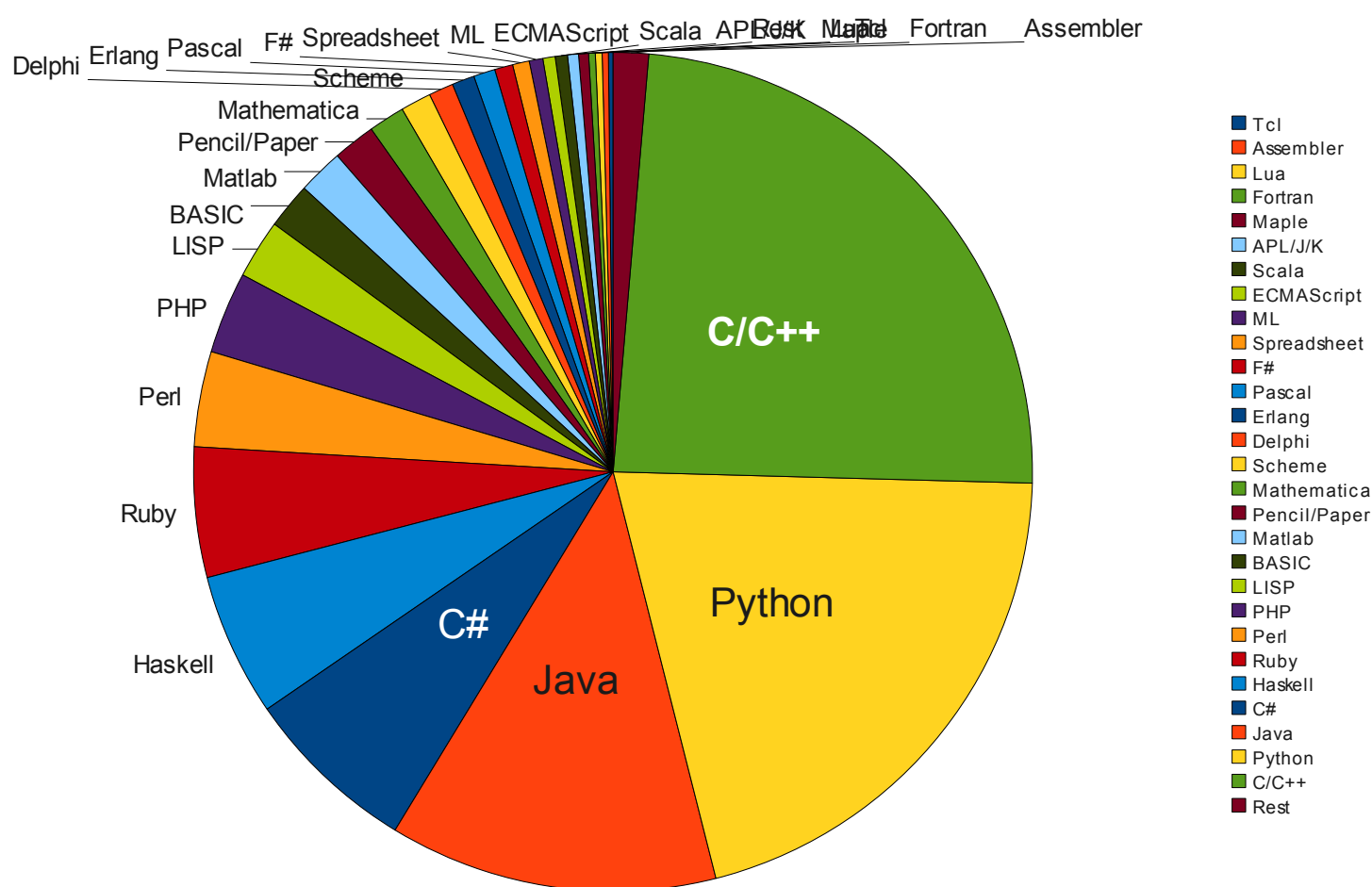


## Wettbewerb Computeralgebra

### I Die Idee

Es gibt viele Möglichkeiten mathematische Problemstellungen zu vereinfachen und im Kopf zu lösen, jedoch benötigt man ab einem gewissen Punkt Hilfsmittel. Anfangs können Stift und Papier genügen, später wird man zu Taschenrechner und Formelsammlung greifen und zu guter Letzt sind Mathematikprogramme und größere Informationsquellen wie das Internet gefragt. Bei den Mathematikprogrammen gibt es wiederum eine sehr große Auswahl: Spezielle Programme wie Mathematica oder Maple sowie Programmiersprachen.



