Elementare Angewandte Mathematik

Götz Kersting, WS 2007/08

Vorlesungsskript für L1, L2, L5-Studierende

Inhaltsverzeichnis

1 Bäume **3**
1.1 Binärbäume 3
1.2 Suchbäume 5
1.3 Huffman-Codes 9

2 Graphen **15**
2.1 Kreise in Graphen 15
2.2 Das Problem des Handlungsreisenden 17
2.3 Planare Graphen 22

3 Fehlerkorrigierende Codes **27**

4 Kryptographie **32**
4.1 Modulares Rechnen 32
4.2 Öffentliche Chiffriersysteme 36

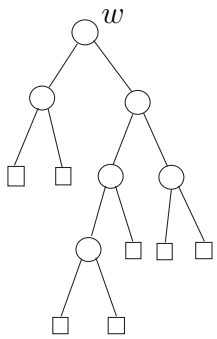
5 Zufall **40**
5.1 Wahrscheinlichkeiten 40
5.2 Variabilität 47
5.3 Kann das Zufall sein? Statistische Tests 52

1 Bäume

Bäume sind spezielle Graphen, die für praktische Zwecke besonders wichtig sind. Bei ihrer Untersuchung greifen wir auch auf Logarithmen zurück.

1.1 Binärbäume

Unter einem *Baum* verstehen wir hier einen Graphen der folgenden Gestalt.



Er besteht aus endlich vielen *Knoten*, die durch *Kanten* verbunden sind. Man spricht von einem Baum, weil sich keine geschlossenen ‚Rundwege' (‚Kreise') vorfinden. Die Knoten werden unterschieden in *interne Knoten* (dargestellt als Kreise), und *externe Knoten* (dargestellt als Quadrate). Der Knoten w an der Spitze heißt *Wurzel*. Die externen Knoten nennt man auch *Blätter*. Aus ihnen wachsen keine weiteren Knoten. Da aus jedem internen Knoten nach unten höchstens 2 Kanten herausgehen, spricht man von einem *Binärbaum*. Wir betrachten in diesem Kapitel (wie im Bild) nur den Fall von genau 2 Kanten, den Fall von *vollen Binärbäumen*. Jeder Knoten k besitzt eine *Tiefe* $\ell(k)$, das ist die Anzahl der Kanten zwischen k und der Wurzel. Die Maximaltiefe der Knoten

$$h := \max_{k \in K} \ell(k)$$

heißt *Höhe* des Binärbaumes (K bezeichnet die Menge der Knoten). Im Beispiel ist $h = 4$.

Bemerkung. Da in der Graphentheorie auch Bäume ohne Wurzel betrachtet werden, wäre es genauer, hier von binären *Wurzelbäumen* zu sprechen. Wir behandeln in diesem Kapitel nur Bäume mit Wurzel.

Sei

$$m := \text{Anzahl der Blätter}\ , \quad n := \text{Anzahl der internen Knoten}$$

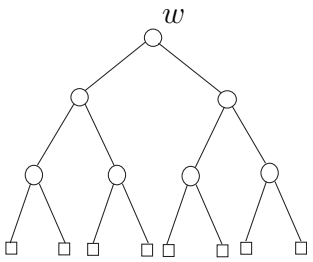
(im Beispiel $m = 7, n = 6$).

Satz. *Für jeden vollen Binärbaum gilt*

$$m = n + 1 \; . \tag{1}$$

Beweis. Stellen wir uns vor, dass man einen vollen Binärbaum aus seiner Wurzel (oder irgend einem Knoten) heraus Schritt für Schritt ‚wachsen' lässt. Bei diesem Prozess verwandelt sich in jedem Schritt ein Blatt in einen internen Knoten, gleichzeitig wachsen aus diesem neuen internen Knoten zwei neue Blätter. In der Bilanz vergrößert sich die Anzahl der internen Knoten um 1, und genauso die Anzahl der Blätter um $2 - 1 = 1$. Die Differenz $n - m$ bleibt also unverändert. Sie beträgt am Anfang 1, dann ist die Wurzel externer Knoten, und $n - m = 1 - 0 = 1$. Daher gilt also, wie behauptet, immer $n - m = 1$. □

Das folgende Bild zeigt einen *vollständigen Binärbaum* der Höhe $h = 3$.



Er ist dadurch ausgezeichnet, dass alle Ebenen einer vorgegebenen Tiefe, bis zur letzten Ebene, vollständig mit Knoten aufgefüllt ist. Die Blätter haben alle Tiefe h, und die internen Knoten Tiefen kleiner als h.

Die Gesamtzahl der Knoten ist leicht ermittelt. Von einer Ebene zur nächsten verdoppelt sich die Knotenzahl, sie wächst also geometrisch, 1, 2, 4, 8 .... In der Tiefe t finden sich folglich 2^t Knoten. Die Anzahl der Blätter ist folglich $m = 2^h$ und nach (1) die Anzahl der internen Knoten $n = 2^h - 1$. Genausogut kann man die internen Knoten ebenenweise aufzusummieren und ihre Gesamtzahl als

$$n = 1 + 2 + 4 + \cdots + 2^{h-1}$$

bestimmen. Wir erhalten also mit unseren Ansatz nebenbei die Formel

$$1 + 2 + 4 + \cdots + 2^{h-1} = 2^h - 1 \; .$$

Man kann sie natürlich auch direkt beweisen, ohne binäre Bäume, etwa durch Induktion oder auch auf folgende Weise:

$$\begin{aligned} 1 + 2 + 4 + \cdots + 2^{h-1} &= (1 + 2 + 4 + \cdots + 2^{h-1}) \cdot (2 - 1) \\ &= (2 + 4 + 8 + \cdots + 2^h) - (1 + 2 + 4 + \cdots + 2^{h-1}) = 2^h - 1 \; . \end{aligned}$$

Satz. (Fano-Kraft) *Sei B die Menge der Blätter in einem vollen Binärbaum. Dann gilt*

$$\sum_{k\in B} 2^{-\ell(k)} = 1 \ . \tag{2}$$

Beweis. Stellen wir und vor, dass wir, ausgehend von der Wurzel, zufällig durch den Baum wandern, bis wir in einem Blatt landen. Der Weg wird durch Münzwurf gewählt, damit entscheiden wir, ob wir die rechte oder linke Kante nach unten laufen. Um zum Blatt k der Tiefe $\ell(k)$ zu gelangen, sind $\ell(k)$ Münzwürfe erforderlich, jedesmal mit dem „richtigen“ Resultat. Dies geschieht mit Wahrscheinlichkeit

$$\underbrace{\frac{1}{2}\cdot\frac{1}{2}\cdots\frac{1}{2}}_{\ell(k)\text{-mal}} = 2^{-\ell(k)} \ .$$

und diese Wahrscheinlichkeiten summieren sich zu 1 auf, da wir mit Wahrscheinlichkeit 1 in irgend einem Blatt landen. □

1.2 Suchbäume

Aus der Informatik stammt folgendes Problem: n Nummern $a_1, \ldots, a_n$ sind so in einem Rechner abzuspeichern, dass man sie schnell wiederfinden kann. Man kann etwa an die Matrikelnummern von Studierenden denken und sich vorstellen, dass man am Speicherort mit einer Nummer auch direkten Zugriff zu anderen Daten der Person erhält.

Uns interessieren hier nicht die Details der Programmierung von Computern sondern eine idealisierte Situation. Wir betrachten binäre Bäume, an dessen Knoten die Nummern abgelegt sind, zusammen mit einer Suchstrategie, bei der die Suchzeit für eine Nummer (grob) der Tiefe des entsprechenden Knotens entspricht. Aus Gründen, die später deutlich werden, sehen wir nur interne Knoten als Plätze für die Nummern vor.

Idealerweise könnte man versuchen, die Nummern in einem *vollständigen* Binärbaum der Höhe h unterzubringen. Er hat viel Platz: Wie wir gesehen haben, hat er $n = 2^h - 1$ interne Knoten. So lassen sich in einem Baum der Höhe 20 immerhin schon $2^{20} - 1 = 1.048.575$ Nummern unterbringen, und für jede Nummer beträgt dann die Suchzeit höchstens 19. Die mittlere Tiefe eines internen Knotens im vollständigen Binärbaum ist, notieren wir mit I die Menge aller internen Knoten, gegeben durch

$$\frac{1}{n}\sum_{i\in I} t(i) = \frac{1}{n}\sum_{t=0}^{h-1} t\cdot 2^t \ .$$

Sie errechnet sich (etwa per Induktion) als

$$h - 2 + \frac{h}{n}$$

und unterscheidet sich von der maximalen Tiefe $h - 1$ interner Knoten um weniger als 1.

Vollständige Bäume sind aber nur als Ideal nützlich. Es bleibt ungeklärt, wie sich darin die Nummern Einsortieren lassen, so dass man sie schnell wiederfindet.

Wir betrachten nun *volle* binäre Bäume als Sortierschema. Seien $a_1, \ldots, a_n$ paarweise verschiedene Zahlen. Der zugehörige Suchbaum wird schrittweise nach einem Verfahren aufgebaut, bei dem die größeren Zahlen weiter rechts und die kleineren Zahlen weiter links untergebracht sind. Dann kann man offenbar eine gesuchte Zahl schneller wiederfinden. Dies erfordert, dass man zwischen Kanten unterscheidet, die von einem Knoten entweder nach rechts unten oder nach links unten abzweigen. Die Bilder machen dies evident (der Fachausdruck in der Mathematik ist *geordneter* Baum oder *planarer* Baum).

Genauer geht man folgendermaßen vor:

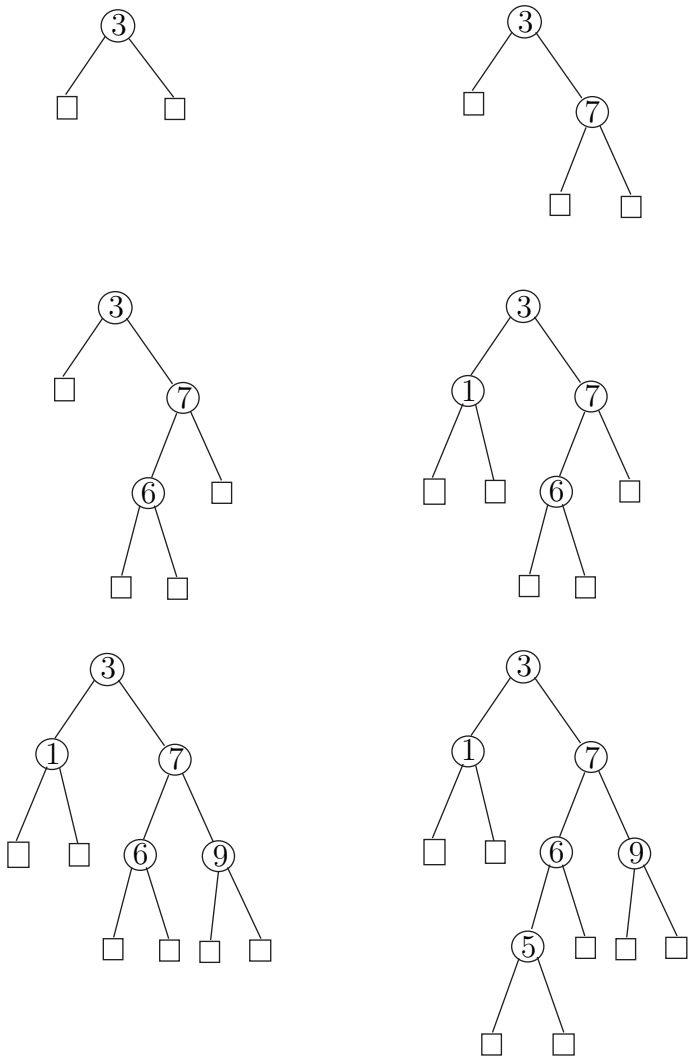
1. Plaziere a_1 an der Wurzel. Ergänze diesen Knoten in Tiefe 1 mit zwei Blättern. Sie sind mögliche Plätze für a_2.

2. Gilt $a_2 > a_1$, so gehe von der Wurzel nach rechts unten, lege a_2 in dem erreichten Blatt ab und verwandle dies zu einem neuen internen Knoten, indem man am Knoten nach unten zwei neue Blätter der Tiefe 2 anfügt. Gilt $a_2 < a_1$, so gehe stattdessen nach links unten und verfahre analog.

3. Allgemeiner: Sind schon $a_1, \ldots, a_{j-1}$ im Baum untergebracht, so durchlaufe den Baum von der Wurzel aus nach unten, bis ein Blatt erreicht ist. An jedem besuchten internen Knoten vergleiche a_j mit der dort abgelegten Zahl a. Im Fall $a_j > a$ setzte den Weg nach rechts unten fort, im Fall $a_j < a$ nach links unten. Lege a_j im erreichten Blatt ab und mache es zum internen Knoten, indem man es nach unten durch zwei Blätter ergänzt.

Die folgenden Bilder veranschaulichen den Prozess für die Zahlen $3, 7, 6, 1, 9, 5$ (in dieser Reihenfolge!).

Man bemerke, dass die gefundenen Bäume davon abhängen, in welcher Reihenfolge die Zahlen stehen. Jeder interne Knoten ist mit einer Zahl versehen, die Blätter sind freie Plätze, in denen nachkommende Zahlen einsortiert werden können.



Die Suche nach einer Zahl a funktioniert wie das Einsortieren. Man vergleicht a erst an der Wurzel mit a_1 und geht nach rechts oder links in die Tiefe, je nachdem ob $a > a_1$ oder $a < a_1$ gilt. So stößt man schließlich auf a (oder aber man stellt fest, dass sie gar nicht im Baum gespeichert ist).

Offenbar sind diese Suchbäume im Allgemeinen nicht mehr vollständige sondern nur noch volle binäre Bäume. Eine interessante Frage ist es, die Bäume mit „möglichst vollständigen“ Bäumen der Höhe h zu vergleichen, die dieselbe Anzahl n von inneren Knoten hat. Die Blätter eines solchen Baumes haben alle die Tiefe h oder $h-1$, also gilt $2^{h-1} < m \leq 2^h$ bzw.

$$h - 1 < \log_2 m \leq h \ .$$

$\log_2 m$ ist also die Messlatte.

Mit diesem Wert wollen wir die Höhe von Suchbäumen vergleichen. Dazu machen wir die *Annahme*, dass die Zahlen eine zufällige Reihenfolge besitzen. Wenn bereits $k-1$ Zahlen im Baum einsortiert sind, besitzt er $k-1$ interne Knoten und k Blätter. Genauso gibt es zwischen den $k-1$ Zahlen, auch rechts und links von ihnen, k Positionen, in denen sich eine neue Zahl einordnen kann. Die Blätter und die Positionen entsprechen einander. Wir nehmen nun an, dass a_k jede dieser k Positionen mit derselben Wahrscheinlichkeit einnimmt. (Man darf sich das auch so vorstellen, dass jede Reihenfolge von $a_1, \ldots, a_n$ dieselbe Wahrscheinlichkeit hat.) Im k-ten Schritt wird also jedes Blatt mit derselben Chance zum internen Knoten.

Wir haben es nun also mit zufälligen Suchbäumen zu tun. Zu seiner Beurteilung betrachten wir

$$e_m \;:=\; \text{mittlere Tiefe eines Blattes im Suchbaum mit } m \text{ Blättern}$$

Satz. $e_m = 2\Big(\frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{m}\Big)$ *für* $m \geq 2$.

