Inhaltsverzeichnis

1 Einleitung

Die Diskrete Mathematik ist ein noch relativ junges Teilgebiet der Mathematik, das sich als eigenständige Disziplin erst in der zweiten Hälfte des 20. Jahrhunderts etabliert hat. Dabei war die Diskrete Optimierung mit ihren beachtlichen Anwendungserfolgen die treibende Kraft. Untersuchungsgegenstand der Diskreten Mathematik sind im Wesentlichen endliche Strukturen. Die Diskrete Mathematik hat aber auch einen engen Bezug zur theoretischen Informatik. Viele Beweise sind konstruktiv und damit einem Algorithmus verwandt. Die meisten Probleme und Fragestellungen sind einfach zu verstehen, die Lösungen aber oft sehr schwierig.

Das Bonner Forschungsinstitut für Diskrete Mathematik hat die Entwicklung der Diskreten Mathematik in den vergangenen Jahrzehnten maßgeblich mitgestaltet. Die Forschungsschwerpunkte des Instituts sind die Kombinatorische Optimierung und Anwendungen im Chip-Design. Theorie und Anwen-

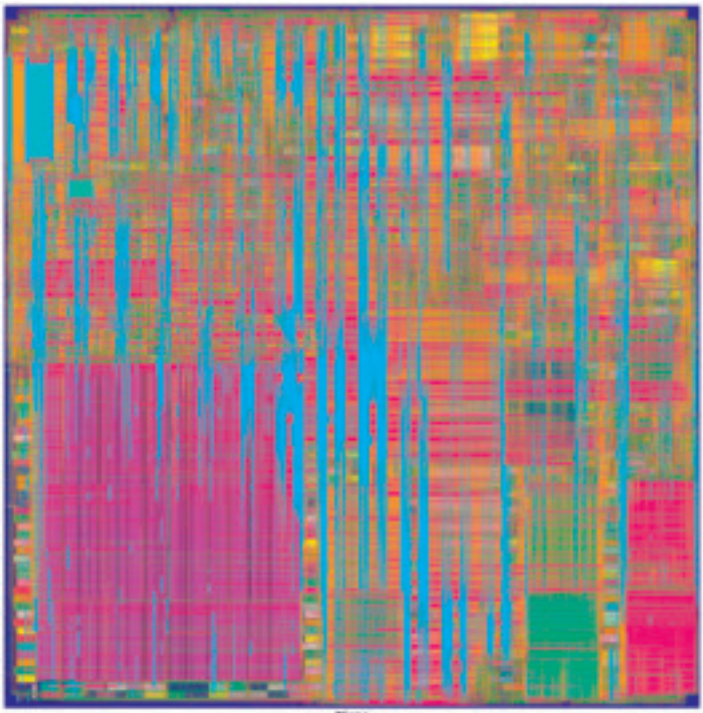
dung sind hier aufs engste verzahnt. Nicht zuletzt die Anwendungserfolge ermöglichen eine exzellente Ausstattung des Instituts, unter anderem mit einer der weltbesten Bibliotheken des Gebiets und natürlich dem Arithmeum. Dieses zeigt mit seiner einzigartigen Sammlung historischer Rechenmaschinen und -bücher bis hin zu neuester Chiptechnologie die Entwicklung des Rechnens und bereichert darüberhinaus mit Konzerten und Kunstausstellungen die Bonner Kulturlandschaft.

Die Diskrete Mathematik ist in Bonn Teil der Bachelor- und Masterstudiengänge in Mathematik und Informatik. In beiden Fächern ist eine Spezialisierung auf Diskrete Mathematik möglich. Das Lehrveranstaltungsangebot besteht aus regelmäßigen Vorlesungen, Seminaren und Praktika sowie natürlich den Abschlussarbeiten. Als studentische Hilfskräfte und im Rahmen ihrer Abschlussarbeiten wirken viele Studentinnen und Studenten in den Forschungsprojekten des Instituts mit. Diese Arbeiten tragen oft wesentlich zu neuen Verfahren bei, mit denen zukünftige Chips optimiert werden.

Diese Broschüre soll als erste Einführung in die Diskrete Mathematik, das Bonner Institut und das hiesige Studium dienen. Sie kann selbstverständlich nicht alle Fragen beantworten. Bei weiterem Interesse finden Sie auf den letzten Seiten zusätzliche Informationsquellen und Kontaktadressen. Gerne beantworten wir Ihre Fragen in einem persönlichen Gespräch.

2 Was ist Diskrete Mathematik?

Die Diskrete Mathematik beschäftigt sich mit endlichen Mengen und Strukturen. Diese sehr allgemeine Beschreibung der Diskreten Mathematik wollen wir in den nächsten fünf Abschnitten mit mehr Leben füllen. Wir werden einige ausgewählte, aber typische Fragestellungen der Diskreten Mathematik vorstellen. Dabei beschränken wir uns nicht nur auf die Beschreibung der Probleme, sondern geben vielfach auch vollständige Beweise sowie Algorithmen zur Lösung der Probleme an. Gerade der algorithmische Aspekt in der Diskreten Mathematik ist wesentlicher Antrieb zum Beweis neuer struktureller Aussagen. Daher haben konstruktive Beweise in der Diskreten Mathematik eine viel größere Bedeutung als in vielen anderen Teilgebieten der Mathematik.



2.1 Der Vier-Farben-Satz

1852 stellte Francis Guthrie dem Mathematikprofessor De Morgan eine scheinbar sehr einfache Frage: Ist es stets möglich, mit vier oder weniger Farben die Länder einer beliebigen Landkarte so zu färben, dass benachbarte Länder unterschiedliche Farben erhalten? Die Länder sollen dabei zusammenhängende Gebiete sein (Exklaven oder Inseln sind also nicht erlaubt), und zwei Länder sollen nur dann als benachbart betrachtet werden, wenn sie eine gemeinsame Grenzlinie haben. Beispiele für solche Färbungen sehen wir in Abbildung 1. Zum einen ist dort eine Deutschlandkarte mit ihren 16 Bundesländern (aber ohne die Exklave Bremerhaven und die Inseln) mit vier Farben gefärbt. Zum anderen ist dort eine kompliziertere künstliche Landkarte zu sehen, die mit vier Farben gefärbt ist.

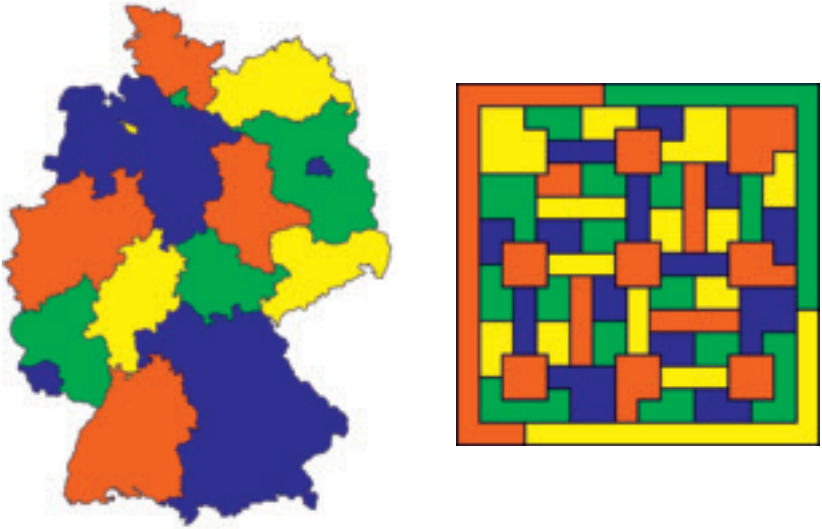


Abbildung 1: Mit vier Farben gefärbte Landkarten

Es hat über hundert Jahre gedauert, bis die Antwort auf Guthries Frage endlich gefunden wurde: Im Jahre 1976 bewiesen Appel und Haken das folgende Ergebnis, das „Vier-Farben-Satz“ genannt wird und einer der bekanntesten Sätze der Diskreten Mathematik ist.

Satz 1 ***(Vier-Farben-Satz)*** *Die Länder einer jeden Landkarte können so mit vier Farben gefärbt werden, dass benachbarte Länder unterschiedliche Farben erhalten.*

Der Beweis dieses Satzes von Appel und Haken ist mehr als 700 Seiten lang und umfasst eine riesige Fallunterscheidung, die so umfangreich ist, dass nur mit Hilfe eines Computers alle diese Fälle überprüft werden können.

1996 haben Robertson, Sanders, Seymour und Thomas einen modifizierten Beweis gefunden, der mit einer kleineren Fallunterscheidung auskommt. Aber auch hier sind noch so viele Fälle zu überprüfen, dass dies nur mit Computerhilfe möglich ist.

Wie lässt sich nun ein Ergebnis wie der Vier-Farben-Satz mit Methoden der Diskreten Mathematik beweisen? Zunächst stellt man fest, dass die Form und Größe der Länder offensichtlich keine Rolle spielt. Allein wichtig für das Problem ist zu wissen, welche Länder es gibt und welche Länder eine gemeinsame Grenzlinie haben, die erfordert, dass beide Länder mit unterschiedlichen Farben gefärbt werden müssen. Damit kann man das Problem als ein Problem in Graphen formulieren.

Ein *Graph* besteht aus einer Menge von *Knoten* und einer Menge von *Kanten*, wobei jede Kante ein Paar von Knoten ist. Alle für das Färbungsproblem relevanten Informationen lassen sich nun einfach in einem Graphen darstellen, indem man als Knoten die Länder nimmt und als Kanten alle Paare von Ländern, die eine gemeinsame Grenzlinie haben. Graphen lassen sich gut veranschaulichen, indem man für jeden Knoten einen Punkt malt und für jede Kante eine Linie malt, die diese beiden Punkte verbindet. Die Linien dürfen dabei beliebige Form haben, und sie dürfen sich auch in endlich vielen Punkten schneiden. Allerdings darf jede Linie außer den beiden Knoten, die sie verbindet, keine weiteren Knoten enthalten. Motiviert durch diese Darstellung von Graphen, sagt man daher auch, dass eine Kante zwei Knoten *verbindet*. Abbildung 2 zeigt den Graphen, den man beispielsweise aus der Deutschlandkarte von Abbildung 1 erhält.

Jede Landkarte kann auf diese Art in einen Graphen umgewandelt werden, aber nicht zu jedem Graphen existiert auch eine Landkarte. Nimmt man zum Beispiel einen Graphen, der fünf Knoten hat und in dem je zwei dieser Knoten durch eine Kante verbunden sind (Abbildung 3), so gibt es keine Landkarte, die diesen Graphen liefern würde. Denn es gibt keine Landkarte mit fünf

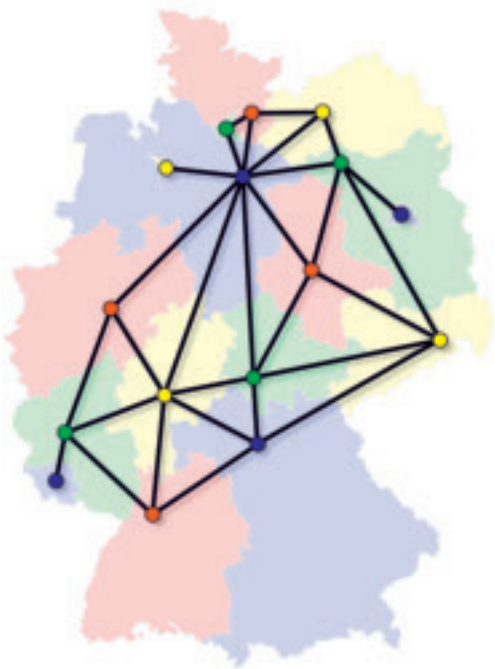


Abbildung 2: Der Graph der Deutschlandkarte

Ländern, in der je zwei Länder eine gemeinsame Grenzlinie haben. Dies zu beweisen ist jedoch nicht ganz einfach.

Offensichtlich muss ein Graph, der aus einer Landkarte entstehen kann, die Eigenschaft haben, dass man ihn so in der Ebene zeichnen kann, dass sich keine zwei seiner Kanten kreuzen. Graphen mit dieser Eigenschaft nennt man *planar*. Statt die Länder einer Landkarte gilt es nun also, die Knoten eines Graphen so zu färben, dass Knoten, die durch eine Kante verbunden sind, unterschiedliche Farben haben. Allgemein heißt ein Graph *k-färbbar*, falls man die Knoten des Graphen so mit k Farben färben kann, dass je zwei Knoten, die durch eine Kante verbunden sind, unterschiedliche Farben haben. Man kann den Vier-Farben-Satz daher in der Sprache der Graphentheorie auch einfach so formulieren:

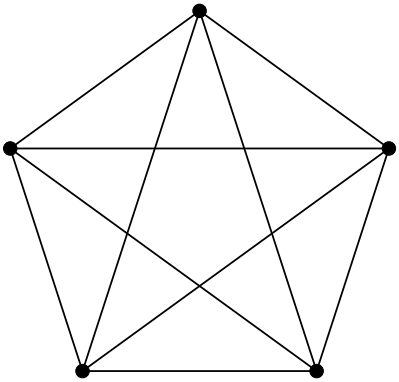


Abbildung 3: Ein Graph, zu dem es keine Landkarte gibt.

