Disclaimer: this is a draft and probably contains many typos and mistakes. Please report them to Manuel.Bodirsky@tu-dresden.de.
## Contents

Chapter 1. Graphs  
1.1. Colorability 6  
1.2. Matchings 7  
1.3. Applications 9  

Chapter 2. Duality 11  
2.1. Duality in linear algebra 11  
2.2. Weighted Matchings 12  
2.2.1. Maximum matching as an integer linear program 12  
2.2.2. A relaxation 12  
2.3. The duality theorem 14  
2.3.1. Example first 14  
2.3.2. The dual linear program in general 15  
2.3.3. Optimality via feasibility 16  
2.3.4. Fourier-Motzkin elimination 16  
2.3.5. The Farkas lemma 17  
2.3.6. Proving the duality theorem 18  
2.3.7. The dualization recipe 19  
2.4. Applications 19  
2.4.1. Flows in networks 19  
2.4.2. The easychair problem 21  
2.4.3. Von Neumann minimax theorem 21  
2.4.4. Simple stochastic games 24  

Chapter 3. The Probabilistic Method 27  
3.1. Tournaments 27  
3.2. O-notation 28  
3.3. Random Graphs 29  
3.3.1. Introducing Random Graphs 29  
3.3.2. The Erdős-Rényi Evolution 30  
3.3.3. The Void 30  
3.3.4. The k-th Day 31  
3.3.5. Day ω 31  
3.3.6. Past the Double Jump 31  
3.3.7. Beyond Connectivity 31  
3.3.8. Powers of n 32  
3.4. High Girth and High Chromatic Number 32  

Bibliography 35  

Appendix A. What every mathematician needs to know about complexity theory 37
CHAPTER 1

Graphs

There are directed und undirected Graphs. We start with undirected graphs. In some contexts it is natural or even important to admit multiple edges; but for the beginning we do not need this.

We mostly follow the notation of the text book “Graph Theorie” of Reinhard Diestel. For a set $S$ we write $\binom{S}{2}$ for the set of all two-element subsets of $M$. The notation is motivated by the following identity

$$\binom{|S|}{2} = \frac{|S|(|S| - 1)}{2}.$$

**Definition 1.0.1.** A {simple undirected} graph $G$ is a pair $(V, E)$ where $V = V(G)$ is a set, called the vertex set and where $E = E(G) \subseteq \binom{V}{2}$ is called the edge set.

The elements of the vertex set of a graph $G$ are also called the vertices or the nodes of $G$. A graph $G$ is called finite if $V(G)$ is finite. If $\{u, v\} \in E(G)$ then $u$ and $v$ are called adjacent, and $v$ is called a neighbour of $u$. The number of neighbours of $x$ in $G$ is called the degree of $x$. We give a couple of examples of fundamental graphs along with their names. Let $n \in \mathbb{N}$ be a positive natural number.

- $K_n$ denotes the graph $(V, E)$ with $V := \{1, 2, \ldots, n\}$ and $E := \binom{V}{2}$. This graph is called the $n$-element clique.
- $I_n$ denotes the graph $(\{1, 2, \ldots, n\}, \emptyset)$ and is called independent set (or stable set) of size $n$.
- $P_n$, for $n \geq 2$, denotes the path of length $n$, that is, the graph $(V, E)$ with $V := \{1, \ldots, n\}$ and $E := \{\{1, 2\}, \{2, 3\}, \ldots, \{n - 1, n\}\}$. Warning: in some books and articles $P_n$ denotes the graph with $n$ edges, and not, as here, the path with $n$ nodes.
- $C_n$, for $n \geq 3$, denotes the graph

$$\{(0, 2, \ldots, n - 1), \{i, j\} | (i - j) \equiv 1 \pmod{n}\}$$

called cycle (with $n$ nodes and $n$ edges).

The complement of a graph $G = (V, E)$ is the graph $\overline{G} = (V, \binom{V}{2} \setminus E)$. For instance the complement of $I_n$ is $K_n$. Obviously, $(\overline{G}) = G$.

**Definition 1.0.2** (Isomorphism). Two graphs $G$ and $H$ are called isomorphic if there exists a bijection $f : V(G) \to V(H)$ such that $\{u, v\} \in E(G)$ if and only if $\{f(u), f(v)\} \in E(H)$.

For example the complement of $C_5$ is isomorphic to $C_5$.

**Definition 1.0.3** (subgraph). A graph $H$ is called a subgraph of $G$ if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G) \cap \binom{V(H)}{2}$. An induced subgraph of $G$ is a graph $H$ with $V(H) \subseteq V(G)$, and $E(H) = E(G) \cap \binom{V(H)}{2}$.

\footnote{For the moment, our graphs also don’t have loops; graphs without multiple edges or loops (whatever this is) are called simple.}
For \( V \subset V(G) \) we write \( G[V] \) for the (uniquely determined) induced subgraph of \( G \) with vertex set \( V \), and call \( G[V] \) the subgraph of \( G \) induced by \( V \). A sequence \( (u_1, u_2, \ldots, u_l) \) of nodes of a graph \( G \) is called a walk from \( u_1 \) to \( u_l \) in \( G \) if \( \{u_i, u_{i+1}\} \in E(G) \) for all \( i \in \{1, \ldots, l-1\} \). We allow the case \( l = 1 \); in this case the walk has only one vertex and no edges. A walk \( (u_1, u_2, \ldots, u_l) \) is called closed if \( u_1 = u_l \), and otherwise open. A walk \( (u_1, u_2, \ldots, u_l) \) is called a path from \( u_1 \) to \( u_l \) if \( u_i \neq u_j \) for distinct \( i, j \in \{1, \ldots, l\} \). Note that if there is a walk from \( u \) to \( v \) then clearly there is also a path from \( v \) to \( u \).

A cycle is a walk \( (u_0, u_1, \ldots, u_l, u_0) \) with \( u_0 = u_l \), \( l \geq 3 \) and \( u_i \neq u_j \) for all distinct \( i, j \in \{1, \ldots, l-1\} \). Note that a graph contains a cycle if and only if it contains a subgraph isomorphic to \( C_n \) for some \( n \geq 3 \).

**Definition 1.0.4 (Connectedness).** A graph \( G \) is called connected if for all \( s, t \in V(G) \) there is a walk from \( s \) to \( t \) in \( G \).

Let \( G = (V, E) \) and \( H = (V, F) \) be two graphs with disjoint vertex sets. Then \( G \cup H \) denotes the graph \( U \cup V, E \cup F \), called the disjoint union of \( G \) and \( H \). The following is easy to see.

**Lemma 1.0.5.** A graph \( G \) is connected if and only if it cannot be written as \( H_1 \cup H_2 \) for graphs \( H_1, H_2 \) with at least one vertex.

A connected component of a graph \( G \) is a connected induced subgraph \( C \) of \( G \) such that for any \( v \notin C \), the graph \( G[V \cup \{v\}] \) is not connected. Clearly, every graph can be written as a disjoint union of its connected components.

### 1.1. Colorability

A k-colouring of a graph \( G \) is a function

\[
f : V(G) \to \{0, 1, \ldots, k-1\}
\]

such that \( f(u) \neq f(v) \) for all \( \{u, v\} \in E(G) \). A graph \( G \) is called k-colorable (or k-partite) if there exists a k-colouring of \( G \). When is a graph 2-colorable?

**Proposition 1.1.1.** A finite graph \( G \) is 2-colorable if and only if it contains no odd cycles (i.e., cycles of odd size).

**Proof.** Odd cycles are certainly not 2-colourable, and neither are graphs that contain odd cycles. So suppose that \( G = (V, E) \) has no odd cycles. Note that \( G \) is 2-colourable if and only if all its connected components are 2-colourable. We color a connected component \( C \) of \( G \) as follows:

1. Select an arbitrary vertex \( u \in C \) and define \( f(u) := 0 \).
2. For all \( v \in N(u) \) define \( f(v) := 1 \).
3. If \( f(v) \) is defined for all \( v \in C \) then \( f \) is the desired colouring.
4. Otherwise, suppose that \( f(w') = i \) and \( w \in N(w') \). Define \( f(w) := 1 - i \).
5. Continue with step 3.

Since \( C \) is finite, this procedure terminates after finitely many steps, and we have found the desired colouring. \( \square \)

Note that the proof shows that there exists an efficient algorithm (which performs at most linearly many operations in \( n + m \)) that determines for a given graph whether it is 2-colorable.

Two-colorable graphs are also called bipartite. In other words, a graph \( G \) is bipartite if its vertex set can be partitioned into two independent sets \( A \) and \( B \). The set \( \{A, B\} \) is called a bipartition of \( G \), and \( A \) and \( B \) are called partition classes (or colour classes).
**Example 1.1.2.** \(K_{n,m}\) denotes the complete bipartite graph with partition classes \(P_1 := \{1, \ldots, n\}\) and \(P_2 := \{n+1, \ldots, m+n\}\), that is, \(K_{n,m} = (P_1 \cup P_2, \{\{u,v\} \mid u \in P_1, v \in P_2\})\).

