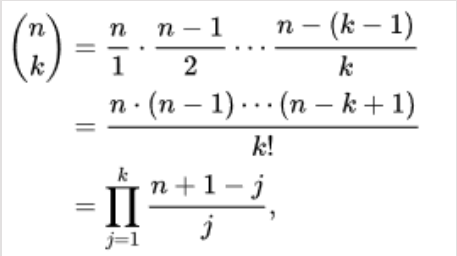


$$\begin{aligned}\binom{n}{k} &= \frac{n}{1} \cdot \frac{n-1}{2} \cdots \frac{n-(k-1)}{k} \\ &= \frac{n \cdot (n-1) \cdots (n-k+1)}{k!} \\ &= \prod_{j=1}^{k} \frac{n+1-j}{j},\end{aligned}$$

Entwickler-Notizen

Mathe für Programmierer

kurz & bündig

TEKTUR

Version: 58

de-DE

Dieses Buch wurde mit Tektur CCMS erstellt. Tektur ist ein einfach zu bedienender kollaborativer Editor um DITA[1)] Inhalte erstellen, als PDF ausgeben und pflegen zu können. Die Eingabe erfolgt dabei per WYSIWYG[2)] mit geführter Benutzerinteraktion. Die Inhalte werden als einzelne Topics verwaltet, die in verschiedenen Maps referenziert werden können; Stichwort: Topic Based Authoring[3)].

Sonstige Features: Rechte- und Rollensystem, Versionskontrolle, konfigurierbarer Workflow mit Review & Approval Funktionen. Auf dem Entwicklerblog[4)] kann man sich über den Fortschritt informieren.

► *Dieses Buch dient in erster Linie als Test für Tektur CCMS. Der Inhalt ist recht schnell entstanden, so dass der Feinschliff an mancher Stelle vielleicht noch fehlt... Trotzdem könnte es für den einen oder anderen XML Entwickler ganz interessant sein. Für mich dient der Text in erster Linie als Gedächtnisstütze.*

Die Quelltexte sind aus "didaktischen" Gründen an mancher Stelle in deutsch gehalten, was natürlich für ein richtiges Programm nicht soviel Sinn machen würde.

1) https://de.wikipedia.org/wiki/Darwin_Information_Typing_Architecture
2) https://de.wikipedia.org/wiki/WYSIWYG
3) https://en.wikipedia.org/wiki/Topic-based_authoring
4) http://www.tekturcms.de

Inhalt

1 Wahrscheinlichkeiten

In diesem Abschnitt ist zentrale Formel die Berechnung des Binominalkoeffizienten, mit dem man ausrechnen kann, auf wieviele verschiedene Arten man eine Teilmenge - mit einer fixen Zahl von Elementen - aus einer Obermenge auswählen kann. Das gewählte Beispiel mit den Kleidungsstücken ist mir willkürlich eingefallen. Der Klassiker ist natürlich das sogenannte Urnenmodell[5].

Der Binominalkoeffizient, kurz n über k, hat `k!` *im Nenner und im Zähler die "Umkehrung bis n" - das ist jetzt meine Eselsbrücke - diesen Ausdruck wird man in der Literatur wohl so nicht finden...*

$$\binom{n}{k} := \frac{n(n-1)(n-2)\cdot\ldots\cdot(n-k+1)}{1\cdot 2\cdot\ldots\cdot(k-1)k} = \frac{\prod_{i=0}^{k-1}(n-i)}{k!}.$$

Bild 1: Binominalkoeffizient als Bruch geschrieben

1. Variation mit Wiederholung

   Klassisches Abzählproblem: Auf wieviel verschiedene Arten können sich 12 Personen kleiden, wenn insgesammt 8 verschiedene Kleidungsstücke pro Person zur Auswahl stehen und alles getragen werden muss?

   ► 8^{12}

2. Variation ohne Wiederholung

5) *https://de.wikipedia.org/wiki/Urnenmodell*

Wahrscheinlichkeiten

Beispiel wie vorher, allerdings zieht jede Person nur ein Kleidungsstück an und es stehen nur ingesamt 20 zur Verfügung.

