Elementare Zahlentheorie mit Maple (01202) WS 2011/12

Klausur am 18.02.2012:

Musterlösungen

Aufgabe 1

1. Wegen $\mathrm{ggT}(x, y) = 1$ haben x und y keine gemeinsamen Primfaktoren. Analog haben x und z keine gemeinsamen Primfaktoren. Es folgt, dass x und yz keine gemeinsamen Primfaktoren haben, und damit gilt $\mathrm{ggT}(x, yz) = 1$.
2. Sei $d = \mathrm{ggT}(a + b, a)$. Dann gilt $d|(a + b)$ und $d|a$, also $d|(a + b - a)$ und somit $d|a$ und $d|b$. Wegen $\mathrm{ggT}(a, b) = 1$ folgt $\mathrm{ggT}(a + b, a) = 1$. Analog gilt $\mathrm{ggT}(a + b, b) = 1$. Mit $x = a + b$, $y = a$ und $z = b$ folgt aus dem ersten Teil der Aufgabe, dass $\mathrm{ggT}(a + b, ab) = 1$ ist.

Aufgabe 2

In Kongruenzen ausgedrückt lautet unsere Aufgabe, die kleinstmögliche natürliche Zahl x zu finden, für die gilt:

$$x \equiv 3(\mathrm{mod}11) \text{ und } x \equiv 8(\mathrm{mod}10) \text{ und } x \equiv 0(\mathrm{mod}9).$$

Da die Reste 11, 10 und 9 paarweise teilerfremd sind, können wir den Chinesischen Restsatz anwenden. Wir setzen $M = 11 \cdot 10 \cdot 9 = 990$, $M_1 = 10 \cdot 9 = 90$, $M_2 = 11 \cdot 9 = 99$ und $M_3 = 11 \cdot 10 = 110$.

Jetzt berechnen wir x_1 mit $90x_1 \equiv 1(\mathrm{mod}11)$. Dazu verwenden wir (sofern uns keine anderen Tricks einfallen) den erweiterten Euklidischen Algorithmus:

$$\begin{aligned} 90 &= 8 \cdot 11 + 2 \\ 11 &= 5 \cdot 2 + 1, \end{aligned}$$

also

$$1 = 11 - 5 \cdot 2 = 11 - 5(90 - 8 \cdot 11) = -5 \cdot 90 + 41 \cdot 11.$$

Eine Lösung x_1 der Kongruenz ist dann $x_1 = -5 + 11 = 6$. Als nächstes berechnen wir x_2 mit $99x_2 \equiv 1(\mathrm{mod}10)$:

$$\begin{aligned} 99 &= 9 \cdot 10 + 9 \\ 10 &= 1 \cdot 9 + 1, \end{aligned}$$

also

$$1 = 10 - 9 = 10 - (99 - 9 \cdot 10) = -99 + 10 \cdot 10.$$

Eine Lösung x_2 der Kongruenz ist dann $x_2 = -1 + 10 = 9$. Zum Schluss berechnen wir x_3 mit $110x_3 \equiv 1(\mathrm{mod}9)$:

$$\begin{aligned} 110 &= 12 \cdot 9 + 2 \\ 9 &= 4 \cdot 2 + 1, \end{aligned}$$

also

$$1 = 9 - 4 \cdot 2 = 9 - 4(110 - 12 \cdot 9) = -4 \cdot 110 + 49 \cdot 9.$$

Eine Lösung x_3 der Kongruenz ist dann $x_3 = -4 + 9 = 5$. Als Ergebnis der simultanen Kongruenz erhalten wir

$$\begin{aligned} x &= (3 \cdot 90 \cdot 6 + 8 \cdot 99 \cdot 9 + 0) \bmod 990 \\ &= (1620 + 7128) \bmod 990 \\ &= (630 + 198) \bmod 990 \\ &= 828. \end{aligned}$$

Aufgabe 3

Sei n eine gerade vollkommene Zahl. Dann ist n von der Form $n = 2^{p-1}(2^p - 1)$, wobei p und $2^p - 1 = q$ Primzahlen sind. Die Teiler d mit $1 < d < n$ von n sind die Zweierpotenzen $2, 2^2, \ldots, 2^{p-1}$ und $q, 2q, \ldots, 2^{p-2}q$.