When is a graph 3-colourable? For this we do not have a similarly elegant description as in Proposition 1.1.1. It is an (important) open problem (in fact, it is one of the Millennium Problems of the Clay Mathematics Institute; [http://www.claymath.org/millennium-problems/p-vs-np-problem](http://www.claymath.org/millennium-problems/p-vs-np-problem)) whether there exists an efficient algorithm that tests for a given graph whether it is 3-colourable.

**Exercises.**

1. Show that the Clebsch graph (see Figure 1.1) is 4-colourable, but not 3-colourable.
2. Suppose that a graph has 12 edges and 6 vertices, each of which has degree 3 or 5. How many vertices are there of each degree?
3. If a graph \(G\) has \(p\) vertices, and the degree of every vertex is at least \(\lceil \frac{p-1}{2} \rceil\), then \(G\) is connected.

### 1.2. Matchings

Let \(G\) be a graph. A **matching** (in \(G\)) is a subset \(M\) of \(E(G)\) of pairwise disjoint edges: that is, for all \(u, v \in M\) we have \(u \cap v = \emptyset\). A **perfect matching** is a matching \(M\) with \(2|M| = |V|\). If \(\{x, y\} \in M\) then \(y\) is called the partner of \(x\). If \(S \subseteq V(G)\) then \(M\) is a matching of \(S\) if every element of \(S\) appears in an edge of \(M\).

How can we find in \(G\) a matching of maximal size? Let \(M\) be any matching in \(G\). A path in \(G\) whose edges alternate between edges from \(E \setminus M\) and edges from \(M\) is called an alternating path. An alternating path \(P\) is called augmenting with respect to \(M\) if both the first and the last vertex of \(P\) have no partners in \(M\). Augmenting paths can be used to obtained larger matchings than \(M\). To formalise this we need the notion of a symmetric difference of two sets \(A\) and \(B\): this is the set \(A \Delta B := \{x \in A \cup B \mid x \notin A \cap B\}\).

**Lemma 1.2.1.** Let \(M_1\) and \(M_2\) be matchings in \(G = (V,E)\). Then the graph \((V, M_1 \Delta M_2)\) consists of a disjoint union of cycles of even length and of paths.

**Lemma 1.2.2 (Lemma of Berge).** Let \(G\) be a finite graph. A matching \(M\) in \(G\) is of maximal size if and only if there is no augmenting path with respect to \(M\) in \(G\).

**Proof.** We have already seen that a matching with an augmenting path cannot be of maximal size. To prove the converse, suppose that \(G\) has a matching \(M'\) in \(G\) with \(|M'| > |M|\). Since \(|M'| > |M|\) the graph \((V, M' \Delta M)\) has a component with
more edges from \( M' \) than from \( M \). By Lemma 1.2.1 such a component must be an augmenting path with respect to \( M \).

If \( S \subseteq V \) we write \( N(S) \) for \( \{ n \in V \mid \{ n, s \} \in E, \text{ for some } s \in S \} \), the neighborhood of \( S \) in \( G \). An obvious necessary condition for the existence of a matching of \( S \subseteq V \) in \( G \) is \( |N(S)| \geq |S| \). This condition is in general not sufficient (example?). Theorem 1.2.3 presents a necessary and sufficient condition for the existence of a matching of \( S \) in \( G \) in bipartite (i.e., two-colorable) graphs.

In the following, let \( G = (V, E) \) be a bipartite graph with a fixed bipartition \( A, B \), so that \( A \cup B = V \) and \( A, B \) are independent sets.

**Theorem 1.2.3 (Hall’s Marriage theorem).** A bipartite graph \( G \) has a matching of \( A \subseteq V(G) \) if \( |N(S)| \geq |S| \) for all \( S \subseteq A \).

**Manuel’s favourite proof.** Let \( M \) be a matching of \( G \) such that \( a_0 \in A \) remains without partner. We will construct an augmenting path with respect to \( M \). Let \( a_0, b_1, a_1, b_2, a_2, \ldots \) be a sequence of maximal length of distinct vertices \( a_i \in A \) and \( b_i \in B \) such that

1. \( \{b_i, a_i\} \in M \), and
2. \( b_i \) has an edge to a vertex \( a_{f(i)} \in \{a_0, \ldots, a_{i-1}\} \).

By assumption this sequence cannot end in a vertex from \( A \): the \( i \) nodes \( a_0, \ldots, a_{i-1} \) together have at least \( i \) neighbours in \( B \), so we can always find an edge \( \{a, b\} \in E \) such that \( a \in \{a_0, \ldots, a_{i-1}\} \) and \( b \in B \setminus \{b_1, \ldots, b_{i-1}\} \). Let \( b_k \in B \) be the last vertex of this sequence. Because of the two properties \( P := b_k a_{f(k)} b_{f(k)} a_{f^2(k)} b_{f^2(k)} \ldots a_{f^t(k)} \) with \( f^t(k) = 0 \) is an alternating path.

We claim that \( b_k \) has no partner in \( M \). If \( a \) would be a partner of \( b_k \) and \( a = a_i \) for an \( i \in \{1, \ldots, k-1\} \), then \( b_k = b_i \), since \( M \) is a matching, contradiction. If \( a \neq a_i \) for all \( i \in \{1, \ldots, k-1\} \), then \( a_k := a \) would prolong our sequence, contradicting its maximality. Hence, \( b_k \) is without partner and \( P \) is an augmenting path.

**Andrés’ favourite proof.** Suppose for a contradiction that there is no matching for \( A \) in \( G \) and \( |N(S)| \geq |S| \) for all \( S \subseteq A \). Let \( M \) be a matching of \( G \) of maximal size. Since \( M \) is not a matching for \( A \), there is some \( u \in A \) without a partner in \( M \). Note that \( N(u) \neq \emptyset \) because \( |N(u)| \geq \{|u|\} = 1 \) by hypothesis, and \( N(u) \) contains endpoints of edges in \( M \) by maximality of \( M \). In the rest of this proof, the adjective “alternating” will refer to the matching \( M \).

Let \( X \) be the set containing all vertices that can be reached by alternating paths that start in \( u \), and let \( B' := X \cap B \) and \( A' := X \cap A \). We claim that every \( b \in B' \) is the partner of some \( a \in A' \setminus \{u\} \): by the definition of \( B' \), there is an alternating path \( P \) from the unmatched \( u \) to \( b \). By Lemma 1.2.2 the path \( P \) cannot be augmenting, so \( b \) must have a partner \( a \) in \( M \). Then \( a \in A' \) since \( a \) appended at the path \( P \) gives an alternating path from \( u \) to \( a \). We also have \( a \neq u \) because \( u \) has no partner in \( M \). Similarly, each vertex in \( A' \setminus \{u\} \) is the partner of some vertex in \( B' \). Therefore, \( |A'| = |B'| + 1 \).

Now we claim that \( N(A') \subseteq B' \). Suppose that \( w \in B \) is connected to some \( d \in A' \). There are two cases to consider:

1. If \( \{w, d\} \in M \), then \( w \in B' \) by the argument above, and
2. if \( \{w, d\} \notin M \), then \( w \) is on an alternating path starting from \( u \), so that \( w \in B' \).

From this we get \( |N(A')| = |A'| - 1 \), contradicting the hypothesis.

**Exercises.**
(4) The goal of this exercise is to show that the marriage theorem is false for infinite bipartite graphs. Let $G$ be the graph with vertex set $\mathbb{Z}$ and edge set
\[ \{(a, -a) \mid a \in \mathbb{Z} \setminus \{0\}\} \cup \{\{0, a\} \mid a \in \mathbb{N} \setminus \{0\}\}. \]
Show that the conditions in Hall’s marriage theorem are satisfied for $A := \{a \in \mathbb{Z} \mid a \leq 0\}$, but that $A$ has no matching in $G$.

(5) Let $G$ be a bipartite graph with partition classes $A$ and $B$, and let $A' \subseteq A$ and $B' \subseteq B$. Suppose that $A'$ has a matching in $G$ and $B'$ has a matching in $G$. Prove that there exists a matching of $A' \cup B'$ in $G$. Does this statement hold if $G$ is infinite?

1.3. Applications

A graph $G$ is called $k$-regular if every node in $G$ has degree $k$.

Corollary 1.3.1. Every bipartite $k$-regular graph, for $k \geq 1$, has a perfect matching.