► **20*19*18*17*16*15*14*13*12*10*8*7**

3. Kombination mit Wiederholung

Es stehen 20 Kleidungsstück-Arten zur Verfügung und die Personen können mehrere übereinander anziehen und auch dieselben, aber maximal 4. Auf wieviel verschiedene Arten kann sich die Gruppe kleiden?

►

$$\binom{n+k-1}{k}$$

4. Kombination ohne Wiederholung

Gleiches Szenario wie vorher, jedoch stehen nicht unendlich viele Kleidungsstücke zu Verfügung, sondern nur 20.

►

$$\frac{n!}{(n-k)! \cdot k!} = \binom{n}{k}$$

5. Permutation

Für die 8 verschiedenen Personen stehen 8 Badehosen zur Verfügung, die alle eine unterschiedliche Farbe haben. Jede Per-

son will bekleidet sein, wieviele verschiedene Möglichkeiten der Zuordnung gibt es?

► **8!**

6. Fixpunktfreie Permutation

Wenn wir nun davon ausgehen, dass jede der 8 Personen eine Lieblingsfarbe hat - wir aber keinen bevorzugen wollen - also erreichen wollen, dass jeder eine Badehose anzieht, die nicht in seiner Lieblingsfarbe ist, wieviele Möglichkeiten der Bekleidung gibt es dann?Abstrakt gesehen wollen wir die "fixpunktfreien Permutationen" zählen.

►

$$D(n) = \sum_{i=0}^{n} (-1)^i \frac{n!}{i!}.$$

Auf die Formel kommt man, indem man die diejenigen Permutationen zählt, die eine bestimmte Anzahl `k` von Elementen fest lässt. Das sind genau die Permutationen der übrigen `n-k` Elemente, also `(n-k)!` Stück. Mit Hilfe von ein bisschen Mengenleere und dem Satz von Inklusion und Exklusion kann man dann die Formel weiter herleiten.

7. Zerlegung einer Permutation

Jede Permutation lässt sich in disjunkte Zyklen zerlegen. Für unser Bekleidungsbeispiel könnte man sich das so vorstellen, dass jede Person ihre Badehose auszieht und diese an eine bestimmte, zuvor schon fest ausgesuchte Person weitergibt.

Wahrscheinlichkeiten

► **Wären die Zyklen nicht disjunkt, so könnte es vorkommen, dass eine Person nackt dasteht und eine Person mehrere Badehosen übereinander trägt.**

```
==================================
Alex | Horst  |  Anne   | Sepp
----------------------------------
rot  |  grün  |  blau   | braun
grün |  blau  |  braun  | orange
==================================
Dieter| Gerd     | Alfons | Jutta
----------------------------------
orange| schwarz | gelb    | ocker
ocker | gelb    | schwarz | rot
```

Betrachten wir die folgenden zwei Zuordnungen, welche eine Permutation darstellen. Zerlegt in Zyklen bekommen wir:

<rot, grün, blau, braun, orange, ocker, rot> **als ersten Zyklus und** *<schwarz, gelb, schwarz>* **als zweiten Zyklus.**

8. Lottoziehung

► Der Klassiker in der Wahrscheinlichkeitsrechung ist die Lottoziehung. Jeder weiss, dass die Wahrscheinlichkeit einen 6er im Lotto zu haben irgendwas mit *1:14 Mio* ? ist. Das entspricht einer Kombination ohne Wiederholung bzw. dem Binominlakoeffizienten "n über k" - 49 über 6 - oder "Wie oft kann man eine unterschiedliche 6-elementige Teilmenge aus einer 49-elementigen Menge auswählen?

► Ist die Aufgabenstellung dagegen etwas anders gestellt, sieht man sich schnell mit komplizierteren Abzählungen

konfrontiert, bspw: **Wie hoch ist die Wahrschlichkeit 5 Richtige mit Zusatzzahl zu bekommen?**

> *Für die Ziehung der Superzahl hat man 10 Möglichkeiten, da diese zwischen 0 und 9 liegen muss. Kombinert mit der Gesamtwahrscheinlickeit und unter Berücksichtigung, dass tatsächlich nur eine Möglichkeit gezogen wird, berechnet sich die Wahrscheinlichkeit für das erfolgreiche Ereignis anhand dieser Formel:*

$$\frac{1}{10 \cdot \binom{49}{6}} = \frac{1}{139\,838\,160}.$$

Das klärt aber noch nicht die Aufgabenstellung. Offensichtlich gibt es bei der **5 mit Zusatzzahl** mehr als eine richtige Gewinnnkombination. Diese Kombinationen müssen wir zählen und durch die Anzahl aller Ereignisse teilen, um auf die gesuchte Wahrscheinlichkeit zu kommen.

