

Table of contents

1

Matchings

- Perfect Matchings - Tutte's Theorem
- Maximum Cardinality Matchings - Berge's Theorem
- Maximum Cardinality Matchings - Hopcroft-Karp Algorithm

2

Network flows

- Flow network
- Flow, flow value, maximum flow problem

3

Exercises for the 8th seminar (november 24 - 28 week)

Let G be a graph. Clearly, $|M| \leq |G|/2, \forall M \in \mathcal{M}_G$. A **perfect matching** (or **1-factor**) in G is a matching M with the property $|M| = |G|/2$ (i. e., $S(M) = V(G)$).

A connected component of graph G is even (odd) if the number of its vertices is even (odd). We denote by $q(G)$ the number of odd connected components of G .

Remark

Let G be a graph having a perfect matching. Clearly, each connected component of G is even. Hence, $q(G) = 0$; moreover, if $S \subseteq V(G)$, then for each odd connected component of the graph $G - S$ there exists an edge in the perfect matching of the graph with an extremity in this connected component and the other in S . Since the extremities of different edges are distinct, it follows that $|S| \geq q(G - S)$. For $S = \emptyset$, we obtain $q(G - \emptyset) \leq 0$, hence $q(G) = 0$.

Perfect Matchings - Tutte's Theorem

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

Proof. The necessity of condition (T) has been already observed in the above discussion. We prove by induction on $n = |G|$ that if a graph $G = (V, E)$ satisfies (T), then G has a perfect matching.

For $n = 1, 2$ the theorem obviously holds. In the inductive step, let G be graph with $n \geq 3$ vertices which satisfies (T) and suppose that any graph G' with $|G'| < n$ which satisfies (T) has a perfect matching.

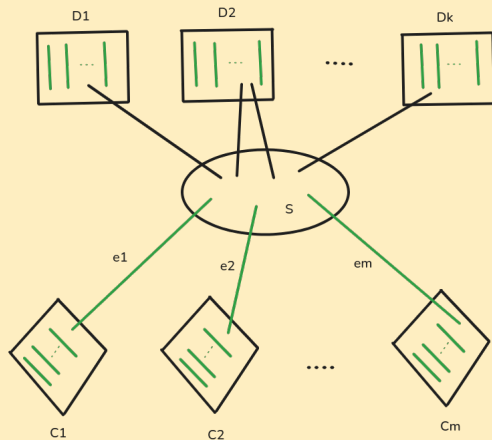
Let $S_0 \subseteq V(G)$ such that $q(G - S_0) = |S_0|$ and maximal (with respect to " \subseteq ") having the property that (T) holds with equality (i. e., for any strictly superset S of S_0 we have $q(G - S) < |S|$).

Note that the family of subsets of $V(G)$ for which (T) is satisfied with equality is non-empty, and therefore S_0 exists (for each vertex v_0 which is not a cut-vertex in its component, we have $q(G - v_0) = 1 = |\{v_0\}|$, since each connected component is even and contains such a vertex v_0).

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

Perfect Matchings - Tutte's Theorem

Let $m = |S_0| > 0$ and C_1, C_2, \dots, C_m be the odd components of $G - S_0$ and D_1, D_2, \dots, D_k be the even components of $G - S_0$ ($k \geq 0$):



We will build a perfect matching in G composed by

Perfect Matchings - Tutte's Theorem

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

- a) a perfect matching in each even connected component D_i ;
- b) a matching with m edges, $\{e_1, \dots, e_m\}$, the edge e_i having an end $s_i \in S_0$ and the other $v_i \in C_i$ ($i = \overline{1, m}$);
- c) a perfect matching in each subgraph $C_i - v_i$ ($i = \overline{1, m}$).

a) For each $1 \leq i \leq k$ the graph $[D_i]_G$ has a perfect matching. Indeed, since $m > 0$, it follows that $|D_i| < n$ and by the induction hypothesis it is sufficiently to show that $G' = [D_i]_G$ satisfies (T).