Satz 2 ***(Graphentheoretische Formulierung des Vier-Farben-Satzes)*** *Jeder planare Graph ist 4-färbbar.*

Der Vier-Farben-Satz ist zunächst nur eine reine Existenzaussage. Der Satz selber liefert noch keinerlei Hinweis darauf, wie man eine 4-Färbung der Knoten eines planaren Graphen finden kann. Natürlich könnte man alle möglichen Färbungen mit vier Farben durchprobieren. Aber die Anzahl der möglichen Färbungen, die man ausprobieren muss, steigt exponentiell in der Anzahl der Knoten des Graphen. Daher ist dieser Ansatz schon für recht kleine Graphen nicht praktikabel. Man möchte für dieses Problem daher einen sogenannten *polynomiellen Algorithmus* angeben können. Dabei handelt es sich um ein Verfahren, für das man ein Polynom P angeben kann, so dass die Anzahl Rechenschritte des Verfahrens für einen beliebigen planaren Graphen mit n Knoten durch $P(n)$ beschränkt ist.

Eine schnelle Möglichkeit, die Knoten eines planaren Graphen zu färben, liefert ein sogenannter *Greedy-Ansatz*. Bei diesem gehen wir davon aus, dass wir als Farben natürliche Zahlen verwenden. Man wählt nun eine beliebige Reihenfolge der Knoten des Graphen aus und färbt die Knoten in dieser Reihenfolge, indem man jedem Knoten die kleinste natürliche Zahl als Farbe gibt, die noch keiner seiner mit ihm durch eine Kante verbundenen Knoten trägt. Dieser Algorithmus ist sehr einfach und auch sehr schnell (man kann zeigen, dass die Anzahl Rechenschritte durch eine lineare Funktion beschränkt ist).

Allerdings kann es passieren, dass der Algorithmus mehr als vier Farben zum Färben eines planaren Graphen benötigt. Die Landkarte in Abbildung 4 zeigt ein sehr einfaches Beispiel hierfür. Wenn man die Länder in der Reihenfolge ihrer Nummerierung färbt, so wird Land 1 die Farbe 1 erhalten, Land 2 die Farbe 2 und Land 3 die Farbe 3 erhalten. Land 4 kann die Farbe 2 erhalten, Land 5 die Farbe 4 und Land 6 muss schließlich die Farbe 5 erhalten.

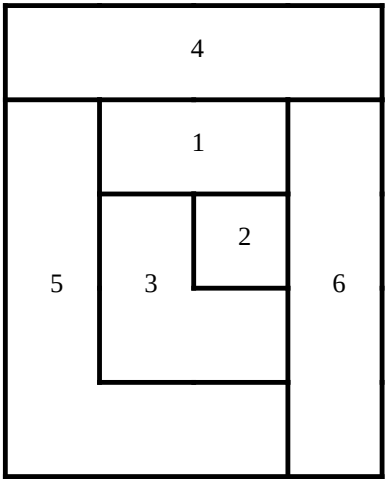


Abbildung 4: Eine Landkarte, für die der Greedy-Algorithmus fünf Farben benutzt.

Die einzigen polynomiellen Algorithmen, die man heutzutage kennt, um einen planaren Graphen mit höchstens vier Farben zu färben, beruhen auf dem Beweis des Vier-Farben-Satzes und sind daher sehr kompliziert.

Eine weitere Frage beim Färben planarer Graphen ist, ob man für einen gegebenen Graphen alle vier Farben benötigt oder ob man mit weniger Farben auskommt. Die Landkarte in Abbildung 5 ist beispielsweise mit nur zwei Farben färbbar. Man kann sich leicht überlegen, wie man zu einer beliebigen Landkarte entscheiden kann, ob sie 2-färbbar ist.

Während man in polynomieller Zeit einen planaren Graphen mit vier Farben färben kann, ist bis heute kein polynomieller Algorithmus bekannt, der feststellen kann, ob man auch mit weniger als vier Farben auskommt. Man

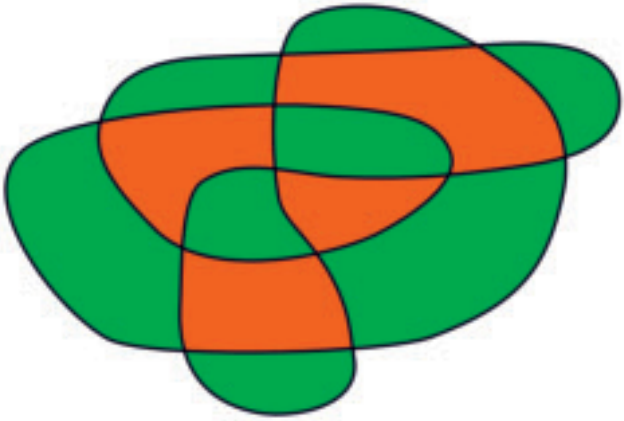


Abbildung 5: Eine Landkarte, die mit nur zwei Farben gefärbt werden kann.

kann zeigen, dass das Problem, zu entscheiden, ob man drei oder vier Farben benötigt, um einen gegebenen planaren Graphen zu färben, zu der Klasse der sogenannten NP-vollständigen Probleme gehört. Bei diesen handelt es sich um eine große Klasse von Problemen, bei denen bis heute für keines dieser Probleme ein polynomieller Algorithmus bekannt ist. Sobald man jedoch für nur eines dieser Probleme einen solchen Algorithmus fände, wüßte man, dass es auch für alle anderen NP-vollständigen Probleme einen polynomiellen Algorithmus gibt. Die Frage, ob NP-vollständige Probleme polynomielle Algorithmen haben, ist unter dem Kürzel „P=NP?“ bekannt und gehört zu den wichtigsten Fragen der Mathematik und der Informatik.

2.2 Bäume zählen

Die Diskrete Mathematik beschäftigt sich auch mit dem Zählen von Dingen. Dabei geht es allerdings weniger darum, konkrete Dinge zu zählen, wie zum Beispiel die Anzahl Münzen in der eigenen Geldbörse, oder die Anzahl Reiskörner in einer Packung Reis. Vielmehr geht es darum, abstrakte Dinge zu zählen, die nicht explizit gegeben sind, sondern sich nur über ihre Eigenschaften definieren.

Als Beispiel betrachten wir die Frage, wie viele Möglichkeiten es gibt, ein Straßennetz für n Städte zu bauen, das aus möglichst wenigen Straßenstücken besteht, die jeweils genau zwei Städte verbinden, so dass jede Stadt von jeder anderen aus erreichbar ist. Nachfolgende Abbildung zeigt ein solches Straßennetz für einige deutsche Städte.

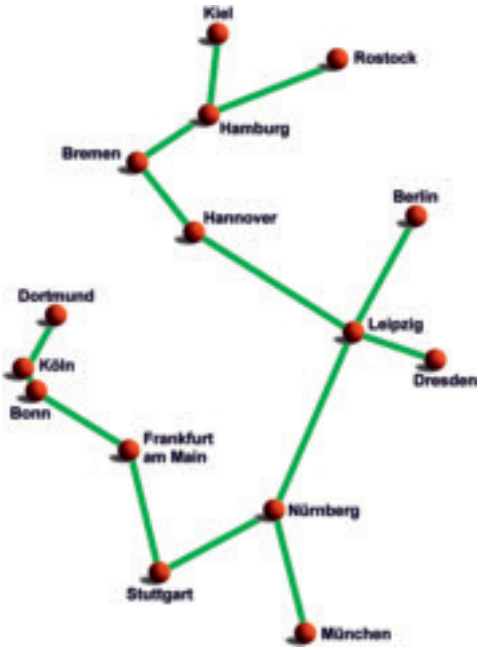


Abbildung 6: Ein Straßennetz für 15 Städte.

Zunächst abstrahieren wir wiederum und stellen fest, dass jedes solche Straßennetz durch einen Graphen repräsentiert werden kann, bei dem die Knoten den Städten und die Kanten den Straßen entsprechen. Da wir in unserem Straßennetz von jeder Stadt zu jeder anderen Stadt gelangen wollen, heißt dies für unseren Graphen, dass wir von jedem Knoten des Graphen zu jedem anderen Knoten des Graphen gelangen können, indem wir entlang von Kanten laufen. Graphen mit dieser Eigenschaft nennt man *zusammenhängend*. Als weitere Forderung haben wir, dass unser Straßennetz möglichst wenige Straßenstücke enthalten soll. Entsprechend muss unser Graph zusammenhängend sein und möglichst wenige Kanten enthalten. Einen Graphen mit dieser Eigenschaft nennt man einen *Baum*. Unser obiges Problem ist also äquivalent dazu, zu bestimmen, wie viele Bäume es auf n Knoten gibt.

Hat ein Baum mindestens zwei Knoten, so findet man in ihm stets mindestens einen Knoten, von dem nur eine Kante abgeht. Entfernt man nun diese Kante, so erhält man einen einzelnen Knoten und einen Baum auf den restlichen Knoten. Iteriert man dieses Vorgehen, so stellt man fest, dass man nach $n-1$ Schritten keine Kante mehr hat. Damit haben wir gezeigt, dass ein Baum mit n Knoten genau $n-1$ Kanten enthalten muss.

Auf zwei Knoten gibt es nur einen möglichen Baum. Auf drei Knoten gibt es drei verschiedene Bäume: Man erhält diese, indem man sich einen der drei Knoten auswählt und diesen mit den beiden anderen Knoten jeweils durch eine Kante verbindet. Auf vier Knoten gibt es bereits 16 mögliche Bäume, die in der folgenden Abbildung gezeigt sind.

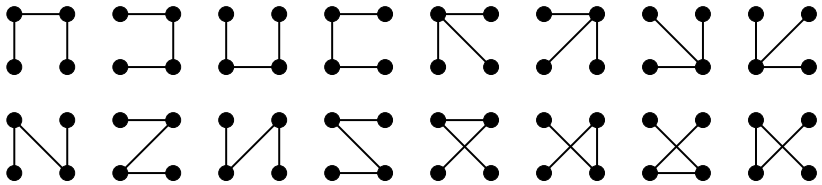


Abbildung 7: Die 16 möglichen Bäume auf vier Knoten.

Wie viele Bäume gibt es auf n Knoten? Diese Frage wurde erstmals 1889 von Arthur Cayley beantwortet und ist daher auch unter dem Namen Cayley-Formel bekannt.

Satz 3 ***(Cayley-Formel)*** *Es gibt n^{n-2} Bäume auf n Knoten.*

Beweis. Um die Anzahl der Bäume auf n Knoten zu zählen, machen wir zunächst etwas scheinbar Unsinniges: Wir zählen nämlich etwas anderes als uns eigentlich interessiert. Und das machen wir dann gleich zweimal. Es handelt sich hierbei aber um eine bekannte Beweistechnik der Diskreten Mathematik: das *doppelte Abzählen*.

Wir bezeichnen mit $f(n)$ die Anzahl der möglichen Bäume auf n Knoten. Wir haben gerade gesehen, dass $f(4) = 16$ gilt. Wir wollen nun durch doppeltes Abzählen von etwas anderen Bäumen die Funktion $f(n)$ bestimmen. Wir betrachten im Folgenden einen Baum mit n Knoten, von denen genau ein Knoten rot ist, während die anderen $n-1$ Knoten alle schwarz sind. Zudem nummerieren wir die $n-1$ Kanten mit den Zahlen $1, 2, \ldots, n-1$ durch.

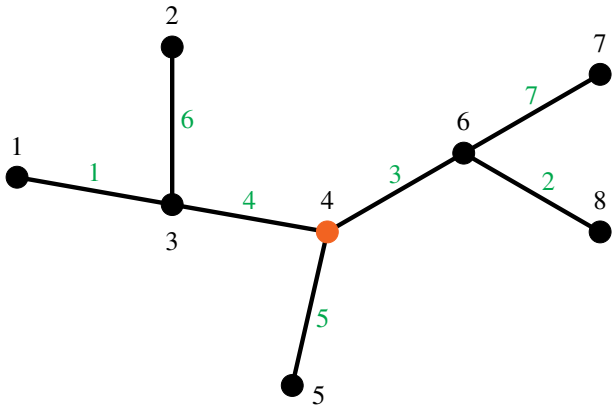


Abbildung 8: Ein Baum mit nummerierten Knoten und Kanten sowie einem roten Knoten.

Wie viele Bäume auf n Knoten mit einem roten und $n-1$ schwarzen Knoten gibt es, bei denen zusätzlich die Kanten mit den Zahlen $1, 2, \ldots, n-1$ nummeriert sind? Bei einem gegebenen Baum auf n Knoten haben wir n verschiedene Möglichkeiten, einen dieser Knoten rot zu färben. Zudem gibt es $(n-1) \cdot (n-2) \cdot \ldots \cdot 1 = (n-1)!$ Möglichkeiten, die $n-1$ Kanten mit den Zahlen $1, \ldots, n-1$ durchzunummerieren. Daher gibt es also

$$f(n) \cdot n \cdot (n-1)!$$

Möglichkeiten, für n Knoten einen Baum mit $n-1$ durchnummerierten Kanten anzugeben und einen der n Knoten rot zu färben. Damit haben wir auf die erste Art die gesuchte Anzahl bestimmt.

Wir zählen nun die gleiche Anzahl auf eine andere Art. Dazu geben wir den $n-1$ Kanten eine Richtung und zwar so, dass man in dem Baum von jedem Knoten zu dem roten Knoten gelangen kann, wenn man sich daran hält, dass eine Kante nur in der vorgegebenen Richtung durchlaufen werden darf.