Proof. Every subset $S \subseteq A$ has exactly $k|S|$ edges into $N(S)$. Together there are $|k|N(S)|$ edges to vertices in $N(S)$. Therefore $k|S| \leq |N(S)|$ and $|S| \leq |N(S)|$. The Marriage Theorem gives us a matching of $A$ in $G$. In regular bipartite graphs we clearly have $|A| = |B|$. Hence we have found a perfect matching for $G$. □

Another consequence of the marriage theorem is the important theorem of König. Let $G = (V, E)$ be a graph. A set $U \subseteq V$ is called a covering of $G$ if every edge of $G$ contains a vertex from $U$.

Theorem 1.3.2 (König). Let $G$ be a finite bipartite graph. Then the maximal size of a matching of $G$ equals the minimum size of a covering of $G$.

Proof. Let $U$ be a covering of $G$ of minimal size. Let $M$ be a matching of $G$. We need at least $|M|$ nodes to cover $M$. Hence, $|U| \geq |M|$. We will prove that there exists a matching of size $|U|$ in $G$. Let $U_1 := U \cap A$ and $U_2 := U \cap B$. To verify the marriage condition for $U_1$ in the bipartite graph $G_1 := G[U_1 \cup B \setminus U_2]$, we have to show for an arbitrary $S \subseteq U_1$ that $|S| \leq |N(S)|$. Otherwise, we could have replaced the set $S$ in $U$ by the smaller set $N(S)$, and would still have a cover of $G$, contradicting the minimality of $U$. The Marriage Theorem (Theorem 1.2.3) gives us a matching $M_1$ of $U_1$ in $G_1$. Analogously, we obtain a matching $M_2$ of $U_2$ in the graph $G_2 := G[U_2 \cup A \setminus U_1]$. Then $M_1 \cup M_2$ is a matching in $G$, and $|M_1 \cup M_2| = |U_1| + |U_2| = |U|$. □

Exercises.

(6) A Latin square of order $n$ is matrix $A \in \{1, \ldots, n\}^{n \times n}$ such that each symbol appears exactly once in each row and each column. If $A \in \{1, \ldots, n\}^{m \times n}$ for $m < n$ is such that each symbol appears at most once in every row and every column then $A$ is called a Latin rectangle. Use Hall’s marriage theorem to prove that any $m \times n$ Latin rectangle can be extended to an $n \times n$ Latin square.

(7) A partially ordered set is a pair $(P, \leq)$ where $P$ is a set and $\leq$ is a binary relation that satisfies for all $p, q, r \in P$:  
(a) $p \leq p$,  
(b) if $p \leq q$ and $q \neq p$, then $p = q$, and  
(c) if $p \leq q$ and $q \leq p$, then $p \leq r$.

Two elements $p, q$ of $P$ are called comparable if $p \leq q$ or $q \leq p$, and incomparable otherwise. A subset $S$ of $P$ is called
• a chain if all pairs of elements of $S$ are comparable.
• an antichain if distinct elements of $S$ are incomparable.

Use König’s theorem to prove Dilworth’s theorem: if $(P, \leq)$ is a finite partial order, then the size of the largest antichain of $P$ equals the minimal $k$ such that there are chains $C_1, \ldots, C_k$ in $P$ such that $P = C_1 \cup \cdots \cup C_k$. 
CHAPTER 2

Duality

Proposition 1.1.1 is an example of a duality: for every graph \( G \), either it is 2-colourable, which is easy to verify once we are given the 2-colouring \( f: V(G) \to \{0,1\} \), or it contains an odd cycle, which is easy to verify, too. Even Lemma 1.0.5 can be viewed as a (baby-) duality: either a graph has a decomposition as a disjoint union of non-trivial smaller graphs, or for every pair of vertices \( u, v \in V(G) \) there exists a path from \( u \) to \( v \) in \( G \). The marriage theorem of Hall and the theorem of Kónig are further examples of dualities.

One way to formalise the similarities in these statements has been proposed by Jack Edmonds. A class of finite graphs \( C \) (or a class of more general mathematical structures) is said to have a good characterisation if

- \( C \) is in the complexity class NP (i.e., there exists a polynomial-time non-deterministic Turing machine that decides membership in \( C \)), and
- the complement of \( C \) is in the complexity class NP, too, i.e., \( C \) is in the complexity class coNP.

Typically, if a class is in \( \text{NP} \cap \text{coNP} \), it is also in the complexity class \( \text{P} \), i.e., it can be solved in polynomial time. Hence, a good characterisation for \( C \) can be taken as an indication that an efficient algorithm for membership in \( C \) exists.

In this section we first revisit a well-known duality from linear algebra. We then find a common generalisation of Kónig’s theorem for matchings and linear algebra duality.

2.1. Duality in linear algebra

Let \( A \in \mathbb{Q}^{m \times n} \) and \( b \in \mathbb{Q}^m \). If \( Ax = b \) has a solution, then this can be shown by simply presenting a solution from \( \mathbb{Q}^n \) for the vector of unknowns. Are there also simple proofs for unsatisfiability of \( Ax = b \) (perhaps even simpler than performing Gaussian elimination)?

**Theorem 2.1.1 (Duality).** Let \( A \in \mathbb{Q}^{m \times n} \), \( x = (x_1, \ldots, x_n) \) an \( n \)-tuple of variables, and \( b \in \mathbb{Q}^m \). Then \( Ax = b \) is unsatisfiable if and only if the system

\[
(A|b)^\top y = (0, \ldots, 0, 1)^\top
\]

is satisfiable.

**Proof.** \( \Leftarrow \) is the easier direction. Suppose we can derive from \( A|b \) by row transformations row \( (0, \ldots, 0, 1) \). Then this means that \( Ax = b \) implies \( 0x_1 + \cdots + 0x_n = 1 \), which is unsatisfiable. Hence, \( Ax = b \) must be unsatisfiable.

Let \( z_1, \ldots, z_m \) be the rows of \( A|b \). If \( (0, \ldots, 0, 1) \in \langle z_1, \ldots, z_m \rangle \) then \( (0, \ldots, 0, 1) \) can be derived from \( A|b \) with row transformations. We have that

\[
(0, \ldots, 0, 1)^\top \in \langle z_1, \ldots, z_m \rangle
\]

\( \Leftrightarrow \) there are \( y_1, \ldots, y_m \in K \) such that \( y_1z_1 + \cdots + y_m z_m = (0, \ldots, 0, 1)^\top \)

\( \Leftrightarrow (A|b)^\top = (0, \ldots, 0, 1)^\top \) has a solution.
The direction $\Rightarrow$ is also easy to show, using the row echelon form. Use row transformations to bring the matrix $(A|b)$ into row echelon form $(C|d)$. If $Ax = b$ is unsatisfiable, then $Cx = d$ is unsatisfiable, and $r := \text{rank}(C) < \text{rank}(C|d)$ by well-known linear algebra. In particular $z_{r+1} = (0,\ldots,0,d_{r+1})$ with $d_{r+1} \in \mathbb{Q} \setminus \{0\}$. Since this row has been derived from $(A|b)$ by row transformations, we have $(0,\ldots,0,1) \in \langle z_1,\ldots,z_m \rangle$. Hence, the system $(A|b)^T y = (0,\ldots,0,1)^T$ has a solution. \hfill $\square$

### 2.2. Weighted Matchings

We would like to find a common generalisation of dualities for matchings and the duality of linear algebra that we have just seen in the previous section. In our first step from matchings towards algebra we consider in this section the weighted matching problem. In this problem, we are given a bipartite graph $G$ with partition classes $A$ and $B$ where each edge $e \in E(G)$ is decorated by a weight $w_e \in \mathbb{Q}$. We are interested in finding a matching $M$ of $A$ in $G$ whose weight $w(M) := \sum_{e \in M} w_e$ is maximal. This problem has numerous applications.

#### 2.2.1. Maximum matching as an integer linear program

We will reformulate the weighted matching problem for $G$ as a linear optimisation problem over a system of linear inequalities. We introduce a variable $x_e$ for each edge $e \in E(G)$; the variable $x_e$ can attain values 0 or 1. They encode the desired matching $M$, where $x_e = 1$ means $e \in M$ and $x_e = 0$ means $e \notin M$. Then $\sum_{e \in M} w_e$ can be written as

$$w(x) := \sum_{e \in E(G)} w_e x_e$$

where $x$ is a tuple listing all the variables; $w(x)$ will be called the objective function. The requirement that a vertex $v \in V$ appears in exactly one edge from $M$ can be expressed by

$$\sum_{e \in E(G), e \ni v} x_e = 1.$$

More generally, a set of linear inequalities over the integers together with a linear objective function is called an Integer Linear Program. Unfortunately, there is in general no efficient algorithm known that solves integer linear programs. However, if the integer linear program comes from a weighted matching problem as described above, a miracle happens, as we will see in the next subsection.

