N}^*$) are the only regular, chordal connected graphs.

Exercise 2. A graph of order at least three is called **confidentially connected** if, for every three distinct vertices a , b and c , it exists a path from a to b such that c differs and is not adjacent with the internal nodes (if any) of this path. (An example of confidential connected graph is the complete graph K_n , with $n \geq 3$.)

Show that a connected, incomplete graph, $G = (V, E)$, with at least three vertices is confidential connected if and only if:

- (i) for every $v \in V$, $N_G(v) \neq \emptyset$ and induces a connected subgraph;
- (ii) any edge of G is part of an induced C_4 or is a mid-edge of an induced P_4 .

Exercise 3. Prove that a connected p -regular bipartite graph is 2-connected.

Exercise 4. Let $G = (V, E)$ be a digraph. Prove that:

- G is strongly connected if and only if for every $S \subsetneq V$, $S \neq \emptyset$, there exists an arc leaving S .
- If G is strongly connected and can be disconnected by removing at most p arcs (i. e., $\exists A \subseteq E$, $|A| \leq p$ such that $G - A$ is not strongly connected), then G can be disconnected by reversing at most p arcs (that is $\exists B \subseteq E$, $|B| \leq p$ such that $G' = (V, (E \setminus B) \cup \{uv : vu \in B\})$ is not strongly connected).

Exercise 5. Let G be a 2-edge-connected graph ($G - e$ is connected, $\forall e \in E(G)$). Define the following binary relation $e \asymp f$ if $e = f$ or $G - \{e, f\}$ is not connected.

- Prove that $e \asymp f$ if and only if e and f belong to the same cycles.
- Show that an equivalence class $[e]_{\asymp}$ is included in a cycle.
- By removing the edges of an entire equivalence class $[e]_{\asymp}$ the connected components of the remaining graph are 2-edge-connected.

Exercise 6.

- (a) Let G be a graph with at least 3 vertices. If G is 2-connected, then we can orient its edges in such a way that the resulting oriented graph is strongly connected.
- (b) Is the converse true?

Exercise 7.

- (a) Let G be an incomplete 2-connected graph and $xy \in E(G)$. Prove that $G - xy$ or $G|xy$ is 2-connected.
- (b) Give examples of graphs G and edges $xy \in E(G)$ such that: (b1) $G - xy$ and $G|xy$ are both 2-connected; (b2) $G - xy$ is not 2-connected and $G|xy$ is 2-connected; (b3) $G - xy$ is 2-connected and $G|xy$ is not 2-connected;

Exercise 8. For a given connected graph G we perform the following algorithm:

```

 $Q \leftarrow \{G\};$  //  $Q$  is a queue;
while ( $Q \neq \emptyset$ ) {
   $H \leftarrow \text{pop}(Q);$ 
  let  $A \subseteq V(H)$  a minimal cut-set in  $H$ ;
  let  $G_1, \dots, G_k$  the connected components of  $H - A$ ;
  for ( $j = 1$  to  $k$ )
    push( $Q, [A \cup V(G_j)]_G$ );
}

```

Observe that if G is a complete graph, then in Q we do not push any other graph.

- Show that any graph which is pushed in Q is a connected one.
- Prove that the total number of graphs pushed in the queue Q is at most $|G|^2$.

Exercises for the 6th 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 9. Let $G = (V, E)$ be a connected graph and T_1, T_2 be two spanning trees of G ($T_1, T_2 \in \mathcal{T}_G$).

- (a) Prove that T_1 can be transformed into T_2 by repeatedly applying the following operation: remove an edge and add another edge to the current tree.
- (b) If, in addition, G is 2-connected show that T_1 can be transformed into T_2 by repeatedly applying the following operation: remove an edge uv and add another edge uw to the current tree.

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

Exercise 10. Prove that the set of edges of a complete graph K_n ($n \geq 2$) can be partitioned in $\lceil n/2 \rceil$ subsets each representing the set of edges of a tree (subgraph in K_n).

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

Exercise 13. We consider the street network of a given city. Prove that if we can remove all the cycles in this network by creating at most p blockings (blocking means obstructing one way of a street), then we can remove all the cycles in the city network by reversing one way of at most p streets.

(Reversing one way of a given two ways street means to transform it into a one-way-street; reversing an one-way-street means to transform it into the other one-way-street.)

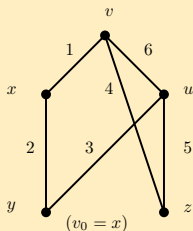
Exercise 14. How many spanning trees has a complete bipartite graph $K_{n,n}$? Same question for $K_{p,q}$.