Let $S' \subseteq D_i$. If $q(G' - S') > |S'|$, then we get the following contradiction:

$$q(G - (S_0 \cup S')) = q(G - S_0) + q(G' - S') = |S_0| + q(G' - S') > |S_0 \cup S'|.$$

Hence, $q(G' - S') \leq |S'|$, $\forall S' \subseteq D_i$, i. e., G' satisfies (T).

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

Perfect Matchings - Tutte's Theorem

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

b) Let $H = (S_0, \{C_1, \dots, C_m\}; E')$ be the bipartite graph with a class of the bipartition S_0 , the other class being the set of odd connected components of $G - S_0$, and $\{s, C_i\}$ is an edge of H if $s \in S_0$ is adjacent in G with $v \in C_i$.

H has a perfect matching. Indeed, we show that for H the Hall's property holds, implying the existence of matching M_0 that saturates $\{C_1, \dots, C_m\}$.

Let $A \subseteq \{C_1, \dots, C_m\}$. Then $B = N_H(A) \subseteq S_0$, and by the construction of H , in the graph G we have no edge from a vertex $v \in S_0 - B$ to a vertex $w \in C_i \in A$. Hence the odd connected components from A remains odd connected components in $G - B$; hence $q(G - B) \geq |A|$. Since G satisfies Tutte's condition (T), we have $|B| \geq q(G - B)$. We obtained that $|N_H(A)| = |B| \geq |A|$.

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

Perfect Matchings - Tutte's Theorem

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

By Hall's theorem H has a matching M_0 , which saturates $\{C_1, \dots, C_m\}$; since $|S_0| = m$, M_0 is perfect.

$$M_0 = \{s_1 v_1, s_2 v_2, \dots, s_m v_m\}, S_0 = \{s_1, \dots, s_m\}, v_i \in C_i, \forall i = \overline{1, m}.$$

c) $\forall i \in \{1, \dots, m\}$ the graph $G' = [C_i - v_i]_G$ has a perfect matching. Using the induction hypothesis, it will be enough to prove that G' satisfies (T).

Let $S' \subseteq C_i - v_i$. If $q(G' - S') > |S'|$, then, since $q(G' - S') + |S'| \equiv 0 \pmod{2}$ (because $|G'|$ is even), it follows that $q(G' - S') \geq |S'| + 2$. If $S'' = S_0 \cup \{v_i\} \cup S'$, we have

$$\begin{aligned} |S''| &\geq q(G - S'') = q(G - S_0) - 1 + q(G' - S') = |S_0| - 1 + q(G' - S') \geq \\ &\geq |S_0| - 1 + |S'| + 2 = |S''|, \end{aligned}$$

i. e., $q(G - S'') = |S''|$, contradicting the choice of S_0 (since $S_0 \subsetneq S''$).

Maximum Cardinality Matchings - Berge's Theorem

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

Let $G = (V, E)$ be a graph and $M \in \mathcal{M}_G$ be a matching in G .

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.

Definition

An **alternating path** of G with respect to the matching M is any path

$$P : v_0, v_0 v_1, v_1, \dots, v_{k-1}, v_{k-1} v_k, v_k$$

such that $\{v_{i-1} v_i, v_i v_{i+1}\} \cap M \neq \emptyset, \forall i = \overline{1, k-1}$.

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.

Note that, since M is a matching, if P is an alternating path w.r.t. M , then from any two consecutive edges of P exactly one belongs to M (the edges belong alternatively to M and $E \setminus M$).

In the following, when we will refer to a path P we will understand its set of edges.

Maximum Cardinality Matchings - Berge's Theorem

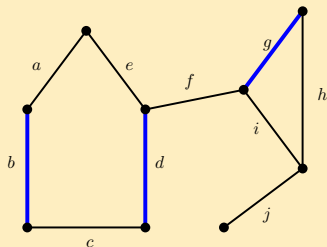
C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms
* C. Croitoru - Graph Algorithms * C. Croitoru - Graph Algorithms * C. Croitoru - Graph

Definition

An **augmenting path** of G with respect to the matching M is an alternating path joining two distinct exposed vertices with respect to M .