Beweis. Wir betrachten erst einen festen vollen Binärbaum mit $k-1$ Blättern, die die Tiefen $t_1, \ldots, t_{k-1}$ haben. Seine mittlere Tiefe ist dann

$$\frac{t_1 + \cdots + t_{k-1}}{k-1}.$$

Hängen wir an das i-te Blatt eine Gabel mit zwei neuen Blättern, so haben beide die Tiefe t_i+1, während ein Blatt der Tiefe t_i verschwindet. Die mittlere Blatttiefe im neuen Baum ist nun

$$\frac{t_1 + \cdots + t_{i-1} + t_{i+1} + \cdots + t_{k-1} + 2(t_i+2)}{k} = \frac{t_1 + \cdots + t_{k-1}}{k} + \frac{t_i}{k} + \frac{2}{k}$$

Da über $i = 1, \ldots, k-1$ gemittelt wird, erhalten wir als mittlere Tiefe

$$\frac{t_1 + \cdots + t_{k-1}}{k} + \frac{\frac{t_1 + \cdots + t_{k-1}}{k-1}}{k} + \frac{2}{k} = \frac{t_1 + \cdots + t_{k-1}}{k-1} + \frac{2}{k}$$

Die mittlere Tiefe vergrößert sich also um $2/k$. Dies Resultat überträgt sich offenbar auf zufällige Suchbäume, es folgt also $e_k = e_{k-1} + 2/k$. Durch Iteration folgt

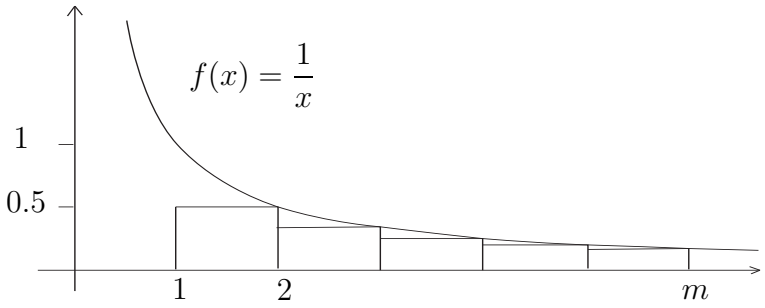
$$e_m = e_{m-1} + \frac{2}{m} = e_{m-2} + \frac{2}{m-1} + \frac{2}{m} = \cdots = e_2 + \frac{2}{3} + \cdots + \frac{2}{m}\ .$$

Wegen $e_2 = 1$ folgt die Behauptung. □

Weiter gilt

$$\frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{m} \leq \int_1^m \frac{1}{x}\, dx = \ln m \ ,$$

wie man dem folgenden Bild entnimmt.



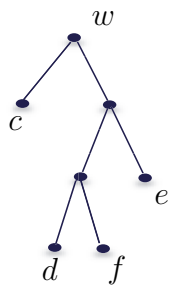
Außerdem gilt $\ln m = \ln 2 \log_2 m = 0.69 \log_2 m$. Es folgt

Korollar. $e_m \leq 1.4 \log_2 m$.

Zufällige Suchbäume haben also im Vergleich zu vollständigen Binärbäumen eine nur um den Faktor 1.4 größere mittlere Tiefe. Dies belegt die Effizienz des beschriebenen Verfahrens.

1.3 Huffman-Codes

Jetzt betrachten wir endliche Mengen $\mathcal{A}$ und zugehörige Binärbäume, wobei jedem Element $a \in \mathcal{A}$ ein Blatt zugewiesen sei. Zum Beispiel kann man für $\mathcal{A} = \{c, d, e, f\}$ den folgenden Baum betrachten.



Der Weg von der Wurzel nach $a \in \mathcal{A}$ verläuft mal nach rechts, mal nach links, was sich in eine endliche Folge $k(a)$ von 1en und 0en übersetzt. Im Bild gilt $k(c) = 0, k(d) = 100, k(e) = 11, k(f) = 101$. Die Länge der Folge $k(a)$

bezeichnen wir mit $\ell(a)$, sie ist gleich der Tiefe des zugehörigen Blattes. Im Beispiel gilt $\ell(c) = 1, \ell(d) = 3, \ell(e) = 2, \ell(f) = 3$.

An folgende Interpretationen kann man denken.

1. $\mathcal{A}$ steht für ein Alphabet und der Binärbaum für einen digitalen Code, der a in den 01-string $k(a)$ überträgt. Solche Codes sind grundlegend für Computer. Weil nur Blätter Buchstaben tragen, ist keine Codierung eines Buchstaben ein Anfangstück der Codierung eines anderen Buchstaben. Solche Codes heißen ‚präfixfrei', sie erlauben eindeutige Entzifferung von ganzen Wörtern.

2. Alternativ lassen sich die Bäume als Fragestrategie auffassen, um ein Element $a \in \mathcal{A}$ mit Ja-Nein-Fragen zu identifizieren. Die erste Frage lautet „Fängt $k(a)$ mit einer 1 an?", die zweite „Ist die zweite Stelle von $k(a)$ eine 1?" etc. Dann ist $\ell(a)$ die Anzahl von Fragen, um a zu bestimmen.

3. Man kann auch mit mehr als zwei Symbolen arbeiten. Beim Morsen benutzt man drei (kurz, lang, Pause), dann wären ternäre Bäume anstelle binärer zu betrachten.

Treten die Buchstaben von $\mathcal{A}$ alle mit gleicher Häufigkeit auf, so wird man danach trachten, dass die Codierungen alle gleichlang sind. Anders verhält es sich, wenn manche Buchstaben bevorzugt auftreten, andere seltener, wie dies für Sprachen zutrifft. Wir nehmen nun an, dass jeder Buchstabe a eine Wahrscheinlichkeit p_a besitzt, mit der er eintritt. Für die deutsche Sprache sind diese Wahrscheinlichkeiten ziemlich genau bekannt, z.B. hat das e eine Wahrscheinlichkeit von etwa 17%, das d von 5% und das x von weniger als 0.02%.

Zusammengefasst schreibt man $(p_a)_{a\in\mathcal{A}}$ für das System der Wahrscheinlichkeiten und nennt es eine *Wahrscheinlichkeitsverteilung* mit *Gewichten* p_a. Allgemein ist nur festzustellen, dass

$$p_a \geq 0 \ , \quad \sum_{a\in\mathcal{A}} p_a = 1$$

gilt.

Offenbar ist es sinnvoll, sich auf Codes zu beschränken, bei denen häufige Buchstaben kleine Codewörterlänge haben, für die genauer gesagt gilt:

$$p_a < p_b \quad \Rightarrow \quad \ell(a) \geq \ell(b)$$

für alle $a, b \in \mathcal{A}$. Hätte man nämlich Buchstaben a, b mit $p_a < p_b$ und $\ell(a) < \ell(b)$, so könnte man durch Vertauschen der Codewörter die Länge der Codierungen von a verlängert und die von b verkürzt, was günstig wäre.

Wir konstruieren nun optimale Codes. Damit meinen wir Präfixcodes k, die die erwartete Codewortlänge

$$E(k) := \sum_{a \in \mathcal{A}} \ell(a) p_a$$

minimieren. Sie heißen *Huffman-Codes* und werden von den Blättern hin zur Wurzel konstruiert. Sie sind von fundamentaler Bedeutung in der Informatik. Die Konstruktion von Huffman verfolgt man am besten an den Baumdarstellungen der Codes. Sie beruht auf den folgenden einfachen Feststellungen:

1. In einem optimalen Code verzweigt jeder innere Knoten (jeder Knoten, der kein Blatt ist) nach unten in *zwei* Kanten. Wäre da nur eine Kante, so könnte man sie aus dem Baum heraustrennen und erhielte damit verkürzte Codewortlängen.

2. In einem optimalen Code haben Buchstaben großer Wahrscheinlichkeit kurze Codierungen. Genauer: Aus $p_a < p_b$ folgt immer $\ell(a) \geq \ell(b)$. Andernfalls könnte man die Beschriftungen a und b im Baum vertauschen, wodurch sich die erwartete Codewortlänge um

$$\begin{aligned} &\ell(a)p_b + \ell(b)p_a - \ell(a)p_a - \ell(b)p_b \\ &\qquad = -\big(\ell(b) - \ell(a)\big)\big(p_b - p_a\big) < 0 \end{aligned}$$

   verändern würde.

3. Sind also $u, v \in \mathcal{A}$ zwei Buchstaben kleinster Wahrscheinlichkeit,

$$p_u \leq p_v \leq p_a \quad \text{für alle} \quad a \neq u, v,$$

   so folgt für einen optimalen Code

$$\ell(u) = \ell(v) \geq \ell(a) \quad \text{für alle} \quad a \neq u, v.$$

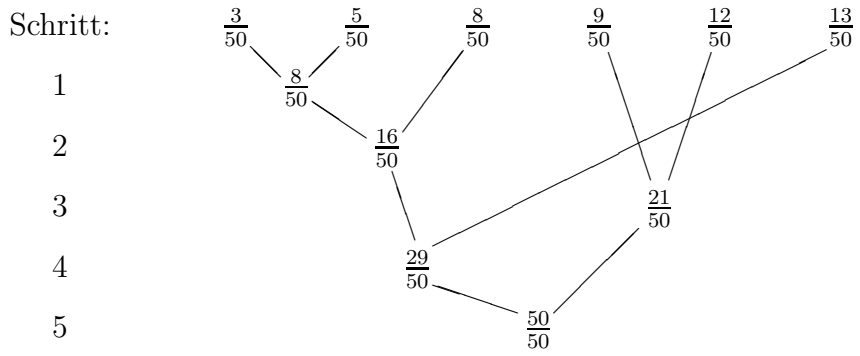
4. Damit dürfen wir nun auch annehmen, dass diese Buchstaben u und v in einem optimalen Code an derselben Gabel sitzen, d.h. dass sich die 01-Wörter $k(u)$ und $k(v)$ nur an der letzten Stelle unterscheiden. In dieser Situation kann man u, v zu einem neuen Buchstaben $\langle uv \rangle$ verschmelzen, zu dem kleineren Alphabet $\mathcal{A}' := S \cup \{\langle uv \rangle\} - \{u, v\}$ übergehen und dabei $\langle uv \rangle$ die Wahrscheinlichkeit $p_u + p_v$ zuweisen. Entsprechend kann man im Baum die u, v-Gabel beseitigen und an dem freiwerdenden Blatt den Buchstaben $\langle uv \rangle$ platzieren. Es ist offenbar: Der ursprüngliche Baum ist optimal für das ursprüngliche Alphabet, falls der reduzierte Baum für das reduzierte Alphabet optimal ist (die erwartete Wortlänge unterscheidet sich um $p_u + p_v$).

Es liegt nun auf der Hand, wir man von der Krone zur Wurzel einen optimalen Code erhält. Man verschmilzt erst die beiden Buchstaben kleinster Wahrscheinlichkeit. Man wiederholt diese Operation im reduzierten Alphabet und führt das Verfahren solange fort, bis alle Buchstaben verschmolzen sind. Nach diesem Verfahren entstehen Huffman-Codes.

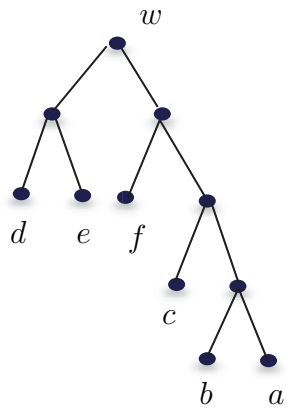
Beispiel. Wir führen das Verfahren exemplarisch für die Verteilung ρ mit den Gewichten

$$\rho(a) = \frac{3}{50},\ \rho(b) = \frac{5}{50},\ \rho(c) = \frac{8}{50},\ \rho(d) = \frac{9}{50},\ \rho(e) = \frac{12}{50},\ \rho(f) = \frac{13}{50}$$

durch. Das folgende Schema zeigt die Reduktionsschritte.



Als Codebaum erhalten wir



Die mittlere Wortlänge berechnet sich als $2,48$.

Wir schätzen nun die erwartete Codewortlänge ab. Dazu definieren wir die *Entropie* der Wahrscheinlichkeitsverteilung $p_a, a \in \mathcal{A}$, als

$$H := -\sum_{a\in\mathcal{A}} p_a \log_2 p_a.$$

Im vorigen Beispiel ist $H = 2,443$.

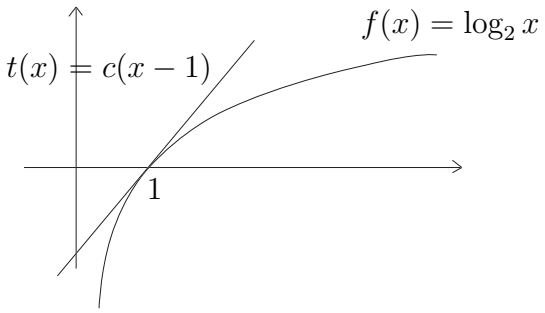
Quellencodierungssatz. *Für die erwartete Codewortlänge $E(k)$ eines Huffman-Codes k gilt*

$$H \le E(k) < H + 1.$$

Beweis. Wir zeigen nur die (einfachere) Abschätzung nach unten. Es gilt

$$H - E(k) = \sum_a p_a \log_2 \frac{2^{-\ell(a)}}{p_a} \ .$$

Nun liegt die Logarithmenfunktion unterhalb ihrer Tangenten $t(x)$ an der Stelle 1.



Folglich gilt $\log_2 x \le c(x-1)$ mit geeignetem $c > 0$, und

$$H - E(k) \le c\sum_a p_a\Big(\frac{2^{-\ell(a)}}{p_a} - 1\Big) \le c\Big(\sum_a 2^{-\ell(a)} - 1\Big) \ .$$

Schließlich gilt $\sum_a 2^{-\ell(a)} = 1$ für einen vollen Binärbaum, wie wir schon im ersten Abschnitt festgestellt haben, und es folgt die $H \le E(k)$. □

Die „Quelle“ ist in der Informationstheorie der Ort, von wo die Nachrichten kommen. Dort benutzt man dann den Huffman-Code, um die Nachrichten in 01-Folgen zu transformieren. Daher rührt die Bezeichnung Quellencodierungssatz.

Bemerkung. Die übliche Interpretation der Entropie ergibt sich aus dem Quellencodierungssatz. Sei X ein „zufälliger Buchstabe“ aus dem Alphabet $\mathcal{A}$ mit Verteilung $p_a, a \in \mathcal{A}$, d.h. die Wahrscheinlichkeit, dass X gleich a ist, ist p_a. In Formeln:

$$\mathbf{P}(X = a) = p_a \ .$$

Stellen wir uns vor, dass wir X nicht kennen. Wie oben dargelegt können wir den Huffman-Code als Fragestrategie auffassen, um mit Ja-Nein-Fragen X von jemanden zu erfragen, der beobachten kann, welches der Buchstabe ist, den X in $\mathcal{A}$ angenommen hat.

Die Entropie gibt nach dem Quellencodierungssatz fast genau die mittlere Anzahl

$$E(k) = \sum_{a \in \mathcal{A}} \ell(a) \mathbf{P}(X = a)$$

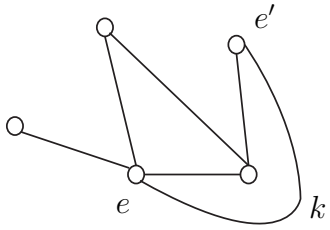
von Ja-Nein Fragen an, die dazu für uns notwendig ist. Dies ist gemeint, wenn man die Entropie beschreibt als den *Grad von Unbestimmtheit* oder *Ungewissheit* über den Wert, den X annimmt. Positiv ausgedrückt kann man auch vom *Informationsgehalt* des Zufallsexperiments sprechen, bei dem X entsteht. Dabei ist Information nicht inhaltlich gemeint, sondern in einem statistischen Sinn: Führt man ein Zufallsexperiment durch, bei dem „Erfolg“ mit Wahrscheinlichkeit p und „Misserfolg“ mit Wahrscheinlichkeit $q = 1 - p$ eintritt, so erfährt man wenig, wenn p nahe bei 0 oder 1 liegt, denn dann ist man sich über den Versuchsausgang schon von vornherein ziemlich sicher. So gesehen ist der Fall $p = 1/2$, der Wurf einer Münze, am informativsten. Da ist dann auch die Entropie maximal. – Die Informationstheorie, von Claude Shannon zur Welt gebracht, vertieft diese Interpretation und das Themengebiet der Nachrichtenübertragung.

2 Graphen

Die Theorie der Graphen hat eine geometrische Komponente, aber genauso eine algorithmische: Manche Fragen lassen sich rechnerisch schnell beantworten, andere nur mit allergrößtem Aufwand.

2.1 Kreise in Graphen

Unter einem Graphen verstehen wir hier (die Nomenklatur ist nicht einheitlich) ein Paar $G = (E, K)$, bestehend aus einer *Eckenmenge* E und einer *Kantenmenge* K. Das folgende Bild mit 5 Ecken und 6 Kanten zeigt, was gemeint ist.