#### 2.2.2. A relaxation

If we leave out the integrality conditions, i.e., if we allow each variable $x_e$ to attain all values in the interval $[0,1]$, we obtain the following:

Maximize $\sum_{e \in E} w_e x_e$

subject to $\sum_{e \in E, v \in e} x_e = 1$ for each $v \in V$ and

$$0 \leq x_e \leq 1$$

for each $e \in E$.

Such an optimisation problem is called a linear program; they can be solved efficiently, and are extremely important in optimisation and theoretical computer science. There is a notion of a dual of a linear program, and this notion of duality has many applications in combinatorics, as we will see in later sections. For more background on linear
programming, we recommend [5], which we have freely used to prepare the following subsections.

Linear programs that are obtained from integer linear programs as described above are called LP relaxations. A linear program might not have any solution at all (if the set of linear inequalities is unsatisfiable); if this happens for an LP relaxation, then the original integer linear problem does not have a solution as well. For example, this might happen if we consider the integer linear program for a bipartite graph which does not have a perfect matching.

Let us now assume that the LP relaxation has an optimal solution \( s \in \mathbb{Q}^n \), i.e., a solution where the value of the objective function is largest possible. Certainly \( s \) provides an upper bound for the objective function of the original integer program. This is because every feasible solution of the integer program is also a feasible solution of the LP relaxation. The mentioned miracle in the case that we started with the integer program for the weighted matching problem is that this upper bound is also tight: it provides an optimal solution of the original problem!

**Theorem 2.2.1.** Let \( G = (V, E) \) be a finite bipartite graph with rational edge weights \( w_e \). If the LP relaxation \( \text{LP} \) has at least one feasible solution, then it has at least one integral optimal solution. This solution is an optimal solution for the integer program as well, and hence provides a maximum weight matching of \( G \).

**Proof.** Let \( s \) be an optimal solution of the LP relaxation, and let \( w(s) = \sum_{e \in E} w_es_e \) be the value of the objective function at \( s \). Let \( k(s) \) be the number of \( e \in E \) such that \( s_e \) is not integral. If \( k(s) = 0 \) there is nothing to be shown. Otherwise, we will produce a new optimal solution \( s' \) such that \( k(s') < k(s) \). Let \( e_1 = \{a_1, b_1\} \in E \) be such that \( 0 < s_{e_1} < 1 \). Since \( \sum_{e \in E} b_{e_1} s_e = 1 \) there exists another edge \( e_2 = \{a_2, b_1\} \in E \) with \( a_2 \neq a_1 \) such that \( s_{e_2} \) nonintegral. Similarly, we can find a third edge \( e_3 = \{a_2, b_2\} \) with \( 0 < s_{e_3} < 1 \). We continue in this manner and obtain a longer and longer path (\( a_1, b_1, a_2, b_2, \ldots \)). Since the graph \( G \) has finitely many vertices, eventually we reach a vertex that we have already visited before. Since \( G \) is bipartite, this means that we have found an even cycle \( C = (c_0, \ldots, c_{2l-1}) \) such that \( 0 < s_{\{c_i, c_{i+1}\}} < 1 \) for all \( i \in \mathbb{Z}^l \) (indices of \( c \) modulo \( l \)). For a small number \( \epsilon \) define

\[
s'_{e} :=
\begin{cases}
    s_{\{c_i, c_{i+1}\}} - \epsilon & \text{for } e = \{c_i, c_{i+1}\} \text{ and } i \text{ even} \\
    s_{\{c_i, c_{i+1}\}} + \epsilon & \text{for } e = \{c_i, c_{i+1}\} \text{ and } i \text{ odd} \\
    s_e & \text{otherwise.}
\end{cases}
\]

Then \( s' \) satisfies for every \( v \in V \) the condition \( \sum_{e \in E, v \in e} s'_e = 1 \). For \( \epsilon \) sufficiently small the condition \( 0 \leq s'_e \leq 1 \) are satisfied, too, and \( s' \) is a feasible solution. The objective function evaluated at \( s' \) is

\[
w(s') = \sum_{e \in E} w_es'_e = w(s) + \epsilon \sum_{i=0}^{l-1} (-1)^i w_{\{c_i, c_{i+1}\}}.
\]

Since \( s \) is optimal, we must have \( \Delta := \sum_{i=0}^{l-1} (-1)^i w_{\{c_i, c_{i+1}\}} = 0 \) since otherwise we could achieve \( w(s') > w(s) \) by choosing \( \epsilon > 0 \) for \( \Delta > 0 \) and by choosing \( \epsilon < 0 \) for \( \Delta < 0 \). This means that \( s' \) is an optimal solution which is feasible for all \( \epsilon \) with a sufficiently small absolute value. Let us now choose the largest \( \epsilon > 0 \) such that \( s' \) is still feasible. Then there has to exist \( i \in \{0, \ldots, l - 1\} \) such that \( s'_{\{c_i, c_{i+1}\}} \in \{0, 1\} \), and \( s' \) has fewer nonintegral components than \( s \). \( \square \)
2.3. The duality theorem

This section presents an extremely powerful and useful duality result that implies many of the dualities that we have seen previously, and many more results in combinatorics.

2.3.1. Example first. Let us start with a concrete example. Consider the linear program (see Figure 2.1)

\[
\begin{align*}
\text{maximize} & \quad x_1 + x_2 \\
\text{subject to} & \quad 3x_1 - x_2 \leq 0 \\
& \quad -x_1 + x_2 \leq 4 \\
& \quad 2x_1 + 4x_2 \leq 40 \\
& \quad x_1, x_2 \geq 0
\end{align*}
\]

Since \(x_1, x_2 \geq 0\) we obtain that

\[
x_1 + x_2 \leq 2x_1 + 4x_2 \leq 40
\]

so the optimum is bounded by 40. We can obtain a better bound by first dividing the third inequality by two:

\[
x_1 + x_2 \leq x_1 + 2x_2 \leq 20.
\]

We can do even better by adding the second and two times the third inequality:

\[
x_1 + x_2 = (3x_1 - x_2) + (-2x_1 + 2x_2) \leq 8.
\]

More generally, from the constraints we are trying to derive an inequality of the form \(c_1x_1 + c_2x_2 \leq h\) where \(c_1, c_2 \geq 0\) and \(h\) is as small as possible. We derive inequalities by choosing nonnegative coefficients \(y_1, y_2, y_3\), obtaining

\[
y_1(3x_1 - x_2) + y_2(-x_1 + x_2) + y_3(2x_1 + 4x_2) \leq y_2 4 + y_3 40
\]

which can be rewritten to

\[
(3y_1 - y_2 + 2y_3)x_1 + (-y_1 + y_2 + 4y_3)x_2 \leq 4y_2 + 40y_3
\]

and thus \(c_1 = 3y_1 - y_2 + 2y_3\), \(c_2 = -y_1 + y_2 + 4y_3\), and \(h = 4y_2 + 40y_3\). Finding such \(y_1, y_2, y_3\) is again a linear program, namely

\[
\begin{align*}
\text{minimize} & \quad 4y_2 + 40y_3 \\
\text{subject to} & \quad 3y_1 - y_2 + 2y_3 \geq 1 \\
& \quad -y_1 + y_2 + 4y_3 \geq 1 \\
& \quad y_1, y_2, y_3 \geq 0
\end{align*}
\]

Clearly, the optimum of the new linear program, which is called the dual linear program, provides an upper bound for the optimum of the original linear program. Note that in the dual LP, we have one variable for each constraint of the original LP, and one constraint for each variable of the original LP.

In fact, the upper bound from the dual is tight! This can be seen from the observation that the maximum of the original linear program is at least 8, because 4 is attained for \(x_1 = 2\) and \(x_2 = 6\). The minimum of the dual is at most 8, because 8 is attained for \(y_1 = 1\), \(y_2 = 2\), and \(y_3 = 0\). So the maximum of the original linear program equals the minimum of the dual linear program.
2.3. The duality theorem

2.3.2. The dual linear program in general. Let \( A \) be a matrix with \( m \) rows and \( n \) columns and entries from \( \mathbb{Q} \) (the same results hold for \( \mathbb{R} \) instead of \( \mathbb{Q} \)). Consider the linear program

\[
\text{maximize } c^\top x \text{ subject to } Ax \leq b \text{ and } x \geq 0 \tag{P}
\]

which we call the primal linear program in the following. Similarly as in Section 2.1, we are trying to combine the \( m \) inequalities of the system \( Ax \leq b \) with some nonnegative coefficients \( y_1, \ldots, y_m \) so that

- the resulting inequality has the \( j \)-th coefficient at least \( c_j \), and
- the right-hand side is as small as possible.

This leads to the dual linear program

\[
\text{minimize } b^\top y \text{ subject to } A^\top y \geq c \text{ and } y \geq 0 \tag{D}
\]

The following is clear from the way we constructed the dual linear program:

**Proposition 2.3.1.** For each feasible solution \( t \) of the dual linear program (D) the value \( b^\top t \) provides an upper bound on the maximum of the objective function of the primal (P).