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

Note that, from the above definition, it follows that if P is an augmenting path w.r.t. M , then $|P \setminus M| = |P \cap M| + 1$.



a, b, c, d - alternating even path

f - alternating odd path

j - augmenting path

g, f, d - alternating odd path

a, b, c, d, e - closed alternating path

a, b, c, d, f, g, h - augmenting path

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

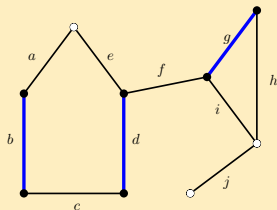
Maximum Cardinality Matchings - Berge's Theorem

Theorem 2

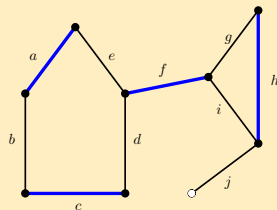
(Berge, 1959) M is a maximum cardinality matching in the graph G if and only if there is no augmenting path in G w.r.t. M .

Proof. " \Rightarrow " Let M be a maximum cardinality matching in G . Suppose that P is an augmenting path in G with respect to M .

Then, $M' = M \Delta P = (P \setminus M) \cup (M \setminus P) \in \mathcal{M}_G$. Indeed, M' can be obtained by interchanging the edges in M with those not in M on the path P :



$P = a, b, c, d, f, g, h$ - augmenting path



$M \Delta P$

Maximum Cardinality Matchings - Berge's Theorem

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

Furthermore, $|M'| = |P \cap M| + 1 + |M \setminus P| = |M| + 1$, contradicting the choice of M .

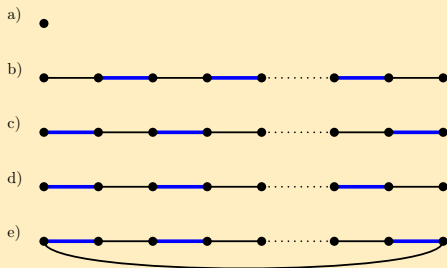
" \Leftarrow " Let M be a matching in G with the property that there are no augmenting paths in G with respect to M .

Let M^* be a maximum cardinality matching in G . We will prove that $|M^*| = |M|$. Let G' be the subgraph spanned by $M \Delta M^*$ in G ($G' = (V, M \Delta M^*)$).

Note that $d_{G'}(v) \leq 2, \forall v \in V$ and therefore the connected components of G' are isolated vertices, paths of length at least one, or cycles. We have five possible cases (see the following figure; blue edges are from M^* , black edges are from M).

Case b) does not occur since it gives an augmenting path with respect to M^* , which is a maximum cardinality matching. Case c) does not occur since it gives an augmenting path with respect to M .

Maximum Cardinality Matchings - Berge's Theorem



If we denote by $m_M(C)$ the number of edges from M in the connected component C of G' and by $m_{M^*}(C)$ the number of edges from M^* in the same connected component of G' , we obtained that $m_M(C) = m_{M^*}(C)$.

Hence,

$$\begin{aligned}
 |M \setminus M^*| &= \sum_{C \text{ conn. comp. of } G'} m_M(C) = \\
 &= \sum_{C \text{ conn. comp. of } G'} m_{M^*}(C) = |M^* \setminus M|
 \end{aligned}$$

Maximum Cardinality Matchings - Berge's Theorem

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

Therefore $|M| = |M^*|$. \square

We get a strategy to find a maximum cardinality matching:

```
let  $M$  be a matching in  $G$  (e. g.,  $M = \emptyset$ );  
while ( $\exists P$  augmenting path w. r. t.  $M$ ) do  
     $M \leftarrow M \Delta P$ ;  
end while
```

At each **while** iteration the cardinality of M increases by 1, therefore in at most $n/2$ iterations we obtain a matching without augmenting paths, that is of maximum cardinality. It remains to implement the **while** loop in polynomial time complexity. This was firstly done by **Edmonds (1965)**.

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

Maximum Cardinality Matchings - Hopcroft-Karp Algorithm

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

Lemma 1

Let $M, N \in \mathcal{M}_G$, $|M| = r$, $|N| = s$ and $s > r$. Then in $M \Delta N$ there are at least $s - r$ vertex-disjoint augmenting paths with respect to M .