Die Elemente aus E heißen *Ecken* oder *Knoten*, im Bild z.B. e und e'. Die Elemente $k \in K$ schreibt man gern in der Gestalt $k = \{e, e'\}$ mit Ecken $e, e' \in E$, $e \neq e'$. Wir interpretieren k als *Kante* zwischen e und e'. e, e' heißen dann *Nachbarn*. Es muss nicht jedes Paar von Ecken durch eine Kante verbunden sein – ist dies der Fall, so sprechen wir von einem *vollständigen Graphen*. Genauer gesprochen handelt es sich hier um *ungerichtete Graphen*, denn den Kanten ist keine Richtung gegeben. Gibt es Schleifen oder mehrere Kanten zwischen zwei Ecken, so spricht man von einem *Multigraphen*.

Der *Grad* $g(e)$ einer Ecke e ist die Anzahl ihrer Nachbarn,

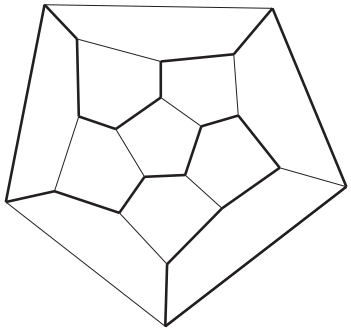
$$g(e) := \#\{e' : \{e, e'\} \in K\}.$$

Im Beispiel $g(e) = 4$, $g(e') = 2$.

Eine Folge $e_0, e_1, \ldots, e_l$ von Ecken heißt *Weg der Länge* l, falls e_{i-1}, e_i immer Nachbarn sind, falls also $\{e_{i-1}, e_i\} \in K$ für $i = 1, \ldots l$ gilt. Genauso kann man natürlich den Weg durch die Folge $k_1 = \{e_0, e_1\}, \ldots, k_l = \{e_{l-1}, e_l\}$ von Kanten gegeben denken. Gilt zusätzlich $e_0 = e_l$, handelt es sich also um einen geschlossenen Weg, so spricht man von einem *Kreis* (oder *Zyklus*).

Ein Graph heißt *zusammenhängend*, wenn es zwischen je zwei Ecken e, e' immer einen Verbindungsweg gibt, also einen Weg $e_0, e_1, \ldots, e_l$ mit $e_0 = e$ und $e_l = e'$. Zusammenhängende Graphen ohne Kreise haben wir schon kennengelernt, sie heißen *Bäume*.

Definition. Ein *Eulerkreis* ist ein Kreis, der jede Kante genau einmal passiert. Ein *Hamiltonkreis* ist ein Kreis, der jede Ecke genau einmal besucht.



ein Hamiltonkreis durch das ‚Dodekaeder'

Die Fragestellungen, ob ein Graph einen Eulerkreis bzw. einen Hamiltonkreis enthält, erscheinen auf den ersten Blick wenig unterschiedlich. Dies täuscht. Für Eulerkreise gibt es ein einfaches Kriterium.

Satz. *Ein Graph (und allgemeiner Multigraph) G enthält genau dann einen Eulerweg, wenn er zusammenhängend ist und der Grad einer jeden Ecke eine gerade Zahl ist.*

Beweis. Die Bedingungen sind offenbar notwendig: Existiert ein Eulerkreis, so auch ein Verbindungsweg zwischen je zwei Ecken. Außerdem muss der Eulerkreis, wenn er eine Ecke e besucht, sie auch wieder verlassen. Dies geschieht über verschiedene Kanten, damit hat e dann zwei Nachbarn. Besucht der Eulerkreis e zweimal, so hat e vier Nachbarn usw.

Die Bedingung ist auch hinreichend: Man wähle für den zu konstruierenden Eulerweg einen Startpunkt e und bilde (nach irgendeiner Regel) einen Weg durch G, der keine Kante doppelt durchläuft. Wegen der Gradbedingung kann jede Ecke $e' \neq e$ wieder verlassen werden. Man gelangt also irgendwann nach e zurück. Es entsteht ein Kreis K_1, der aber noch nicht alle Ecken durchlaufen haben muss. Gibt es eine Ecke e' außerhalb K_1, so gibt es auch einen Weg von e nach e', und damit einen Weg von einer Ecke von K_1 nach e', der keine Kreiskante mehr benutzt. Wieder wegen der Gradbedingung kann man diesen Weg zu einem neuen Kreis K_2 fortsetzen, ohne Kanten aus K_1 zu benutzen. Beide Kreise kann man zu einem einzigen zusammenfassen. Dies setzt man fort, bis alle Ecken erfasst sind. □

Den zweiten Teil des Beweises kann man auch als Algorithmus ausge-

stalten. Dies ist der *Hierholzer-Algorithmus* aus 1873, er ist ein schneller Algorithmus.

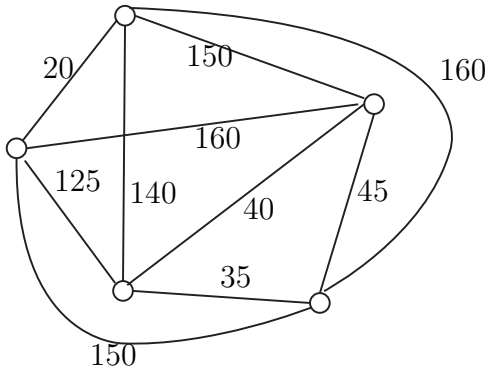
Für Hamiltonkreise gibt es kein ähnlich einfaches Kriterium. Auch gibt es keinen schnellen Algorithmus, um Hamiltonkreise zu finden. Im Gegenteil: Das Finden von Hamiltonkreisen ist Prototyp für Probleme, die algorithmisch besonders großen Aufwand erfordern. Würde man nämlich einen schnellen Algorithmus erfinden, so hätte man, wie sich herausstellt, auch schnelle Algorithmen für eine ganze Schar anderer notorisch schwieriger Probleme (alle sog. NP-vollständigen Probleme). Dass es solche Algorithmen nicht gibt, daran zweifelt heute wohl niemand mehr, auch wenn es bisher dafür keinen Beweis gibt. (Die Frage gehört zu den wichtigsten offenen Fragen der Mathematik.)

2.2 Das Problem des Handlungsreisenden

Wir kommen nun auf ein anderes notorisch schwieriges algorithmisches Problem zu sprechen, das *Problem des Handlungsreisenden* (traveling salesman problem, TSP). Jemand möchte eine Tour durch n Städte antreten und dann an den Ausgangsort zurückkehren. Die Kosten (Distanzen) für die Fahrt zwischen je 2 Städte sind gegeben (dabei seien die Kosten unabhängig davon, in welche Richtung man fährt, ob von A nach B oder B nach A. Man spricht deswegen genauer von einem symmetrischen TSP).

Aufgabe: Finde eine Rundtour mit mimimalen Gesamtkosten.

Dieser Situation liegt ein Graph zugrunde.



Die Ecken sind die Städte, die Kanten die Verbindungswege dazwischen. Der Graph ist vollständig, d.h. je zwei Ecken sind direkt duch eine Kante verbunden. Jede Kante k besitzt ein *Gewicht* $d(k)$, das die Kosten (Distanz) für die Benutzung dieser Verbindungslinie angibt. Gesucht ist ein Hamiltonkreis $k_1, k_2, \ldots, k_n$ mit minimalen Gesamtkosten $d(k_1) + \cdots + d(k_n)$.

Ein genauerer Blick auf das Beispiel zeigt, dass in diesem Beispiel der direkte Weg zwischen zwei Ecken immer der kürzeste ist. In Formeln ausgedrückt bedeutet dies

$$d(\{e_1, e_3\}) \leq d(\{e_1, e_2\}) + d(\{e_2, e_3\}) \quad (\textit{Dreiecksungleichung})$$

für beliebige Ecken e_1, e_2, e_3. Man spricht dann von einem *metrischen TSP*. Wir setzen dies im folgenden voraus.

Das Problem bei einem TSP ist nicht, einen Hamiltonkreis zu finden. In einem vollständigen Graphen ist das trivial, immer kann man eine beliebige, noch nicht besuchte Ecke ansteuern. Das Problem ist, dass so immens viele Hamiltonkreise existieren: Bei n Ecken hat man, ausgehend von einer Startecke, für die erste Fahrt $n-1$ Wahlmöglichkeiten, für die zweite dann noch $n-2$, usw. Insgesamt sind das

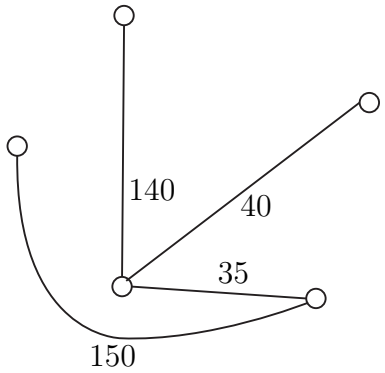
$$(n-1) \cdot (n-2) \cdots 2 \cdot 1 = (n-1)!$$

Hamiltonkreise. Für $n = 10$ sind das immerhin schon 362.880 verschiedene Rundfahrten! Dabei könnte man noch je zwei Touren zusammenfassen, die sich durch Umkehrung der Fahrtrichtung auseinander ergeben, aber das hilft auch nicht viel weiter.

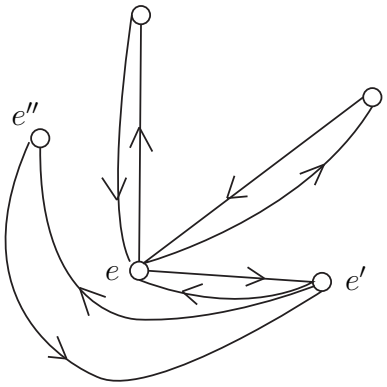
Um die optimale Rundtour zu finden, scheint kein besserer Algorithmus bekannt zu sein, als im wesentlichen alle Kreise durchzugehen. Verschiedene Strategien (die Informatiker sprechen von *Heuristiken*) wurden vorgeschlagen, um vergleichsweise gute Rundwege zu erhalten. Man kann etwa den Weg schrittweise aufbauen und dabei immer um eine möglichst kurze Strecke verlängern (im Beispiel: $20 + 125 + 35 + 45 + 150 = 375$). Man spricht dann von einem „greedy“, gierigen Verfahren. Eine andere Strategie besteht darin, von einer kürzesten Tour der Länge 3 auszugehen (im Beispiel $35 + 40 + 45$) und die schrittweise durch Hinzunahme einer weiteren Ecke zu immer größeren Kreisen auszudehnen, wobei pro Schritt der Längenzuwachs so klein wie möglich gehalten wird. Es stellt sich heraus, dass solche Heuristiken sehr schlechte Ergebnisse produzieren können.

Bemerkenswert ist, dass es schnelle Algorithmen gibt, die Rundwege produzieren, deren Länge *bis auf einen Faktor* optimal sind. Wir wollen ein solches Verfahren behandeln, bei dem dieser Faktor 2 ist. Damit erhält man also Rundtouren, deren Länge höchstens doppelt so lang wie die kürzeste Rundfahrt.

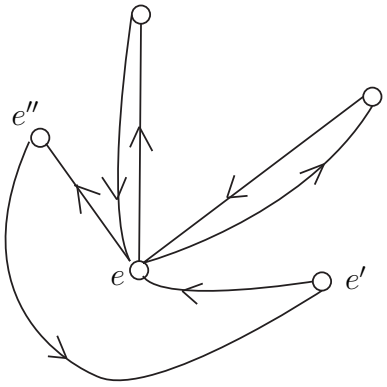
Für dieses Verfahren betrachtet man Spannbäume. Ein *Spannbaum* in einem Graphen ist ein Teilgraph, der alle n Ecken des Graphen enthält und der (weil er ein Baum ist) keine Kreise hat.



Jeder Spannbaum ergibt durch Verdoppeln jeder Kante einen Rundweg, bei die Ecken mehrfach besucht werden.



Jede mehrfach besuchte Ecke kann dann durch Wahl einer Abkürzung schrittweise zu einer einfach besuchten Ecke umgewandelt werden, ohne dass eine Ecke ausgelassen wird. Ist nämlich e, e', e'' ein Wegstück aus drei Ecken, und wird e' zweimal besucht, so kann man das Wegstück ohne weiteres durch e, e'' ersetzen, ohne das eine Stadt ausgelassen wird.



Bei einem metrischen TSP verkürzt sich dadurch die Tour. Die Gesamtlänge der am Ende resultierenden Rundtour mit Kanten $r_1, \dots r_n$. ist höchstens so lang wie die die doppelte Summe der Gewichte im Spannbaum.

Diese Vorgehensweise ist natürlich besonders günstig, wenn wir von einem *minimalen Spannbaum* ausgehen, dessen Summe über die Kantengewichte also unter allen Spannbäumen am kleinsten ist. Wir können dann folgendermaßen abschätzen:

Seien $k_1, \dots, k_n$ die Kanten einer optimalen Rundtour durch die n Ecken. Weil sie ein Kreis bilden, sind $k_1, \dots, k_{n-1}$ die Kanten eines Spannbaums. d' sei die Summe der Kantengewichte dieses Baums.

Sei andererseits d'' die Summe der Kantengewichte eines minimalen Spannbaumes und $r_1, \dots, r_n$ eine wie oben aus ihm resultierende Rundtour. Dann folgt nach unseren Überlegungen

$$\frac{d(r_1) + \dots + d(r_n)}{2} \leq d'' \leq d' \leq d(k_1) + \dots + d(k_n) \ .$$

Wir fassen zusammen:

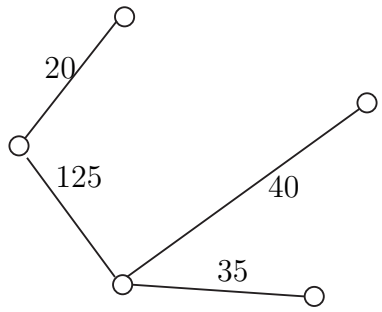
Satz. *Eine aus einem minimalen Spannbaum erhaltene Rundtour hat eine Länge, die höchstens doppelt so lang ist wie diejenige einer kürzesten Rundtour.*

Diese Vorgehensweise ist natürlich nur dann von Nutzen, wenn es einen schnellen Algorithmus gibt, um minimale Spannbäume zu finden. Der „greedy“ Algorithmus für dieses Problem erweist sich als geeignet.

Algorithmus für die Wahl eines minimalen Spannbaums. *Wähle die Kanten $k_1, k_2, \dots$ des Baums schrittweise nach folgenden Regeln.*

1. *Wähle k_1 als eine Kante von kleinstem Gewicht.*
2. *Sind $k_1, \dots, k_{j-1}$ schon bestimmt, so wähle k_j als eine von den Kanten $k \neq k_1, \dots, k_{j-1}$, so dass der von $k_1, \dots, k_j$ gebildete Graph keinen Kreis enthält. Unter diesen Kante wähle eine von kleinstem Gewicht.*

Das Verfahren bricht ab, wenn ein Spannbaum entstanden ist. In unserem Beispiel werden der Reihe nach die Kanten mit den Gewichten $20, 35, 40, 125$ ausgewählt, und der entstehende Spannbaum hat die Gestalt



Offenbar ist der Algorithmus schnell und leicht durchzuführen. Auch ist es plausibel, dass er korrekt ist, d.h. dass er immer einen minimalen Spannbaum liefert. Dieser letzte Tatbestand ist aber nicht offensichtlich und bedarf eines Beweises. Vorbereitend zeigen wir:

Satz. *Ein Baum mit n Ecken hat $n-1$ Kanten.*

Beweis. Die Behauptung erkennt man, indem man sich vorstellt, dass der Baum schrittweise aus einer Ecke heraus wächst. Am Anfang hat er eine Ecke und keine Kante, und pro Schritt kommt eine Ecke und eine Kante dazu, weil keine Kreise entstehen dürfen. □

Wir wollen nun den vom Algorithmus erzeugten Baum mit Kanten $k_1, \ldots, k_{n-1}$ vergleichen mit irgend einem anderen Spannbaum mit Kanten $k'_1, \ldots, k'_{n-1}$, dabei seien diese Kanten so aufgezählt, dass $d(k'_1) \leq \cdots \leq d(k'_{n-1})$ gilt.