► **Für einen Lotto-5er gibt es 6*43 Möglichkeiten = 258. Also ist die Wahrscheinlichkeit 258 / 139838160, was wohl sehr, sehr unwahrscheinlich ist - ohne das jetzt näher auszurechnen.**

2 Abbildungen

Hier geht es um Abbildungen - wie bei Funktionen wird eine Eigabe mittels einer Funktion (oder einem Programm) in eine Ausgabe überführt.

Wir unterscheiden zwischen injektiven, surjektiven und bijektiven Abbildungen. Bijektive Abbildungen sind beides, injektiv und surjektiv.

1. Surjektive Abbildungen

   Bei einer surjektiven Abbildungen wird jedes Element der Bildmenge getroffen. D.h. es gibt keine "herrenlosen" Werte in der Ergebnismenge.

   ► **f(M) = N**

2. Injektive Abbildungen

   Bei einer injektiven Abbildung ist sichergestellt, dass ein Wert des Definitionsbereichs genau auf einen eindeutigen Wert des Bildbereichs abgebildet wird.

   ►

   $$x \neq y \Rightarrow f(x) \neq f(y),$$

3. Bijektive Abbildungen

   Bijektive Abbildungen sind injektiv und surjektiv

3 Bäume

Ein Baum ist ein zusammenhängender Graph, der keine Kreise enthält. Viele Datenstrukturen sind als Baumstruktur realsiert, z.B. eine Binärbaum[6] auf Algorithmusebene oder XML auf Modellierungsebene.

► *Ein minimaler ausfspannender Baum ist ein Teilgraph, der alle Knoten des Baums besucht, mit einer minimaler Anzahl von aufsummierten Kantengewichten - im Vergleich zu anderen aufspannenden Bäumen.*

1. Zum Auffinden eines MST könnte man nach folgenden Algorithmus vorgehen:
   - 1.1. Zuerst ist die Menge MST leer
   - 1.2. Sortiere die Kanten des Graphen mach aufsteigendem Kantengewicht
   - 1.3. Nimm die erste Kante weg und füge sie zu MST hinzu - nur wenn dadurch in MST kein Kreis entsteht
   - 1.4. ahre solange bis MST alle Knoten des Graphen enthält

Ein Kantengewicht könnte z.B. ein Preis für eine bestimmte Rohstoffart sein, die nach einem bestimmten Prozess (als Entscheidungsgraph modelliert) ein Ergebnisprodukt liefert.

3.1 Vergleich von Bäumen

Wenn man zwei Bäume vor sich hat, ist es nicht immer leicht zu erkennen, ob sie den gleichen Sachverhalt darstellen. Bspw. kann ein

6) https://de.wikipedia.org/wiki/Bin%C3%A4rbau

Bäume

Entscheidugsbaum so in der Ebene verdreht sein, dass er nicht mit einem anderen Baum vergleichbar ist.

► *Aus diesem Grund ist es ratsam die Baumdarstellung zu normalisieren. Grds. sind zwei Bäume gleich, wenn die folgenden Kriterien erfüllt sind:*

1. Die beiden Bäume haben dieselbe Anzahl von Knoten
2. Die beiden Bäume haben dieselbe Anzahl von Kanten
3. Eine Kanze in Baum ein, muss auch genauso eine Kante in Baum zwei haben

   ► **Der Vergleich funktioniert natürlich nur, wenn man die Knoten der Bäume geeignet durchnumeriert.**

4. Man kann auch den sog. Code des Baums bestimmen. Das ist eine wohldefinierte Klammerfolge:
   - 4.1. Es gibt genau soviele öffnende wie schliessende Klammern
   - 4.2. Es existiert eine Totalordnung auf den Klammerausdrücken, d.h. A < B, wenn