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

Proof. Let $G' = (V, M \Delta N)$ and C_i ($i = \overline{1, p}$) be the connected components of G' . For each $1 \leq i \leq p$, we denote by $\delta(C_i)$ the difference between the number of edges of N in C_i and the number of edges of M in C_i :

$$\delta(C_i) = |E(C_i) \cap N| - |E(C_i) \cap M|.$$

Note that, since M and N are matchings, C_i are paths or cycles. Hence $\delta(C_i) \in \{-1, 0, 1\}$. $\delta(C_i) = 1$ if and only if C_i is an augmenting path with respect to M .

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

Maximum Cardinality Matchings - Hopcroft-Karp Algorithm

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

Proof cont'd. Since

$$\sum_{i=1}^p \delta(C_i) = |N \setminus M| - |M \setminus N| = s - r,$$

it follows that there are at least $s - r$ connected components of G' with $\delta(C_i) = 1$, that is, there are at least $s - r$ vertex-disjoint augmenting paths contained in $M \Delta N$. \square

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

Lemma 2

If $\nu(G) = s$ and $M \in \mathcal{M}_G$ with $|M| = r < s$, then there exists in G an augmenting path with respect to M of length at most $2\lceil r/(s-r) \rceil + 1$.

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

Proof. Let $N \in \mathcal{M}_G$ with $|N| = s = \nu(G)$.

Maximum Cardinality Matchings - Hopcroft-Karp Algorithm

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

Proof cont'd. By Lemma 1, there are $s - r$ edge-disjoint augmenting paths (vertex-disjoint paths are also edge-disjoint) contained in $M \Delta N$. It follows that at least one of them has at most $\lceil r/(s - r) \rceil$ edges from M . The length of this augmenting path is at most $2\lceil r/(s - r) \rceil + 1$. \square

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

Definition

Let $M \in \mathcal{M}_G$. A **minimum augmenting path** with respect to M in G is an augmenting path of minimum length over all augmenting paths with respect to M in G .

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

Lemma 3

Let $M \in \mathcal{M}_G$, P be a minimum augmenting path w. r. t. M , and P' be an augmenting path w. r. t. $M \Delta P$. Then $|P'| \geq |P| + 2|P \cap P'|$.

Maximum Cardinality Matchings - Hopcroft-Karp Algorithm

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

Proof. Let $N = (M \Delta P) \Delta P'$. Then $M \Delta N = P \Delta P'$ and $|N| = |M| + 2$. By Lemma 1, there are two edge-disjoint augmenting paths w. r. t. M , P_1 and P_2 , contained in $M \Delta N$. Since P is a minimum augmenting path w.r.t. M , we have $|P \Delta P'| \geq |P_1| + |P_2| \geq 2|P|$ and therefore $|P| + |P'| - 2|P \cap P'| \geq 2|P|$. \square

Let us consider the following algorithm:

$M_0 \leftarrow \emptyset; i = 0;$

while (\exists augmenting paths w. r. t. M_i) do

 let P_i a minimum augmenting path w.r.t. M_i ;

$M_{i+1} \leftarrow M_i \Delta P_i$;

$i++$;

end while

Let $P_0, P_1, \dots, P_{\nu(G)-1}$ be the sequence of minimum augmenting paths constructed by this algorithm.

Theorem 3

(Hopcroft, Karp, 1973) Let G be a graph and $\nu(G) = s$. The number of distinct integers in the sequence $|P_0|, |P_1|, \dots, |P_{s-1}|$ (P_i are the minimum augmenting paths constructed by the above algorithm) is not greater than $2\lfloor\sqrt{s}\rfloor + 2$.

Proof. Let $r = \lceil s - \sqrt{s} \rceil$. Then $|M_r| = r$ and, by Lemma 2,

$$|P_r| \leq 2\lceil r/(s-r) \rceil + 1 = 2\lceil \lceil s - \sqrt{s} \rceil / (s - \lceil s - \sqrt{s} \rceil) \rceil + 1 < 2\lfloor\sqrt{s}\rfloor + 3.$$

Hence, for every $i < r$, $|P_i|$ is one of $\lfloor\sqrt{s}\rfloor + 1$ odd integers not greater than $2\lfloor\sqrt{s}\rfloor + 1$.