Wir wollen zeigen, dass dann

$$d(k_i) \leq d(k'_i)$$

für alle $i = 1, \ldots, n-1$ gilt. Für $i = 1$ ist das klar nach Konstruktion des Algorithmus. Für $i > 1$ betrachten wir den von den Kanten $k_1, \ldots, k_{i-1}$ erzeugten Graphen. Nach Konstruktion enthält er keine Kreise, das bedeutet, er bildet er einen „Wald“, er zerfällt also in j verschiedene Bäume. Deren Ecken bilden Mengen $E_1, \ldots, E_j$ von den Mächtigkeiten $n_1, \ldots, n_j$. Nach dem Satz gilt

$$i - 1 = (n_1 - 1) + \cdots + (n_j - 1) \; .$$

Nun betrachten wir auch den Wald, der von $k'_1, \ldots, k'_i$ aufgespannt ist. Innerhalb von E_l kann er nach dem Satz höchstens $n_l - 1$ Kanten besitzen, $l = 1, \ldots, j$, sonst hätten wir in dem anderen Baum einen Kreis. Nach der letzten Gleichung gibt es also mindestens ein $j \leq i$, so dass k'_j keine zwei Ecken in einer der Mengen E_j verbindet.

Diese Kante k'_j steht damit im i-ten Schritt bei Durchführung des Algorithmus zur Verfügung, und es folgt wie behauptet

$$d(k_i) \leq d(k'_j) \leq d(k'_i) \ .$$

Insgesamt stellen wir fest, dass

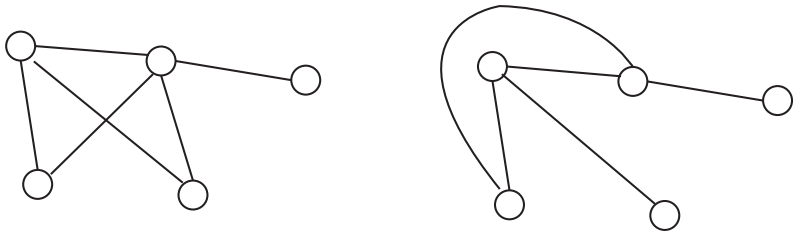
$$d(k_1) + \cdots + d(k_{n-1}) \leq d(k'_1) + \cdots + d(k'_{n-1})$$

gilt. Wir fassen zusammen:

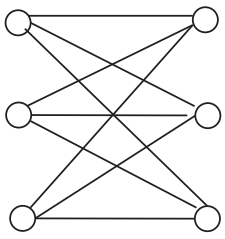
Satz. *Der Algorithmus zur Erzeugung eines minimalen Spannbaums ist korrekt.*

2.3 Planare Graphen

Wir wollen nun noch stärker geometrischen Aspekten von Graphen nachgehen. Das folgende Beispiel zeigt ein und denselben Graphen in zwei Darstellungen. Sie sind gleich, weil sie dieselben Anzahl von Ecken haben und entsprechende Paare von Ecken benachbart sind.

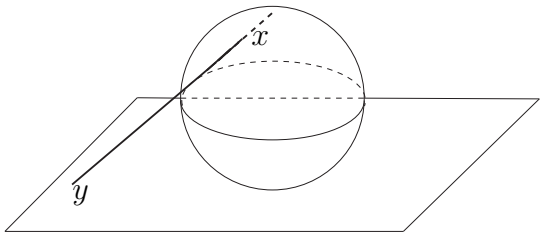


Im ersten kreuzen sich zwei Kanten, im zweiten ist das vermieden. Man kann sicher nicht immer erreichen, das es keine sich kreuzende Kanten gibt, aber wie ist das z. B. in der folgenden Situation (dieser Graph heißt $K_{3,3}$)?

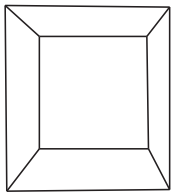


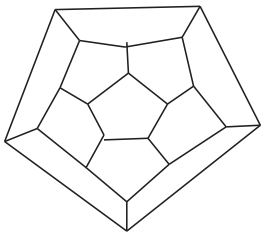
Definition. Ein Graph heißt *planar* (*plättbar*), falls sich seine Ecken und Kanten so in die Ebene legen lassen, dass sich keine zwei Kanten schneiden.

Genausogut kann man Graphen auf die Kugelsphäre zeichnen. Planare Graphen lassen sich dann mit *Polyedern* identifizieren. Weil man aber Elemente x der Kugelsphäre (ohne den Nordpol n) und Elemente y der Ebene eins zu eine miteinander identifizieren lassen, macht das keinen wesentlichen Unterschied. Die Identifikation lässt sich mit der *stereographischen Projektion* bewerkstelligen.



So lassen sich z.B. Würfel und Dodekaeder in die Ebene einbetten und in planare Graphen verwandeln.





Ein planarer Graph zerlegt die Ebene in *Zellen* (Teilflächen), einschließlich einer äußeren, die den Graphen umfasst. Dabei gilt eine wichtige und berühmte Geseztmäßigkeit, die zudem einen hübschen Beweis hat. Dabei werden die folgenden Größen zueinander in Beziehung gesetz:

$f :=$ Anzahl der Zellen, einschließlich der äußeren ,
$e :=$ Anzahl der Ecken ,
$k :=$ Anzahl der Kanten .

Satz. Eulersche Polyederformel. *Für zusammenhängende planare Graphen gilt*

$$e + f = k + 2 \ .$$

Beweis. Wir beseitigen schrittweise alle Kanten, nach dem folgenden Schema: Erst beseitigen wir Kanten, so dass sich immer wieder zwei Zellen zu einer einzigen vereinigen. Dies sind $f-1$ Schritte, denn am Anfang gab es f Zellen, am Ende nur noch ein.

Die restlichen Kanten bilden einen Spannbaum für die e Ecken. Wir beseitigen sie Schritt für Schritt, so dass der Baum immer kleiner wird. Dazu benötigen wir $e-1$ Schritte, denn am Anfang sind e Ecken verbunden, am Ende bleibt eine Ecke übrig.

Am Ende sind alle Kanten beseitigt. Die Gesamtanzahl der Kanten ist also $k = (f-1) + (e-1)$. Die ergibt die Behauptung. □

Beispiel. Ein nicht-planarer Graph. Wir zeigen, dass der obige Graph $K_{3,3}$ nicht planar ist. Wir wollen einen Widerspruchsbeweis führen. Nehmen wir an, der Graph wäre planar. Dann hätte jede Zelle mindestens 4 begrenzende Kanten, denn in $K_{3,3}$ gibt es keine Kreise der Länge 3. Da jede Kante an 2 Flächen grenzt, würde

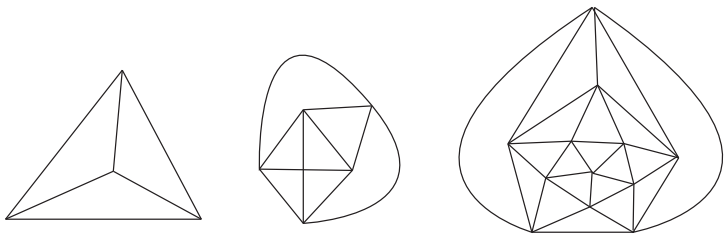
$$4f \leq 2k$$

folgen. Außerdem gilt $e = 6$ und $k = 9$. Mit der Polyederformel erhielten wir

$$11 = k + 2 = e + f \leq e + \frac{k}{2} = 10,5 \ ,$$

ein Widerspruch. Die Annahme kann also nicht richtig sein.

Beispiel. Regelmäßige Graphen. Wir nennen einen Graphen regelmäßig, wenn seine Ecken alle gleichviele Nachbarn haben, nämlich $p \geq 3$ Stück, und seine Zellen (einschließlich der äußeren) gleichviele begrenzende Kanten haben, nämlich $q \geq 3$ Stück. Die Patonischen Polyeder geben Beispiele. Würfel und Dodekaeder haben wir oben betrachtet, die anderen Polyeder sind Tetraeder, Oktaeder und Ikosaeder, die sich alle aus Dreiecken zusammensetzen.



Wir wollen zeigen, dass es keine weiteren gibt.

Da jede Kante zwei Ecken als Anfang und Ende hat und an zwei Flächen angrenzt, gilt

$$2k = pe \ , \quad 2k = qf \ .$$

Aufgelöst nach e und f und eingesetzt in die Polyederformel erhalten wir

$$\frac{2}{p}k + \frac{2}{q}k = k + 2 \ ,$$

und es folgt

$$\frac{2}{p} + \frac{2}{q} > 1.$$

Diese Ungleichung wird nur in den Fällen $p = q = 3$ oder $p = 4, q = 3$ oder $p = 3, q = 4$ oder $p = 5, q = 3$ oder $p = 3, q = 5$ erfüllt, sonst ist der linke Ausdruck, beginnend mit $p = q = 4$, kleiner oder gleich 1. Aus p und q berechnet sich dann aus der vorletzten Formel k und aus den beiden Formeln davor e und f. Dies entspricht genau den 5 Graphen, die wir bereits betrachtet haben.

3 Fehlerkorrigierende Codes

Wir wenden uns nun stärker algebraisch ausgerichteten Teilen angewandter Mathematik zu.

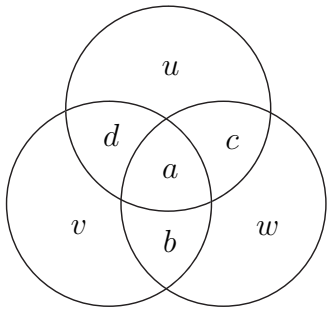
Wenn Nachrichten gesendet werden, können sich Übertragungsfehler einschleichen. Bei Texten bestehen da kaum Probleme, selbst bei vielen Fehlern ist man hinterher noch in der Lage, sie zu erkennen und zu beheben. Bei kodierten Nachrichten (etwa nach der Methode von Huffman) ist das Problem dagegen gravierend, und genauso bei der Übertragung von Daten. Moderne CD-Spieler etwa würden nicht funktionieren, wenn man nicht über *fehlerkorrigierende Codes* verfügte. Die Grundidee ist, dass man die Nachrichten redundant codiert, d.h. mehr Bits benutzt, als eigentlich (angesichts des Huffman-Codes) erforderlich wären.

Man unterscheidet zwischen *fehlererkennenden* und *fehlerkorrigierenden* Kodes. Ein fehlererkennender Code entsteht z. B., wenn man einer 01-Folge $a \dots d$ ein Prüfbit $e \in \{0, 1\}$ anhängen, so dass $s := a + \cdots + d + e$ eine gerade Zahl ist („parity check“). Ähnlich geht man z.B. bei der ISBN-Codierung von Büchern vor. Stellt man jedoch fest, dass s ungerade, also sicher ein Bit falsch ist, so kann man noch lange nicht den Fehler - allein anhand der codierten Nachricht - beseitigen. Solche Methoden sind etwa bei der Datenübertragung von Satelliten oder für CD-Spieler unbrauchbar.

Eine anderes Rezept zur Fehlererkennung ist, jedes Bit a doppelt zu übertragen: *aabbccdd* statt *abcd*. Fehlerkorrektur wird möglich, wenn man jedes Bit dreifach gesendet wird: Die empfangene Nachricht *acabbbccaddd* wird man dann als *abcd* entziffern, und nicht als *cbad*.

Hamming hat eine elegantere und effizientere Methode zur Fehlerkorrektur erfunden, die weite Anwendung findet und mit der wir uns nun eingehender befassen wollen. Sie ist ein $(7, 4)$-*Blockcode*, d.h. sie ergänzt jede Folge *abcd* aus 0en und 1en der Länge 4 zu einer 01-Folge *abcduvw* der Länge 7 mit drei zusätzlichen Prüfbits, so dass ein einzelner Fehler erkannt und korrigiert werden kann.

Das folgende Diagramm mit drei Kreisen erläutert das Vorgehen. Das Symbol a kommt in den Schnitt der 3 Kreise, b, c, d dorthin, wo je 2 Kreise sich schneiden. In den freigebliebenen Teilen der Kreise werden die Prüfbits u, v, w eingetragen, sie werden so gewählt, dass die Summe in jedem Kreis geradzahlig ist.



In Formeln: Zu $a, b, c, d \in \{0, 1\}$ wähle $u, v, w \in \{0, 1\}$, so dass

$$\begin{aligned} a + b + c + w &= \text{geradzahlig} \\ a + c + d + u &= \text{geradzahlig} \\ a + b + d + v &= \text{geradzahlig} \end{aligned} \tag{3}$$

Also: Für $abcd = 1001$ ist $uvw = 001$.

Der Clou ist, dass ein Fehler in der Reihe $abcduvw$ entdeckt und korrigiert werden kann. Dann sind manche dieser Summen ungeradzahlig: Bei a alle drei Summen, bei b, c, d jeweils 2 Summen und bei u, v, w jeweils eine Summe. Das falsche Bit lässt sich in jedem Fall ermitteln. Wir halten dies fest:

Satz. *Der Hammingcode kann einen Fehler korrigieren.*

Anders ausgedrückt: Ist $ABCDUVW$ eine beliebige 01-Folge, so gibt es genau eine Folge $abcduvw$, die Gleichungen (3) erfüllen und sich nur um höchstens 1 Bit von der vorgegebenen Folge unterscheidet. Sind etwa die Summen $A + B + C + W$, $A + C + D + U$, $A + B + D + V$ alle ungerade, so muss man A verändern, um die Gleichungen zu erfüllen.

Damit wird eine übersichtliche Struktur des Raumes $\{0, 1\}^7$ aller 01-Folgen der Länge 7 erkennbar. Insgesamt gibt es 2^7 derartige Folgen. Darunter sind 2^4 Folgen Codewörter. Sie entsprechen jeder möglichen Wahl von a, b, c, d. Um jedes Codewort gibt es eine „Umgebung“ benachbarter 01-Folgen, die sich um genau 1 Bit vom Codewort unterscheiden. Diese 7 Folgen werden bei Fehlerkorrektur in das Codewort übersetzt. Korrekte Übertragung führt natürlich auch zu korrekter Entschlüsselung, damit haben $7+1 = 8 = 2^3$ Folgen dieselbe Entschlüsselung. Alle 01-Folgen der Länge 7 sind erfasst, in der Tat gilt

$$2^4 \cdot 2^3 \;=\; 2^7 \; .$$

Damit ist jede Folge der Länge 7 entweder ein Codewort, oder es unterscheidet sich von genau einem Codewort an genau einer Stelle. Keine Folge wird dabei übersehen. Besser lässt sich der Raumes der Folgen nicht aufteilen, deswegen nennt man den Hammingcode einen *perfekten Code*.

Der Gewinn lässt sich in einer einfachen Rechnung mit Wahrscheinlichkeiten erkennen. Wird ein Bit mit Wahrscheinlichkeit $p = 0,05$ vertauscht - von Bit zu Bit unabhängig -, so gibt der Hammingcode das richtige Wort mit Wahrscheinlichkeit

$$q^7 + 7pq^6 \;=\; 0,956\ ,$$

mit $q = 1 - p$. Dagegen wird ein Wort *abcd* ohne Korrekturmöglichkeit mit Wahrscheinlichkeit

$$q^4 \;=\; 0,81$$

richtig übertragen.

Die geometrische Darstellung des Hammingcodes mittels Schnitt von 3 Kreisen ist suggestiv, für den Rechner aber ungeeignet. Für ihn ist ein algebraischer Zugang besser geeignet, der das Rechnen mit geraden und ungeraden Zahlen stärker formalisiert. Wir beschreiben dies nun etwas genauer.

Die Menge $\mathbb{Z}$ der ganzen Zahlen zerfällt in die beiden Klassen $\mathbb{G}$ der geraden und $\mathbb{U}$ der ungeraden Zahlen. Wie üblich schreiben wir 0 für $\mathbb{G}$ und 1 für $\mathbb{U}$, betonen aber, dass nun mit 0 und 1 nicht mehr die gewöhnlichen ganzen Zahlen gemeint sind. In der Mathematik schreibt man dann für die Menge beider Klassen

$$\mathbb{Z}_2 \;=\; \{0, 1\}\ .$$

Die arithmetischen Eigenschaften von $\mathbb{Z}_2$ sind folgendermaßen. Zwei gerade - oder zwei ungerade - Zahlen addiert ergeben eine gerade Zahl, eine gerade und eine ungerade Zahl addiert ergibt eine ungerade Zahl. Dies ergibt folgende Additionstabelle.

$+$	0	1
0	0	1
1	1	0

Genauso lassen sich gerade und ungerade Zahlen multiplizieren.