Note that this implies that if (P) is unbounded (from below), then (D) must be infeasible, and if (D) is unbounded (from above) then (P) is infeasible. The following, on the other hand, requires proof.

**Theorem 2.3.2 (LP duality).** Exactly one of the following possibilities occurs.

1. Neither (P) nor (D) has a solution.
2. (P) is unbounded and (D) has no solution.
3. (P) has no solution and (D) is unbounded.
4. Both (P) and (D) have a solution, and if \( s \) is an optimal solution to (P) and \( t \) is an optimal solution to (D) then
   \[
   c^\top s = b^\top y.
   \]

**Example 2.3.3.** The problem of finding small vertex covers can also be formulated as an integer linear program, namely as

\[
\text{minimize } \sum_{v \in V} y_v \text{ subject to } y_v \geq 0 \text{ for all } v \in V
\]

\[
\sum_{v \in V, v \in e} x_e \geq 1 \text{ for all } e \in E
\]

Similarly as for matchings it turns out that the LP relaxation provides the optimum also for the integer linear program. Moreover, the linear program for matching that
we have already seen for the weighted case

$$\text{maximize } \sum_{e \in E} x_e \text{ subject to } x_e \geq 0 \text{ for all } e \in E$$

$$\sum_{e \in E, v \in e} x_e \leq 1 \text{ for all } v \in V$$

is precisely the dual of the linear program for the vertex cover problem! So linear programming duality implies König’s theorem.

2.3.3. Optimality via feasibility. In this section we use the duality theorem to reduce the task of deciding whether a given linear program (P) has an optimal solution to the task of deciding whether a set of linear inequalities has any solution (which can be solved in polynomial time by, for example, the so-called ellipsoid method; see [7] for more details). We simply combine the constraints from (P), the constraints from (D), and add an inequality between the objective functions, obtaining the following system of linear inequalities:

$$Ax \leq b$$
$$A^\top y \geq c$$
$$c^\top x \geq b^\top y$$
$$x, y \geq 0$$

For each feasible solution \((s, t)\) for the variables \((x, y)\) of this system, \(s\) is an optimal solution of the linear program \((P)\).

2.3.4. Fourier-Motzkin elimination. Our proof of LP duality uses the lemma of Farkas, which in turn we prove using Fourier-Motzkin elimination. Fourier-Motzkin elimination is a systematic procedure for eliminating variables from systems of linear inequalities. If we eliminate all variables from the system, we might end up with an inequality of the form \(a \leq b\) for a value \(a\) which is strictly larger than \(b\), a contradiction. In this case we know that the original system was infeasible, and otherwise the original system was feasible. Geometrically speaking, we compute in each step a system that describes the projection of the solution space of the system to a subset of the variables. The way this is done is very similar in spirit to Gaussian elimination that we already used in Section 2.1. We look at an example first. Consider the linear system

$$x - y \leq 0$$
$$-x - 3y \leq -6$$
$$y + x \leq 2$$
$$-x + 3y \leq 0$$

To eliminate the variable \(y\), we collect lower bounds on \(y\) in terms of \(x\), and upper bounds on \(y\) in terms of \(x\), so we rewrite the equation system into:

$$y \geq x$$
$$y \geq 2 - \frac{1}{3}x$$
$$y \leq 2 - x$$
$$y \leq \frac{1}{3}x$$

Note that each upper bound must be larger than each lower bound, so the system implies that
\[ x \leq 2 - x \quad \text{(combining (5) and (7))} \]
\[ x \leq \frac{1}{3} x \quad \text{(combining (5) and (8))} \]
\[ 2 - \frac{1}{3} x \leq 2 - x \quad \text{(combining (6) and (7))} \]
\[ 2 - \frac{1}{3} x \leq \frac{1}{3} x \quad \text{(combining (6) and (8))} \]

Rewriting again, we obtain
\[ x \leq 1 \]
\[ x \leq 0 \]
\[ x \leq 0 \]
\[ x \geq 3 \]

Again combining lower with upper bounds, we obtain a contradiction, so the original system was unsatisfiable. Motzkin’s theorem states that if we cannot derive a contradiction by this procedure, then the original system was satisfiable. This simply follows from the observation that at each step of the procedure, any solution to the new system can be extended to a solution to the old system (by picking any value that lies between the lower and the upper bounds).

The Fourier-Motzkin procedure is not very efficient: in each step, the number of inequalities can grow quadratically, which might lead to an exponential growth in general. However, our current goal is theoretical: we want to prove the duality theorem, and do not care about efficiency. Because of the simplicity of the procedure, it is a very good starting point for proving LP duality.

We mention that the results in this section have straightforward generalisations to the situation where some of the inequalities are strict.

2.3.5. The Farkas lemma. We use Fourier-Motzkin Elimination to prove the following important lemma, the lemma of Farkas. There are numerous variants for this lemma, we give two. They are easily seen to be equivalent. The first variant is the more natural one to prove from Fourier-Motzkin elimination and the second one is the one needed in our proof of the LP duality theorem.

**Lemma 2.3.4 (Lemma of Farkas in two variants).** Let \( A \in \mathbb{Q}^{m \times n} \) and let \( b \in \mathbb{Q}^m \).

1. The system \( Ax \leq b \) has a solution if and only if every nonnegative \( y \in \mathbb{Q}^m \) with \( y^\top A = 0^\top \) also satisfies \( y^\top b \geq 0 \).
2. The system \( Ax \leq b \) has a nonnegative solution if and only if every nonnegative \( y \in \mathbb{R}^m \) with \( y^\top A \geq 0 \) also satisfies \( y^\top b \geq 0 \).

**Proof.** We prove variant (1) using Motzkin, and then derive (2) from (1).

First we prove the easy direction of variant (1). If \( Ax \leq b \) has some solution \( \tilde{x} \), and \( y \geq 0 \) satisfies \( y^\top A = 0^\top \), we get \( y^\top b \geq y^\top A\tilde{x} = 0^\top \tilde{x} = 0 \). For the interesting direction of (1) we assume that \( Ax \leq b \) has no solution. Our task is to construct a vector \( y \geq 0 \) satisfying \( y^\top A = 0^\top \) and \( y^\top b < 0 \). We find such a witness of infeasibility by induction on the number of variables. In the base case the system \( Ax \leq b \) has no variables, so it is of the form \( 0 \leq b \) with \( b_i < 0 \) for some \( i \leq m \). Then \( y = e_i \) (the \( i \)-th unit vector) clearly satisfies the requirements for \( y \). If \( x \leq b \) has at least one variable, we perform a step of the Fourier-Motzkin elimination. This yields an infeasible system \( A'x' \leq b' \) with one variable less. So inductively we find an unfeasibility witness \( y' \) for it. Recall that all inequalities of \( A'x' \leq b' \) are positive linear combinations of original
inequalities; equivalently, there is an $m \times m$ matrix $M$ with all entries nonnegative and $(0|A') = MA, b' = Mb$. We claim that $y = M^\top y'$ is a witness of infeasibility for the original system $Ax \leq b$. Indeed, we have $y^\top A = y'^\top MA = y'^\top (0|A') = 0^\top$ and $y^\top b = y'^\top Mb = y'^\top b' < 0$ since $y'$ is a witness of infeasibility for $A'x' \leq b'$. The condition $y \geq 0$ follows from $y' \geq 0$ by the nonnegativity of $M$.

To prove that (1) implies (2) we have to find an equivalent condition for $Ax \leq b$ having a nonnegative solution. Let $\bar{A} = (\begin{array}{c} A \\ -I_n \end{array})$, where $I_n$ is the $n \times n$-unit matrix, and $\bar{b} = (b | 0)$. Note that $Ax \leq b$ has a nonnegative solution if and only if $\bar{A}x \leq \bar{b}$ has any solution. The latter is equivalent, by (1), to the condition that all $\bar{y} \geq 0$ with $\bar{y}^\top \bar{A} = 0^\top$ satisfy $\bar{y}^\top \bar{b} \geq 0$. Writing $\bar{y} = (\bar{y}_v, \bar{y}_c)$ where $\bar{y}_v$ is a vector with $m$ components, we have

$$\bar{y} \geq 0 \text{ and } \bar{y}^\top \bar{A} = 0^\top \text{ if and only if } y \geq 0 \text{ and } y^\top A \geq 0^\top.$$

Moreover, $\bar{y}^\top \bar{b} = y^\top b$. Hence, $Ax \leq b$ has a nonnegative solution if and only if all $y \geq 0$ with $y^\top A \geq 0^\top$ satisfy $y^\top b \geq 0$, which is what we had to show.