Es gilt $\sigma(2^i) = 2^{i+1} - 1 < 2 \cdot 2^i$, und es folgt, dass Teiler der Form 2^i mit $1 \leq i \leq p-1$ nicht vollkommen sind.

Es ist $\sigma(2^p - 1) = 2^p - 1 + 1 = 2^p$, denn $2^p - 1$ ist eine Primzahl. Ferner ist $2(2^p - 1) = 2^{p+1} - 2 > 2^p$, denn $2^{p+1} - 2 - 2^p = 2^p - 2 > 0$. Somit ist $d = 2^p - 1$ nicht vollkommen.

Da σ multiplikativ ist, folgt für Teiler $d = 2^i(2^p - 1)$ mit $1 \leq i \leq p-2$

$$\sigma(d) = \sigma(2^i)\sigma(2^p - 1) = (2^{i+1} - 1)2^p = 2^{i+1+p} - 2^p.$$

Ferner ist

$$2d = 2^{i+1}(2^p - 1) = 2^{i+1+p} - 2^{i+1} > 2^{i+1+p} - 2^p,$$

denn $i + 1 < p$. Es folgt, dass d nicht vollkommen ist.

Hier hätten wir auch anders (eleganter) argumentieren können. Der einzige ungerade Teiler einer geraden vollkommenen Zahl $n = 2^{p-1}(2^p - 1)$ ist $2^p - 1$. Dass dieser nicht vollkommen ist, zeigt man wie oben. Die geraden Teiler von n, die kleiner als n sind, sind $2^i(2^p - 1)$ mit $1 \leq i \leq p-2$ und 2^j mit $1 \leq j \leq p-1$. Wären diese vollkommen, so wären sie von der Form $2^{p'-1}(2^{p'} - 1)$, wobei p' und $2^{p'} - 1$ Primzahlen sind. Das ist für die oben genannten Teiler aber nicht der Fall, und es folgt, dass die echten Teiler einer geraden vollkommenen Zahl nicht vollkommen sind.

Aufgabe 4

Sei x eine natürliche Zahl, sodass $(x, x+1, x+2)$ ein pythagoreisches Tripel ist. Dann gilt

$$x^2 + (x+1)^2 = (x+2)^2, \text{ also } x^2 + x^2 + 2x + 1 = x^2 + 4x + 4, \text{ und damit } x^2 - 2x - 3 = 0.$$

Die quadratische Gleichung $x^2 - 2x - 3 = 0$ hat die Lösungen $x = 3$ und $x = -1$. Für ein pythagoreisches Tripel $(x, x+1, x+2)$ benötigen wir ein $x > 0$. Also ist $x = 3$ die einzig mögliche Zahl, für die $(x, x+1, x+2)$ ein pythagoreisches Tripel ist.

Aufgabe 5

Jede Primzahl $p \neq 3$ ist kongruent zu 1 oder 2 modulo 3. Damit ist p^2 für $p \neq 3$ kongruent zu 1 modulo 3.

Sei q eine Primzahl mit $q = p_1^2 + p_2^2 + p_3^2$, wobei p_1, p_2 und p_3 Primzahlen sind. Dann ist $q > 3$. Wenn jedes p_i^2, $1 \leq i \leq 3$, kongruent zu 1 modulo 3 ist, so ist q als Summe kongruent zu 0 modulo 3, also durch 3 teilbar. Das ist ein Widerspruch, denn $q > 3$. Somit ist mindestens ein p_i^2 nicht kongruent zu 1 modulo 3, und dieses p_i muss dann 3 sein.