Dieses Diagramm habe ich mit Daten von [projecteuler.net](http://projecteuler.net) erstellt. Dies verdeutlicht nochmals die enorme Auswahl an Hilfsmitteln in der Mathematik und wie viele Personen in diesem Projekt welche Hilfsmittel nutzen. Jedoch haben die meisten dieser Hilfsmittel ein großes Manko: Sie sind, verglichen mit Stift und Papier, sehr schwer zu bedienen und müssen erst installiert werden bevor sie genutzt werden können. Aus diesem Grund habe ich mit PHP und MySQL Webanwendungen geschrieben, die man einfach über den Browser bedienen kann.

Ein Anreiz mathematische Probleme mithilfe einer Programmiersprache zu lösen waren die Aufgaben auf [projecteuler.net](http://projecteuler.net). Hier will ich nun meinen Lösungsweg zu einigen Aufgaben beschreiben.

**Vorarbeit:**

Ich habe mir eine Liste der ersten Millionen Primzahlen<sup>1</sup> besorgt und diese folgendermaßen in eine MySQL Datenbank eingelesen:

```
include('db.inc.php');
$primes = file('primes.txt');
$anzahl = count($primes);
for($i=0;$i<$anzahl;++$i){
    $sql = "INSERT INTO `primzahlen` (`primzahl`) VALUES ('".$primes[$i]."')";
    mysql_query($sql);
}
```

Nun können einige Aufgaben über SQL gelöst werden:

**Problem 7: Wie lautet die 10.000ste Primzahl?**

```
SELECT primzahl FROM primzahlen WHERE id = 10000
```

Das Ergebnis: 104729

**Problem 10: Berechnen Sie die Summe aller Primzahlen die kleiner als 2.000.000 sind.**

```
SELECT sum( primzahl ) FROM primzahlen WHERE primzahl <2000000
```

Das Ergebnis: 142913828922

**Problem 37: Finden Sie die Summe aller Primzahlen mit folgender Eigenschaft: Wenn man Schritt für Schritt von Links nach rechts (oder umgekehrt) die Ziffern entfernt muss jede der so entstehenden neuen Zahlen auch eine Primzahl sein.**

Hierfür benötigt man sowohl PHP als auch die Datenbank

```
for($i=11;$i<1000000;$i+=2){
    $k = is_truncable($i);
    if($k){echo $k.'<br/>';}
}

function is_truncable($n){
    $l = strlen($n);
    $return = $n;
    for($i=0;$i<$l;++$i){
        $result = mysql_query("SELECT id FROM primzahlen WHERE
primzahl=".substr($n, $i));
        if(!mysql_num_rows($result)){return false;}
    }
    if($return){
        for($i=1;$i<=$l;++$i){
            $result = mysql_query("SELECT id FROM primzahlen WHERE
primzahl=".substr($n, 0, $i));
            if(!mysql_num_rows($result)){return false;}
        }
    }
    return $return;
}
```

<sup>1</sup> <http://primes.utm.edu/lists/small/millions/>

Somit findet man folgende Primzahlen:

23 , 37 , 53 , 73 , 313 , 317 , 373 , 797 , 3137 , 3797 , 739397

Addiert ergibt das 748317

Problem 41: Finde die größte Primzahl, die die Ziffern von 1 bis n genau einmal beinhaltet!

*(% ist ein Platzhalter und kann für eine beliebige Anzahl an Zeichen stehen)*

```
SELECT primzahl FROM primzahlen WHERE
primzahl NOT LIKE('%1%1%') AND
primzahl NOT LIKE('%2%2%') AND
primzahl NOT LIKE('%3%3%') AND
primzahl NOT LIKE('%4%4%') AND
primzahl NOT LIKE('%5%5%') AND
primzahl NOT LIKE('%6%6%') AND
primzahl NOT LIKE('%7%7%') AND
primzahl NOT LIKE('%8%8%') AND
primzahl NOT LIKE('%9%9%') AND
primzahl LIKE('%1%') AND
primzahl LIKE('%2%') AND
primzahl LIKE('%3%') AND
primzahl LIKE('%4%') AND
primzahl LIKE('%5%') AND
primzahl LIKE('%6%') AND
primzahl LIKE('%7%') AND
primzahl NOT LIKE('%9%') AND
primzahl NOT LIKE('%0%')
```

Zu guter Letzt noch eine „falsche“ Anwendung von Computeralgebra: Das Ziegenproblem<sup>2</sup>

In einer Quizshow sind drei Türen. Hinter einer Tür ist der Gewinn, hinter den zwei anderen ein Trostpreis. Man entscheidet sich für eine Tür, dann öffnet der Moderator eine der verbliebenen beiden Türen. Nun kann man sich noch umentscheiden, die andere, nicht geöffnete Tür zu nehmen.