In the sub-sequence $|P_r|, \dots, |P_{s-1}|$ there are at most $s - r \leq \lfloor\sqrt{s}\rfloor + 1$ distinct integers. It follows that in the sequence $|P_0|, |P_1|, \dots, |P_{s-1}|$ there are no more than $2\lfloor\sqrt{s}\rfloor + 2$ distinct integers. \square

$$2\lceil [s - \sqrt{s}] / (s - \lceil s - \sqrt{s} \rceil) \rceil + 1 < 2\lfloor \sqrt{s} \rfloor + 3 \Leftrightarrow$$

$$2\lceil [s - \sqrt{s}] / (s - \lceil s - \sqrt{s} \rceil) \rceil < 2\lfloor \sqrt{s} \rfloor + 2 \Leftrightarrow$$

$$\lceil [s - \sqrt{s}] / (s - \lceil s - \sqrt{s} \rceil) \rceil \leq \lfloor \sqrt{s} \rfloor + 1 \Leftrightarrow$$

$$(s - \lfloor \sqrt{s} \rfloor) / \lfloor \sqrt{s} \rfloor \leq \lfloor \sqrt{s} \rfloor + 1 \Leftrightarrow s \leq \lfloor \sqrt{s} \rfloor^2 + 2\lfloor \sqrt{s} \rfloor$$

If $\lfloor \sqrt{s} \rfloor = k \in \mathbb{N}$, then $k \leq \sqrt{s} < k + 1$, hence $s = (\sqrt{s})^2 < k^2 + 2k + 1$ which is equivalent with $s = (\sqrt{s})^2 \leq k^2 + 2k$ - the inequality from above.

If the above algorithm is decomposed in stages such that in each stage a maximal (w.r.t. inclusion) set of vertex-disjoint minimum augmenting paths is found, then - by Lemma 4 - the length of the minimum augmenting paths in the next stage will not decrease (otherwise, the set of minimum augmenting paths constructed in the current stage is not maximal). By Theorem 3, the number of stages is not greater than $2\lfloor \sqrt{\nu(G)} \rfloor + 2$.

Maximum Cardinality Matchings - Hopcroft-Karp Algorithm

Hence we have the following algorithm to find a maximum cardinality matching in G :

$M \leftarrow \emptyset$;

repeat

 find \mathcal{P} a maximal set of vertex-disjoint minimum augmenting paths
 w.r.t. M ;

 for ($P \in \mathcal{P}$) do

$M \leftarrow M \Delta P$;

 end for

until ($\mathcal{P} = \emptyset$)

The time complexity of the above algorithm is $\mathcal{O}(\sqrt{n}A)$, where A is the time needed for finding the set \mathcal{P} .

In the case of bipartite graphs, Hopcroft and Karp shown that this can be obtained in $\mathcal{O}(n + m)$ time and therefore the entire algorithm has $\mathcal{O}(m\sqrt{n})$ running time.

Maximum Cardinality Matchings - Hopcroft-Karp Algorithm

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 -

This result has been extended to arbitrary graphs by **Micali** and **Vazirani (1980)** using an elaborate data structure to maintain the labels associated to vertices in order to construct minimum augmenting paths.

Consider the case of bipartite graphs: $G = (S, T; E)$ and $M \in \mathcal{M}_G$. Starting from one of the classes, say S , we consider the set of initial extremities of augmenting paths $S \cap E(M)$. From each such vertex we start, in parallel, the construction of alternating paths in a **bfs** manner. The first obtained augmenting path stops the construction, giving the minimum length of an augmenting path. The set \mathcal{P} is obtained using the labels and the adjacency lists in $\mathcal{O}(n + m)$.

Details are omitted, an example is given on the next slide.

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 *

A **flow network (transportation network)** with source s and sink t is a tuple $R = (G, s, t, c)$ where:

- $G = (V, E)$ is a digraph,
- $s, t \in V; s \neq t; d_G^+(s) > 0; d_G^-(t) > 0,$
- $c : E \rightarrow \mathbb{R}_+; c(e)$ is the capacity of the arc e .