Aufgabe 6

Es gilt

$$\begin{aligned} a(a^{p-2}b) \bmod p &= a^{p-1}b \bmod p \\ &= (a^{p-1} \bmod p) \cdot (b \bmod p) \\ &= (1 \bmod p) \cdot (b \bmod p) \text{ mit dem kleinen Satz von Fermat} \\ &= b \bmod p. \end{aligned}$$

Es folgt $a(a^{p-2}b) \equiv b \pmod{p}$, das heißt, $a^{p-2}b \bmod p$ ist Lösung von $aX \equiv b \pmod{p}$.

Aufgabe 7

1. Unabhängig von der jeweiligen Eingabe `a` werden zunächst in der 2. Zeile die lokalen Variablen `i` und `M` deklariert und in den folgenden beiden Zeilen 3 und 4 die Menge `M` als leere Menge und die Laufvariable `i` mit 1 initialisiert. In Zeile 5 treten wir in eine `while`-Schleife ein, wobei überprï¿½ft wird, ob das aktuelle `i` kleiner als `a` ist. Dies ist für `a` = 6 und `i` = 1 der Fall. In Zeile 6 wird geprüft, ob $6 \bmod 1 = 0$ gilt. Da dies der Fall ist, wird in Zeile 7 die 1 der Menge `M` hinzugefügt. Anschließend wird in Zeile 9 die Laufvariable `i` um 1 erhï¿½ht, hat also nun den Wert 2. Da $2 < 6$ gilt, wird die `while`-Schleife erneut durchlaufen. Auch die 2 wird der Menge `M` hinzugefügt, da $6 \bmod 2 = 0$ gilt. Die `while`-Schleife wird mit den Werten 3 bis 5 durchlaufen. Dabei gilt $6 \bmod 3 = 0$ und $6 \bmod 4, 6 \bmod 5 \neq 0$. Die Schleife bricht ab, sobald `i` den Wert 6 (= `a`) erhält. Mit dem `print`-Befehl in Zeile 11 wird die Menge `M` ausgegeben. Die Ausgabe lautet $\{1, 2, 3\}$. Für eine Eingabe $a \in \mathbb{N}$ gibt die Prozedur alle $1 \leq i < a$ aus, für die $a \bmod i = 0$ gilt, die also ein Teiler von a sind. Es gilt also $\texttt{M} = \{1 \leq i < a \mid i \text{ ist ein Teiler von } a\}$.
2. Die Prozedur `klausur1` gibt nicht alle Teiler der Eingabe `a` aus, sondern nur die, die ungleich `a` sind. Also muss die Prozedur so abgeändert werden, dass auch die Zahl `a` selbst mit ausgegeben wird. Dafür gibt es zwei Möglichkeiten: Entweder wir lassen die `while`-Schleife von 1 bis `a` laufen oder wir fügen nach der `while`-Schleife die Zahl `a` der Menge `M` hinzu. Wir entscheiden uns für die erste Variante und ändern Zeile 5 folgendermaßen ab: `while i <= a do`. Danach sieht die Prozedur folgendermaßen aus:

```
>  # Die Zeilennummerierung ist zu Ihrer Orientierung
01.  klausur1a:=proc(a::posint) #gibt alle positiven Teiler von a aus
02.    local i, M;
03.    M:={};
04.    i:=1;
05.    while i <= a do
06.      if a mod i = 0 then
07.        M:=M union{i};
08.      fi;
09.      i:=i+1;
10.    od;
11.    print(M);
12.  end:
```

Die Ausgabe für a = 6 lautet dann $\{1, 2, 3, 6\}$.

Aufgabe 8

Eine mögliche Prozedur könnte folgendermaßen aussehen:

```
>  klausur2:=proc(a::complex,b::complex) # gibt aus, ob zwei Gauß'sche
Zahlen assoziiert sind
     if a = 1*b then
       return(true);
     fi;
     if a = (-1)*b then
       return (true);
     fi;
     if a = b*I then
       return (true);
     fi;
     if a = b*(-I) then
       return(true);
     fi;
     return(false);
    end:
```