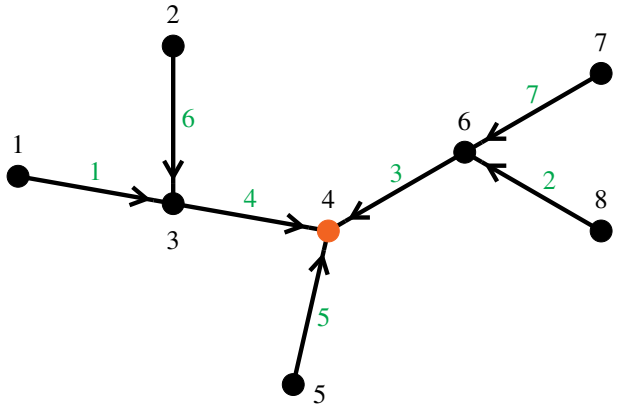


Abbildung 9: Ein Baum, bei dem alle Kanten zu dem roten Knoten hin orientiert sind.

Für jeden unserer $f(n)$ möglichen Bäume gibt es genau eine Möglichkeit, sich Richtungen für die Kanten zu wählen, so dass der rote Knoten von jedem anderen Knoten aus erreichbar ist. Der rote Knoten ist dann in diesem Baum der einzige Knoten, von dem aus man zu keinem weiteren Knoten über eine Kante gelangen kann. Wählt man umgekehrt in einem Baum die Richtungen der Kanten so aus, dass es genau einen Knoten gibt, von dem aus man zu keinem anderen Knoten über eine Kante gelangt, so ist dieser Knoten eindeutig bestimmt als derjenige, der rot gefärbt werden muss. Wir fragen uns jetzt also: Wie viele Möglichkeiten gibt es, den Kanten eines Baumes eine Richtung zu geben, so dass es genau einen Knoten gibt, von dem man nicht mehr weggelangt?

Um diese Anzahl zu bestimmen, wollen wir schrittweise $n-1$ mit einer Richtung versehene Kanten zwischen unsere n Knoten einfügen und dabei zählen, wie viele solche Möglichkeiten wir haben. Die erste Kante, die wir einfügen, kann in einem beliebigen der n Knoten beginnen und kann in einem beliebigen anderen Knoten enden, das heißt also für die erste Kante haben wir $n \cdot (n-1)$ Möglichkeiten. Wie viele Möglichkeiten haben wir für die nächsten Kanten?

Wir nehmen an, dass wir bereits k Kanten eingefügt haben. Der Graph besteht dann aus $n-k$ zusammenhängenden Teilen. Wollen wir eine weitere Kante einfügen, so muss sie zwei verschiedene solche Teile verbinden. In jedem Teil des Graphen gibt es genau einen Knoten, in dem noch keine Kante beginnt. Eine neu einzufügende Kante kann in einem beliebigen der n Knoten enden, sie kann jedoch nur in einem von $n-k-1$ Knoten beginnen, nämlich in einem Knoten aus den $n-k-1$ Teilen, die den Endknoten nicht enthalten und in denen noch keine Kante beginnt. Wir haben also $n \cdot (n-k-1)$ Möglichkeiten, um die $(k+1)$-te Kante einzufügen.

Insgesamt ergibt sich somit für unsere gesuchte Anzahl die Formel:

$$n \cdot (n-1) \cdot n \cdot (n-2) \cdot \ldots \cdot n \cdot 1 = n^{n-1} \cdot (n-1)!$$

Vergleichen wir nun die so erhaltene Formel mit der oben erhaltenen Formel, so können wir die gesuchte Anzahl $f(n)$ leicht ermitteln, denn es gilt:

$$f(n) \cdot n \cdot (n-1)! \;=\; n^{n-1} \cdot (n-1)!$$

und daraus folgt

$$f(n) = n^{n-2}.$$

Das heißt also, wir haben damit bewiesen, dass es n^{n-2} Bäume mit n Knoten gibt. Damit ist der Beweis der Cayley-Formel beendet. □

Die Cayley-Formel besagt, dass es $15^{13} = 1\,946\,195\,068\,359\,375$ verschiedene Möglichkeiten gibt, ein Straßennetz für 15 Städt zu bauen! Dies zeigt, dass man bei der Suche nach einem billigsten solchen Straßennetz nicht einfach alle Möglichkeiten durchprobieren kann; das würde viel zu lange dauern. Wie es schneller geht, sehen wir im nächsten Abschnitt.

2.3 Optimale Bäume

Im vorigen Abschnitt haben wir gesehen, dass es n^{n-2} verschiedene Bäume auf n Knoten gibt. Demzufolge gibt es auch n^{n-2} Möglichkeiten, ein Straßennetz für n Städte zu bauen. Wir nehmen nun an, dass wir für jede mögliche Straße die Kosten kennen, die ihr Bau verursacht. Welches unter den n^{n-2} möglichen Straßennetzen ist dann das billigste? Wir werden in diesem Abschnitt zeigen, wie man ein solches Straßennetz sehr schnell bestimmen kann, ohne alle möglichen Straßennetze durchprobieren zu müssen.

Wir formulieren unser Problem zunächst wieder für Bäume. Wir haben einen Graphen G gegeben, bei dem wir für je zwei Knoten wissen, wie hoch die Kosten für eine Kante zwischen diesen beiden Knoten ist. Insgesamt hat dieser Graph $\frac{n(n-1)}{2}$ Kanten, da eine Kante jeweils zwei Knoten verbindet und es für jeden der n Knoten $n-1$ mögliche andere Knoten gibt, mit denen er durch eine Kante verbunden sein kann. Die Kantenmenge von G sei mit $E(G)$ bezeichnet. Nun wollen wir uns aus $E(G)$ eine Menge F von $n-1$ Kanten so auswählen, dass diese einen Baum mit den n Knoten bilden und dabei insgesamt möglichst geringe Kosten haben. Einen solchen Baum bezeichnet man als *minimalen aufspannenden Baum*.

Wenn man in einem Graphen in einem Knoten startet, dann stets entlang Kanten läuft, ohne einen Knoten (außer dem Startknoten) mehrfach zu besuchen, und schließlich zum ersten Mal wieder am Startknoten ankommt, so nennt man den durch die benutzten Knoten und Kanten gebildeten Graphen einen *Kreis*. In einem Baum kann es niemals einen Kreis geben, denn wir haben gefordert, dass ein Baum zusammenhängend sein muss und möglichst wenige Kanten enthält. Hat man aber einen zusammenhängenden Graphen, der einen Kreis enthält, so kann man eine beliebige Kante des Kreises entfernen, und der Graph bleibt zusammenhängend.

Im Folgenden geben wir einen sehr einfachen Greedy-Algorithmus an, mit dem man einen minimalen aufspannenden Baum bestimmen kann.

Die Kosten einer Kante e seien mit $c(e)$ bezeichnet. Zunächst sortieren wir alle Kanten, entsprechend ihren Kosten: die billigste zuerst, die teuerste zuletzt. Nennen wir die Kanten in dieser Reihenfolge $e_1, e_2, \ldots, e_m$, wobei natürlich $m = \frac{n(n-1)}{2}$ ist. Zur Vorbereitung auf den Hauptteil des Algorithmus setzen wir noch $F := \emptyset$.

Wir gehen dann alle Indizes $i = 1, \ldots, m$ durch und entscheiden jeweils, ob wir e_i zu F hinzufügen. Schauen wir uns eine solche Kante e_i an, die die beiden Knoten v und w verbinde. Wenn wir bereits eine Möglichkeit haben, durch Benutzen der Kanten in F von v nach w zu gelangen, dann dürfen wir e_i nicht hinzufügen, denn sonst entstünde ein Kreis. Andernfalls fügen wir e_i aber zu F hinzu.

Diese Vorgehensweise wurde erstmals 1956 von Joseph B. Kruskal vorgeschlagen und der Algorithmus wird daher auch *Kruskals Algorithmus* genannt. Nachfolgend geben wir den Pseudocode dieses Algorithmus an.

Kruskal $(G,\ c : E(G) \to \mathbb{R})$
1 sortiere $E(G)$, so dass $c(e_1) \leq c(e_2) \leq \ldots \leq c(e_m)$
2 $F := \emptyset$
3 **for** $i = 1$ **to** m **do**
4 **if** $F \cup \{e_i\}$ ist kreisfrei **then** $F := F \cup \{e_i\}$

Satz 4 *Kruskals Algorithmus liefert einen minimalen aufspannenden Baum.*

Beweis. Zunächst beobachten wir, dass wir keinen Kreis in der Kantenmenge F haben. Außerdem erhalten wir einen zusammenhängenden Graphen. Denn nehmen wir an, dass es zwei Knoten v und w gibt, so dass man nicht über eine Folge von Kanten von v nach w gelangt. Dann hätte der Algorithmus in dem Schritt, in dem er die Kante zwischen v und w betrachtet, diese zu F hinzugefügt. Dann gelangt man aber doch von v nach w.

Die Kantenmenge F enthält also keinen Kreis, und sie ist die Kantenmenge eines zusammenhängenden Graphen. Es handelt sich also wie gewünscht um einen Baum. Wir müssen nun noch zeigen, dass dieser Baum ein minimaler aufspannender Baum ist.

Diesen Beweis führen wir indirekt, das heißt wir nehmen an, es gäbe einen besseren Baum, und führen diese Annahme zum Widerspruch.

Unter allen minimalen aufspannenden Bäumen sei F^* die Kantenmenge eines solchen, so dass der kleinste Index i für den $e_i \in F$ aber $e_i \notin F^*$ größtmöglich ist. Falls dieser Index nicht existiert, so gilt $F = F^*$ und wir sind fertig. Ansonsten betrachte die Kantenmenge $F^* \cup \{e_i\}$. Diese enthält einen Kreis. Da F keinen Kreis enthält muss dieser Kreis eine Kante $e_j \notin F$ enthal-

ten. Die Kanten, die vor e_i in F eingefügt wurden, sind nach Definition von i auch alle in F^* enthalten. Zu dem Zeitpunkt, zu dem Kruskals Algorithmus die Kante e_i zu F hinzufügt, hätte auch e_j zu F hinzugefügt werden können, ohne einen Kreis mit den aktuell in F enthaltenen Kanten zu schließen. Da e_j aber nicht in F ist, muss $j > i$ gelten und damit wegen der Sortierung der Kanten $c(e_j) \geq c(e_i)$. Dann bildet die Kantenmenge $F^* \cup \{e_i\} \setminus \{e_j\}$ auch die Kantenmenge eines minimalen spannenden Baumes. Dies widerspricht aber der Wahl von F^*. □

Wir haben also bewiesen, dass der Algorithmus immer einen minimalen spannenden Baum berechnet. Aber wie schnell ist er? Ein Teilproblem besteht darin, m Zahlen zu sortieren, und ein anderes darin, zu entscheiden, ob es einen Kreis in der Kantenmenge $F \cup \{e_i\}$ gibt. Wir wollen hier nicht auf die Details eingehen, aber das eine geht leicht in $c \cdot m^2$ Rechenschritten, das andere in $c \cdot m$ Rechenschritten, wobei c eine Konstante ist, die vom Computer und der Implementierung, aber nicht von der Größe des gegebenen Graphen

abhängt. Da das zweite Problem m mal gelöst werden muss, ist die Gesamtlaufzeit höchstens $2c \cdot m^2$. Tatsächlich kann der Algorithmus sogar so implementiert werden, dass er nur $c \cdot m \cdot \log m$ Rechenschritte benötigt.

Das Berechnen eines minimalen aufspannenden Baumes ist ein typisches Problem der kombinatorischen Optimierung. Stets gilt es, aus einer implizit gegebenen endlichen (aber im Allgemeinen sehr großen) Menge ein bestes Element zu finden. Die grundlegende Fragestellung besteht darin, ob das wesentlich schneller als durch Ausprobieren geht, z.B. mit einem polynomiellen Algorithmus. Im Falle des Berechnens eines minimalen aufspannenden Baumes lautet die Antwort Ja, wie wir gesehen haben.

Hätten wir statt nach einem Baum aber nach einer Rundreise gefragt, die jeden der n Knoten genau einmal besucht und zum Ausgangspunkt zurückkehrt (das ist das berühmte *Traveling Salesman Problem*), so könnten wir diese Frage nicht beantworten. Denn dieses Problem ist NP-schwer, das heißt, es gibt genau dann einen polynomiellen Algorithmus, wenn P=NP gilt (vergleiche Seite 11).

2.4 Kürzeste Wege

Wir wollen uns nun mit der Frage beschäftigen, wie ein Navigationsgerät die kürzeste Route zwischen zwei Punkten bestimmt. Das geht mit Methoden der Diskreten Mathematik relativ einfach. Auch hier ist die Abstraktion einer Straßenkarte durch einen Graphen sehr hilfreich.

Nehmen wir also an, wir haben eine Straßenkarte, auf der sowohl Start- als auch Zielpunkt verzeichnet sind. Als Knotenmenge V des Graphen nehmen wir die Straßenkreuzungen auf dieser Karte. Ein Knoten v wird durch eine Kante mit einem weiteren Knoten w verbunden, wenn man auf direktem Wege von v nach w gelangen kann. Zum Beispiel werden also zwei durch ein Stück Straße verbundene Kreuzungen miteinander durch eine Kante verbunden. Natürlich braucht man für das Zurücklegen der Strecke entlang einer Kante $e \in E$ eine gewisse Zeit, und diese nennen wir Kosten (oder Gewicht) dieser Kante und bezeichnen sie mit $c(e)$. Auch Einbahnstraßen kann man so im Graphen abbilden: wir geben jeder Kante eine Richtung, entlang der wir fahren dürfen. Für eine Straße, die in beide Richtungen befahrbar ist, müssen wir dann zwei solche gerichtete Kanten in den Graphen einfügen: eine für jede Richtung.