### 2.3.6. Proving the duality theorem.

**Proof of Theorem 2.3.2.** Let us assume that the linear program (P) has an optimal solution $x^*$. We show that the dual (D) has an optimal solution as well, and that the optimum values of both programs coincide. Let $\gamma = c^\top x^*$ be the optimum value of (P). Then we know that the system of inequalities

$$Ax \leq b, c^\top x \geq \gamma$$

has a nonnegative solution, but for any $\epsilon > 0$, the system

$$Ax \leq b, c^\top x \geq \gamma + \epsilon$$

has no nonnegative solution. If we define an $((m + 1) \times n)$-matrix $\hat{A}$ and a vector $\hat{b}_v \in \mathbb{Q}^{m+1}$ by

$$\hat{A} := \begin{pmatrix} A \\ -c^\top \end{pmatrix}, \hat{b}_v := \begin{pmatrix} b \\ -\gamma - \epsilon \end{pmatrix}$$

then (9) is equivalent to $\hat{A}x \leq \hat{b}_0$ and (10) is equivalent to $\hat{A}x \leq \hat{b}_v$.

We now apply variant (2) of the Farkas lemma and conclude that there is a nonnegative vector $\bar{y} = (u, z) \in \mathbb{Q}^{m+1}$ such that $\bar{y}^\top \hat{A} \geq 0^\top$ but $\bar{y}^\top \hat{b}_v < 0$. These conditions boil down to

$$A^\top u \geq zc, b^\top u < z(\gamma + \epsilon),$$

Applying the Farkas lemma for the case $\epsilon = 0$ we see that the very same vector $\bar{y}$ must satisfy $\bar{y}^\top \hat{b}_0 \geq 0$, and this is equivalent to

$$b^\top u \geq z\gamma.$$ 

It follows that $z > 0$, since $z = 0$ would contradict the strict inequality in (11). But then we may set $v := \frac{1}{z}u \geq 0$, and (11) gives

$$A^\top v \geq c, b^\top v < \gamma + \epsilon.$$ 

In other words, $v$ is a feasible solution of (D), with the value of the objective function smaller than $\gamma + \epsilon$.

We have already observed that every feasible solution of (D) has value of the objective function at least $\gamma$. Hence (D) is a feasible and bounded linear program, and so we know that it has an optimal value $y^*$. Its value $b^\top y^*$ is between $\gamma$ and $\gamma + \epsilon$ for every $\epsilon > 0$, and thus it equals $\gamma$. □
2.3.7. The dualization recipe. The linear program (P) in the duality theorem, Theorem 2.1.1, has a very particular shape: all variables must be nonnegative, and we only have inequality conditions, no equalities. We can easily transform a general linear program into a program of this shape: for each variable \( x \), we introduce two new variables \( x^+ \) and \( x^- \), add the constraints \( x^+ \geq 0 \) and \( x^- \geq 0 \), and we substitute \( x \) by \( x^+ - x^- \) everywhere. Moreover, an equality \( a^T x = b \) can be rewritten as a conjunction of two inequalities, \( a^T x \leq b \) and \( a^T x \geq b \). Finally, inequalities of the form \( a^T x \geq b \) can be turned by using \( -a^T x \leq -b \) instead. So we can transform every linear program into one that has the shape as in the duality theorem, and then apply the duality theorem.

However, it is always possible to read off the obtain dual LP directly from the original LP, without doing the transformation. For this, we use the following recipe (details can be worked by the readers themselves):

<table>
<thead>
<tr>
<th>Variables</th>
<th>Primal</th>
<th>Dual</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matrix</td>
<td>( A )</td>
<td>( A^T )</td>
</tr>
<tr>
<td>Right-hand side</td>
<td>( b )</td>
<td>( c )</td>
</tr>
<tr>
<td>Objective function</td>
<td>( \max c^T x )</td>
<td>( \min b^T y )</td>
</tr>
<tr>
<td>Constraints</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( i )-th constraint ( \leq )</td>
<td>( y_i \geq 0 )</td>
<td></td>
</tr>
<tr>
<td>( i )-th constraint ( \geq )</td>
<td>( y_i \leq 0 )</td>
<td></td>
</tr>
<tr>
<td>( i )-th constraint ( = )</td>
<td>( y_i \in \mathbb{Q} )</td>
<td></td>
</tr>
<tr>
<td>( x_j \geq 0 )</td>
<td>( j )-th constraint ( \geq )</td>
<td></td>
</tr>
<tr>
<td>( x_j \leq 0 )</td>
<td>( j )-th constraint ( \leq )</td>
<td></td>
</tr>
<tr>
<td>( x_j \in \mathbb{Q} )</td>
<td>( j )-th constraint ( = )</td>
<td></td>
</tr>
</tbody>
</table>

A minimisation problem problem can be turned into a maximisation problem by changing the sign of the objective function. Hence, we can compute the dual of the dual when the dual is phrased as a maximisation problem. It is then an easy observation that the dual of the dual equals the primal!

2.4. Applications

2.4.1. Flows in networks. A network \((V, E, s, t, w)\) consists of

- a set of nodes \( V \);
- a set of directed edges \( E \subseteq V^2 \);
- a source \( s \in V \) (there are no outgoing edges \((s, u) \in E\));
- a sink \( t \in V \) (there are no incoming edges \((u, t) \in E\));
- a non-negative capacity function \( c: E \to \mathbb{Q}_{\geq 0} \).

**Definition 2.4.1 (Flows).** A flow in a network \((V, E, s, t, c)\) is a non-negative function

\[
 f: E \to \mathbb{R}_{\geq 0}
\]

such that for every \( v \in V \setminus \{s, t\} \):

\[
 \sum_{(u,v) \in E} f(u,v) = \sum_{(v,u) \in E} f(v,u) \quad \text{('What flows in needs to flow out')}
\]

A flow is admissible if \( f(u,v) \leq c(u,v) \) for every \((u, v) \in E\). If \( f \) is a flow and \( U \subseteq V \setminus \{s, t\} \), then summing over the elements in \( U \) yields

\[
 \sum_{(u,v) \in E, u \notin U, v \in U} f(u,v) = \sum_{(v,u) \in E, v \in U, u \notin U} f(v,u) \quad \text{('Flow preservation').}
\]
Choosing $U = V \setminus \{s, t\}$ we obtain in particular that what leaves the source is what enters the sink:

$$\sum_{(s,u) \in E} f(s,u) = \sum_{(u,t) \in E} f(u,t).$$

This amount is written $||f||$ and called the strength of the flow $f$.

**Definition 2.4.2.** A cut in $N$ is a set $S \subseteq E$ such that the directed graph $(V, E \setminus S)$ does not have a directed walk from $s$ to $t$.

In other words, $S$ is a cut if every walk from the source to the sink contains at least one edge from $S$. The capacity of a cut $S$ is defined as

$$c(S) := \sum_{e \in S} c(e).$$

**Lemma 2.4.3.** Let $S$ be a cut in a finite network $(V, E, s, t, w)$ and $f$ an admissible flow, then $||f|| \leq c(S)$.

**Theorem 2.4.4 (Ford and Fulkerson; Max Flow = Min Cut).** Let $(V, E, s, t, w)$ be a finite network. Then

$$\max_{f \text{ admissible flow}} ||f|| = \min_{S \text{ cut}} c(S)$$

In words: the strength of the biggest admissible flow equals the capacity of the smallest cut.

**Exercises.**

(8) Write the max-flow problem as an ILP.

(9) Write the min-cut problem as an ILP.

(10) Show that the LP for the max-flow problem is the dual of the LP relaxation of an LP for the min-cut problem.

The translation of the max-flow problem into linear programming is very robust in the sense that it can be adapted to also capture generalisations of the flow problem, for instance the generalisation where each edge $e \in E$ does not only have a capacity $c(e)$, but also a payoff $p(e)$; the payoff of the flow is then defined to be $\sum_{e \in E} p(e)f(e)$. We want to find a flow with maximum payoff (rather than a flow with maximum strength $\sum_{e \in E, s \in E} f(e)$). The LP program for the flow problem can be easily adapted to this problem.
2.4.2. The easychair problem. Suppose you are the chair of a scientific conference with peer reviewed submissions of papers; you are leading a program committee whose task is to select 80 papers from the submissions that will be admitted for presentation at the conference. Suppose that 400 papers have been submitted. Each paper will be assigned to at least 3 program committee (PC) members for peer review. Your program committee consists of 60 experts, so that each expert has to write 20 reports (all these numbers are quite realistic). The PC members can select for each paper one of the following responses.

0. No, I don’t want to review this paper (I don’t feel qualified).
1. I might review this paper.
2. Yes, I would like to review this paper.

(This is how things actually happen e.g. within the easychair system).