We will suppose that $V = \{1, 2, \dots, n\}$ ($n \in \mathbb{N}^*$) and $|E| = m$. We extend the function c to $c : V \times V \rightarrow \mathbb{R}_+$ by

$$c((i, j)) = \begin{cases} c(ij), & \text{if } ij \in E \\ 0, & \text{otherwise} \end{cases}$$

and will denote $c((i, j)) = c_{ij}$.

Definition

A **flow** in $R = (G, s, t, c)$ is a function $x : V \times V \rightarrow \mathbb{R}$ s. t.

- (i) $0 \leq x_{ij} \leq c_{ij}, \forall (i, j) \in V \times V,$
- (ii) $\sum_{j \in V} x_{ji} - \sum_{j \in V} x_{ij} = 0, \forall i \in V \setminus \{s, t\}.$

Remarks

- If $ij \in E$ then x_{ij} is referred as the **flow (transported) on ij** .
- Constraints (i) require that **the flow on each arc is non-negative and does not exceed the capacity**.
- Constraints (ii), (**equilibrium constraints**), require that **the sum of flows on the arcs entering vertex i is equal to the sum of flows on the arcs leaving i (flow in equals flow out)**.

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 -

- Note that, by the way the function c have been extended to $V \times V$, the constraints (i) implies that $x_{ij} = 0$ whenever $ij \notin E$. Hence, a flow is in fact a function on E . We prefer its extension on $V \times V$ to simplify the notations.

Let x be a flow in $R = (G, s, t, c)$. If we add all constraints (ii) (for $i \in V \setminus \{s, t\}$) we get

$$\begin{aligned}
 0 &= \sum_{i \neq s, t} \left(\sum_{j \in V} x_{ji} - \sum_{j \in V} x_{ij} \right) = \sum_{i \neq s, t} \sum_{j \neq s, t} x_{ji} - \sum_{i \neq s, t} \sum_{j \neq s, t} x_{ij} + \\
 &\quad + \sum_{i \neq s, t} x_{si} + \sum_{i \neq s, t} x_{ti} - \sum_{i \neq s, t} x_{is} - \sum_{i \neq s, t} x_{it} =
 \end{aligned}$$

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

$$= \left(\sum_{i \in V} x_{si} - \sum_{i \in V} x_{is} \right) - \left(\sum_{i \in V} x_{it} - \sum_{i \in V} x_{ti} \right),$$

Definition

The **value** of the flow x in $R = (G, s, t, c)$ is

$$v(x) = \sum_{i \in V} x_{it} - \sum_{i \in V} x_{ti}.$$

In words, $v(x)$ is the **net flow reaching the sink** of the network or, as we proved above, the **net flow leaving the source** of the network.

Note that in any network $R = (G, s, t, c)$ there is a flow: x^0 , the **null flow**, with $x_{ij}^0 = 0, \forall ij \in E$ and $v(x^0) = 0$.

Maximum flow problem

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

Maximum Flow Problem: Given $R = (G, s, t, c)$ a flow network, find a flow of maximum value.

The maximum flow problem can be viewed as an LP problem:

$$\begin{aligned} \max \quad & v \\ & \sum_{j \in V} x_{ji} - \sum_{j \in V} x_{ij} = 0, \forall i \neq s, t \\ & \sum_{j \in V} x_{js} - \sum_{j \in V} x_{sj} = -v \\ & \sum_{j \in V} x_{jt} - \sum_{j \in V} x_{tj} = v \\ & 0 \leq x_{ij} \leq c_{ij}, \forall ij \in E \end{aligned}$$

However, we will consider a direct combinatorial approach which is important when some integrality constraints are imposed to the variables (flows on the directed edges).

Exercises for the 8th seminar

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

Exercise 1. Let X be a finite set, $X_1, \dots, X_n \subseteq X$, and $d_1, d_2, \dots, d_n \in \mathbb{N}$. Prove that there are n disjoint subsets $Y_i \subseteq X_i$, $|Y_i| = d_i$, $\forall i = \overline{1, n}$ if and only if

$$\left| \bigcup_{i \in I} X_i \right| \geq \sum_{i \in I} d_i,$$

for all $I \subseteq \{1, \dots, n\}$.