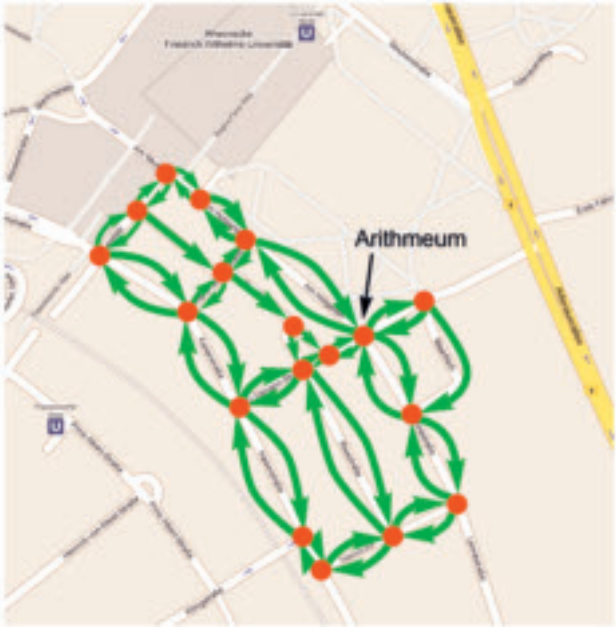


Abbildung 10: Ausschnitt einer Straßenkarte und der daraus abgeleitete gerichtete Graph.

Wir können nun jeden Weg in der Karte als Weg W im Graphen darstellen, nämlich indem wir die Knoten, die diesen Weg ausmachen, in einer Liste aufschreiben: $W = v_1, v_2, \ldots, v_k$ mit $v_i \in V$. Für einen Weg sollte natürlich zwischen zwei aufeinanderfolgenden Knoten auch eine entsprechende Kante im Graphen existieren. Dementsprechend nennen wir eine Liste von Knoten auch nur dann einen Weg, wenn diese Zusammenhangsbedingung erfüllt ist und wir auch keine Kante entgegen ihrer Richtung ablaufen. Die Länge des Weges erhalten wir, indem wir die Kosten dieser Kanten entlang des Weges addieren.

Seien nun s und t zwei Knoten. Um einen kürzesten Weg von s nach t zu finden, können wir alle Wege, die von s nach t führen, betrachten und uns den kürzesten merken. Bei großen Straßenkarten wäre aber auch ein Computer schnell damit überfordert, da es extrem viele verschiedene Wege von s nach t geben kann. Wir wollen daher den sogenannten *Algorithmus von Dijkstra* beschreiben. Dieser berechnet ausgehend vom Startknoten s einen kürzesten Weg zu allen anderen Knoten des Graphen. Die Länge eines solchen kürzesten Weges von s zu einem Knoten t bezeichnen wir als den *Abstand* von s nach t.

Der Algorithmus von Dijkstra markiert im Laufe seiner Ausführung bestimmte Knoten mit zwei Werten. Zum einen markiert er jeden Knoten mit der Länge eines kürzesten momentan bekannten Weges von s bis zu diesem Knoten. Diese Länge bezeichnen wir mit $l(x)$ für einen Knoten x und setzen $l(x) := \infty$, falls kein Weg von s zu dem Knoten x bekannt ist. Den Wert $l(x)$ nennen wir auch den l-Wert des Knotens x.

Zusätzlich wird jeder Knoten mit einer von drei möglichen Farben markiert. Diese Farbe gibt an, in welchem Zustand sich der Knoten befindet:

- Alle Knoten, deren kürzester Abstand zu s bekannt ist, markieren wir *grün*. Für diese Knoten ist der l-Wert gleich dem Abstand von s.
- Alle Knoten, für die wir schon einen Weg kennen, aber noch nicht wissen, ob dieser Weg wirklich der kürzeste ist, färben wir *gelb*. Der Knoten s wird zu Beginn gelb gefärbt.
- Alle Knoten, für die wir noch keinen Weg von s aus kennen, markieren wir *rot*. Für diese Knoten ist der l-Wert ∞. Zu Beginn sind alle Knoten, bis auf den Startknoten s, rot.

Der Algorithmus von Dijkstra geht wie folgt vor:

Dijkstra $(G, s \in V(G), c : E \to \mathbb{R}_+)$

```
1   setze l(s) := 0 und l(v) := ∞ für alle v ∈ V(G) \ {s}
2   färbe s gelb und alle anderen Knoten rot
3   while es gibt einen gelben Knoten do
4       wähle einen gelben Knoten v mit minimalem l(v)
5       färbe v grün
6       für alle roten und gelben Knoten w mit (v,w) ∈ E:
7           setze l(w) := min{l(w), l(v) + c((v,w))}
8           färbe w gelb
```

Satz 5 *Der Algorithmus von Dijkstra bestimmt den Abstand von s zu jedem anderen Knoten des Graphen.*

Beweis. In jedem Durchlauf der while-Schleife wird ein gelber Knoten v grün gefärbt, und wenn ein Knoten einmal grün gefärbt ist, bleibt er es auch. Der Algorithmus terminiert also.

Wir zeigen nun induktiv, dass stets für jeden grünen Knoten x der Wert $l(x)$ gleich dem Abstand vom Startknoten s zu x ist. Am Anfang gibt es keinen grünen Knoten, also gilt unsere Aussage. Auch nachdem wir s grün gefärbt haben, stimmt die Aussage.

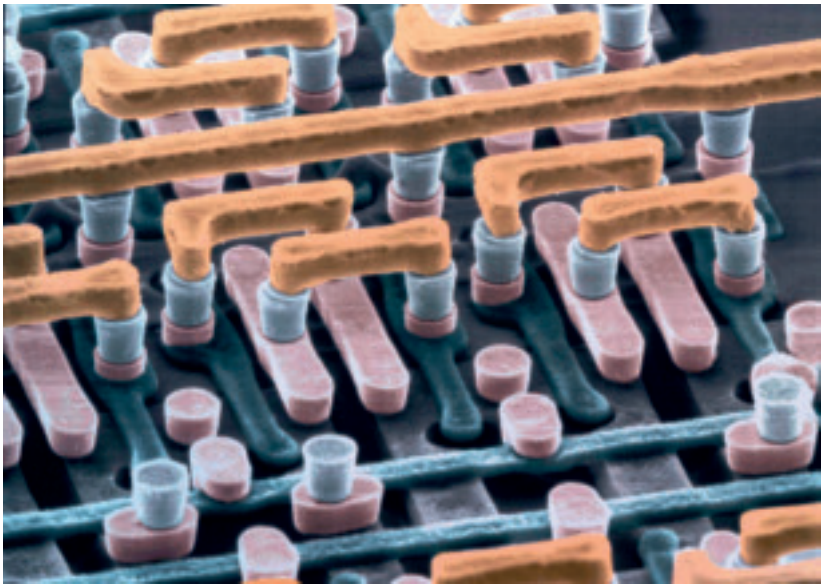
Wenn wir in einem Schritt nun einen gelben Knoten $v \in V \setminus \{s\}$ grün färben, so haben alle Knoten mit kleinerem l-Wert bereits die Farbe grün; wir wählen ja v gerade als gelben Knoten mit kleinstem l-Wert. Nach Induktionsvoraussetzung haben alle diese grünen Knoten den korrekten Abstand als l-Wert gespeichert. Alle Kanten haben positive Länge, also gibt es einen kürzesten Weg von s nach v, in dem alle Knoten außer v grün sind. Ist x nun der Vorgänger von v auf diesem Weg von s nach v, so hat der Algorithmus in dem Schritt, in dem x die Farbe grün erhielt, auch die Kante (x,v) betrachtet und den l-Wert von v auf den Abstand von s nach v gesetzt. □

Ist ein Knoten bei Beendigung des Algorithmus von Dijkstra rot gefärbt, so bedeutet dies, dass es keinen Weg von s zu diesem Knoten gibt.

Wir haben den Algorithmus von Dijkstra benutzt, um die Länge eines kürzesten Weges von s zu jedem anderen Knoten des Graphen zu berechnen. Der Algorithmus von Dijkstra lässt sich leicht so erweitern, dass auch die kürzesten Wege berechnet werden: Immer wenn in Zeile 7 ein l-Wert eines Knotens w verringert wird, wir also einen kürzeren Weg gefunden haben, merken wir uns den grünen Knoten v, der dafür verantwortlich war, als Vorgänger von w. Später rekonstruieren wir durch Zurückverfolgen der Vorgängerverweise die Route vom Ziel zum Start.

Das Berechnen kürzester Wege spielt nicht nur beim Routenplaner im Navigationsgerät eine wichtige Rolle. Auch im Chip-Design, einem zentralen Arbeitsgebiet des Forschungsinstituts für Diskrete Mathematik, spielen kürzeste-Wege-Probleme eine große Rolle. Ein Chip besteht aus vielen Millionen kleinen Bauteilen, die durch Metallbahnen verbunden werden müssen, so dass entsprechend einem Schaltplan Strom zwischen ihnen fließen kann.

Diese Metallbahnen verlaufen alle horizontal oder vertikal, und zwar in bis zu zehn Ebenen. Die möglichen Positionen von Metallbahnen werden durch ein dreidimensionales Gitter repräsentiert. Anschlusspunkte einzelner Bauteile bilden jetzt Start und Ziel für einen kürzesten Weg, der in diesem sogenannten Routinggitter berechnet werden muss. Allerdings gibt es in diesem Gitter bereits Stellen, die durch andere Metallbahnen und Bauteile belegt sind.



Wenn ein Routenplaner mit Straßenkarte ein paar Sekunden für jede Route benötigt, so ist das nicht weiter schlimm, beim Routing eines Chips sieht es ganz anders aus: hat das Netzwerk einer Straßenkarte von Europa etwa 20 Millionen Knoten, so besteht das Routinggitter für einen derzeit aktuellen Chip aus 100 Milliarden Knoten. Auch werden für einen einzigen Chip etwa 10 Millionen unterschiedliche Wege benötigt, die natürlich auch alle berechnet werden müssen. Ein wirklich effizientes Verfahren zur Berechnung kürzester Wege ist also notwendig. Am Forschungsinstitut für Diskrete Mathematik werden ständig weitere Verbesserungen hierfür entwickelt.

2.5 Matroide

Wir haben in den Abschnitten 2.1 und 2.3 jeweils einen Greedy-Algorithmus kennengelernt. Der erste diente uns zum Färben planarer Graphen, der zweite zur Bestimmung optimaler Bäume. Allerdings hat unser Färbungsalgorithmus nicht notwendigerweise eine optimale Lösung gefunden. Wir wollen in diesem Abschnitt ganz allgemeine Bedingungen aufstellen, unter denen ein Greedy-Verfahren stets eine optimale Lösung findet. Dazu benötigen wir zunächst die folgenden Begriffe:

Ein *Unabhängigkeitssystem* ist ein Paar $M = (S, I)$, das folgenden Bedingungen genügt:

1) S ist eine endliche, nichtleere Menge.

2) I ist eine Familie von Teilmengen von S mit der Eigenschaft, dass wenn $A \in I$ und $B \subseteq A$, dann gilt auch $B \in I$. Die Elemente von I heißen *unabhängige Mengen*.

Gilt zudem die folgende, als *Austauscheigenschaft* bezeichnete Bedingung, so nennt man das Unabhängigkeitssystem ein *Matroid*:

3) Falls $A \in I$ und $B \in I$ und $|A| < |B|$, so gibt es ein $x \in B \setminus A$ so dass $A \cup \{x\} \in I$.

Es sei $M = (S, I)$ ein Unabhängigkeitssystem. Eine unabhängige Menge $A \in I$ heißt *maximale unabhängige Menge*, falls $A \cup \{x\} \notin I$ für alle $x \in S \setminus A$, d. h. keine Obermenge von A ist in I enthalten. Wir können nun folgende Aussage beweisen:

Satz 6 *In einem Matroid haben alle maximalen unabhängigen Mengen die gleiche Größe.*

Beweis. Angenommen, es gibt maximale unabhängige Mengen $A, B \in I$ mit $|A| < |B|$. Dann gibt es nach 3) ein $x \in B$ mit $A \cup \{x\}$ ist unabhängig. Dies ist ein Widerspruch zur Maximalität von A. □

Als Anwendung dieses Satzes betrachte einen Graphen G mit Knotenmenge V und Kantenmenge E. Dann ist $M = (E, I)$ ein Matroid, wobei I alle kreisfreien Teilmengen von E beinhaltet. Offensichtlich genügt M den Eigenschaften 1) und 2) und ist somit ein Unabhängigkeitssystem. Der Nachweis der Eigenschaft 3) ist auch nicht weiter schwierig: Betrachte zwei kreisfreie Kantenmengen A und B mit $|A| < |B|$. Dann hat der Graph mit Knotenmenge V und Kantenmenge B weniger Teile als der Graph mit Knotenmenge V und Kantenmenge A. Es gibt somit mindestens eine Kante in B, deren Endknoten in verschiedenen Teilen des Graphen mit der Kantenmenge A liegen. Eine solche Kante kann daher zu A hinzugefügt werden, ohne einen Kreis zu erzeugen. Wir stellen fest, dass in dem Matroid $M = (E, I)$ die maximalen unabhängigen Mengen genau die Kantenmengen der aufspannenden Bäume sind. Wie wir bereits in Abschnitt 2.2 gesehen haben, haben diese in einem Graphen mit n Knoten die Größe $n-1$, sind also insbesondere alle gleich groß.