To turn this into a flow problem we create the following network $N$ where each edge has a payoff value as described at the end of the previous section (taken from [http://corner.mimuw.edu.pl/?p=811](http://corner.mimuw.edu.pl/?p=811)). Besides the source $s$ and the sink $t$ we have a node for each paper and for each PC member. The edges in $E$ are defined as follows.

- The source $s$ is connected with each PC member with an edge of capacity 20 and payoff 0;
- Each PC member is connected with each paper with an edge of capacity 1 and payoff 1 for NO, payoff 10000 for Maybe, and payoff 10001 for Yes.
- Each paper is connected with the sink $t$ with an edge of capacity 3 and payoff 0.

We are interested in a flow with maximum payoff (again, see the remarks at the end of the previous section for the variant of the maximum flow problem with payoffs). An integral flow of size 1 from a PC member to a paper means that the PC member has to write a report for the paper. They payoffs are chosen so that maximum flow assigns as many papers as possible to PC members that chose ‘Yes’ or ‘Maybe’ for that paper. If there are several flows that are equally good with respect to this condition, it prefers flows that have more papers assigned to PC member that chose ‘Yes’ rather than ‘Maybe’.

Note that just optimising the flow for the (global) payoff can lead to very unfair assignments: some PC members might receive many papers that were labelled by No, while others have none. This can be addressed as well; we refer to the discussion in [http://corner.mimuw.edu.pl/?p=811](http://corner.mimuw.edu.pl/?p=811).

2.4.3. Von Neumann minimax theorem. A zero sum game is a game with two players in which each player has a finite set of strategies. The payoff to the first player is determined by the strategies chosen by both players. The payoff to the second player is the negation of the payoff to the first, so the sum of their payoffs is zero. The following Paper-Scissor-Stone game is a zero sum game.

<table>
<thead>
<tr>
<th>Column Player</th>
<th>Paper</th>
<th>Scissor</th>
<th>Stone</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row Player</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Paper</td>
<td>0</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>Scissor</td>
<td>1</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>Stone</td>
<td>-1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

If the row player plays strategy $i$, and the column player plays strategy $j$, the payoff to the column player is $a_{i,j}$. If the row player plays first, she can obtain the profit

$$\max_i \min_j a_{i,j}.$$
If the row player plays last, she can obtain the profit
\[ \min_j \max_i a_{i,j}. \]
In our concrete game Paper-Scissor-Stone, we have that for all strategies \( j \)
\[ \max_i a_{i,j} = 1 \]
and for all strategies \( i \) we have
\[ \min_j a_{i,j} = -1 \]
So it is a big advantage to play second in the above game.

Now we change the game. Each of the players has to expose a probability dis-
\[ \{ x \in \mathbb{Q}^3 \mid \alpha \geq 0, \sum x_i = 1 \} \]
tribution \( \Delta \) on the strategies; these are called mixed strategies. Is it still an advantage to play second in the game? If the row player plays
\[ P_1 := \max_{x \in \Delta} \min_{y \in \Delta} \sum_{i,j} x_i y_j a_{i,j}. \]
If the row player plays second, her profit is
\[ P_2 := \min_{x \in \Delta} \max_{y \in \Delta} \sum_{i,j} x_i y_j a_{i,j}. \]
We will show that \( P_1 = P_2 \). As we will see, \( P_1 \leq P_2 \) is the easy direction. For compact notation, we write \( A = (a_{i,j}) \) for the payoff matrix, so that the expected payoff \( \sum_{i,j} x_i y_j a_{i,j} \) can be written as \( x^\top A y \).

**Theorem 2.4.5 (Von Neumann Min-Max Principle).** Let \( A = (a_{i,j}) \in \mathbb{Q}^{m \times n} \) be a two-person zero-sum game. Let \( \Gamma, \Delta \) be the set of all mixed strategies for row and column player, respectively. Then there are \( \tilde{x} \in \Gamma \) and \( \tilde{y} \in \Delta \) such that
\[ \max_{x \in \Gamma} \min_{y \in \Delta} x^\top A y = \min_{y \in \Delta} \max_{x \in \Gamma} x^\top A y = \tilde{x}^\top A \tilde{y}. \]

The following terminology is important and will help us to give a clear presenta-
tion of the proof. For simplicity, we call player one Alice and player two Bob. The worst-case payoff for a mixed strategy \( x \in \Gamma \) for Alice is defined to be
\[ \alpha(x) := \min_{y \in \Delta} x^\top A y \]
and likewise the worst-case payoff for a mixed strategy \( y \in \Delta \) for Bob is
\[ \beta(y) := \min_{x \in \Gamma} x^\top A y. \]
These are well-defined functions: if we would work over \( \mathbb{R} \) instead of \( \mathbb{Q} \), then \( \Delta \) would be a compact set and hence the minima are attained. It can be shown (e.g. by tools from logic using the fact that the structure \( (\mathbb{Q}; +, \leq, \cdot, 1) \) has ‘quantifier elimination’) that the minima are also attained when working over \( \mathbb{Q} \) instead of \( \mathbb{R} \).

A pair \( (\tilde{x}, \tilde{y}) \) such that
\[ \alpha(\tilde{x}) = \tilde{x} A \tilde{y} = \beta(\tilde{y}) \]
is called a mixed Nash equilibrium. Alice’s mixed strategy \( \tilde{x} \) is called worst-case optimal if \( \alpha(\tilde{x}) = \max_{x \in \Gamma} \alpha(x) \), and we make the analogous definition for a mixed strategy of Bob.

**Proof.** We first show how worst-case optimal mixed strategies \( \tilde{x} \) for Alice and \( \tilde{y} \) for Bob can be found by linear programming. Then we prove that \( \alpha(\tilde{x}) = \beta(\tilde{y}) \) holds.

First notice that Bob’s best response to a fixed mixed strategy \( x \) of Alice can be found by solving a linear program. That is, \( \alpha(x) \), with \( x \) a concrete vector of
$m$ numbers, is the optimal value of the following linear program in the variables $y_1, \ldots, y_n$:

$$\text{Minimize } x^\top A y$$

subject to $\sum_{j=1}^{n} y_j = 1$

$y \geq 0$

Unfortunately, $\alpha(x)$ is not a linear function, so we cannot directly formulate the maximisation of $\alpha(x)$ as a linear program. Fortunately, we can circumvent this issue by using LP duality. The dual of the above LP is

$$\text{Maximize } x_0$$

subject to $A^\top x - 1 x_0 \geq 1$

By the duality theorem, the optimal value of the dual LP equals $\alpha(x)$. In order to maximise $\alpha(x)$ over all mixed strategies $x$ of Alice, we derive a new LP from the dual in which $x_1, \ldots, x_m$ are now regarded as variables.

$$\text{Maximize } x_0$$

subject to $A^\top x - 1 x_0 \geq 1$ \hspace{1cm} (12)

$$\sum_{i=1}^{m} x_i = 1$$

$x \geq 0$

Clearly, there exist feasible solutions to this LP. If $(\tilde{x}_0, \tilde{x})$ denotes an optimal solution, we have by construction that

$$\tilde{x}_0 = \alpha(\tilde{x}) = \max_{x \in \Gamma} \alpha(x).$$

Symmetrically, we can construct an LP for computing a worst-case optimal mixed strategy $\tilde{y}$ for Bob:

$$\text{Minimize } y_0$$

subject to $A^\top y - 1 y_0 \geq 1$ \hspace{1cm} (13)

$$\sum_{j=1}^{n} y_j = 1$$

$y \geq 0$

Now observe that the two linear programs (12) and (13) are dual to each other! This concludes the proof. $\square$

The corresponding problem for two-player (or $n$-player) games that are not necessarily zero-sum is a very interesting problem which is not known to be in P, but believed to not be NP-hard (since if it were NP-hard, then NP=coNP). On the other hand, there is also some evidence that the problem might not be in P; see [3] and the references therein.

**Exercises.**

(11) Find a mixed Nash equilibrium for the game “Papers-Scissors-Stone-Well” which is the modification of Papers-Scissors-Stone where an additional pure strategy “Well” has been added, which wins against Stone and Scissors, but loses against paper.
2.4.4. Simple stochastic games. A simple stochastic game are a special case of stochastic games as introduced by Shapley in 1953 (a grad school friend of Nash from the previous section). They are played on a directed graph $G = (V, E)$ whose vertices are partitioned into three disjoint sets $V_{\text{max}}, V_{\text{min}}, V_{\text{stoch}}$, whose elements are called max-, min-, and stochastic vertices, respectively. Moreover, there is a distinguished start vertex $s$, and two distinguished terminal vertices $t_{\text{max}}$ and $t_{\text{min}}$. Each vertex has at least one outgoing edge, except for $t_1$ and $t_2$, which have no outgoing edge (they are sinks).