     ► **A ist Anfang von B, mit A = () und B = ()**

     ► **Wenn sich die erste Klammer, in der sich A und B unterscheiden, bei A öffnend und bei B schliessend ist.**

5. Um nun zu testen, ob man einen echten Baum vor sich hat, geht man nach den folgenden Schritten vor:
   - 5.1. Kriterium für einen gepflanzten Baum: Es gibt gnausoviele öffnende, wie schliessende Klammern?
   - 5.2. Kriterium für einen Wurzelbaum: Sind die Unterbäume jedes Knotens lexikographisch sortiert?

5.3. Kriterium für einen echten Baum: Hat die Wurzel des Baum minimale Exzentrität (= größter Abstand eines Knotens zu einem anderen)

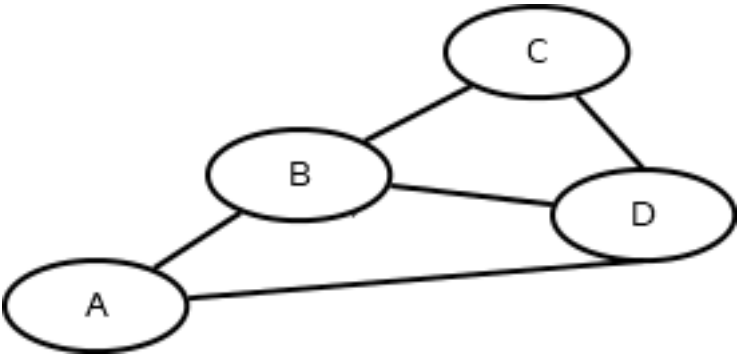
► **Minimale Exzentrität bedeutet also, dass man die Wurzel eines Baumes vor sich hat. Ein nach den obigen Kriterien gefundener Baum ist eindeutig definiert.**

4 Eigenschaften von Graphen

Graphem können ganz interessante Eigenschaften haben, die man mit cleveren Algorithmen bestimmen kann. Wenn man z.B. von einem Knoten zu einem anderen läuft, dann unternimmt man einen Spaziergang.

Anzahl der Spaziergänge mit bestimmter Länge

Es gibt eine recht interessante Methode, wie man die Anzahl aller möglichen, unterschiedlichen Spaziergänge zwischen zwei Knoten mit der Länge k bestimmen kann. Dazu stellt man die Adjazenzmatrix auf, diese sieht bei einem Graphen mit 4 Knoten `(A,B,C,D)` folgendermassen aus:



	A	**B**	**C**	**D**
A	0	1	0	1
B	1	0	1	1

C	0	1	0	1
D	1	1	1	0

Um die Anzahl der Spaziergänge mit Länge k zu bestimmen, führt man eine Martizenmultiplikation der Länge `k` aus.

Für die Anzahl der möglichen Spaziergänge der Läge `3` wird also `A * A * A` bestimmt. Eine Matrizenmultiplikation nach dem Falkschen Schema[7] so:

Matrizenmultiplikation nach dem Falkschen Schema

-	-	-	-	-	**A**	**B**	**C**	**D**
-	-	-	-	-	0	1	0	1
-	-	-	-	-	1	0	1	1
-	-	-	-	-	0	1	0	1
-	-	-	-	-	1	1	1	0
-	0	1	0	1	2	1	2	1
-	1	0	1	1	1	3	1	2
-	0	1	0	1	2	1	2	1
-	1	1	1	0	1	2	1	3
A	0	1	0	1	2	5	2	**5**

7) https://de.wikipedia.org/wiki/Falksches_Schema

Eigenschaften von Graphen

B	1	0	1	1	5	4	5	5
C	0	1	0	1	2	5	2	5
D	1	1	1	0	**5**	5	5	4

In der Teil-Tabelle rechts unten kann man nun ablesen, dass es `5` Möglichkeiten für Spaziergänge von `A` nach `D` gibt.