$\cdot$	0	1
0	0	0
1	0	1

Entscheidend ist, dass die üblichen Rechenregeln der ganzen Zahlen - Assoziativgesetz, Distributivgesetz, Kommutativgesetz - sich auf $\mathbb{Z}_2$ übertragen. Wer's nicht einsehen will, muss es nachrechnen.

Damit geht (3) in ein Gleichungssystem über, nämlich

$$\begin{aligned} a+b+c+w &= 0 \\ a+c+d+u &= 0 \\ a+d+b+v &= 0\ , \end{aligned}$$

ein ganz gewöhnliches, abgesehen davon, dass nun in $\mathbb{Z}_2$ gerechnet wird und nicht mehr in den rationalen Zahlen. Für diejenigen, die sich mit Matrizen auskennen, kann man das auch so schreiben:

$$\begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} \mathrm{a} \\ \mathrm{b} \\ \mathrm{c} \\ \mathrm{d} \\ \mathrm{u} \\ \mathrm{v} \\ \mathrm{w} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \tag{4}$$

Die Matrix heißt *Hammingmatrix*, man kann sich merken, dass sie als Spalten alle 01-Folgen der Länge 3 enthält, bis auf 000. Man bezeichnet sie üblicherweise mit H. Dass es beim Übertragen des Codewortes einen Übertragungsfehler gibt, kann man nun mittels Vektoraddition beschreiben. Wird etwa a falsch übertragen, so gilt

$$\begin{pmatrix} \mathrm{A} \\ \mathrm{B} \\ \mathrm{C} \\ \mathrm{D} \\ \mathrm{U} \\ \mathrm{V} \\ \mathrm{W} \end{pmatrix} = \begin{pmatrix} \mathrm{a} \\ \mathrm{b} \\ \mathrm{c} \\ \mathrm{d} \\ \mathrm{u} \\ \mathrm{v} \\ \mathrm{w} \end{pmatrix} + \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Bei einem Übertragungsfehler enthält der letzte Spaltenvektor f genau eine 1, und zwar an der Stelle, wo der Fehler eingetreten ist. Unter Beachtung von (4) folgt

$$\begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} \mathrm{A} \\ \mathrm{B} \\ \mathrm{C} \\ \mathrm{D} \\ \mathrm{U} \\ \mathrm{V} \\ \mathrm{W} \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Der rechte Ausdruck ergibt eine Spalte der Länge 3, die sich in der Hammingmatrix genau dort befindet, wo der Übertragungsfehler ist.

Zusammenfassend ergibt sich folgendes *Rezept zur Korrektur*: Multipliziere die Hammingmatrix H mit dem empfangenen Wort, aufgefasst als Spaltenvektor. Ergibt sich der Nullvektor, so nimm keine Korrektur vor. Ergibt sich ein anderer Vektor, so korrigiere an der Stelle, wo sich dieser Spaltenvektor in der Hammingmatrix befindet.

Man beachte, dass bei mehr als einem Übertragungsfehler das Verfahren nicht mehr funktioniert und sogar zusätzliche Fehler produzieren kann.

Zusammenfassend halten wir fest: Geometrisch gesehen ist der Hammingcode suggestiv. Algebraisch kann ihn der Rechner besser verarbeiten. Er ordnet sich dann in den Rahmen der Linearen Algebra ein ($\{0,1\}^7$ wird zum Vektorraum der Dimension 7 mit Skalarbereich $\mathbb{Z}_2$) und erfährt auf diese Weise auch die Möglichkeit der Verallgemeinerung. Lineare Codes sind ein wichtiges Thema der Codierungstheorie.

4 Kryptographie

Die moderne Kryptographie, die Lehre von der geheimen Nachrichtenübertragung, beruht darauf, dass es Rechenaufgaben gibt, die ohne Hilfestellung nicht bewältigt werden können, die aber mit einer Zusatzinformation leicht zu handhaben sind. Diese Rechnungen finden nicht in den von der Schule her bekannten Zahlsystemen statt, man muss in neue Bereiche vorstoßen.

4.1 Modulares Rechnen

Modulares Rechnen geht auf Gauß zurück. Es ist das Rechnen mit Resten. Man gibt sich eine natürliche Zahl $m > 1$, den „Modul“ vor und ersetzt jede ganze Zahl a durch ihren Rest r, der bei Division von a durch m,

$$a = mb + r \ , \quad 0 \leq r < m \ ,$$

übrigbleibt. Die Zahlen werden also zu Klassen zusammenfasst, zwei Zahlen, die denselben Rest “modulo m“ haben, werden nicht mehr unterschieden. Man spricht hier von *Restklassen*. Die Restklasse, in der a liegt, wird auch mit $[a]$ bezeichnet, und es gilt $[a] = [b]$, falls a und b denselben Rest haben. Man schreibt dann auch

$$a \equiv b \bmod m \ .$$

Für $m = 11$ z. B. gilt $[3] = [25] = [-8]$ bzw. $3 \equiv 25 \equiv -8 \bmod 11$.

Modulares Rechnen ist in der Zahlentheorie entstanden. Heutzutage ist es insbesondere für das Rechnen auf den Computer wichtig, wir kommen darauf zu sprechen.

Modulares Rechnen wird nur dann vernünftig, wenn man mit den Resten genauso rechnen kann wie mit normalen Zahlen. Den Spezialfall $m = 2$ haben wir im letzten Kapitel kennengelernt, dann hat man zwei Restklassen, die Menge der geraden Zahlen $\mathbb{G} = [0]$ und die Menge der ungeraden Zahlen $\mathbb{U} = [1]$. Die zugehörige Arithmetik haben wir bereits erörtert, wir wollen sie verallgemeinern.

Kann man zwei Restklassen addieren? Naheliegend ist folgende Regel:

$$[a] + [b] := [a + b] \ ,$$

also etwa $[7] + [8] = [15] = [4]$ für $m = 11$. Aber geht das gut auf? Es gilt $[7] = [-4]$ und $[8] = [30]$, und wie steht es mit $[7] + [8] = [-4] + [30]$? Damit man sich beim Addieren von Restklassen nicht in Widersprüche verwickelt, ist es notwendig, dass die Eigenschaft

$$[a] = [c] \ , \ [b] = [d] \quad \Rightarrow \quad [a + b] = [c + d]$$

erfüllt ist. Sie ist in der Tat richtig. Der Beweis: $[a] = [c]$ ist äquivalent zur Aussage, dass $a - c$ durch m teilbar ist (denn durch Differenzbildung heben sich die gleiche Reste weg). Genauso ist dann $b - d$ durch m teilbar. Es folgt, dass auch $(a + b) - (c + d) = (a - c) + (b - d)$ durch m teilbar ist, und dies bedeutet, dass $a + b$ und $c + d$ beide denselben Rest haben, dass also $[a + b] = [c + d]$ gilt.

Genauso lassen sich Restklassen multiplizieren:

$$[a] \cdot [b] := [a \cdot b] \ .$$

Auch dies ist stimmig: Ist $a - c$ und $b - d$ durch m teilbar, so auch $ab - cd = (a - c)b + c(b - d)$.

Die Additions- und Multiplikationstabelle sehen im Fall $m = 5$ so aus:

$+$	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

$\cdot$	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

Die Rechenregeln für das Addieren und Multiplizieren, Assoziativität, Distributivität, Kommutativität, bleiben gültig. Um etwa das Distributivgesetz zu verifizieren, vollziehe man die folgende Rechnung nach:

$$\begin{aligned}[a] \cdot ([b] + [c]) &= [a] \cdot [b + c] = [a(b + c)] \\ &= [ab + ac] = [ab] + [ac] = [a] \cdot [b] + [a] \cdot [c] \ .\end{aligned}$$

Die anderen Rechenregeln folgen analog.

So erschließen sich ganz neue Rechenbereiche, mit bekannten, dann aber auch unerwarteten neuen Eigenschaften. Man bezeichnet sie als

$$\mathbb{Z}_m = \{[0], [1], \ldots, [m - 1]\}$$

und spricht von *Restklassenringen modulo m*.

Beispiele.

1. Wir wollen die beiden letzten Dezimalstellen von 24^{2008} berechnen. Es gilt $24^2 = (25 - 1)^2 = 25 \cdot 25 - 2 \cdot 25 + 1 \equiv 25 - 2 \cdot 25 + 1 \equiv -24 \bmod 100$ und folglich (per Induktion) $24^k \equiv (-1)^{k-1} \cdot 24 \bmod 100$. Wir erhalten also $24^{2008} \equiv -24 \bmod 100$. Die letzten beiden Stellen von 24^{2008} sind 76.

2. **Neunerprobe.** Jemand hat $40752 \cdot 32111 = 1308587572$ ausgerechnet. Er rechnet die Quersummen aus und erhält 0, 8 und 1. Wegen $0 \cdot 8 \neq 1$ liegt ein Fehler vor. – Es handelt sich um Rechnen modulo 9. Es gilt $10 \equiv 1 \bmod 9$, also $40752 = 4 \cdot 10^4 + 7 \cdot 10^2 + 5 \cdot 10 + 2 \equiv 4 + 7 + 5 + 2 = 18 = 10 + 8 \equiv 1 + 8 = 9 \equiv 0 \bmod 9$ etc. Aus $40752 \cdot 32111 = 1308587572$ müsste also $0 \cdot 8 = 1$ folgen, ein Widerspruch.

3. Modulares Rechnen wird für den Computer wichtig, wenn mit sehr großen Zahlen operiert wird, die auch vom Computer nicht mehr ohne weiteres verarbeitbar sind. Man wählt dann verschiedene Moduln $m_1, \cdots, m_r$ und rechnet parallel mit ihnen. Am Ende muss man die verschiedenen Reste wieder zu einer Zahl zusammensetzen. Das geht ohne Schwierigkeiten, wenn $m_1, \ldots, m_r$ paarweise teilerfremd sind. Dann bestimmt sich aus den Restklassen modulo $m_1, \ldots, m_r$ eine eindeutige Restklasse modulo $m = m_1 \cdots m_r$, in der das Rechenresultat liegt („chinesischer Restsatz“), und sie legt die Zahl eindeutig fest, wenn nur m groß genug ist. – Zur Illustration multiplizieren wir die Zahlen 1010 und 997. Dazu setzen wir $m_1 = 99, m_2 = 100, m_3 = 101$. Es gilt $1010 \cdot 997 \equiv 20 \cdot 7 \equiv 41 \bmod 99$, $1010 \cdot 997 \bmod 10 \cdot (-3) \equiv -30 \bmod 100$, $1010 \cdot 997 \equiv 0 \cdot (-13) = 0 \bmod 101$. Der chinesische Restsatz ergibt $1010 \cdot 997 \equiv 7070 \bmod 999900$. Also $1010 \cdot 997 = 7070 + 999900 = 1006970$.

In ein paar Punkten unterscheidet sich das Rechnen modulo m vom gewöhnlichen Rechnen mit ganzen Zahlen. Sie machen die Restklassenringe interessant und auch für Anwendungen bedeutsam.

Nullteiler und inverse Restklassen. Im Gegensatz zu den ganzen Zahlen kann das Produkt zweier Restklassen $[a], [b] \neq [0]$ als Ergebnis die Restklasse $[0]$ ergeben, z. B. für $m = 14$ gilt $[2] \cdot [7] = [0]$. Solche Restklassen heißen *Nullteiler*. Dies hat zur Konsequenz, dass man Faktoren ungleich $[0]$ nicht ohne weiteres wegkürzen darf, z. B. gilt für $m = 14$

$$[1] \cdot [2] = [8] \cdot [2] \quad \text{aber} \quad [1] \neq [8] \ .$$

Andere Elemente $[a]$ von $\mathbb{Z}_m$ besitzen ein *Inverses* $[b]$, d.h. es gilt

$$[a] \cdot [b] = [1] \ ,$$

z. B. $[3] \cdot [5] = [1]$ modulo $m = 14$. Man schreibt dann $[b] = [a]^{-1}$. Restklassen, die ein Inverses besitzen, kann man aus Gleichungen mit Restklassen (durch Multiplikation mit ihrem Inversen) immer wegkürzen. Gilt $[a]^{-1} = [a]$, so heißt die Restklasse $[a]$ eine *Einheit*. z. B. ist $[4]$ Einheit modulo 15.

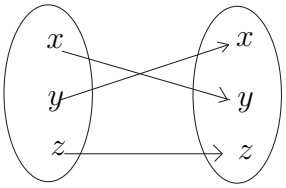
Es ist nicht schwer zu zeigen, dass jede Restklasse $[a] \neq [0]$ entweder Einheit oder Nullteiler ist, je nachdem, ob a und m teilerfremd sind oder nicht.

Prime Moduln. Die folgenden Resultate über den Fall primer Moduln benötigen wir für die folgenden Anwendungen in der Kryptographie.

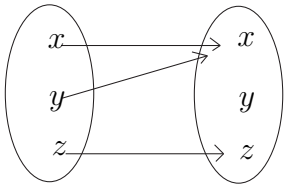
Satz. *Ist p eine Primzahl, so hat jede Restklasse $[a] \neq [0]$ in $\mathbb{Z}_p$ ein Inverses.*

Beweis. Sei $[a] \neq 0$ fest gewählt. Wir betrachten die Abbildung $\varphi : \mathbb{Z}_p \to \mathbb{Z}_p$, gegeben durch $[b] \mapsto \varphi([b]) := [a] \cdot [b]$. Sie ist injektiv auf $\mathbb{Z}_p$, d.h. keine zwei Restklassen haben dasselbe Bild unter φ. Anders ausgedrückt, aus $\varphi([b]) = \varphi([c])$ folgt $[b] = [c]$. In der Tat: Aus $\varphi([b]) = \varphi([c])$ bzw. $[ab] = [ac]$ folgt, dass $ab - ac = a(b - c)$ durch p teilbar ist. Die Annahme $[a] \neq 0$ bedeutet, dass a nicht durch p teilbar ist. Da p Primzahl ist, muss folglich p ein Teiler von $b - c$ sein, bzw. b und c bei Division durch p denselben Rest geben. Wir erhalten also $[b] = [c]$.

Nun ist bekanntlich eine injektive Abbildung auf einer endlichen Menge automatisch surjektiv, d.h. alle Restklassen sind Bilder, für jede Restklasse $[c]$ gibt es eine Restklasse $[b]$ mit $\varphi([b]) = [c]$.



injektiv und surjektiv



nicht injektiv, nicht surjektiv

Insbesondere folgt, dass es ein $[b]$ gibt mit $\varphi([b]) = [1]$ bzw. $[a] \cdot [b] = [1]$. $[b]$ ist dann das Inverse von $[a]$. □

Satz von Fermat. *Ist p eine Primzahl und $a \not\equiv 0 \bmod p$, so folgt*

$$a^{p-1} \equiv 1 \bmod p \ .$$

Beweis. Zu vorgegebenem a bilden wir wie im vorigen Beweis die bijektive Funktion φ. Mit ihr können wir das Produkt aller Restklassen ungleich $[0]$ verschieden aufschreiben:

$$[1] \cdot [2] \cdots [p-1] = \varphi([1]) \cdot \varphi([2]) \cdots \varphi([p-1]) = [a]^{p-1} \cdot [1] \cdot [2] \cdots [p-1] \ .$$

Nun können wir der Reihe nach durch Multiplikation mit den entsprechenden Inversen die Faktoren $[1]$ bis $[p-1]$ eliminieren und erhalten die Behauptung. □

4.2 Öffentliche Chiffriersysteme

Die **Kryptographie** ist die Lehre von den Chiffriersystemen. Stellen wir uns vor, dass eine Person A eine geheime Nachricht an die Person B übermitteln möchte. A kodiert sie deswegen mittels einer bijektiven Kodierabbildung

$$\kappa : \mathcal{N} \to \mathcal{K}.$$

Anstelle der Nachricht $a \in \mathcal{N}$ im Klartext sendet A die chiffrierte Nachricht $\kappa(a)$. Der Empfänger dekodiert mittels der inversen Abbildung

$$\kappa^{-1} : \mathcal{K} \to \mathcal{N}.$$

Ein bekanntes Verfahren beruht auf der Addition modulo 2. Wir nehmen an, dass die Nachrichten als 01-Folgen der Länge n vorliegen, wählen also $\mathcal{N} = \mathcal{K} = \{0,1\}^n$. Wir fassen die Folgen als Vektoren der Läne n über dem Körper $\mathbb{Z}_2$ auf. Zum Kodieren wird ein 01-String $s = s_1 s_2 \dots s_n$ verwendet. Wir setzen

$$\kappa(a) \;:=\; a + s,$$

die beiden 01-Folgen a und s werden also komponentenweise modulo 2 addiert. Der Schlüssel ist hier so lang wie die Nachricht. Es gilt $\kappa^{-1} = \kappa$, denn $s + s$ ist die nur aus Nullen bestehende Vektor. Nachrichten werden daher nach demselben Verfahren kodiert und dekodiert. Dieses klassische Verfahren hat den Namen ‚One-time-pad' (Einmalverschlüsselung). Es gilt $a+\kappa(a) = s$, gelingt es daher, eine kodierte Nachricht $k(a)$ zu entschlüsseln, so kennt man s und damit bereits die vollständige Kodier- und Dekodiervorschrift. Dies bedeutet, dass das Verfahren sicher ist, man hat keine Chance, eine Nachricht zu dechiffrieren, wenn man auf s keinen Zugriff hat. Ist s rein zufällig aus $\{0,1\}^n$ gewählt, so gilt dies auch für $\kappa(a)$. Die Methode ist perfekt, wenn s nur einmal verwendet wird.

