MATHEMATIQ

Der Newsletter der MathSIG
(Interessensgruppe innerhalb der Mensa Österreich)

Ausgabe 18

http://www.hugi.scene.org/adok/mensa/mathsig/

Editorial

Liebe Leserinnen und Leser!

Dies ist die achtzehnte Ausgabe von MATHEMATIQ, dem Newsletter der MathSIG. Die MathSIG wurde gegründet, um die spezifischen Interessen mathematisch hochbegabter Menschen zu fördern. In erster Linie soll sie sich also den Themengebieten Mathematik, Informatik, Physik und Philosophie widmen. Beiträge von Lesern sind herzlich willkommen. Wenn in ihnen mathematische Sonderzeichen vorkommen, bitte ich aber, sie zwecks möglichst einfacher und fehlerfreier Formatierung im TEX-Format einzusenden. Als Vorlage ist eine Fassung des jeweils aktuellen Newsletters im TEX-Format auf Anfrage bei mir erhältlich. Außer Artikeln sind natürlich auch Illustrationen für das Titelblatt willkommen. Die Rechte an diesen müssen aber eindeutig bei euch selbst liegen, Kopieren von Bildern aus dem Internet ist nicht erlaubt.

Hinweis: Autoren sind für den Inhalt ihrer Artikel oder Werke selbst verantwortlich. Die in MATHEMATIQ veröffentlichten Beiträge widerspiegeln ausschließlich die Meinung ihrer Autoren und nicht jene des Vereins Mensa. Die Zusendung von Beiträgen gilt auch als Einverständnis zu deren Veröffentlichung in MATHEMATIQ.

Diese Ausgabe setzt das Thema (Algorithmik) fort.

In diesem Sinne: Viel Spaß beim Lesen und Lernen!

Claus D. Volko, cdvolko@gmail.com

Der Sieb des Eratosthenes

Man kennt noch keine allgemeingültige einfache Formel zur Ermittlung von Primzahlen. Wohl gibt es aber einen sehr einfachen Algorithmus dazu. Dieser Algorithmus dient dazu, sämtliche Primzahlen zwischen 2 und einer gegebenen Zahl n zu finden, wobei n im Prinzip beliebig groß, aber nicht unendlich sein darf. Der Algorithmus ist unter der Bezeichnung "Sieb des Eratosthenes" bekannt und ist ein gutes Beispiel für einen immer noch recht einfachen Algorithmus, der uns aber einige neue Konzepte der Algorithmik näherbringen wird.

Die Idee des Algorithmus: Wir schreiben zuerst alle Zahlen von 2 bis n auf, gehen dann die Zahlen der Reihe nach, von der kleinsten bis zur größten, durch und streichen alle Vielfachen, denn diese können ja keine Primzahlen sein. Logischerweise sind alle Zahlen, die am Ende übrig bleiben, Primzahlen.

Wie setzt man das am Computer um? Wir benötigen in diesem Fall eine Datenstruktur, die sich Feld oder englisch Array nennt. Dabei handelt es sich quasi um eine Ansammlung von Kästchen, in die wir Werte stecken können. Jedes Kästchen hat eine Nummer. Diese Nummer ist in diesem Fall die Zahl, und im Kästchen wollen wir speichern, ob diese Zahl Primzahl ist oder nicht.

Man könnte nun meinen, dass es pro Zahl drei Zustände gäbe: Entweder ist noch unbekannt, ob die Zahl Primzahl ist, oder sie ist Primzahl, oder sie ist keine Primzahl. Das würde bedeuten, dass wir pro Zahl 2 Bit speichern müssten, denn die Zahl drei ist als Dualzahl zweistellig. In Wirklichkeit brauchen wir aber nur 1 Bit, denn es genügen zwei Zustände: "undefiniert oder Primzahl" bzw. "keine Primzahl". Das ist deswegen möglich, weil wir in einer Variablen speichern, wie weit wir schon mit dem Algorithmus sind - welche Zahlen wir schon gewählt haben, um die Vielfachen dieser Zahlen zu markieren. Wir können daher jederzeit feststellen, ob die Primzahlen-Eigenschaft einer Zahl undefiniert oder positiv ist: denn jene Zahlen, die wir schon untersucht haben und die als "undefiniert oder Primzahl" markiert sind, sind Primzahlen, und die übrigen Zahlen, die als "undefiniert oder Primzahl" markiert sind, sind noch undefiniert. Wir brauchen also für n Zahlen nur n - 1 Bit. Das ist eine sehr schlanke Datenstruktur.