▶ *Als Programmierer ist diese Information interessant, wenn man z.B. Aussagen über den Aufwand einer Testabdeckung veranschlagen möchte. Hier würde ein Spaziergang von einem Ausgangszustand `A` zu einem Endzustand `D`, der über verschiedene Fallunterscheidungen erreicht werden kann, den Graphen darstellen. Fallunterscheidungen könnten z.B. auch verschiedene Eingabedialoge in einer GUI-Anwendung sein...*

4.1 Gleichwertigkeit von Graphen

Zwei Graphen sind gleich, wenn sie isomorph sind. Man notiert einen Graphen mit der sog. Valenzsequenz:

`(3,2,2,1,1,1)`

Hier sind die Knotengrade (= Anzahl der Kanten an einem Graphen) der Größe nach notiert. Wenn dann die Anzahl der Knoten und die Anzahl der Kanten, der zu vergleichenden Graphen gleich ist, und es jeweils eine Entsprechung der verbunden Kanten in beiden Graphen gibt, dann sind die Graphen gleich bzw. isomorph.

Eigenschaften von Graphen

► *Damit die Valenzsequenz valide ist, müssen die folgenden Bedingungen erfüllt sein:*

1. Die Summe aller Knotengrade ist gerade - eine Regel, die vom sog. Handshake Lemma[8] abgleitet ist.
2. Nach jedem Knoten in der Valenzsequenz gibt es mindestens soviele Knoten, wie (Knotengrad - 1) des aktuell betrachteten Knoten.
3. Jeder Knotengrad muss natürlich positiv sein

   ► **Ein weiterer interessanter Umstand, ist die Tatsache, dass man schrittweise entsprechende Knoten in zwei gleichwertigen Graphen löschen kann, und es entstehen wieder isomporphe Graphen. Umgekehrt kann man sukkzessive isomorphe Graphen konstruieren. Beim Verfahren von Havel und Hakimi[9] macht man sich das zu Nutze, um zu prüfen, ob eine gegebene Valenzsequenz einen validen Graphen darstellt.**

4. Verfahren von Havel und Hakimi

► *Wir betrachten die Valenzsequenz von oben mit* `(3,3,3,2,2,1)` *. Schrittweise werden nun die ersten Knoten entfernt und die Valenzsequenz reduziert. Es muss immer eine valide Valenzsequenz übrig bleiben, ansonsten hat man bewiesen, dass es sich um keine gülige Valenzsequenz für einen Graphen handelt.*

4.1. Schritt 1

8) https://de.wikipedia.org/wiki/Handschlaglemma
9) https://de.wikipedia.org/wiki/Havel-Hakimi-Algorithmus

Eigenschaften von Graphen

► **Die Valenzsequenz unterhalb der zugehörigen Knoten aufgeschrieben, ergibt diese Tabelle:**

v1	v2	v3	v4	v5	v6
3	3	3	2	2	1
	2	2	1	2	1

4.2. Schritt 2

► **Jetzt sortiert man die Sequenz neu und merkt sich dabei die Knotenbezeichner**

v1	v2	v3	v5	v4	v6
	2	2	2	1	1
		1	1	0	0

Schliesslich erhalten wir eine Kante zwischen den Knoten `v3` **und** `v5` **... Mit dieser Information kann man sukzessive den Graphen grafisch aufbauen, indem man bis Schritt 1 zurückgeht.**

► *Mit dem Satz von Erdös und Gallai kann man mittels einer Formel prüfen, ob eine gegebene Valenzsequenz einen validen Graphen darstellt.*

5. Verfahren von Erdös und Gallai

► **Um zu beweisen, dass es sich bei einer gegebenen Sequenz um keinen Graphen handelt, genügt es zu**

beweisen, dass für ein `i` die folgende Formel nicht erfüllt ist.

$$(\sum_{j=1}^{i} d_j) - i(i-1) \leq \sum_{j=i+1}^{n} min\{d_j, i\}$$

Die `dj` stellen dabei die Knotengrade der Sequenz an der Stelle `j` dar.

4.2 Hamiltonkreis vs. Eulertour

Jeder kennt das Königsberger Sieben-Brückenproblem. Der korrekte Fachbegriff lautet dabei Eulertour, bei der jede Kante (=Brücke) eines Graphen genau einmal durchlaufen wird. Ein Spezialfall ist der Hamiltonkreis. Hier darf auch nur jeder Knoten einmal besucht werden.