Exercise 15. Prove that graph with at least three vertices is 2-connected if and only if any two distinct vertices belong to a cycle.

Trees - Generating \mathcal{T}_G

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

- We describe a simple **backtracking method** to generate all spanning trees of a connected graph $G = (V, E)$, where $V = \{1, \dots, n\}$, $|E| = m$.
- The set of edges, E , will be represented as an array $E[1..2, 1..m]$ having entries from V , with the meaning: if $v = E[1, i]$ and $w = E[2, i]$, then vw is the i th edge of G . Furthermore, we assume that the first $d_G(v_0)$ columns in the array E have v_0 in the row 1 ($E[1, i] = v_0, \forall i = \overline{1, d_G(v_0)}$), for $v_0 \in V$.



	1	2	3	4	5	6
1	x	x	y	z	z	u
2	v	y	u	v	u	v

Trees - Generating \mathcal{T}_G

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

- A spanning tree $T \in \mathcal{T}_G$ will be represented as an set of $n - 1$ indexes (in increasing order) of the columns in the array E (designating its edges).
- During the generation, we maintain a vector $T[1..n - 1]$ with entries from $\{1, \dots, m\}$ and a flag $i \in \{1, \dots, n\}$ with the following meaning:

We are searching for all spanning trees of G , with the property that the smallest $i - 1$ edges are: $T[1] < T[2] < \dots < T[i - 1]$.

- In the above example, if $i = 3$, $T[1] = 1$, and $T[2] = 2$, then the spanning trees which must be found are $\{1, 2, 3, 4\}$, $\{1, 2, 3, 5\}$, $\{1, 2, 4, 5\}$, $\{1, 2, 4, 6\}$, and $\{1, 2, 5, 6\}$. But, if $i = 3$, $T[1] = 3$, and $T[2] = 5$ then no spanning tree will be found.

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

Trees - Generating \mathcal{T}_G

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

```
ALL-ST-Gen( $i$ )
```

```
//we generate all spanning trees of  $G$ , with the smallest  $i - 1$  edges:  $T[1], \dots, T[i - 1]$ 
```

```
if ( $i = n$ ) then
```

```
  //  $\{T[1], \dots, T[n - 1]\}$  is a spanning tree
```

```
  process( $T$ ); // print, store etc
```

```
else
```

```
  if ( $i = 1$ ) then
```

```
    for ( $j = 1, \overline{d_G(v_0)}$ ) do
```

```
       $T[i] \leftarrow j$ ; A All-ST-Gen( $i + 1$ ); B
```

```
  else
```

```
    for ( $j = \overline{T[i - 1] + 1, m - (n - 1) + i}$ ) do
```

```
      if ( $\langle\{T[1], \dots, T[i - 1]\} \cup \{j\}\rangle_G$  has no cycle) then
```

```
         $T[i] \leftarrow j$ ; A All-ST-Gen( $i + 1$ ); B
```

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 *

- By the call All-ST-Gen(1) we obtain \mathcal{T}_G .
- To test if the graph $\langle \{T[1], \dots, T[i-1]\} \cup \{j\} \rangle_G$ has no cycle, let us observe that, by construction,

$$\langle \{T[1], \dots, T[i-1]\} \rangle_G$$

has no cycle, hence it is a forest (each connected component is a tree).

- Let $root[1..n]$ be a (global) vector with entries from V and the meaning: $root[v]$ = the root of the connected component containing v (one of its vertices).
- Before the call All-ST-Gen(1), the vector $root$ is initialized to satisfy this property: $root[v] \leftarrow v$ ($\forall v \in V$) (since then, $\{T[1], \dots, T[i-1]\} = \emptyset$).

- During the recursive calls, when we test if the edge j can be added to the set $\{T[1], \dots, T[i-1]\}$ without creating a cycle, let $v = E[1, j]$ and $w = E[2, j]$. Then,
 $\langle \{T[1], \dots, T[i-1]\} \cup \{j\} \rangle_G$ has no cycle if and only if v and w are in different connected components of the forest, i.e., $root[v] \neq root[w]$.
- In order to maintain the vector `root`, in the places marked **A** and **B** in the algorithm, we must make the following changes.
- Instead of **A**:

$S \leftarrow \emptyset; x \leftarrow root[v];$

for $(u \in V)$ do

if $(root[u] = x)$ then

$S \leftarrow S \cup \{u\}; root[u] \leftarrow root[w];$