The game is played by two players, called the max player and the min player. At the start of the game, a token is placed on the start vertex. In each round, the token is moved from a vertex $v$ along some of the outgoing edges at $v$. When the token is positioned on a max vertex, then player max decides along which edge the token is moved, and when the token is on a min vertex then player min decides. When the token is on a stochastic vertex, then each outgoing edge is chosen with equal probability (so in some sense there is a third player, randomness; for this reason, this type of game is also called a $2\frac{1}{2}$-player game). The game ends when the token reaches $t_{\text{max}}$ or $t_{\text{min}}$; in the first case, player max wins, and in the second case, player min wins. If the play continues forever then player min wins. See Figure 2.3 for an example.

A (positional) strategy $\sigma: V_{\text{max}} \to E$ for player max is a function that selects for each vertex $u \in V_{\text{max}}$ one outgoing edge $(u, w)$. Strategies $\tau: V_{\text{min}} \to E$ for player min are defined analogously. It can be shown that if a player can win, it can win following such a positional strategy (see [2]). The value $v_{\sigma, \tau}(u)$ of $u \in V$ with respect to $\sigma$ and $\tau$ is the probability that the pebble reaches $t$ starting from $u$. The optimal value $v(u)$ of a vertex $u \in V$ is defined to be

$$\max_{\sigma} \min_{\tau} v_{\sigma, \tau}(u).$$

The value of the game is defined to be $v(s)$. The primary question about a simple stochastic game is the question: what is its value?

![Figure 2.3. An example of a simple stochastic game with the value of each vertex in red.](image)

There is no polynomial-time algorithm known that solves this problem. In particular, we will be interested in the decision problem whether the value of the game...
is at least 1/2 (i.e., we decide whether min has a greater winning probability than max if min plays optimally). If the game contains no max vertices, then an optimal strategy for player max can be found by linear programming.

**Definition 2.4.6.** An assignment \( \tilde{x} \) that satisfies

- \( \tilde{x}(t_{\text{max}}) = 1, \tilde{x}(t_{\text{min}}) = 0; \)
- \( 0 \leq \tilde{x}(u) \leq 1 \) for all \( u \in V \)
- \( \tilde{x}_u = \min_{(u,w) \in E} \tilde{x}_w \) for \( u \in V_{\text{min}} \) out-neighbours \( w_1, \ldots, w_k \) and
- \( \tilde{x}_u = \frac{\tilde{x}_{w_1} + \cdots + \tilde{x}_{w_k}}{k} \) for \( u \in V_{\text{stoch}} \) with out-neighbours \( w_1, \ldots, w_k \)

is called a solution to the game.

SGGs might not have unique solutions in general. However, it can be shown that the general case can be (efficiently) reduced to the case where solutions are unique; this is technical and out of the scope of this text. We refer to [2] for details. If a game has a unique solution, then the solution can be found by the following linear program.

**Theorem 2.4.7.** The LP above has an optimal solution, \( (x^*_u)_{u \in V} \), and \( x^*_u = v(u) \) as defined above.

**Proof.** It is clear that \( x_u := v(u) \), for all \( u \in V \), is a solution to the game. It is also clear that every solution to the game gives a valid solution to the LP. Since all variables are upper bounded by 1 and the objective is to maximize \( \sum_{u \in V} x_u \) it follows that the LP has an optimal solution \( x^* \).

We claim that every optimal solution to the LP gives a solution to the game. Suppose otherwise that for some \( u \in V_{\text{max}} \) we have \( x^*_u < x^*_w \) for all out-neighbours \( w \) of \( u \). Then we construct a better solution \( x' \) to the LP as follows: \( x'(u) := \min_{(u,w) \in E} (x^*_w) \) and \( x'(w) := x^*(w) \) for all \( w \in V \setminus \{ u \} \). Then the new solution satisfies all the constraints but has a strictly larger objective function, contradicting the maximality of \( x^* \). Now suppose for contradiction that for some \( u \in V_{\text{stoch}} \) with out-neighbours \( w_1, \ldots, w_k \) we have \( x^*_u < \frac{x^*_{w_1} + \cdots + x^*_{w_k}}{k} \). Then similarly as above we can construct a solution with a strictly larger objective function. It follows that if the game has a unique solution, then \( x^* = v(u) \).

This shows that deciding whether the value of a game is larger than a given threshold is in the complexity class NP: we simply guess an outgoing edge for each \( v \in V_{\text{max}} \), remove all other outgoing edges from \( v \), and turn \( v \) into a min vertex. This corresponds to selecting one strategy for min. The resulting game has no more max vertex and we can find an optimal counter-strategy for min by the LP above. We now want to argue that the problem is also in coNP.

The following result does not follow from the Minimax theorem since the players have to play pure strategies.
Theorem 2.4.8 (of [8] and [2]). Let $G$ be a simple stochastic game. Then we have
\[ \max_{\sigma} \min_{\tau} v_{\sigma,\tau}(s) = \min_{\tau} \max_{\sigma} v_{\sigma,\tau}(s). \]

This result suggests that in order to compute $v(s)$ we can swap the roles of the players. A very similar linear program works in the case that the game has no min-nodes:

\[
\text{minimize } \sum_{u \in V} x_u \\
\text{subject to } x_u \geq x_w \quad \text{if } u \in V_{\text{max}} \text{ and } (u, w) \in E \\
x_u \geq \frac{x_{w_1} + \cdots + x_{w_k}}{k} \quad \text{if } u \in V_{\text{stoch}} \\
\quad \quad \quad \text{and } w_1, \ldots, w_k \text{ are the out-neighbours of } u \\
x_u \geq 0 \\
x_{t_{\text{max}}} = 1 \\
x_{t_{\text{min}}} = 0
\]

This shows that deciding the winner in a simple stochastic game is in $\text{NP} \cap \text{coNP}$ (a result of Condon [2]).
Bibliography


APPENDIX A

What every mathematician needs to know about complexity theory

For a set $A$, we write $A^*$ for the set of all words over the alphabet $A$. A word over $A$ can be seen as a function from $\{1, \ldots, n\} \to A$, for some $n \in \mathbb{N}$. We write $\epsilon$ for the empty word (i.e., for the function with the empty domain).

The most classical setting of complexity theory is the study of the computational complexity of functions $f$ from $\{0, 1\}^* \to \{0, 1\}$. Alternatively, we may view $f$ as a set of words, namely that set of words $w$ such that $f(w) = 1$; such sets are also called formal languages. There are several mathematically rigorous machine models to formalise the set of such functions that are computable or efficiently computable. The first insight is that most of these machine models lead to the same, or to closely related classes of functions. Complexity theory maps out the landscape of the resulting classes of functions. Typically the first machine model that is introduced in introductory courses are Turing machines. They strike a good balance between the following two requirements that a theoretician has for those machine models:

- It should be relatively simple, so that it is easy to show that it can be simulated by many other machine models.
- It should be relatively easy to program within the machine model, so that it is easy to show that it can simulate many other machine models.

Turing machines are simple, but still the definition does not easily fit into a few lines. On the other hand, today academics are most likely to already have a very good idea of what a computer program can do (in polynomially many steps); and this coincides with what a Turing machine $M$ can do (in polynomially many computational steps). So we will pretend in the following that the reader already knows what Turing machines $M$ are. Every Turing machine describes a formal language, namely the function $f: \{0, 1\}^* \to \{0, 1\}$ such that $f(w) = 1$ if and only if when running the machine on input $w$ it terminates and accepts. We also say that $M$ computes $f(w)$, and we then sometimes write $M(f)$ instead of $f(w)$. In complexity theory we are interested in the number of computation steps that $M$ needs to perform to compute $f(w)$ (which corresponds to computation time). For example, we say that a Turing machine runs in polynomial time if the number of computation steps is in $O(|w|^k)$ for some $k \in \mathbb{N}$. The class of such functions is denoted by $P$.

In the combinatorics course we have met computational complexity for example in the section about colorability. We mentioned that 2-colourability is in $P$ and that $k$-colourability is NP-hard. But these were problems about finite graphs, whereas in the above we only treated formal languages. But this is just a matter of coding. TODO. Discuss #.

The second most important complexity class is NP.

Definition A.0.1. NP (for nondeterministic polynomial time) stands for the class of all functions $f: \{0, 1\}^* \to \{0, 1\}$ such that there exists a polynomial-time
Turing machine $M$ such that for every $w \in \{0,1\}^*$ there exists a $d \in \{0,1\}^*$ such that $f(w) = M(w\#d)$.

Define the class of all graph classes in NP. Unlike the class P, it is possible to define the class of all graph classes in NP transparently and fully formally in a few lines.

**Theorem A.0.2 (Fagin).** A class of finite graphs $C$ is in NP if and only if there exists an existential second-order sentence $\Phi$ such that for every finite graph $G$ we have

$$G \in C \text{ if and only if } G \models \Phi.$$