Das RSA-Schema

Für klassische Chiffrierverfahren besteht ein Sicherheitsproblem darin, dass man nicht nur die Dekodiervorschrift κ^{-1} geheimhalten muß, sondern auch das Kodierverfahren κ. Wie das Beispiel des One-time-pad zeigt, lassen sich die Nachrichten mit Hilfe der Chiffrier- wie der Dechiffriervorschrift leicht entschlüsseln. Diffie und Hellman haben daher 1976 einen (damals beherzten) Vorschlag gemacht: Man solle Kodierabbildungen κ benutzen, für die $b = \kappa(a)$ aus a leicht berechnet werden kann, umgekehrt aber die Berechnung von $a = \kappa^{-1}(b)$ aus b typischerweise mit einem immensen Rechenaufwand verbunden ist, der praktisch nicht zu bewältigen ist (selbst wenn

einem die Abbildung κ bekannt ist!). In dieser asymmetrischen Situation darf man das Chiffrierverfahren öffentlich bekannt machen, ohne die Sicherheit zu gefährden. Dies ist die Grundidee der modernen, computergestützten *öffentlichen Kodiersysteme*. Es bereitet Mühe, theoretisch zu begründen, dass solche Abbildungen κ, man spricht von *Einweg-(one-way) Abbildungen*, wirklich existieren. Für praktische Zwecke haben sich jedoch verschiedene Abbildungen als brauchbar erwiesen. Wir werden Abbildungen betrachten, bei denen das Dekodieren erst dann praktisch durchführbar ist, wenn man über einen zusätzlichen ‚Schlüssel' verfügt. Dann spricht man von *Falltür-(trapdoor) Abbildungen*.

Zum Beispiel kann man schnell modulo m potenzieren:

$$a^{20} = a^{16+4} = (((a^2)^2)^2)^2 \cdot (a^2)^2.$$

Allgemein berechnet man a^c modulo m, indem man den Exponenten als binäre Zahl darstellt, $c = d_0 + d_1 \cdot 2 + \cdots + d_k \cdot 2^k$ mit $d_i \in \{0, 1\}$, in k Schritten rekursiv die Potenzen $a^{2^i} = (a^{2^{i-1}})^2$ modulo m berechnet und schließlich diejenigen Potenzen modulo m multipliziert, für die $d_i = 1$ gilt. Die Anzahl der Operationen ist von der Ordnung $O(k) = O(\log c)$. – Eine allgemeine Methode, mit der man schnell (schneller als Durchprobieren) die Gleichung $b \equiv a^x \bmod m$ nach x auflöst (‚diskreter Logarithmus'), kennt man dagegen nicht.

Wir beschreiben nun das bekannteste öffentliche Chiffriersystem, das von Rivest, Shamir und Adleman 1978 vorgeschlagene *RSA-System*. Es baut darauf auf, dass es schwer ist, eine Zahl m in ihre Primfaktoren zu zerlegen.

Die RSA-Codierung.

- *Als Nachrichtenmenge $\mathcal{N} = \mathcal{K}$ wird $\mathbb{Z}_m$ gewählt. Dabei sei $m = pq$ das Produkt zweier sehr großer Primzahlen p und q. Wir gehen also davon aus, dass die Nachricht als natürliche Zahl $a < m$ dargestellt ist.*

- *Zum Kodieren wird eine natürliche Zahl $s < (p-1)(q-1)$ gewählt, die teilerfremd zu $(p-1)(q-1)$ ist. Die Kodierung der Nachricht $a \in \mathcal{N}$ ist dasjenige $\kappa(a) \in \mathcal{N}$, so dass*

$$\kappa(a) \equiv a^s \bmod m \ .$$

- *Zum Dekodieren benötigt man einen Schlüssel, eine weitere natürliche Zahl $t < (p-1)(q-1)$ mit der Eigenschaft*

$$s \cdot t \equiv 1 \bmod (p-1)(q-1) \ .$$

  Die Dekodierung von $b \in \mathcal{N}$ ist dasjenige $\kappa^{-1}(b) \in \mathcal{N}$ mit

$$\kappa^{-1}(b) \equiv b^t \bmod m \ .$$

Beispiel. Wir wählen $m = 17 \cdot 23 = 391$, $s = 101$ und $t = 237$. Dann sind in der Tat s und $16 \cdot 22 = 352$ teilerfremd und $st = 23937 \equiv 1 \bmod 352$. Die Nachricht $a = 52$ wird verschlüsselt zu

$$\begin{aligned} 52^{101} &= 52^{1+4+32+64} \\ &= 52 \cdot ((52)^2)^2 \cdot (((((52)^2)^2)^2)^2)^2 \cdot ((((((52)^2)^2)^2)^2)^2)^2 \\ &\equiv 52 \cdot 307 \cdot 188 \cdot 154 \equiv 358 \bmod 391 \end{aligned}$$

und wieder entschlüsselt als

$$\begin{aligned} 358^{237} &= 358^{1+4+8+32+64+128} \\ &= 358 \cdot ((358)^2)^2 \cdot (((358)^2)^2)^2 \cdot (((((358)^2)^2)^2)^2)^2 \\ &\quad \cdot ((((((358)^2)^2)^2)^2)^2)^2 \cdot (((((((358)^2)^2)^2)^2)^2)^2)^2 \\ &\equiv 358 \cdot 18 \cdot 324 \cdot 154 \cdot 256 \cdot 239 \equiv 52 \bmod 391 \ . \end{aligned}$$

Ohne Kenntnis von t müsste man alle Zahlen $a < 391$ durchgehen, ihre 101-te Potenz modulo 391 bilden, um diejenige zu finden, die dann 358 ergibt.

Dieses Verfahren wird auf Computern implementiert mit Moduln m, die mehr als 200-stellig sind (Stand 2008). Man darf dieses Verfahren so einrichten, dass die Nachrichtenmenge $\mathcal{N}$, also m, die Zahl s und auch die kodierte Nachricht $\kappa(a)$ öffentlich zugänglich sind.

Es stellen sich einige Fragen. Wie findet man die Schlüssel? Warum ist es für die Sicherheit ausreichend, dass man nur p, q, und t geheim hält? Wir wollen hier nur zeigen, dass die Entschlüsselung korrekt funktioniert.

Satz. *Seien p, q Primzahlen, $m = pq$ und $st \equiv 1 \bmod (p-1)(q-1)$. Dann gilt*

$$b \equiv a^s \bmod m \quad \Leftrightarrow \quad a \equiv b^t \bmod m \ .$$

Beweis. Zu zeigen ist

$$(a^s)^t \equiv a \bmod m \ .$$

Sei zunächst $a \not\equiv 0 \bmod p$. Dann folgt $a^{p-1} \equiv 1 \bmod p$ nach dem Satz von Fermat. Nach Voraussetzung gibt es eine natürliche Zahl k, so dass $st = 1 + k(p-1)(q-1)$. Es folgt

$$(a^s)^t = a \cdot (a^{p-1})^{k(q-1)} \equiv a \bmod p \ .$$

Offenbar gilt diese Aussage auch für $a \equiv 0 \bmod p$. Genauso folgt die Kongruenz $(a^s)^t \equiv a \bmod q$ für alle a. Insgesamt ist $(a^s)^t - a$ durch p und q teilbar, also auch durch m. Dies ergibt die Behauptung. □

Das RSA-System steht und fällt damit, dass die Primfaktoren p und q geheim bleiben, sonst kann man aus s leicht t bestimmen und die Geheimbotschaft damit knacken. Die Erfahrung zeigt, dass die Faktorisierung von m in seine Primteiler einen hohen Rechenaufwand erfordert und praktisch nicht mehr gelingt, wenn p und q ausreichend groß sind. Dies ist das *Grundpostulat der Kryptographie*: Man geht davon aus, dass es keinen schnellen Algorithmus zum Zerlegen einer Zahl in seine Primfaktoren gibt. Man nennt deshalb in der Kryptographie ein Chiffriersystem ‚beweisbar sicher', wenn es letztlich auf die Faktorisierung großer Zahlen zurückgeführt werden kann.

Signatur von Nachrichten. Das RSA-Schema eignet sich gut zum geheimen Austausch von Nachrichten innerhalb einer Gruppe von Teilnehmern. Jeder Teilnehmer A erhält von einer Zentrale einen öffentlichen Schlüssel $(m(A), s(A))$ und seinen persönlichen geheimen Schlüssel $t(A)$. A teilt B eine geheime Nachricht a als

$$b := \kappa_B(a) \equiv a^{s(B)} \bmod m(B)$$

mit, und B dekodiert mittels

$$\kappa_B^{-1}(b) \equiv b^{t(B)} \bmod m(B) \ .$$

Ein vorheriger Kontakt zwischen A und B ist nicht erforderlich, A braucht lediglich den öffentlich zugänglichen Schlüssel $s(B)$ von B. Ein wichtiger Vorteil des Systems ist, dass es den Teilnehmern erlaubt, Mitteilungen fälschungssicher zu signieren (‚Unterschriftensystem'). A beglaubigt eine öffentliche Nachricht a durch die öffentliche Mitteilung von

$$c := \kappa_A^{-1}(a) \equiv a^{t(A)} \bmod m(A) \ .$$

Alle anderen Teilnehmer können verifizieren, dass $a \equiv c^{s(A)} \bmod m(A)$ gilt.

5 Zufall

5.1 Wahrscheinlichkeiten

Wahrscheinlichkeiten benutzt man dazu, um die Chancen zu quantifizieren, dass *Zufallsereignisse* eintreten. Wir schreiben

$$\mathbf{P}(E)$$

für die Wahrscheinlichkeit des Ereignisses E. Das *Komplementärereignis* (Gegenereignis) ist das Ereignis E^c, das genau dann eintritt, wenn E nicht eintritt. Es hat Wahrscheinlichkeit

$$\mathbf{P}(E^c) = 1 - \mathbf{P}(E) \ .$$

Beispiel. Beim Werfen eines (ungezinkten) Würfels hat das Ereignis

$$E_6 = \{\text{eine 6 wird geworfen}\}$$

zweifellos die Wahrscheinlichkeit

$$\mathbf{P}(E_6) = \mathbf{P}(\text{eine 6 wird geworfen}) = \frac{1}{6}$$

und das Komplementärereignis

$$E_6^c = \{\text{es fällt eine der Zahlen } 1, \ldots, 5\}$$

die Wahrscheinlichkeit 5/6.

Beispiel. Reihenuntersuchungen. Bei der Wahrscheinlichkeit des Ereignisses

$$E_{kr} = \{\text{eine Person trägt die Krankheit K in sich}\}$$

denkt man an den relativen Anteil der Personen, die von dieser Krankheit K befallen sind – etwa unter allen 40–50-jährigen Personen, die aus einer bestimmten Region stammen – und die noch keine Symptome der Krankheit zeigen. In relevanten Fällen gilt

$$\mathbf{P}(E_{kr}) = 0,008 = 0,8\%$$

Für Personen mit Krankheitssymptomen ist diese Wahrscheinlichkeit natürlich nicht anwendbar, sie sind hier beiseite gelassen. Das Gegenereignis

$$E_{ges} = \{\text{eine Person ist frei von K}\}$$

hat dann Wahrscheinlichkeit $0,992$. Wir werden sehen, was solche Zahlen im Kontext von Reihenuntersuchungen bedeuten. Eine genauere Erörterung der Thematik findet sich in dem empfehlenswerten Buch ‚Das Einmaleins der Skepsis' von G. Gigerenzer.

Neben Wahrscheinlichkeiten betrachten wir bedingte Wahrscheinlichkeiten. Die *bedingte Wahrscheinlichkeit von E_1 gegeben E_2*,

$$\mathbf{P}(E_1 \,|\, E_2) \ ,$$

ist anschaulich gesprochen die Wahrscheinlichkeit des Ereignisses E_1, wenn schon bekannt ist, dass das Ereignis E_2 eingetreten ist.

Beispiel. Beim Würfeln gilt

$$\mathbf{P}(\text{eine 6 fällt} \,|\, \text{die geworfene Zahl ist gerade}) = 1/3 \ .$$

Denn wenn man schon weiß, dass die geworfene Zahl gerade ist, ändert sich die Chance für eine 6 offenbar von $1/6$ auf $1/3$.

Beispiel. Reihenuntersuchungen. Bedingte Wahrscheinlichkeiten werden gebraucht, um die Güte von Vorsorgeuntersuchungen („Screenings") für die Krankheit K zu bewerten. Mit hoher Wahrscheinlichkeit hat man einen positiven Befund, gegeben die Testperson ist krank, manchmal ist aber auch bei gesunden Personen der Befund positiv. Bezeichnet

$$E_{po} = \{\text{positiver Befund für die Testperson}\}$$

das Ereignis, das für eine Person der Testbefund positiv ausfällt, so sind typische Werte

$$\mathbf{P}(E_{po} \,|\, E_{kr}) = 0,9 = 90\% \ , \quad \mathbf{P}(E_{po} \,|\, E_{ges}) = 0,07 = 7\% \ .$$

Für eine Person mit positivem Testbefund sind diese bedingten Wahrscheinlichkeiten jedoch weniger interessant, sie möchte vor allem wissen, mit welcher Wahrscheinlichkeit sie krank ist, wie groß also die bedingte Wahrscheinlichkeit

$$\mathbf{P}(E_{kr} \,|\, E_{po})$$

ist. Wir werden sehen, wie sich diese Wahrscheinlichkeit berechnet und dass sie kleiner ist, als man erwarten könnte.

Es geht also um den Zusammenhang zwischen zwei Ereignissen E_1 und E_2. Man kann ihn auf drei verschiedene Weise erfassen.

1. Für das Ereignis, dass beide gleichermaßen eintreten, schreibt man

$$E_1 \cap E_2 := \{E_1 \text{ und } E_2 \text{ treten beide ein}\} \ ,$$

genauso $E_1 \cap E_2^c$ für das Ereignis, dass E_1 eintritt und E_2 nicht eintritt usw. Prinzipiell ist dann durch die vier Wahrscheinlichkeiten des Eintretens bzw. Nichteintretens

$$\begin{aligned} p(e,e) &:= \mathbf{P}(E_1 \cap E_2) \ , \ p(e,n) := \mathbf{P}(E_1 \cap E_2^c) \ , \\ p(n,e) &:= \mathbf{P}(E_1^c \cap E_2) \ , \ p(n,n) := \mathbf{P}(E_1^c \cap E_2^c) \end{aligned}$$

alles festgelegt. Ihre Summe ist 1, denn genau eines der vier Ereignisse tritt sicher ein. Es gilt

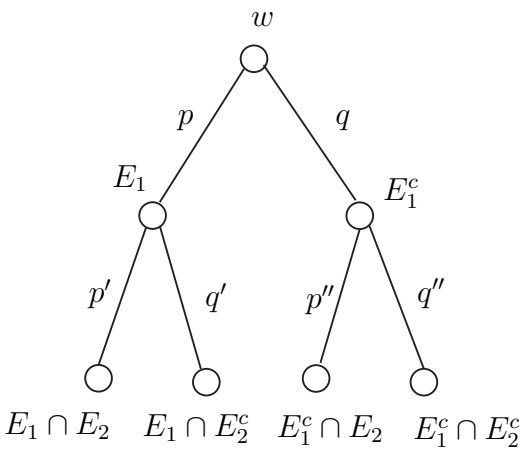
$$\mathbf{P}(E_1) = p(e,e) + p(e,n) \ , \quad \mathbf{P}(E_2) = p(e,e) + p(n,e) \ ,$$

denn E_1 tritt ein, wenn $E_1 \cap E_2$ oder $E_1 \cap E_2^c$ eintritt. Für sich genommen ist diese Sicht noch wenig aufschlussreich.