Der Code sähe in etwa wie folgt aus:

```
char *eratosthenes(int n)
{
  char *array = new char[n];
  int i, j;
  for (i = 0; i < n; i++) array[i] = 0;
  for (i = 0; i < n; i++)
    for (j = i + i; j < n; j += i) array[j] = 1;
  return array;
}
```

Nota bene: Diese Implementierung arbeitet mit Arrays vom Datentyp "char", in dieser Implementierung nimmt also jedes Feldelement 8 Bit ein, obwohl eigentlich 1 Bit reichen würde. Dadurch wird zwar theoretisch viel Speicherplatz verschwendet, in der Praxis ist das Ganze aber auf diese Weise sowohl einfacher zu implementieren als auch schneller, weil man bei modernen Rechnern nicht so einfach einzelne Bits adressieren kann - man müsste dazu so genannte Masken berechnen, was den Algorithmus etwas langsamer machen würde. Eine mögliche Implementierung mit Verwendung solcher Masken wäre jedenfalls:

```
char rol(char x)
{
  x <<= 1;
  if (!x) x = 1;
  return x;
}

char *eratosthenes(int n)
{
  char *array = new char[n / 8 + 1];
  int i, j, k, x;
  for (i = 0; i < n / 8 + 1; i++) array[i] = 0;
  for (i = 0; i < n; i++)
    for (j = i + i, k = j / 8, x = 1 << (j % 8);
              j < n; j += i, k += i / 8)
        {
          array[k] |= x;
          x = rol(x);
          if (x == 1) k++;
        }
  return array;
}
```

Man sieht, dass ein solcher Code schon um Einiges schwieriger zu lesen ist und von Programmierkonzepten Gebrauch macht, die für Anfänger nicht so leicht zu durchschauen sind. Da heutige Computer in der Regel über sehr viel Arbeitsspeicher verfügen, lohnt sich dieser Aufwand, einzelne Bits zu adressieren, nicht.

Wie komplex ist dieser Algorithmus? Wir sehen auf den ersten Blick: Jede Zahl wird einmal genommen, um daraus Vielfache zu bilden. Die Anzahl der Vielfachen beträgt selbst höchstens n (eigentlich sogar einen Bruchteil von n). Der Algorithmus ist also polynomiell. Als grobe obere Schranke lässt sich O(n^2) angeben.

Tatsächlich ist der Algorithmus aber sogar noch eine Spur effizienter. Es ist nämlich so, dass für die Zahl 2 nur $n/2$ Vielfache untersucht werden müssen, für die Zahl 3 nur $n/3$ Vielfache usw. Insgesamt also n * $\sum_{i=2}^{n}$ Vielfache. Die Summe $\sum_{i=2}^{n}$ hat

einen besonderen Namen: Man bezeichnet sie auch als "Harmonische Zahl" H_n. Es gilt $H_n \approx \log(n)$, die n-te Harmonische Zahl ist also asymptotisch ungefähr gleich dem Logarithmus von n. (Das kann man so salopp sagen, ohne die Basis des Logarithmus zu berücksichtigen, weil ja gilt, dass der Logarithmus zu einer beliebigen Basis x ein Vielfaches - oder ein Bruchteil - des Logarithmus zu einer beliebigen anderen Basis y ist: $\log_x a = \log_y a / \log_y x$. Asymptotisch gesehen ist die Basis des Logarithmus also egal.) Eine engere obere Schranke für die Komplexität des Algorithmus "Sieb des Erathostenes" beträgt also O($n \log(n)$).

Ein bisschen ließe sich der Algorithmus noch verbessern: Man muss nämlich nicht von 2 bis n iterieren, sondern nur von 2 bis $\sqrt{n}$, denn zu diesem Zeitpunkt sind bereits alle Nicht-Primzahlen markiert. So lässt sich die Komplexität also auf

O($\sqrt{(}n) \log(n)$)

reduzieren. (Da $\log(\sqrt{(}n)) = 1/2 \log(n)$, ergibt sich $1/2$ als Proportionalitätsfaktor; bei der Komplexitätsanalyse werden diese Faktoren weggelassen, weil sie an der in der "Big-O-Notation" angegebenen Komplexität nichts ändern.)

Claus D. Volko, cdvolko@gmail.com

Neuigkeiten und Weblinks

Zunächst einmal freut es mich, mitteilen zu dürfen, dass unser Mitglied Clemens Raab vor kurzem unter den Auspizien des Bundespräsidenten in Mathematik promoviert wurde. Herzliche Gratulation! Seine Homepage ist unter der Adresse

`http://www.risc.jku.at/home/craab`

zu finden - dort kann auch seine Dissertation heruntergeladen und begutachtet werden. Sein Spezialgebiet ist die Integralrechnung, auf diesem Gebiet gibt es offensichtlich auch noch Einiges zu erforschen - was man in der Schule darüber lernt, ist noch nicht der Weisheit letzter Schluss.

Ich möchte ferner auf eine Website aufmerksam machen, die ich vor kurzem entdeckt habe: Unter der Adresse `http://philosophy.lander.edu/logic/index.html` findet sich ein Kurs "Philosophy 103: Introduction to Logic". Dieser könnte für alle interessant sein, die über ausreichende Englischkenntnisse verfügen und denen mein Artikel in der ersten Ausgabe von MATHEMATIQ über das Thema Logik noch nicht genug war.

Außerdem Dank an Michael Ziegler für einen Link zum Thema Physik: Auf

`https://www.simonsfoundation.org/quanta/`
`20130917-a-jewel-at-the-heart-of-quantum-physics/`

steht: "Physicists have discovered a jewel-like geometric object that dramatically simplifies calculations of particle interactions and challenges the notion that space and time are fundamental components of reality." Was davon zu halten ist, ist eine Frage, mit der ich mich noch beschäftigen muss.

Claus D. Volko, cdvolko@gmail.com