Ein *gewichtetes Matroid* ist ein Matroid $M = (S, I)$ zusammen mit einer Gewichtsfunktion $w : S \to \mathbb{R}$. Für eine Menge $A \in I$ ist $w(A)$ als $\sum_{x \in A} w(x)$ definiert. Wir wollen nun in gewichteten Matroiden maximale unabhängige Mengen kleinsten Gewichts finden. Dies ist nach obiger Bemerkung eine Verallgemeinerung des Problems, einen minimalen aufspannenden Baum in einem Graphen zu finden.

Greedy $(M = (S,I), w : S \to \mathbb{R})$
1 sortiere S, so dass $w(s_1) \leq w(s_2) \leq \ldots \leq w(s_n)$
2 $A := \emptyset$
3 **for** $i = 1$ **to** n **do**
4 **if** $A \cup \{s_i\} \in I$ **then** $A := A \cup \{s_i\}$

Abbildung 11: Der Greedy-Algorithmus zur Berechnung maximaler unabhängiger Mengen kleinsten Gewichts.

Satz 7 *Es sei $M = (S,I)$ mit $w : S \to \mathbb{R}$ ein gewichtetes Matroid. Dann findet der Greedy-Algorithmus eine maximale unabhängige Menge $A \subseteq S$ kleinsten Gewichts.*

Beweis. Wir zeigen, dass die Menge A zu jedem Zeitpunkt des Greedy-Algorithmus die Eigenschaft hat, dass sie Teilmenge einer maximalen unabhängigen Menge kleinsten Gewichts ist. Da dies dann auch bei Beendigung des Algorithmus gilt, folgt daraus, dass der Greedy-Algorithmus eine maximale unabhängige Menge kleinsten Gewichts liefert.

Zu Beginn des Greedy-Algorithmus ist $A = \emptyset$ und damit ist A sicherlich Teilmenge einer maximalen unabhängigen Menge kleinsten Gewichts. Betrachte nun den Zeitpunkt des Algorithmus zu dem ein Element s_i zu A hinzugefügt wird. Sei B eine maximale unabhängige Menge kleinsten Gewichts, die A enthält.

Falls $s_i \in B$, so gilt auch $A \cup \{s_i\} \subseteq B$. Falls $s_i \notin B$, so gibt es eine maximale unabhängige Menge B' mit $A \cup \{s_i\} \subseteq B'$ und $B' \subseteq B \cup \{s_i\}$. Dies folgt aus der Austauscheigenschaft 3), indem man diese auf die Mengen $A \cup \{s_i\}$ und B so lange wie möglich anwendet.

Nach Satz 6 erhalten wir also $B' = (B \cup \{s_i\}) \setminus \{z\}$ für ein Element z von $B \setminus A$. Die Menge $A \cup \{z\}$ ist genau wie die Menge $A \cup \{s_i\}$ unabhängig, da $A \cup \{z\} \subseteq B$ gilt. Da $z \notin A$ ist, kann z noch nicht an der Reihe gewesen sein, es muss also $w(s_i) \leq w(z)$ gelten. Damit gilt dann $w(B') \leq w(B)$, womit auch B' eine maximale unabhängige Menge kleinsten Gewichts ist. □

Wir zeigen nun, dass auch die Umkehrung von Satz 7 gilt, d.h. der Greedy-Algorithmus liefert in Unabhängigkeitssystemen genau dann maximale unabhängige Mengen kleinsten Gewichts, wenn es sich um Matroide handelt.

Satz 8 *Es sei $M = (S, I)$ ein Unabhängigkeitssystem. Falls der Greedy-Algorithmus für beliebige Gewichtsfunktionen $w : S \to \mathbb{R}$ eine maximale unabhängige Menge kleinsten Gewichts findet, so ist M ein Matroid.*

Beweis. Angenommen M ist kein Matroid, d.h. es gibt Mengen $A, B \in I$ mit $|A| < |B|$, aber für alle $x \in B \setminus A$ ist $A \cup \{x\} \notin I$. Definiere eine Gewichtsfunktion w durch

$$w(x) := \begin{cases} -(|B| - 1/2)/|A|, & \text{falls } x \in A \\ -1, & \text{falls } x \in B \setminus A \\ 0, & \text{sonst} \end{cases}$$

Der Greedy-Algorithmus wählt zuerst die Elemente von A, da diese kleinstes Gewicht haben, denn es gilt wegen $|A| < |B|$ auch $-(|B| - 1/2)/|A| < -1$. Alle weiteren Elemente, die zu A hinzugenommen werden können, haben Gewicht 0. Der Greedy-Algorithmus liefert daher eine maximale unabhängige Menge mit Gewicht $-(|B| - 1/2)$. Es gibt jedoch eine maximale unabhängige Menge kleineren Gewichts. Dazu betrachte eine beliebige maximale unabhängige Menge B', die B enthält. Da kein Element positives Gewicht hat, gilt $w(B') \leq w(B) = -|B| < w(A)$, und somit ist B' eine maximale unabhängige Menge, die kleineres Gewicht als A hat. Der Greedy-Algorithmus liefert somit nicht eine maximale unabhängige Menge kleinsten Gewichts. Damit muss unsere Annahme, dass M kein Matroid ist, falsch gewesen sein. □

3 Das Forschungsinstitut für Diskrete Mathematik der Universität Bonn

Das Forschungsinstitut für Diskrete Mathematik wurde im Jahr 1987 durch Senatsbeschluss als zentrale wissenschaftliche Einrichtung der Universität Bonn gegründet. Hierdurch sollte der Arbeitsgruppe von Professor Bernhard Korte ein besonderer Rahmen gegeben werden. Durch Erlass des Ministeriums für Wissenschaft und Forschung wurde Professor Korte zum Direktor dieses Instituts bestellt. Als zentrale wissenschaftliche Einrichtung der Universität Bonn untersteht das Institut direkt dem Senat bzw. Rektorat.

Im Jahr 1997 konnte das Institut einen Neubau an der Ecke Lennéstraße/ Fritz-Tillmann-Straße beziehen. In demselben Gebäude befindet sich auch das Arithmeum.

Das Institutsgebäude enthält zudem einen Hörsaal, einen Seminarraum sowie eine große Bibliothek.

3.1 Professoren und Mitarbeiter

Abbildung 12: Von links nach rechts: Professor Korte, Professor Vygen, Professor Hougardy, Juniorprofessor Nieberg, Dr. Brenner.

Alle am Institut tätigen Professoren sind Mitglieder der Mathematisch-Naturwissenschaftlichen Fakultät, die u.a. für die Studiengänge verantwortlich ist.

Das Institut hat einen wissenschaftlichen Vorstand, dem alle am Institut tätigen Professoren und der gewählte Honorarprofessor, Professor Dr. László Lovász, Budapest, angehören.

Im Einzelnen sind am Institut tätig:

Professor Dr. Stefan Hougardy
Tel. 0228/738746
E-Mail: hougardy@or.uni-bonn.de
Sprechstunden: nach der Vorlesung und nach Vereinbarung

Professor Dr. Dr. h.c. Bernhard Korte
Tel. 0228/738770
E-Mail: dm@or.uni-bonn.de
Sprechstunden: nach der Vorlesung und Dienstag von 11–12 Uhr

Professor Dr. Jens Vygen
Tel.: 0228/738770
E-Mail: vygen@or.uni-bonn.de
Sprechstunden: nach der Vorlesung und nach Vereinbarung

Juniorprofessor Dr. Tim Nieberg
Tel.: 0228/738747
E-Mail: nieberg@or.uni-bonn.de
Sprechstunden: nach der Vorlesung und nach Vereinbarung

Das Lehrveranstaltungsprogramm, insbesondere Übungen, Tutorien und Seminare, wird betreut von

Akad.Rat Dr. Ulrich Brenner
Tel.: 0228/738749
E-Mail: brenner@or.uni-bonn.de

Herr Dr. Brenner ist erster und ständiger Ansprechpartner für die Studierenden in allen Fragen des Studienprogramms, Vorlesungen, Seminare etc.

Das Institut beschäftigt darüber hinaus rund zehn wissenschaftliche Mitarbeiter (die meisten davon Doktoranden), die bei der Betreuung der Studierenden mithelfen.

3.2 Bibliothek des Instituts

Das Institut besitzt eine ausgezeichnete Fachbibliothek zur Diskreten Mathematik mit rund 15.000 Büchern und 160 laufend bezogenen wissenschaftlichen Zeitschriften. Hervorgegangen aus einer von der Deutschen Forschungsgemeinschaft geförderten Literaturstelle, ist diese Bibliothek eine der besten und umfassendsten Literatursammlungen zur Diskreten Mathematik weltweit. Sie ist montags bis freitags von 9 Uhr bis 17 Uhr geöffnet. Ausreichend viele Plätze zum Arbeiten, ein dem Universitätssystem angeschlossener Kartenkopierer, sowie Computer für Literaturrecherchen sind vorhanden.

Der Hauptteil der Bibliothek ist eine Präsenzbibliothek. Daneben gibt es auch eine Studienbibliothek als Ausleihbibliothek für Studierende der Diskreten Mathematik, insbesondere für höhere Vorlesungen und Teilnehmer der Seminare. In dieser Bibliothek ist die einschlägige Literatur von Vorlesungen und Seminaren auch mehrfach vorhanden. Studierende können aus dieser Bibliothek Bücher ausleihen.

Das Institut verfügt ferner gemeinsam mit dem Arithmeum über eine umfangreiche Sammlung historischer Mathematik- und Rechenbücher, vornehmlich aus dem 16. und 17. Jahrhundert. Diese gehört zu den besten ihrer Art im deutschsprachigen Raum. Einige bibliophile Rara sind nur in dieser Sammlung nachgewiesen.

3.3 Forschung am Institut

Die Arbeitsgruppe „Diskrete Optimierung“, die Professor Korte vor mehr als 30 Jahren an der Universität Bonn aufgebaut hat und aus der das Forschungsinstitut für Diskrete Mathematik hervorgegangen ist, war sicherlich prägend für die Entwicklung dieses mathematischen Arbeitsgebietes in Deutschland und auch weltweit. Sie wurde in ihren Anfängen durch mehrere Sonderforschungsbereiche der Deutschen Forschungsgemeinschaft gefördert. Ein langjähriges bilaterales Forschungsprojekt zwischen der Ungarischen Akademie der Wissenschaften und der Deutschen Forschungsgemeinschaft hat insbesondere die Kooperation mit ungarischen Wissenschaftlern und der dortigen exzellenten Schule in der Diskreten Mathematik gefördert. Lehrstühle an zahlreichen deutschen Universitäten und im Ausland sind aus der Bonner Forschungsgruppe hervorgegangen. Mehr als 50 Forschungsstipendiaten und Forschungspreisträger der Alexander-von-Humboldt-Stiftung haben sich das Bonner Institut als Gastinstitution ausgewählt. Zahlreiche Gastwissenschaftler haben längere oder kürzere Forschungsaufenthalte in Bonn verbracht. Der Stifterverband für die Deutsche Wissenschaft hat zwei Stiftungsprofessuren (John-von-Neumann-Professuren) am Bonner Institut eingerichtet. Das Forschungsinstitut für Diskrete Mathematik und die hier tätigen Professoren und Wissenschaftler sind Mitglieder des Hausdorff Center for Mathematics, des Exzellenzclusters Mathematik.

Die Forschungsschwerpunkte des Instituts sind die Kombinatorische Optimierung und Anwendungen im Chip-Design. Gegenstand der Kombinatorischen Optimierung sind Probleme, bei denen aus einer endlichen Menge (mit einer kombinatorischen Struktur) ein bestes Elements (im Sinne einer Zielfunktion) gesucht wird. Einfache Beispiele haben wir in den Kapiteln 2.3 bis 2.5 behandelt. Wie wir dort gesehen haben, ist das Ausprobieren aller Möglichkeiten indiskutabel; stattdessen sucht man Algorithmen mit polynomieller Laufzeit, die die kombinatorische Struktur des Problems ausnutzen.

Die Wissenschaftler des Instituts haben in vielen Bereichen der Kombinatorischen Optimierung wichtige Beiträge geleistet. Zu nennen sind hier Netzwerkflussalgorithmen, Multicommodity flows, Steinerbäume, kürzeste und disjunkte Wege, Matchings, Facility location, Matroide und submodulare Funktionen, um nur einige zu nennen. Das am Institut entstandene Standardwerk „Combinatorial Optimization: Theory and Algorithms“ ist bereits

in vielen Auflagen erschienen und in mehrere Sprachen übersetzt worden. Es bildet auch die Hauptgrundlage mehrerer Lehrveranstaltungen des Instituts.

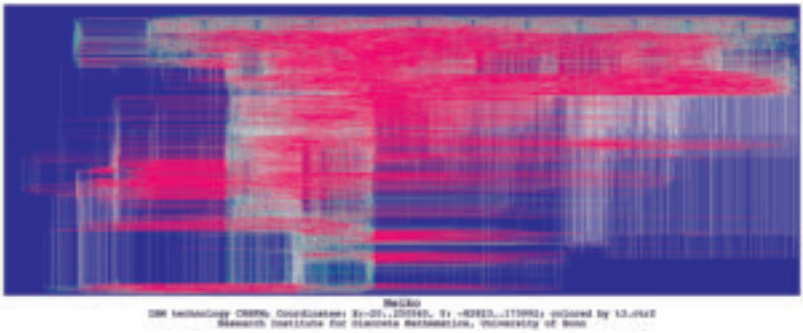
Die Bedeutung der Kombinatorischen Optimierung liegt nicht zuletzt darin, dass die betrachteten Probleme in vielen Anwendungen eine zentrale Rolle spielen, z.B. in Verkehrsplanung, Logistik, Telekommunikation und vielen anderen Bereichen. Das vielleicht spannendste und vielseitigste Anwendungsgebiet ist das Chip-Design. Seit mehr als zwanzig Jahren kooperiert das Institut mit verschiedenen Firmen der Computerindustrie, allen voran IBM, und entwickelt Algorithmen für den Entwurf höchstkomplexer Chips.