2. In vielen Fällen ist der Ausgangspunkt ein anderer. Man stellt sich vor, dass sich erst herausstellt, ob E_1 eintritt, und dass, in Abhängigkeit davon, sich dann das Eintreten von E_2 entscheidet. In diesem Szenario sind statt $p(e,e), \ldots, p(n,n)$ die unbedingten und bedingten Wahrscheinlichkeiten

$$p := \mathbf{P}(E_1) \ , \quad p' := \mathbf{P}(E_2 \,|\, E_1) \ , \quad p'' := \mathbf{P}(E_2 \,|\, E_1^c)$$

vorgegeben. Der Entscheidungsverlauf über das Eintreten von E_1 und E_2 wird im folgenden Baum dargestellt:



Ausgehend von der Wurzel w geht man mit Wahrscheinlichkeit p nach links unten und mit der Gegenwahrscheinlichkeit $q := 1 - p$ nach rechts unten. Im zweiten Schritt geht man dann mit Wahrscheinlichkeit p' bzw. p'' wieder nach links und mit den Gegenwahrscheinlichkeiten q' bzw. q'' nach rechts. Die Wahrscheinlichkeiten, in den vier Blättern zu landen, ergeben sich durch Multiplikation der Kantengewichte als

$$p(e,e) = pp' \ , \ p(e,n) = pq' \ , \ p(n,e) = qp'' \ , \ p(n,n) = qq'' \ ,$$

und die Wahrscheinlichkeit, dass E_2 eintritt, ermittelt sich als

$$\mathbf{P}(E_2) = p(e,e) + p(n,e) = pp' + qp''$$

bzw. als

$$\mathbf{P}(E_2) = \mathbf{P}(E_2 \mid E_1)\mathbf{P}(E_1) + \mathbf{P}(E_2 \mid E_1^c)\mathbf{P}(E_1^c) \ . \tag{5}$$

Diese Formel heißt der *Satz von der totalen Wahrscheinlichkeit*. Umgekehrt bestimmen sich aber auch p, p', p'' aus den Wahrscheinlichkeiten $p(e,e), \ldots, p(n,n)$, gemäß

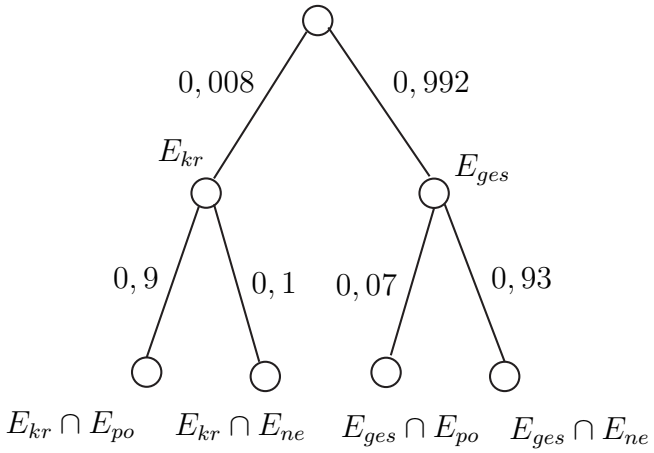
$$p = p(e,e) + p(e,n) \ , \quad p' = \frac{p(e,e)}{p} \ , \quad p'' = \frac{p(n,e)}{1-p} \ .$$

Die zweite Formel können wir auch schreiben als

$$\mathbf{P}(E_2 \,|E_1) = \frac{\mathbf{P}(E_1 \cap E_2)}{\mathbf{P}(E_1)} \ , \tag{6}$$

was der üblichen Definition von bedingten Wahrscheinlichkeiten entspricht.

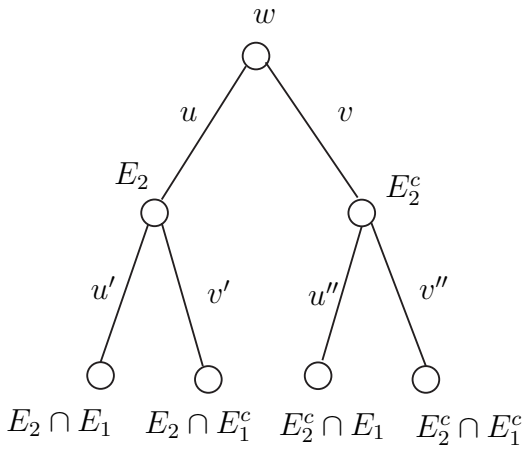
Beispiel. Reihenuntersuchungen. In diesem Fall sieht, mit $E_{ne} := E_{po}^c$, der Baum so aus:



Der Satz von der totalen Wahrscheinlichkeit ergibt

$$\mathbf{P}(E_{po}) = 0,9 \cdot 0,008 + 0,07 \cdot 0,992 = 0,077 \ .$$

3. Wenn aber alles von $p(e,e), \ldots, p(n,n)$ bestimmt ist, dann können wir unter 2. auch die Rolle von E_1 und E_2 vertauschen und uns vorstellen, dass erst entschieden wird, ob E_2 eintritt, und dann in Abhängigkeit davon sich entscheidet, ob E_1 eintritt. Zu dem obigen Baum tritt also ein äquivalenter Baum



Die neuen Kantenwahrscheinlichkeiten u, $v = 1 - u$, ... berechnen sich in analoger Weise aus $p(e,e) \ldots, p(n,n)$.

Die Äquivalenz der drei Sichtweisen bedeutet insbesondere, dass sich die Wahrscheinlichkeiten aus Abschnitt 2. in diejenigen aus Abschnitt 3. umrechnen lassen und insbesondere die bedingte Wahrscheinlichkeit $\mathbf{P}(E_1 \,|E_2)$ aus den Wahrscheinlichkeiten $\mathbf{P}(E_1)$, $\mathbf{P}(E_2 \,|E_1)$ und $\mathbf{P}(E_2 \,|E_1^c)$ berechnen lässt. Diese Erkenntnis war durchaus ein Aha-Erlebnis in der Geschichte er Wahrscheinlichkeitsrechnung. Die folgende Formel leistet die Umrechnung.

Satz. Die Formel von Bayes. *Es gilt*

$$\mathbf{P}(E_1 \,|\, E_2) = \frac{\mathbf{P}(E_2 \,|\, E_1)\mathbf{P}(E_1)}{\mathbf{P}(E_2 \,|\, E_1)\mathbf{P}(E_1) + \mathbf{P}(E_2 \,|\, E_1^c)\mathbf{P}(E_1^c)} \ .$$

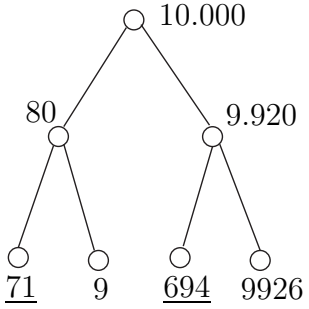
Beweis. Nach (6) gilt $\mathbf{P}(E_1 \,|\, E_2)\mathbf{P}(E_2) = \mathbf{P}(E_1 \cap E_2) = \mathbf{P}(E_2 \,|\, E_1)\mathbf{P}(E_1)$. Die Behauptung folgt dann per Division durch $\mathbf{P}(E_2)$ unter Beachtung der Formel (5). □

Beispiel. Reihenuntersuchungen. Mit $E_1 = E_{kr}$ und $E_2 = E_{po}$ ergibt sich

$$\mathbf{P}\big(\text{Person ist krank} \,|\, \text{positiver Befund}\big) = \frac{0,9 \cdot 0,008}{0,9 \cdot 0,008 + 0,07 \cdot 0,992} = 0,09 \ .$$

Dies Resultat ist bemerkenswert. Wir erkennen: Wenn man ein positives Testresultat hat, ist die Chance viel größer, einer von den vielen Gesunden zu sein, bei dem der Test einmal nicht richtig funktioniert, als einer der wenigen Kranken. Betroffenen mit positivem Testbefund kann man also nur raten, die Ruhe zu bewahren und sich erst einmal von einem Spezialisten gründlich untersuchen zu lassen.

Das Resultat lässt sich plastisch in absoluten Häufigkeiten bei 10.000 Testpersonen wie folgt darstellen:



Das Simpson Paradoxon. Der Satz von der totalen Wahrscheinlichkeit ist auch der Schlüssel, um ein paradoxes Phänomen aufzuklären.

Der folgende Datensatz betrifft den Erfolg zweier Methoden A und B zur Behandlung bei Nierensteinen. In Patientengruppen von unterschiedlichem Umfang wurde die relative Anzahl der erfolgreichen Behandlungen festgestellt. Eine Gruppe K besteht aus Patienten mit kleinen Nierensteinen, hier sind die Erfolgswahrscheinlichkeiten

$$\text{A:}\ \frac{81}{87} = 0,93 \ , \quad \text{B:}\ \frac{234}{270} = 0,87 \ .$$

In einer anderen Gruppe G mit großen Nierensteine sind die Werte

$$\text{A:}\ \frac{192}{263} = 0,73 \ , \quad \text{B:}\ \frac{55}{80} = 0,69 \ .$$

Methode A ist also erfolgreicher. Legt man jedoch beide Gruppen zusammen und unterscheidet nicht mehr nach der Größe der Steine, so kehrt sich das Verhältnis um:

$$\text{A:}\ \frac{273}{350} = 0,78 \ , \quad \text{B:}\ \frac{289}{350} = 0,83 \ .$$

Nun scheint Methode B besser zu sein. Dies ist ein Trugschluss: Kurz gesagt kombinieren sich beim Zusammenfassen die Erfolgswahrscheinlichkeiten für A und B, aber in verschiedenen Verhältnissen, wodurch die Umkehrung entsteht. Wir wollen dies genauer erörtern.

Wir betrachten die Ereignisse

$$\begin{aligned} E &= \{\text{Patient hat kleine Steine}\} \\ E^c &= \{\text{Patient hat große Steine}\} \\ E' &= \{\text{die Behandlung ist erfolgreich}\} \end{aligned}$$

Nach den Daten gilt bei Methode A

$$\mathbf{P}_A(E' \,|\, E) = 0,93 \ , \quad \mathbf{P}_A(E' \,|\, E^c) = 0,73 \ .$$

Außerdem erkennt man, welcher relative Anteil der mit Methode A behandelten Patienten der Gruppe K bzw. G angehört:

$$\mathbf{P}_A(E) = \frac{87}{350} = 0,25 \ , \quad \mathbf{P}_A(E^c) = 0,75$$

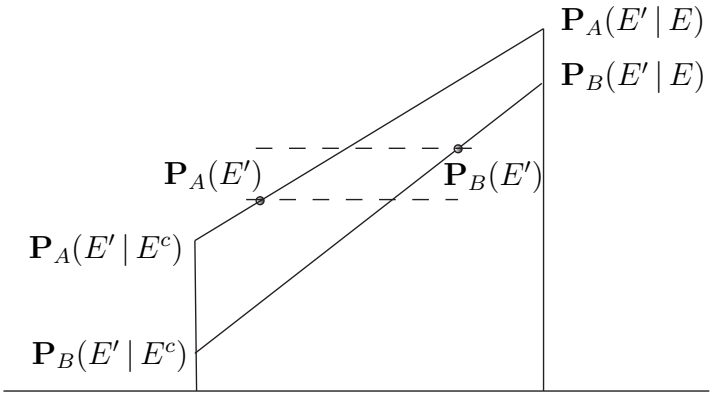
Mit diesen Wahrscheinlichkeiten und dem Satz von der totalen Wahrscheinlichkeit

$$\mathbf{P}_A(E') = \mathbf{P}_A(E' \mid E)\mathbf{P}_A(E) + \mathbf{P}_A(E' \,|\, E^c)\mathbf{P}_A(E^c)$$

erhalten wir erneut den oben angegebenen Wert

$$\mathbf{P}_A(E') = 0,78 \ .$$

Der Wert entsteht durch Mittelung aus den beiden bedingten Wahrscheinlichkeiten, denn $\mathbf{P}_A(E) + \mathbf{P}_A(E^c) = 1$. Er liegt näher bei der Wahrscheinlichkeit für die Patienten aus Gruppe G, da sie hier mit einem Anteil von 75% dominieren.



Genauso können wir die Wahrscheinlichkeiten $\mathbf{P}_B$ für die Behandlung B miteinander kombinieren. Der Unterschied ist, dass nun

$$\mathbf{P}_B(E) = \frac{234}{350} = 0,67 \ , \quad \mathbf{P}_B(E^c) = 0,33$$

gilt. Hier dominieren die Patienten aus der Gruppe K, entsprechend liegt der Wert

$$\mathbf{P}_B(E')$$

näher bei deren Erfolgswahrscheinlichkeit. Da offenbar Patienten mit kleinen Steinen besser zu behandeln sind (gleichgültig ob mit Methode A oder B), entsteht die festgestellt Umkehrung. Die Illustration macht dies anschaulich.

5.2 Variabilität

Bei der Befragung einer Bevölkerung möchte man feststellen, welches der relative Anteil p der Personen ist, die einer bestimmten Ansicht zustimmt („Ist Ihnen Freiheit wichtiger als Gerechtigkeit?“). Dazu nimmt man eine zufällige Stichprobe der Länge n und betrachtet

$$X := \text{ Anzahl der Ja-Antworten in der Stichprobe .}$$

X ist eine Zufallsvariable, wie ihr Wert ausfällt, hängt von der Wahl der Stichprobe ab.

Es liegt auf der Hand, wie man dann p schätzt, nämlich durch

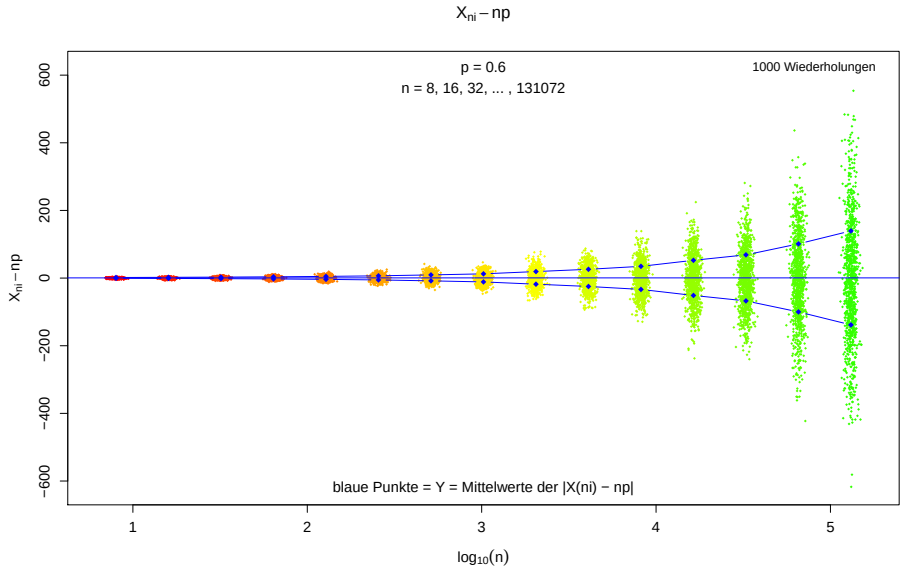
$$\widehat{p} := \frac{X}{n} \ ,$$

der relativen Häufigkeit der Ja-Antworten. Dagegen ist gar nicht offensichtlich, wie groß man n zu wählen hat, um die zufällige Variabilität von $\widehat{p}$ ausreichend klein zu bekommen. Es stellt sich etwa die Frage: In welchem Maß muss man n vergrößern, um die die Variabilität zu halbieren? Wir werden sehen, dass es dazu nicht langt, n zu verdoppeln.