1. Es gibt u.a. zwei Merkmale die bewiesen werden müssen
   - 1.1. Der Graph ist zusammenhängend, das heisst es gibt einen Weg von jedem Koten zu jedem Knoten.
   - 1.2. Die Anzahl der Kanten, die vom Knoten hin-/wegführen ist gerade.

   ► **Paradebeispiel für das Finden einer Eulertour ist das Haus vom Nikolaus. Da das Haus vom Nikolaus 2 Knoten vom Grad 3 hat, gibt es keine Eulertour, sondern nur einen Eulerkreis.**

2. Zum Auffinden einer Eulertour kann man nach dem folgendem Schema vorgehen:
   - 2.1. Starte mit einer Kante eines beliebigen Knoten
   - 2.2. Markiere die Kante und merke Sie Dir in einer Liste

2.3. Laufe auf dieser Kante zum nächsten Knoten

2.4. Wiederhole die Prozedur solange es noch Knoten gibt, die nicht in der Liste sind

► *Wichtg ist das die Liste an der Stelle aktualisiert wird, von der man einen neuen Knoten als Startknoten sucht. Es werden sozusagen Kreise gefunden und aneineinandergereiht.*

3. Praktisch könnte man den Algorithmus im folgenden Szenario einsetzen:

Die Geister eines Pac-Man-artigen Spiels laufen auf einer Eulertour um sich nicht gegenseitig zu blockieren, bei unterschiedlichen Startpunkten mit gleicher Geschwindigkeit.

4.3 Matchingprobleme

Bei einem bipartiten Graphen, bei dem die Knoten in zwei "Farb"-klassen aufgeteilt werden können, findet man sog. Matchingprobleme. Jeder Knoten einer Farbklasse kann nämlich einem Knoten aus der anderen Farbklasse zugeordnet (= gematcht) werden.

1. Identifikation eines bipartiten Graphen

1.1. Markiere abwechselnd besuchte Knoten mit zwei Farben.

1.2. Stösst man auf einen Knoten mit derselben Farbe, in der man markieren will so ist der Graph nicht bipartit.

1.3. Fahre solange fort bis alle Knoten markiert sind.

► **Man stellt dabei fest, dass ein Graph genau dann bipartit ist, wenn er keine Kreise ungerader Länge enthält.**

2. Heiratssatz von Hall

2.1. Men Propose - Women Dispose

2.2. Damen und Herren verfassen Präferenzlisten bzgl. des anderen Geschlechts.

2.3. Die Liste der Herren wird durchlaufen und der aktuelle Herr macht der ersten Dame auf seiner Liste einen Antrag. Lehnt sie ab, streicht er sie von seiner Liste ansonsten geht das Paar eine Beziehung ein, solange kein Herr der Dame einen Antrag macht, den sie besser findet.

2.4. Fahre solange fort bis die Präferenzlisten der Herren durchlaufen sind oder kein Pärchen mehr gebildet werden kann.

2.5. Diese Algorithmus liefert eine für den Mann optimale Pärchenbildung.

3. Perfektes Matching

3.1. **Das Matching ist maximal**, wenn man keinen augmentierenden Weg mehr findet. Dabei spaziert man auf einem augmentierender Weg, wenn man abwechselnd gematchete und ungematchete Knoten besucht, sowie auf letzteren startet und endet.

3.2. **Satz vom König:** Die Anzahl der Matchingkanten in einem maximalen Matching stimmt mit der Anzahl der Knoten einer minimalen Knotenüberdeckung überein. (Knotenüberdeckung: Die Knoten müssen an allen Kanten des Matchings beteiligt sein)

3.3. **Heiratssatz von Hall:** Die Nachbarschaftsmenge einer jeden Teilmenge muss größer sein, als die Teilmenge selbst.

5 Lösen von Gleichungssystemen

Gleichungssysteme kann man lösen, indem man schrittweise nach einer Variablen auflöst und diese dann in den anderen Gleichungen einsetzt.