Man kann beweisen, dass man seine Chancen verbessert wenn man die Tür wechselt. Natürlich kann man das auch mit einem Programm machen:

```
for($i=0;$i<1000000;++$i){
    $ich_nehme = rand(1,3);
    $preis = rand(1,3);
    if($preis == $ich_nehme){$ohne_umentscheiden_gewonnen++;}
}
```

Wenn man sich nun `$ohne_umentscheiden_gewonnen` ausgeben lässt und dies mit der Differenz von 1.000.000 und `$ohne_umentscheiden_gewonnen` vergleicht kann man auch sagen was besser ist und wird sogar ziemlich genau auf das richtige Verhältnis kommen. Allerdings hätte man das auch mit ein bisschen nachdenken geschafft.

### Faktorisierung von Primzahlen

Die Faktorisierung von Primzahlen ist vor allem in der Verschlüsselung (RSA-Algorithmus) eine interessante Aufgabe. Ich habe einen einfachen Algorithmus implementiert, der für kleine Primzahlen geeignet ist. Dieser sucht, bis zur Wurzel der zu faktorisierenden Zahl, nach Teilern. Dabei fängt man mit 2 an, macht mit 3 weiter und erhöht dann den möglichen Teiler immer um zwei. Wurde ein Teiler gefunden, so teilt man die Zahl so lange durch diesen Teiler bis das Ergebnis nicht mehr ganzzahlig ist und macht dann weiter.

Der implementierte Algorithmus ist aus mehreren Gründen nur für kleine Zahlen geeignet:

<sup>2</sup> <http://de.wikipedia.org/wiki/Ziegenproblem>

1. Es wurden Integer-Variablen verwendet, deren Wert je nach System, maximal  $2^{31}$  ist. Dies könnte man mit den BC-Funktionen<sup>3</sup> beheben
2. PHP ist, verglichen mit C++, langsam.
3. PHP hat normalerweise eine maximale Ausführungszeit von 30 Sekunden und maximal 10 MB Arbeitsspeicher zur Verfügung. RSA-Zahlen können jedoch 617 Stellen<sup>4</sup> haben und fünf Monate<sup>5</sup> mit sehr guter Hardware und einem sehr gutem Algorithmus zum faktorisieren benötigen

### Rechnen mit Polynomen

Nachdem ich das Horner-Schema<sup>6</sup> gesehen habe, hatte ich die Idee Polynome ähnlich am PC zu speichern. Jedes Polynom wird in einem Array gespeichert, wobei der Index die x-Potenz und der Wert der Faktor ist.

Hier bin ich leider nicht so weit gekommen wie gewünscht.

### Sonstiges

In diesem Umschlag ist der Programmcode einer Datei „in\_entwicklung.php“ (steht ganz oben). Die Programmteile sind noch nicht fertiggestellt, habe ich jedoch mitgeschickt da es mir geraten wurde. Ich würde diese zwei Blätter nicht lesen, da sie nicht Dokumentiert sind und der Programmcode noch fehlerhaft ist.

### Zu PHP und MySQL:

Unter dem Betriebssystem Ubuntu sind diese sehr leicht zu installieren. Man muss nur

```
sudo apt-get install apache2 php5 mysql-server phpmyadmin
```

in die Konsole eingeben und schon hat man einen kleinen Webserver. Falls Sie Ubuntu testen wollen, so können Sie die beiliegende CD nutzen. Allerdings sollten Sie nur dann auf „Installieren“ klicken, wenn alle Daten auf der Festplatte gesichert wurden, da es sich bei Ubuntu um ein Betriebssystem (ähnlich wie Microsoft Windows) handelt.

Support für Ubuntu findet man unter [wiki.ubuntuusers.de](http://wiki.ubuntuusers.de).

Eine Funktionsreferenz für PHP ist unter <http://de3.php.net/manual/de/> und ein Tutorial unter <http://tut.php-quake.net/de/>

### Weitere Links zur Computeralgebra:

Die folgenden Links haben genau das, was ich implementieren wollte:

- Funktionsplotter: <http://www.mathe-fa.de/de>
- Berechnen der Integralfunktion: <http://integrals.wolfram.com/index.jsp>
- Differenzieren: <http://calc101.com/webMathematica/Ableitungen.jsp>
- Zahlenfolgen erkennen: <http://www.research.att.com/~njas/sequences/>

Auf meiner Website <http://martin-thoma.de/mathe/computeralgebra.php> kann man die Skripte sofort nutzen.

Mit freundlichen Grüßen,

---

<sup>3</sup> <http://de3.php.net/manual/de/ref.bc.php>

<sup>4</sup> [http://en.wikipedia.org/wiki/RSA\\_Factoring\\_Challenge](http://en.wikipedia.org/wiki/RSA_Factoring_Challenge)

<sup>5</sup> <http://www.rsa.com/rsalabs/node.asp?id=2964>

<sup>6</sup> <http://de.wikipedia.org/wiki/Horner-Schema>