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

Exercise 2. Every p -regular bipartite graph has a perfect matching ($p \geq 1$).

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

Exercise 3. Let $G = (S, T; E)$ a bipartite graph. Use Hall's theorem to prove that, for every $0 \leq k \leq |S|$, G has a matching of cardinality at least $|S| - k$ if and only if $|N_G(A)| \geq |A| - k$, $\forall A \subseteq S$.

Exercise 4. Using Tutte's theorem show that a 2-edge connected, 3-regular graph has perfect matching.

Exercise 5. Let $G = (V, E)$ a graph. A subset $A \subseteq V$ is called *m-independent* if G has a matching M which saturates all the vertices from A . Show that, for any two *m-independent* sets A and B with $|A| < |B|$, we can find a vertex $b \in B \setminus A$ such that $A \cup \{b\}$ is also *m-independent* (\Rightarrow all maximal *m-independent* sets have the same cardinality).

Exercise 6. Let $T = (V, E)$ a rooted tree; we denote by r its root and by $\text{parent}(v)$ the direct ancestor of any $v \neq r$. A matching M of T is called *proper* if any exposed vertex $v \neq r$ has a brother w such that $w \text{parent}(v) \in M$.

- Show that any *proper* matching is a maximum cardinality matching.
- Devise an algorithm for finding a *proper* matching in $\mathcal{O}(n)$ time complexity.

Exercise 7. Let $G = (V, E)$ be a p -regular bipartite graph ($p \geq 1$) and consider the following algorithm

for ($e \in E$) do

$a(e) \leftarrow 1$;

end for

$E^+ \leftarrow \{e \in E : a(e) > 0\}$;

while ($G^+ = (V, E^+)$ contains a cycle C) do

 let $C = M_1 \cup M_2$, where M_1 and M_2 are matchings with $a(M_1) \geq a(M_2)$;

 // for any $F \subseteq E$, $a(F) = \sum_{e \in F} a(e)$;

 for ($e \in E(C)$) do

 if ($e \in M_1$) then

$a(e) ++$;

 else

$a(e) --$;

 end if

 end for

$E^+ \leftarrow \{e \in E : a(e) > 0\}$;

end while

return E^+ ;

Exercise 7. (cont'd) Let $f(E^+) = \sum a^2(e)$. Prove that

- (a) after each **while** iteration $f(E^+)$ is an integer number and increases with at least $|C|$;
- (b) after each **while** iteration $\sum_{uv \in E^+} a(uv) = p$, for every $u \in V$;
- (c) the algorithm doesn't end as long as there are edges e with $0 < a(e) < p$; at the end of the algorithm $a(e) = p, \forall e \in E^+$, and E^+ is a perfect matching in G ;
- (d) the **while** loop ends, at the end of the algorithm $f(E^+) = np^2/2 = pm$, and the total length of all cycles is at most pm ;
- (e) all the cycles can be found in $\mathcal{O}(\sum_{C \text{ cycle}} |C|)$ time complexity using **dfs** traversals;
- (f) the overall time complexity of the algorithm is $\mathcal{O}(pm)$.

Exercise 8. Let $G = (S, T; E)$ be a non-null bipartite graph. Prove that the following are equivalent:

- (i) $G - \{x, y\}$ has a perfect matching, $\forall x \in S, \forall y \in T$.
- (ii) G is connected and every edge of G belongs to a perfect matching.
- (iii) $|S| = |T|$, and $\emptyset \neq A \subsetneq S$, $|N_G(A)| > |A|$.

Exercise 9. Let $G = (V, E)$ be a connected graph which has a perfect matching. Devise an $\mathcal{O}(|V| + |E|)$ time complexity algorithm that constructs a spanning tree T of G such that $V(T)$ admits a bipartition in two stable sets of maximum cardinality in T .

Exercise 10. Two kids play the following game on a given graph G : they alternatively pick a new vertex v_0, v_1, \dots such that, for every $i > 0$, v_i is adjacent with v_{i-1} . The player which can not choose another vertex will loose the game. Prove that the player which starts the game has always a winning strategy if and only if G has not a perfect matching.