Mit diesem Vorgehen gelangt man schnell in eine Sackgasse, wenn man mit der falschen Variablen anfängt, oder die Gleichung zu komplex ist.

Es gibt verschiedene algorithmische Verfahren, wie man dieses Problem gut in den Griff bekommen kann.

Gauss-Jordan Algorithmus

Beim Gauss-Jordan-Algorithmus bringt man das Gleichungssystem zunächst in Matrixform. Dabei werden die Koeffizienten (K) der Variablen in den linken Spalten aufgelistet und die Konstanten der Ergebniswerte (E) in der Spalte ganz rechts.

Das sieht dann so z.B. so aus:

K	K	-K	K	-K	E
-K	K	K	.K	K	E
K	K	K	-K	-K	E
-K	-K	-K	K	K	E

Jetzt versucht man einen Koeffizienten zu einer 1 zu machen (wenn man keinen findet), indem man ein Vielfaches des Koeffizienten von allen Zeilen abzieht. Beispielsweise könnte man alle Zeilen durch zwei teilen, wenn man eine 2 aber keine 1 findet.

Hat man eine Spalte mit einer 1 gefunden oder erzeugt, dann macht man alle anderen Werte dieser Spalte zu einer 0, indem man ein Vielfaches der 1-Spalte von den anderen abzieht.

So verfährt man weiter, bis man in allen Spalten jeweils eine 1 und sonst nur Nullen erhält und hat das Gleichungssystem gelöst.

LU-Zerlegung

Bei der LU-Zerlegung teilt man die Gleichungs-Matrix in eine untere (L) und eine obere (O) Koeffizienten-Dreiecksmatrix auf, das sieht dann z.B. so für die untere Matrixhälfte aus:

1	0	0
K	1	0
K	K	1

und so für die obere:

K	K	K
0	K	K
0	0	K

Um diese Matrizen zu erhalten fängt man mit der unteren an und versucht schrittweise Nullen unter die 1 in der ersten Zeile zu bringen, indem an wieder ein Vielfaches der ersten Zeile von den anderen abzieht.

Hat man die erste Spalte in Form gebracht, so macht man dasselbe in der zweiten Spalte.

Das gesuchte Gleichungssystem löst man dann, indem man die zwei Teil-Gleichungssystem der Dreiecksmatrizen ineinander auflöst, nämlich:

Lösen von Gleichungssystemen

`L` *steht für die untere Dreiecksmatrix (lower).* `y` *sind die Koeffizienten in der Matrix.* `b` *ist der Lösungsvektor des Gleichungssystems.*

$$Ly = b$$

und

`U` *steht für die untere Dreiecksmatrix.* `x` *sind die Koeffizienten des ursprünglichen Gleichungssystems.* `y` *sind die Koeffizienten, die wir mit* `Ly = b` *schon bestimmt haben.*

$$Ux = y$$

Cholesky Faktorisierung

Die Cholesky Faktorisierung ist ein Spezialfall der LU-Zerlegung. Sie ist genau dann gegeben, wenn die obere und untere Dreiecksmatrix gleich sind:

`T` *gibt an, dass die Matrix* `L` *transponiert ist.*

$$U = L^T$$

Lösen von Gleichungssystemen

Die Eingabematrix `A` ist somit das Matrixprodukt aus unterer `L` und transponierter unterer Dreiecksmatrix.

$$A = LL^T$$

Die obere Dreiecksmatrix ist die Transponierte der unteren. Ist dieses Szenario gegeben, dann kann man das Gleichungssystem mittels Formeln lösen:

$$\ell_{i,j} = 0$$

$$\ell_{i,i} = \sqrt{a_{i,i} - \sum_{k=1}^{i-1} \ell_{i,k}^2}$$

$$\ell_{i,j} = \frac{1}{\ell_{j,j}} \left(a_{i,j} - \sum_{k=1}^{j-1} \ell_{i,k}\ell_{j,k} \right)$$

Mit diesen Formeln kann man auch bestimmen, ob eine gegeben Matrix positiv definit ist. Das ist genau dann der Fall, wenn die Formeln ein sinnvolles Ergebnis liefern, und die Berechnung möglich ist. Bspw. darf unter der Wurzel kein negativer Wert stehen.