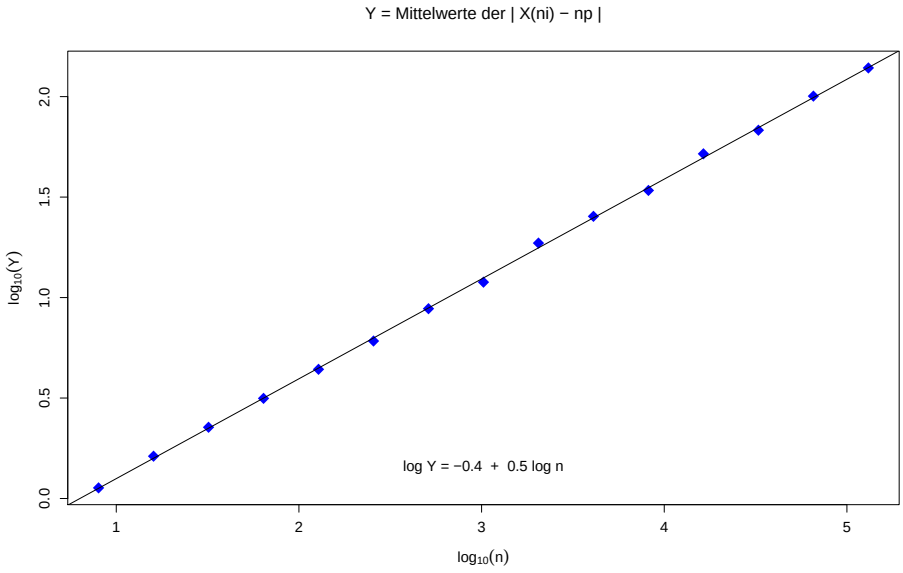
Um die Variabilität innerhalb X aufzuspüren, kann man ein Computerexperiment machen: Man erzeugt mit dem Rechner eine Anzahl künstlicher Stichproben $X_{n1}, \ldots, X_{nw}$, sagen wir $w = 1000$ Stück, und zwar von der Länge n und zu der Erfolgswahrscheinlichkeit p (oder, wenn p nicht bekannt ist, zur beobachteten Erfolgswahrscheinlichkeit $\widehat{p}$). Die Variabilität in den X_{ni} bei festem n wird dann durch ihre mittlere Abweichung von np angegeben, also durch die Zufallsvariable

$$Y_n := \frac{1}{w} \sum_{i=1}^{w} \left| X_{ni} - np \right| \ .$$

Wie ein Computerexperiment zeigt, wächst diese Größe nicht linear in n.



Auf logarithmischer Skala liegen die Werte recht genau auf einer Geraden der Steigung $1/2$, d.h. Y_n wächst proportional zu $\sqrt{n}$.



Ein vertieftes Verständnis entsteht, wenn man die Zufallsvariable von einem theoretischen Standpunkt betrachtet. Es ist nicht schwer, die Wahrscheinlichkeit

$$\mathbf{P}(X = k) \ ,$$

dass X den Wert k annimmt, auszurechnen ($k = 0, 1, \ldots, n$). Aber das brauchen wir hier gar nicht. Es geht uns nur um den mittleren Wert und die Variabilität von X.

Dazu berechnet man erstens den *Erwartungswert* von X. Er ist das gewichtete Mittel der möglichen Werte $k = 0, 1, \ldots, n$ von X, gewichtet mit ihren Eintrittswahrscheinlichkeiten,

$$\mathbf{E}[X] := \sum_{k=0}^{n} k\mathbf{P}(X = k) \ .$$

Wir werden später zeigen, dass sich

$$\mathbf{E}[X] = np$$

ergibt, ein Resultat, das einem auch der Hausverstand sagt. Mittels Division durch n folgt

$$\mathbf{E}[\widehat{p}] = \frac{\mathbf{E}[X]}{n} = p \ .$$

Man sagt, $\widehat{p}$ ist ein erwartungstreuer Schätzer von p. Im Mittel gibt also $\widehat{p}$ den richtigen Wert.

Um zweitens auch ein Maß für die Variabilität von X zu erhalten, wäre es wünschenswert, eine ähnlich einfache Formel für den Erwartungswert

$$\mathbf{E}\big[|X - np|\big] = \sum_{k=0}^{n} |k - np|\mathbf{P}(X = k)$$

zu erhalten. Exakt ist dies noch niemanden gelungen, und approximativ auch nur mit Aufwand. Deswegen betrachtet man ersatzweise die *Varianz* von X, den mittleren *quadratischen* Abstand zwischen X und np,

$$\mathbf{Var}[X] := \mathbf{E}\big[(X - \mathbf{E}[X])^2\big] = \sum_{k=0}^{n} (k - np)^2\mathbf{P}(X = k) \ .$$

Der Grund ist einfach, dass sich diese Größe rechnerisch viel besser erschließt, nicht nur in unserem Zusammenhang. Wir werden später sehen, dass sich

$$\mathbf{Var}[X] = npq \ , \quad \text{mit } q := 1 - p$$

ergibt.

Die Varianz ist noch ungeeignet als Kennzahl für die Variabilität von X, die Wirkung des Quadrierens muss man erst wieder rückgängig machen. Dies führt zur Wurzel der Varianz

$$\sqrt{\mathbf{Var}[X]} = \sqrt{npq} \ ,$$

sie ist gut geeignet. Sie heißt die *Standardabweichung* oder *Streuung* von X. Bemerkenswert ist, dass sie mit $\sqrt{n}$ wächst, ähnlich, wie wir das schon im Computerexperiment erfahren haben.

Daraus ergibt sich weiter, dass die Standardabweichung von $\widehat{p}$ gleich

$$\sqrt{\mathbf{Var}[\widehat{p}]} = \frac{\sqrt{\mathbf{Var}[X]}}{n} = \frac{\sqrt{pq}}{\sqrt{n}}$$

ist. Wir sehen: Um die Genauigkeit der Schätzung zu verdoppeln, muss man n vervierfachen!

Beispiel. Verstecken hinter dem Zufall. Bei heiklen Themen („Waren Sie schon untreu?“) kann man den relativen Anteil π der schwarzen Schafe nicht durch direktes Befragen ermitteln, die Verfälschung ist unkontrollierbar. Hier kann man sich wie folgt aushelfen: Die Testperson wirft zunächst eine gezinkte Münze, die Kopf und Wappen mit Wahrscheinlichkeit ρ bzw. $1-\rho$ zeigt. Bei Kopf soll die Frage richtig mit Ja oder Nein beantwortet werden, bei Wappen aber falsch, die Person soll Nein statt Ja und Ja statt Nein antworten. Hat die Testperson die Münze heimlich geworfen, so ist ihr mit ihrer Antwort nicht mehr auf die Schliche zu kommen, bei $\pi = 1/3$ etwa kann sie sich sicher fühlen (was ist mit $\pi = 1/2$?). Die Wahrscheinlichkeit für eine Ja-Antwort ist dann

$$p = \pi\rho + (1-\pi)(1-\rho) \ .$$

Es folgt

$$\pi = \frac{p+\rho-1}{2\rho-1} \ ,$$

ein natürlicher Schätzer für π ist daher

$$\widehat{\pi} = \frac{X/n+\rho-1}{2\rho-1} = \frac{X}{n(2\rho-1)} + \frac{\rho-1}{2\rho-1} \ .$$

Ihre Streuung ist dieselbe wie die von $X/n(2\rho-1)$. Nun zeigt ein Rechnung, dass

$$p(1-p) = \rho(1-\rho) + \pi(1-\pi)(2\rho-1)^2 \ ,$$

und für die Streuung von $\widehat{\pi}$ ergibt sich

$$\frac{\sqrt{p(1-p)}}{n(2\rho-1)} = \frac{1}{\sqrt{n}}\sqrt{\frac{\rho(1-\rho)}{(2\rho-1)^2} + \pi(1-\pi)} \ .$$

Der Preis also, der für den Schutz der Befragten (und damit eine verlässliche Auswertung) zu zahlen ist, ist eine kräftig vergrößerte Streuung. Der Faktor liegt für $\rho = 1/3$ zwischen $\sqrt{2} = 1,41$ und $\sqrt{2+1/4} = 1,5$. Man kann sich nun überlegen, wie hoch n zu wählen ist, um eine gewisse Genauigkeit zu erhalten.

Berechnung von Erwartungswert und Varianz. Man kann für die Zufallsvariable X die Wahrscheinlichkeiten $\mathbf{P}(X = k)$ und daraus den Erwartungswert und die Varianz direkt berechnen. Aufschlussreicher ist ein anderer Weg.

Wir arbeiten mit der Zerlegung

$$X = Z_1 + \cdots + Z_n$$

mit

$$Z_i := \begin{cases} 1\ , & \text{falls die } i\text{-te Person mit „Ja“ antwortet ,} \\ 0 & \text{andernfalls .} \end{cases}$$

Für jede Ja-Antwort wird also eine 1 gezählt, und das gibt in der Summe die X Ja-Antworten.

Nun gilt eine fundamentale Eigenschaft des Erwartungswertes, seine Linearität. Sie lautet

$$\mathbf{E}[cX + dY] = c\mathbf{E}[X] + d\mathbf{E}[Y]$$

für zwei reellwertige Zufallsvariable X, Y und reelle Zahlen c, d. Der Beweis ist nicht schwer zu führen (zumindest für Zufallsvariable, die nur endlich viele verschiedene Werte annehmen können), er findet sich in jedem Lehrbuch für elementare Stochastik.

Da in unserem Fall Z_i jeweils mit Wahrscheinlichkeit p den Wert 1 annimmt, gilt

$$\mathbf{E}[Z_i] = 0 \cdot \mathbf{P}(Z_i = 0) + 1 \cdot \mathbf{P}(Z_i = 1) = p$$

und deswegen aufgrund der Linearität

$$\mathbf{E}[X] = \mathbf{E}[Z_1] + \cdots + \mathbf{E}[Z_n] = p + \cdots + p = np\ .$$

Das ist bereits die erste Behauptung.

Für die Varianz läuft die Berechnung analog, nur etwas umfänglicher. Es gilt

$$\begin{aligned} \mathbf{E}[(Z_i - p)^2] &= (1-p)^2\mathbf{P}(Z_i = 1) + (0-p)^2\mathbf{P}(Z_i = 0) \\ &= q^2p + p^2q = pq(q+p) = pq \end{aligned}$$

und für $i \neq j$

$$\begin{aligned} &\mathbf{E}[(Z_i - p)(Z_j - p)] \\ &\quad = (1-p)^2\mathbf{P}(Z_i = 1, Z_j = 1) + (0-p)(1-p)\mathbf{P}(Z_i = 0, Z_j = 1) \\ &\qquad + (1-p)(0-p)\mathbf{P}(Z_i = 1, Z_j = 0) + (0-p)^2\mathbf{P}(Z_i = 0, Z_j = 0) \\ &\quad = q^2p^2 - pqqp - qppq + p^2q^2 = 0\ . \end{aligned}$$

Damit folgt, wieder unter Beachtung der Linearität

$$\begin{aligned}
\mathbf{E}\big[(X-np)^2\big] &= \mathbf{E}\big[\big((Z_1-p)+\cdots+(Z_n-p)\big)^2\big] \\
&= \mathbf{E}\Big[\sum_{i=1}^{n}\sum_{j=1}^{n}(Z_i-p)(Z_j-p)\Big] \\
&= \sum_{i=1}^{n}\sum_{j=1}^{n}\mathbf{E}\big[(Z_i-p)(Z_j-p)\big] \\
&= \sum_{i=1}^{n}\mathbf{E}\big[(Z_i-p)^2\big] + \sum_{i=1}^{n}\sum_{j\neq i}\mathbf{E}\big[(Z_i-p)(Z_j-p)\big] \\
&= npq\ .
\end{aligned}$$

Dies ist die Behauptung.

5.3 Kann das Zufall sein? Statistische Tests

Eine Grundidee der Statistik ist es, Daten als Realisierungen von Zufallsvariablen aufzufassen, über deren Wahrscheinlichkeitsverteilung man aus den Daten lernen will. Beim *statistischen Testen* trifft man eine *Hypothese* über die Verteilung und fragt: Liegen die beobachten Daten „im Rahmen“, oder ist hier ein Ereignis eingetreten, das unter der Hypothese so unwahrscheinlich ist, dass wir begründeten Zweifel am Zutreffen der Hypothese hegen sollten? Wir erläutern das Vorgehen an einem Beispiel.

Eine Botschaft ein und desselben Inhalts, es ging um den Vergleich des Erfolgs zweier Therapiemethoden T1 und T2, wurde in zwei unterschiedliche Darstellungsformen verpackt. In Form A wurde herausgestellt, wie groß jeweils der Prozentsatz der Patienten ist, bei denen Behandlung T1 erfolglos bzw. Behandlung T2 erfolgreich war, in Form B wurde der Akzent gerade umgekehrt gesetzt.

Von insgesamt 167 Ärzten, die an einer Sommerschule teilnahmen, wurden rein zufällig 80 ausgewählt, denen die Botschaft in der Form A vermittelt wurde, die restlichen 87 bekamen die Botschaft in der Form B mitgeteilt. Jeder der Ärzte hatte sich daraufhin für die Bevorzugung einer der beiden Therapiemethoden zu entscheiden. Das Ergebnis war:

	für Methode T1	für Methode T2	Summe
A	40	40	80
B	73	14	87
Summe	113	54	167

Die Daten zeigen: In der A-Gruppe gibt es im Verhältnis weniger Befürworter der Therapiemethode T1 als in der B-Gruppe (nämlich 40 : 40 gegen 73 : 14). Haben sich die Ärzte in ihrer Entscheidung durch die Form der Darstellung beeinflussen lassen? Ein Skeptiker könnte einwenden: „Ach was, auch ohne Beeinflussung kann ein derartiges Ergebnis zustande kommen, wenn der Zufall es will."

Um damit umzugehen, treffen wir folgende Hypothese: Die Form der Botschaft habe keinen Einfluss auf die Meinungsbildung der 167 Ärzte; es wäre so, als ob die einen 80 die Botschaft auf weißem, die anderen 87 eine wörtlich gleichlautende Botschaft auf blauem Papier bekommen hätten. Die Aufteilung der $80 + 87$ Formulare auf die 113 Befürworter von T1 und die 54 Befürworter von T2 wäre rein zufällig zustande gekommen. Wie wahrscheinlich ist dann eine so extreme Aufteilung wie die beobachtete?

Eine Veranschaulichung: Wenn aus einer Urne mit 80 weißen und 87 blauen Kugeln rein zufällig 113 Kugeln gezogen werden, wie wahrscheinlich ist dann ein so extremes Ergebnis wie das, nur 40 weiße Kugeln zu ziehen?

Die Anzahl X der weißen Kugeln ist eine Zufallsvariable, für die man die Wahrscheinlichkeiten $\mathbf{P}(X = k)$ ohne weiteres berechnen kann. X neißt hypergeometrisch verteilt. Mit $g = 80 + 87 = 167,\ w = 80,\ n = 113$ ergibt sich für ihre Erwartungswert

$$\mathbf{E}[X] = n \cdot \frac{w}{g} = 54.1 \ .$$

Die Wahrscheinlichkeit, ein Ergebnis zu erhalten, das mindestens so weit von 54 weg ist wie der beobachtete Wert 40, ist

$$\mathbf{P}\big(|X - 54| \geq |40 - 54|\big) = \mathbf{P}(X \leq 40) + \mathbf{P}(X \geq 68) = 5.57 \cdot 10^{-6} \ .$$

Wir halten fest: Angenommen die Hypothese trifft zu. Dann tritt ein Ergebnis, das so extrem ist wie das beobachtete, 6 mal in einer Million auf. Damit wird die Hypothese mehr als fragwürdig.

Man nennt die berechnete Wahrscheinlichkeit *den zu den Daten gehörigen p-Wert* oder auch das *beobachtete Signifikanzniveau*, zu dem die Hypothese abgelehnt wird.

Die eben beschriebene Vorgangsweise, bekannt als *Fishers exakter Test auf Unabhängigkeit*, ist ein typisches Beispiel eines *Permutationstests*: Man schreibt die beobachteten Daten in einem Gedankenexperiment dem reinen Zufall zu, indem man jede andere Aufteilung der A-und B-Formulare auf die T1- und T2-Befürworter (jede andere Permutation der 167 Formulare) als ebenso wahrscheinlich ansieht wie die beobachtete.