Die Kooperation zwischen der IBM und der Universität Bonn bzw. dem Forschungsinstitut für Diskrete Mathematik ist eine der langfristigsten und auch erfolgreichsten Kooperationen zwischen Wissenschaft und Wirtschaft, die es bisher weltweit gegeben hat. Mehr als 1.000 höchstkomplexe Chips sind bisher mit den Algorithmen und Methoden der Bonner Diskreten Mathematik entworfen worden, darunter bekannte Mikroprozessoren, Network Switches und System Controller, die sogar im Guinness-Buch der Rekorde erwähnt wurden. Die in Bonn entwickelten Algorithmen und Methoden, die als Softwarepaket unter dem Namen „BonnTools“ bekannt wurden, sind in Designcentern auf der ganzen Welt ständig im Einsatz.

In den letzten zwanzig Jahren hat sich die Komplexität der Logikchips und Mikroprozessoren dramatisch vergrößert. Zu Anfang hatten die fingernagelgroßen Chips etwa eine Million Transistoren und eine gesamte Verdrahtungslänge dazwischen von circa 15 Metern. Heute findet man auf einem gleich großen Chip rund eine Milliarde Transistoren, deren gesamte Verdrahtung in einem dreidimensionalen Gitter mit zehn Lagen mehr als einen Kilometer Länge haben kann. Bedingt durch die immense Steigerung der Komplexität treten im Chipdesign immer wieder neue und bisher völlig unbekannte Probleme auf, die auch dazu führen, dass Methoden und Theorie der Diskreten Optimierung wesentlich weiterentwickelt werden müssen.

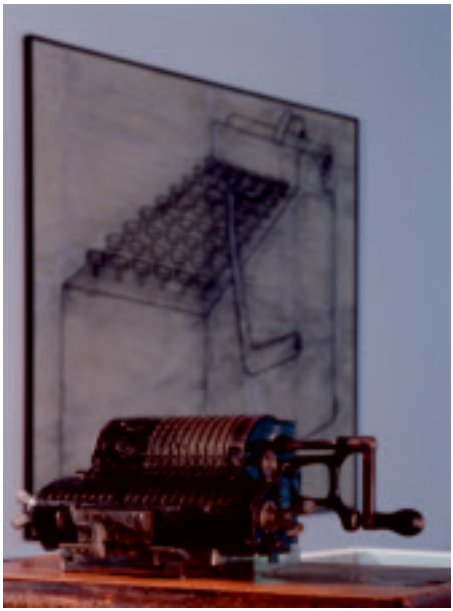
Die Taktfrequenz moderner Mikroprozessoren liegt heute über vier Gigahertz. Selbst einfache PCs haben Taktfrequenzen von mehr als zwei Gigahertz. Das optimale Timing eines solchen Chips kann nur noch mit komplizierten Verfahren aus der Diskreten Optimierung behandelt werden. Um dem Motto „immer kleiner, immer schneller“, von dem die Chipindustrie getrieben wird, Rechnung zu tragen, werden stets neue technologische Entwicklungen vorangetrieben. Die Materialien und Fertigungsprozesse werden verbessert

und optimiert. Durch solche technologischen Verbesserungen kann man dann zum Beispiel Verbesserungen in der Taktfrequenz (Rechengeschwindigkeit) von etwa 20 bis 25 Prozent erreichen. Da hierfür aber neue Materialien und vollständig neue Fertigungstechnologien notwendig sind, ist das nur möglich, indem neue Fabriken und neue Fertigungsgeräte für die nächste Chipgeneration gebaut werden. Das erfordert Investitionen in der Größenordnung von fünf Milliarden Dollar. Mit Methoden der Diskreten Mathematik konnten die Bonner Forscher mehrfach ähnlich große Verbesserungen von 25 Prozent und mehr erreichen. Für den Einsatz der Mathematik sind null Investitionen notwendig, nur richtiges Nachdenken! Ein schöner und nachvollziehbarer Beweis für den Wert der Mathematik!



3.4 Arithmeum

Das Arithmeum ist ebenfalls eine zentrale Einrichtung der Universität Bonn, die durch Senatsbeschluss begründet wurde, um der umfangreichen Sammlung historischer Rechenmaschinen und Rechengeräte von Professor Korte eine besondere Heimat zu geben. Schon als Student begann Professor Korte mechanische Rechenmaschinen zu sammeln, weil zu der Zeit bereits alle großen Rechner elektronisch waren und es abzusehen war, dass mechanische Rechengeräte vollständig obsolet werden würden. Die erste Maschine der Sammlung ist diejenige (Brunsviga-) Rechenmaschine, mit der Professor Korte die Übungsaufgaben in der praktischen Mathematik gelöst hat. Die umfangreiche Sammlung mechanischer Rechenmaschinen war zunächst eine private Sammlung, die dann Professor Korte gemeinsam mit anderen Sammlungsteilen (konstruktive Kunst und historische Bücher) durch Schenkungsvertrag der Universität und dem Land Nordrhein-Westfalen übereignet hat. Im Gegenzug wurden dann zusätzliche Finanzmittel des Bundes (Ausgleichsmittel) für den Museumsteil im Neubau des Instituts zur Verfügung gestellt.



Die Sammlung des Arithmeums ist nun mit mehr als 5.000 Objekten die bei Weitem umfassendste und bedeutendste Sammlung mechanischer Rechenmaschinen weltweit. Viele Objekte sind Unikate, die nur in der hiesigen Sammlung zu finden sind. Aus Platzgründen hatte sich die Sammlung zunächst ausschließlich auf mechanische Rechengeräte konzentriert. Für große elektronische Rechner aus der Frühzeit des Computers fehlte der Platz. Seit der Neubau und zusätzliche Lagerflächen zur Verfügung stehen, werden auch elektronische Rechengeräte in die Sammlung des Arithmeums aufgenommen. So verfügt das Arithmeum über einen der wenigen (zwei oder drei) Zuse-Computer, die noch voll funktionsfähig sind. Bei Spezialführungen im Arithmeum wird dieser Zuse-Rechner vorgeführt.

Wie oben schon erwähnt, verfügt das Arithmeum über eine bedeutende Sammlung bibliophiler Bücher zur Mathematik und zum Rechnen. Newtons *Principia Mathematica* ist in der Erstauflage und weiteren Auflagen vertreten. Die Ausgabe von Diophantes Werken mit der Annotation von Pierre de Fermat ist ebenfalls vertreten. Hier findet man bekanntlich die berühmte Randnotiz von Fermat, dass er für das sogenannte Fermatsche Problem ($a^n + b^n = c^n$ ist nur für $n = 2$ lösbar) eine schöne Lösung habe, aber dass der Rand der Seite zu klein sei, um sie dort niederzuschreiben. Es hat bekanntlich mehr als 300 Jahre gedauert, bis Andrew Wiles hierfür einen Beweis geliefert hat, der sich aber nicht auf den Rand einer Buchseite schreiben lässt!

Darüber hinaus verfügt das Arithmeum über eine umfangreiche Sammlung von Stuhlklassikern und über eine bedeutende Sammlung konstruktiver Kunst, die zu den drei besten Sammlungen im deutschsprachigen Raum gehört. Diese Kunstobjekte werden im Wechsel mit Sonderausstellungen zur konstruktiven Kunst im Arithmeum gezeigt. Schließlich veranstaltet die Deutsche Welle gemeinsam mit dem Arithmeum die Konzertserie „concerto discreto“, die nunmehr schon in der zehnten Saison stattfindet. Neben der Ausstellung von Rechengeräten hat sich das Arithmeum somit zu einem kulturellen Kleinod der Universität entwickelt.

Im Arithmeum finden laufend Kinder-, Schüler- und Jugendprogramme statt. Jeden Sonntag gibt es kostenlose Führungen durch die Sammlung und zur konstruktiven Kunst. Das Arithmeum ist dienstags bis sonntags von 11 bis 18 Uhr geöffnet. Frau Dr. Ina Prinz (Telefon 0228-738770, E-Mail: prinz@or.uni-bonn.de) ist die Direktorin des Arithmeums.

4 Diskrete Mathematik in Bonn studieren

An der Universität Bonn kann die Diskrete Mathematik als Schwerpunktgebiet im Rahmen eines Mathematik- oder Informatikstudiums gewählt werden. Das Studium beginnt mit dem Bachelorprogramm, im Anschluss daran kann ein Masterprogramm durchlaufen werden. Für besonders an der Forschung Interessierte bietet sich darüberhinaus die Möglichkeit eine Promotion anzuschließen.

Die Bachelorprogramme in Mathematik und Informatik zielen darauf hin, den Studentinnen und Studenten eine grundlegende wissenschaftliche Ausbildung in dem jeweiligen Fachgebiet zu vermitteln. Ein Teilgebiet — wie zum Beispiel die Diskrete Mathematik — kann als Schwerpunktgebiet gewählt werden, indem besonders viele Veranstaltungen aus diesem Teilgebiet belegt werden. Zudem ist das Studium eines Nebenfachs im Rahmen des Bachelorprogramms vorgeschrieben. Das Bachelorstudium dauert in der Regel 6 Semester (3 Jahre) und endet mit dem Verfassen einer Bachelorarbeit. Bei erfolgreichem Studienabschluss wird der akademische Grad „Bachelor of Science (B. Sc.)“ verliehen.

Die Masterprogramme *Mathematics* bzw. *Computer Science* dienen dazu, sich in aktuelle Forschungsthemen des jeweiligen Fachgebietes vertieft einzuarbeiten. Ein wesentlicher Bestandteil ist das Verfassen einer Masterarbeit. Als Zulassung zum Masterprogramm ist ein Bachelorabschluss erforderlich. Die Regelstudienzeit für das Masterprogramm beträgt 2 Jahre (4 Semester). Die Veranstaltungen des Masterprogramms werden in englischer Sprache abgehalten. Bei erfolgreichem Studienabschluss wird der akademische Grad „Master of Science (M. Sc.)“ verliehen.

Im Rahmen einer Promotion werden unter der engen Betreuung einer Professorin oder eines Professors aktuelle Forschungsthemen bearbeitet. Die Ergebnisse werden in einer Doktorarbeit aufgeschrieben und stellen ein eigenständiges wissenschaftliches Werk dar. Bei erfolgreichem Abschluss wird der akademische Grad „Doctor rerum naturalium (Dr. rer. nat.)“ verliehen. Zulassungsvoraussetzung zur Promotion ist in der Regel ein Masterabschluss, in Ausnahmefällen kann aber auch der Bachelorabschluss ausreichen.

4.1 Lehrveranstaltungen

Herr Dr. Ulrich Brenner ist erster Ansprechpartner für Studierende in allen Fragen des Studiums. Selbstverständlich können sich die Studierenden auch mit Fragen zum Fach Diskrete Mathematik, zu den Vorlesungsinhalten und zu Abschlussarbeiten (Bachelor- und Masterarbeiten) jederzeit an alle Professoren des Instituts wenden.

In jedem Semester bietet das Forschungsinstitut für Diskrete Mathematik **Vorlesungen** sowohl für den Bachelorstudiengang als auch für den Masterstudiengang an. Die meisten Vorlesungen sind vierstündig und werden durch zweistündige Übungen begleitet. Den Hörerinnen und Hörern der Vorlesungen werden jede Woche Übungsaufgaben gestellt, für die sie in der Regel eine Woche Bearbeitungszeit haben und die dann in den Übungen besprochen werden. Die Aufgaben können Theorieaufgaben sein (einfache Beweise oder Entwicklung von Algorithmen) oder in der Implementierung von Algorithmen bestehen. Am Ende des Semesters gibt es eine Modulprüfung in Form einer Klausur oder mündlichen Prüfung. Zulassungsvoraussetzung für die Modulprüfung ist die erfolgreiche Teilnahme an den Übungen. Daneben

gibt es im Masterstudiengang auch vertiefende zweistündige und vierstündige Vorlesungen ohne begleitende Übungen.

Sämtliche Übungen zu den Vorlesungen im Bachelorprogramm werden von Tutoren in Kleingruppen durchgeführt. Als Tutoren können sich Studierende, die die entsprechende Vorlesung bereits erfolgreich absolviert haben, jederzeit bei Herrn Dr. Brenner (Tel. 0228/738749) bewerben. Eine Tutorentätigkeit kann in den Studiengängen der Mathematik auch als Tutorenpraktikum angerechnet werden. Herr Dr. Brenner betreut auch die Tutoren und veranstaltet laufend Tutorenbesprechungen.

Seminare im Studienprogramm Diskrete Mathematik werden grundsätzlich als Vortragsseminare durchgeführt. Am Ende des vorhergehenden Semesters werden in einer Vorbesprechung die Vortragsthemen vergeben. Hierbei handelt es sich entweder um Kapitel aus geeigneten Monographien oder um aktuelle Forschungsarbeiten aus der Zeitschriftenliteratur oder als Preprint. Die Vortragenden werden von wissenschaftlichen Mitarbeitern des Instituts betreut. Diese können jederzeit bei der Vortragsvorbereitung kontaktiert werden. Vor dem eigentlichen Vortrag im Seminar wird (außer beim für das zweite Semester vorgesehenen Seminar S1G1) ein „Probevortrag“ und eine ein- bis zweiseitige Kurzzusammenfassung erwartet. Hierbei soll festgestellt werden, ob die Vortragenden den Inhalt verstanden haben und in entsprechender

Klarheit auch präsentieren können, so dass sichergestellt ist, dass alle Seminarteilnehmerinnen und Seminarteilnehmer dem Vortrag mit Verständnis folgen können. Eine schriftliche Ausarbeitung des Vortrags wird in der Regel nicht erwartet. Hilfsmittel (Projektor, Handouts) sind möglich, jedoch soll der wesentliche Vortragsinhalt an der Tafel entwickelt werden.

Das Forschungsinstitut bietet regelmäßig **Programmierpraktika** an, in denen die Studierenden lernen sollen, Algorithmen zu implementieren, die etwas umfangreicher und anspruchsvoller sind als die in den Übungsaufgaben behandelten Verfahren. Es gibt in der Regel eine leichtere Einarbeitungsaufgabe, die bis zur Mitte des Semesters zu lösen ist. Die weiteren Teilaufgaben sind dann üblicherweise bis zum Ende der Vorlesungszeit zu bearbeiten.

4.2 Studienpläne

Im Rahmen des Bachelor- und des Masterprogramms besuchen die Studentinnen und Studenten sogenannte *Module*, wie zum Beispiel Vorlesungen oder Seminare, die jeweils ein Semester dauern und an deren Ende eine Prüfung abgelegt wird. Für jede erfolgreich abgelegte Prüfung werden Leistungspunkte vergeben, deren Höhe vom Arbeitsaufwand für das entsprechende Modul abhängt.

Für den erfolgreichen Abschluss des Bachelorprogramms sind 180 Leistungspunkte erforderlich. Im Rahmen des Bachelorprogramms in Mathematik oder Informatik gibt es eine Reihe von Pflichtmodulen, die von jeder Studentin und jedem Studenten belegt werden müssen. Darüberhinaus können unter Einhaltung einiger Regeln weitere Module frei gewählt werden. Alle Regeln im Detail hier aufzulisten wäre zu aufwendig, sie können in den entsprechenden Prüfungsordnungen nachgelesen werden (siehe Abschnitt 5). Stattdessen geben wir hier beispielhafte Studienpläne für ein Mathematikstudium mit Nebenfach Informatik und ein Informatikstudium mit Nebenfach Mathematik an. In beiden Studienplänen sind dabei möglichst viele Module der Diskreten Mathematik vertreten, wobei in der Informatik mehr Pflichtmodule und damit weniger Wahlmöglichkeiten existieren.

In jeder Zeile der Studienpläne sind die in dem entsprechenden Semester zu besuchenden Module aufgeführt. Alle rot gekennzeichneten Module sind Pflichtmodule, die von jeder Studentin und jedem Studenten im Bachelorstudiengang belegt werden müssen. Die Module der Diskreten Mathematik sind gelb. Sonstige, weitgehend frei wählbare Module des Hauptfaches bzw. des Nebenfaches sind grün bzw. blau gekennzeichnet. Module, die zur Bachelor- bzw. Masterarbeit gehören, sind orange hervorgehoben. Jedes Modul erfordert eine gewisse zeitliche Präsenz, die in runden Klammern unterhalb des Modulnamens angegeben ist. Dabei stehen die Buchstaben „V“, „Ü“, „S“ und „P“ für die Lehrformen „Vorlesung“, „Übung“, „Seminar“, bzw. „Praktikum“. Die hinter den Buchstaben stehenden Zahlen geben die wöchentliche Präsenzzeit in Stunden an. Zum Beispiel steht (V4+Ü2) für eine Veranstaltung, die insgesamt 6 Stunden Präsenz in der Woche erfordert, die sich in 4 Stunden für eine Vorlesung und 2 Stunden für eine Übung aufteilen. In eckigen Klammern ist neben der Modulnummer die Anzahl Leistungspunkte angegeben, die für das Modul bei erfolgreichem Abschluss vergeben werden.

Beispiel für einen Studienplan im Bachelorstudiengang Mathematik mit Nebenfach Informatik

1	V1G1 [9] Analysis I (V4 + Ü4)	V1G3 [9] Lineare Algebra I (V4 + Ü4)	V1G5 [9] Algorithmische Mathematik I (V4 + Ü4)		
2	V1G2 [9] Analysis II (V4 + Ü2)	V1G4 [9] Lineare Algebra II (V4 + Ü2)	V1G6 [9] Algorithmische Mathematik II (V4 + Ü2)	S1G1 [4] Seminar (S2)	
3	V2A1 [9] Gruppen, Ringe, Moduln (V4 + Ü2)	V2C1 [9] Einführung in die Diskrete Mathematik (V4 + Ü2)	V2E1 [9] Einführung in die numerische Mathematik (V4 + Ü2)		NI014 [4] Algorithm. Denken u. imperative Programm. (V2 + Ü3)
4		V2C3 [9] Kombinatorik, Graphen, Matroide (V4 + Ü2)	P3C1 [8] Programmierpraktikum Diskrete Optimierung (P4)		NI023 [8] Systemnahe Informatik (V4 + Ü2)
5	V2F1 [9] Einführung i. d. Wahrscheinlichkeitstheorie (V4 + Ü2)	V2C2 [9] Lineare und ganzzahlige Optimierung (V4 + Ü2)	T3G1 [12] Bachelorarbeit	S2C1 [6] Hauptseminar Diskrete Opt. (S4)	NI032 [8] Algorith. u. Berechnungskomplexität I (V4 + Ü2)
6		P2G1 [8] Tutorenpraktikum (P4)		S3G1 [6] Begleitseminar zur Bachelorarbeit (S4)	NI118 [8] Einführg. i. d. Informations- und Lerntheorie (V4 + Ü2)

Beispiel für einen Studienplan im Bachelorstudiengang Informatik mit Nebenfach Mathematik

1	BA-INF 011 [8] Logik und Diskrete Strukturen (V4 + Ü2)	BA-INF 012 [6] Informationssysteme (V3 + Ü2)	BA-INF 013 [8] Technische Informatik (V4 + Ü2)	BA-INF 014 [6] Algorithm. Denken u. imperative Programm. (V2 + Ü3)	BA-INF 015 [4] Techniken des wiss. Arbeitens (V1+Ü2)
2	BA-INF 021 [8] Lineare Algebra (V4 + Ü2)	BA-INF 022 [8] Analysis (V4 + Ü2)	BA-INF 023 [8] Systemnahe Informatik (V4 + Ü2)	BA-INF 024 [6] Objektorientierte Softwareentwicklung (V2 + Ü3)	
3	BA-INF 032 [8] Algorithmen u.Berechnungskomplexität I (V4 + Ü2)	BA-INF 107 [8] Einführung in die Diskrete Mathematik (V4 + Ü2)	BA-INF 031 [8] Angewandte Mathematik (V4 + Ü2)	BA-INF 033 [8] Softwaretechnologie (V4 + Ü2)	
4	BA-INF 041 [8] Algorithmen u.Berechnungskomplexität II (V4 + Ü2)	BA-INF 112 [8] Grundlagen d. digitalen Signalverarbeitung (V4 + Ü2)	BA-INF 108 [4] Geschichte des masch. Rechnens (V2 + Ü1)		BA-INF MM14 [10] Kombinatorik, Graphen und Matroide (V4 + Ü2)
5	BA-INF 034 [6] Systemnahe Programmierung (V2 + Ü3)	BA-INF 106 [8] Lineare und ganzzahlige Optimierung (V4 + Ü2)	BA-INF 103 [4] Algorithmische Lerntheorie (V2 + Ü1)	BA-INF 051 [10] Projektgruppe (S2 + P4)	
6	BA-INF 116 [4] Algorithmen auf Strings (V2 + Ü1)			BA-INF 061 [14] Bachelorarbeit + Begleitseminar (S2)	BA-INF MM6 [10] Mathematische Logik (V4 + Ü2)

Für den erfolgreichen Abschluss des Masterprogramms *Mathematics* oder *Computer Science* sind insgesamt 120 Leistungspunkte erforderlich. Im ersten Semester werden eine Reihe von vertiefenden Einführungsveranstaltungen in unterschiedliche Gebiete des entsprechenden Studienfachs angeboten. Darauf aufbauend können Spezialvorlesungen gewählt werden, die das Wissen aus aktuellen Forschungsgebieten vermitteln und als Grundlage für die Anfertigung der Masterarbeit dienen. Obwohl außer den Modulen zur Masterarbeit alle anderen Module frei wählbar sind, gibt es eine Reihe von Regeln, die zum Beispiel sicherstellen, dass das Studium hinreichend breit ausgelegt ist. Wir geben hier wiederum nicht alle Regeln im Detail wieder, sondern zeigen anhand von Beispielen, wie ein Masterstudium *Mathematics* oder *Computer Science* mit Schwerpunkt auf der Diskreten Mathematik aussehen könnte.

Beispiel für einen Studienplan im Masterstudiengang *Mathematics*

1	V4A3 [9] Higher Set Theory (V4 + Ü2)	V4C1 [9] Combinatorial Optimization (V4 + Ü2)	V4F1 [9] Stochastic Analysis (V4 + Ü2)		
2	V4C3 [9] Chip Design (V4 + Ü2)	V4C2 [9] Approximation Algorithms (V4 + Ü2)		S4C1 [6] Graduate Sem. on Discrete Optim. (S4)	P5C1 [8] Combinatorial Algorithms (P4)
3		V5C1 [7] Advanced Topics in Discrete Math. (V4)	T5G1 [30] Master Thesis	S4C2 [6] Graduate Seminar on Chip Design (S4)	P5C2 [8] Algorithms for Chip Design (P4)
4		V5C2 [4] Selected Topics in Discrete Opt. (V2)		S5G1 [6] Master Thesis Seminar (S4)	

Beispiel für einen Studienplan im Masterstudiengang Computer Science

1	MA-INF 1102 [8] Combinatorial Optimization (V4 + Ü2)	MA-INF 2101 [8] Graphics, Vision, and Audio (V4 + Ü2)	MA-INF 3102 [8] Information Systems Engineering (V4 + Ü2)	MA-INF 4102 [8] Intellig. Learning and Analys. Syst. (V4 + Ü2)	
2	MA-INF 1202 [8] Chip Design (V4 + Ü2)	MA-INF 1205 [4] Discrete Optimization (S2)	MA-INF 1201 [8] Approx.Algorithms for NP-hard Probl. (V4 + Ü2)	MA-INF 1207 [8] Combinatorial Algorithms (P2)	
3	MA-INF 1308 [8] Algorithms for Chip Design (P2)	MA-INF 1305 [4] Graduate Seminar on Chip Design (S2)	MA-INF 2307 [8] Graphics, Vision, and Audio (P4)	MA-INF 2205 [4] Geometry Processing I (V2 + Ü1)	MA-INF 4205 [4] Probabilistic Graphical Models (V2 + Ü1)
4		MA-INF 0401 [30] Master Thesis	MA-INF 0402 [2] Master Thesis Seminar		

4.3 Module der Diskreten Mathematik

Im Rahmen des Bachelor- und Masterstudiums in Mathematik oder Informatik kann eine Vielzahl von Modulen aus der Diskreten Mathematik gewählt werden, die wir im Folgenden detaillierter beschreiben. Hinter jedem Modulnamen stehen in eckigen Klammern jeweils die Modulnummern für das Studienfach Mathematik bzw. Informatik. Einige Module der Diskreten Mathematik können im Rahmen des Informatikstudiums nicht gewählt werden. Entsprechend fehlt bei diesen die Angabe der zweiten Modulnummer. In runden Klammern steht hinter jedem Modul jeweils die wöchentliche Präsenzzeit mit den oben beschriebenen Buchstabenkürzeln, die die Art der Veranstaltung angeben.

Module des Bachelorstudiums

Einführung in die Diskrete Mathematik [V2C1] [BA-INF 107] (V4+Ü2)
Die Vorlesung vermittelt ein vertieftes Verständnis diskreter Strukturen sowie wichtiger Algorithmen für grundlegende kombinatorische Optimierungsprobleme. Zudem erlernen die Studierenden die Bewertung verschiedener algorithmischer Lösungen und die geeignete Modellierung und Implementierung praktischer Probleme als kombinatorische Optimierungsprobleme.

Lineare und ganzzahlige Optimierung [V2C2] [BA-INF 106] (V4+Ü2)
Die Vorlesung vermittelt das Verständnis der grundlegenden Zusammenhänge der Polyedertheorie und der Theorie der linearen und ganzzahligen Optimierung sowie die Kenntnis der wichtigsten Algorithmen. Zudem erlernen die Studierenden die Fähigkeit praktische Probleme als mathematische Optimierungsprobleme zu modellieren und algorithmisch zu lösen.

Kombinatorik, Graphen, Matroide [V2C3] [BA-INF MM14] (V4+Ü2)
Die Vorlesung vermittelt ein tieferes Verständnis diskreter Strukturen, grundlegende Fragestellungen und Lösungsansätze der Kombinatorik sowie die Kenntnis der Grundlagen von Graphen- und Matroidtheorie.

Hauptseminar Diskrete Optimierung [S2C1] (S4)
Im Rahmen dieses Seminars erarbeiten die Studentinnen und Studenten ein aktuelles Thema der Diskreten Optimierung. Dazu wird Wissen in der Literaturrecherche und der Lektüre von Originalarbeiten vermittelt. Die Ergebnisse werden didaktisch aufbereitet in einer Präsentation vorgestellt und diskutiert.

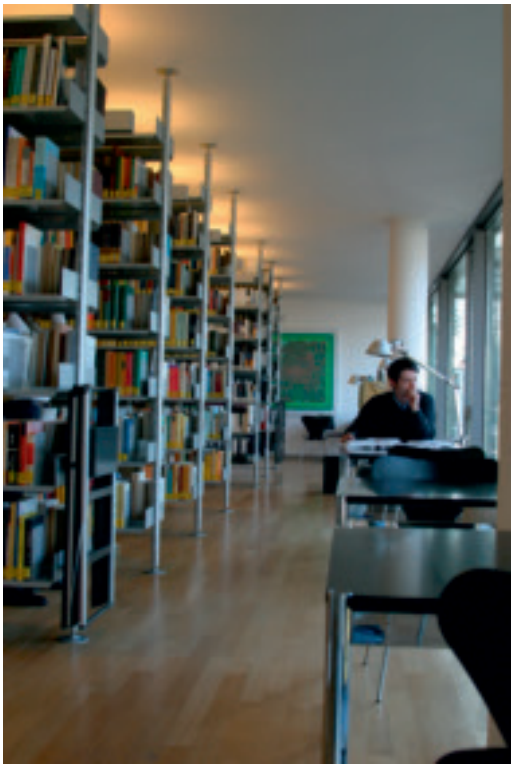
Programmierpraktikum Diskrete Optimierung [P2C1] (P4)
Das Programmierpraktikum vermittelt die Fähigkeit Algorithmen der Diskreten Optimierung zu implementieren. Dazu gehören die Wahl geeigneter Datenstrukturen sowie Kenntnisse in Test und Dokumentation.

Geschichte des maschinellen Rechnens [NI108][BA-INF 108](V2+Ü2)
Es wird ein Überblick über die wesentlichen Erfindungen in der Geschichte des maschinellen Rechnens vermittelt. Dazu gehören nicht nur theoretische Grundlagen, sondern auch das selbständige Untersuchen historischer Objekte. Die Studierenden werden dazu befähigt, aktuelle Entwicklungen der Informatik historisch einzuordnen.

Module des Masterstudiums

Combinatorial Optimization [V4C1] [MA-INF 1102] (V4+Ü2)
Die Vorlesung vermittelt fortgeschrittene Kenntnisse der kombinatorischen Optimierung und die Fähigkeit zur geeigneten Modellierung und Entwicklung von Lösungsstrategien für kombinatorische Optimierungsprobleme.

Approximation Algorithms [V4C2] [MA-INF 1201] (V4+Ü2)
In der Vorlesung werden die wichtigsten Approximationsalgorithmen für NP-schwere kombinatorische Optimierungsprobleme vorgestellt. Darüberhinaus werden Techniken zum Beweis von unteren und oberen Schranken vermittelt und auf Randomisierung beruhende Ansätze vorgestellt.



Chip Design [V4C3] [MA-INF 1202] (V4+Ü2)
Die Vorlesung stellt die zentralen Probleme und Algorithmen im Chip-Design vor. Sie vermittelt zudem das Verständnis für die Entwicklung und Anwendung mathematischer Methoden zur Lösung realer Probleme unter Berücksichtigung technischer Rahmenbedingungen. Schließlich werden Techniken zur Entwicklung und Implementierung effizienter Algorithmen für sehr große Instanzen vorgestellt und Einblicke in die Praxis des Chip-Designs gegeben.

Advanced Topics in Discrete Mathematics [V5C1] [MA-INF 1302] (V4)
Die Vorlesung behandelt ein fortgeschrittenes, aktuelles Forschungsthema der Diskreten Mathematik.

Selected Topics in Discrete Optimization [V5C2] [MA-INF 1303] (V2)
Die Vorlesung behandelt ein fortgeschrittenes, aktuelles Forschungsthema der Diskreten Optimierung.

Graduate Seminar on Discrete Optimization [S4C1] [MA-INF 1205] (S4)
In diesem Seminar wird ein jeweils wechselndes, aktuelles Forschungsgebiet der Diskreten Optimierung anhand neuerer Originalliteratur vertiefend behandelt.

Graduate Seminar on Chip Design [S4C2] [MA-INF 1305] (S4)
Es werden jeweils wechselnde Themen des Chip-Design und verwandter Anwendungen anhand aktueller Originalliteratur vertiefend behandelt.

Combinatorial Algorithms [P5C1] [MA-INF 1207] (P4)
In diesem Praktikum wird die Fähigkeit zur Implementierung von schwierigeren kombinatorischen Algorithmen sowie die Handhabung nichttrivialer Datenstrukturen erlernt. Darüberhinaus werden fortgeschrittene Softwaretechniken erlernt bzw. vertieft.

Algorithms for Chip Design [P5C2] [MA-INF 1308] (P4)
In diesem Praktikum wird die Fähigkeit zur Implementierung von Algorithmen für das Chip-Design sowie der effiziente Umgang mit sehr großen Instanzen vermittelt. Darüberhinaus werden fortgeschrittene Softwaretechniken erlernt bzw. vertieft.

4.4 Abschlussarbeiten

Die meisten Abschlussarbeiten in der Diskreten Mathematik kombinieren eine theoretische Fragestellung mit einer Anwendung, in der Regel im Bereich des Chip-Design. Je nach Neigung und Fähigkeiten kann der Schwerpunkt dabei stärker auf der Theorie oder stärker auf der Anwendung liegen. Dies ist zu Beginn der Arbeit meist nicht exakt festgelegt.

Bachelorarbeiten haben einfache, in wenigen Monaten zu bearbeitende Themen zum Gegenstand. In der Masterarbeit werden anspruchsvollere Fragestellungen bearbeitet. Für die Anfertigung einer Masterarbeit sind unter Berücksichtigung der notwendigen Einarbeitungszeit etwa 6 – 12 Monate zu veranschlagen.

4.5 Weitere Angebote für Schüler und Studenten

Das Institut wird auch ein **Exkursionsprogramm** für Studierende der Diskreten Mathematik durchführen. Hierdurch soll den Studierenden schon in jungen Jahren ein Einblick in die berufliche Praxis und in die breiten und interessanten Anwendungsmöglichkeiten der Diskreten Mathematik gegeben werden. Es ist beabsichtigt, zum Beispiel mathematische Abteilungen in Produktionsunternehmen, Beratungsunternehmen, Fluggesellschaften, IT-Unternehmen und Unternehmen der Computerindustrie in dieses Exkursionsprogramm einzubeziehen. Der Besuch von ausgewählten technischen und mathematisch-naturwissenschaftlichen Museen im In- und Ausland ist ebenfalls vorgesehen. Das Programm wird gemeinsam mit dem Arithmeum durchgeführt.

Das Institut betreibt auch ein besonderes **Mentorenprogramm**. Mentoren können Studierende werden, die sich durch sehr gute und ausgezeichnete Prüfungsergebnisse oder Seminarleistungen ausgewiesen haben. Sie werden von den am Institut tätigen Professoren ausgewählt und für eine Förderungsdauer von mindestens einem Jahr ernannt. Während dieser Förderungsdauer erhalten die Mentoren ein Stipendium von EUR 1.500,00 pro Jahr. Dieses Stipendium wird unabhängig von der finanziellen und ökonomischen Situation der Studierenden vergeben. Es ist insofern primär eine Auszeichnung für exzellente Studienleistungen. Von den Mentoren wird — wie der Name schon sagt — erwartet, dass sie jüngere Studierende beraten, zum Beispiel in Form von zwanglosen Studentenzirkeln u.Ä. Im Gegensatz zu den Tutoren wird von den Mentoren aber keine spezifische Dienstleistung erwartet, da sich das Mentorenprogramm primär als Auszeichnung für herausragende Studienleistungen versteht.

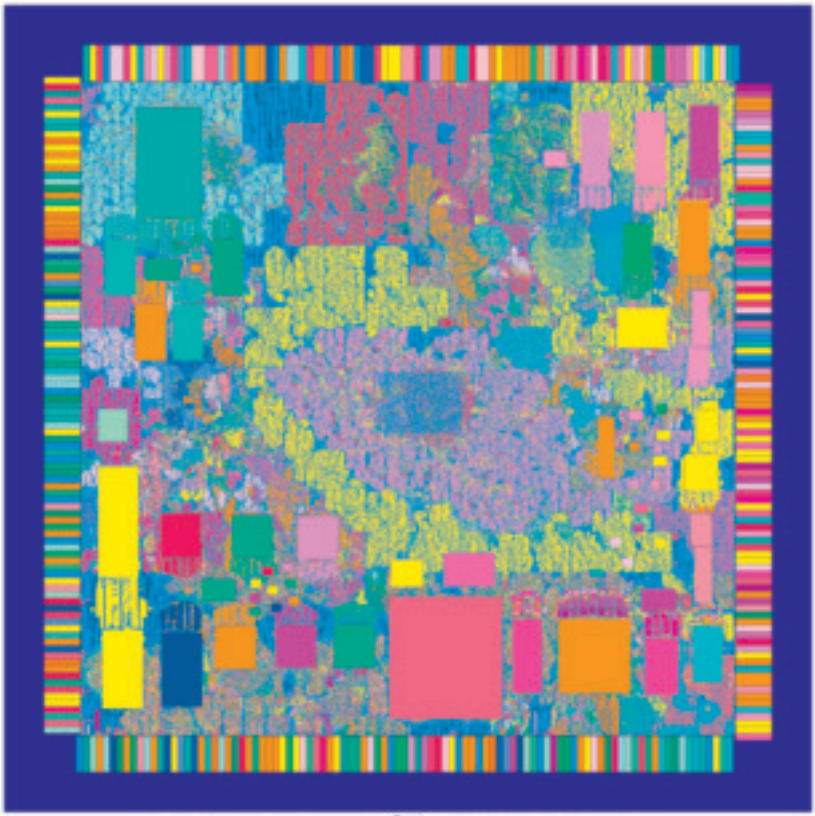
Studierende, die eine Abschlussarbeit (Bachelor- oder Masterarbeit) in der Diskreten Mathematik schreiben wollen, sollten sich möglichst frühzeitig um eine Stelle als **studentische Hilfskraft** am Institut bemühen. Die Vergütung für eine studentische Hilfskraft richtet sich nach den Standardsätzen der Universität und ist so bemessen, dass sie zur Bestreitung von Lebensunterhalt und Wohnen ausreicht. Eine studentische Hilfskraft erhält einen Arbeitsplatz am Institut. Sie kann an den Forschungsseminaren und allen sonstigen Aktivitäten des Instituts teilnehmen. Sie wird von einem wissenschaftlichen Mitarbeiter betreut und mit relativ einfachen Arbeiten an Forschungsthemen des

Instituts zu einer Abschlussarbeit hingeführt. Es ist der Regelfall, dass Studierende, die eine Abschlussarbeit in der Diskreten Mathematik schreiben, auch eine Stelle als studentische Hilfskraft am Institut haben. So ist eine intensive Betreuung gewährleistet.

Das Institut veranstaltet gemeinsam mit dem Arithmeum ein eigenständiges **Schülerprogramm**. Dadurch sollen Schülerinnen und Schüler (vorzugsweise der 9. Klasse) eine erste Einführung in die Diskrete Mathematik erhalten, da dieses neue mathematische Gebiet üblicherweise nicht Gegenstand des Schulstoffs ist. Zu diesem Programm sollen sich die Klassen über Ihren Mathematik- oder Klassenlehrer anmelden. Es können dann Termine für das

Programm, das in den Räumen des Instituts stattfindet, vereinbart werden. In seiner jetzigen Form findet das Programm an vier Tagen mit jeweils 1,5 Stunden statt. Teilnehmerinnen und Teilnehmer dieses Schülerprogramms erhalten ein Teilnahmezertifikat.

Mit diesem Programm soll erreicht werden, Schülerinnen und Schüler möglichst frühzeitig für das spannende neue mathematische Gebiet der Diskreten Mathematik zu interessieren und sie bei entsprechender Begabung zu motivieren, sich für dieses Fach auch im Studium zu interessieren.



5 Weiterführende Informationen

Internet

http://www.or.uni-bonn.de
Die Webseite des Instituts.

http://www.or.uni-bonn.de/teaching.de.html
Enthält alle Informationen zu den Lehrveranstaltungen der Diskreten Mathematik sowie nützliche Informationen zum Bachelor- und Master-Studiengang an der Universität Bonn und am Institut.

http://www.arithmeum.uni-bonn.de
Die Webseite des Arithmeums.

http://www.mathematics.uni-bonn.de
http://www.informatik.uni-bonn.de
Mathematik bzw. Informatik an der Uni Bonn. Hier finden sich insbesondere auch offizielle Dokumente, wie Prüfungsordnungen und Modulhandbücher.

Bücher, die zum Selbststudium geeignet sind:

Gritzmann, Brandenberg: *Das Geheimnis des kürzesten Weges. Ein mathematisches Abenteuer.* 3. Auflage, Springer 2005

Matoušek, Nešetril: *Diskrete Mathematik. Eine Entdeckungsreise.* 2. Auflage, Springer 2007

Aigner, Ziegler: *Das BUCH der Beweise.* 2. Auflage, Springer 2004

Lovász, Pelikán, Vesztergombi: *Diskrete Mathematik.* Springer 2005