Zeitschrift: Elemente der Mathematik
Herausgeber: Schweizerische Mathematische Gesellschaft
Band: 42 (1987)
Heft: 6

Artikel: Kombinatorik mit dem Computer: Partitionen und Frankaturen
Autor: Jeger, M.
DOI: https://doi.org/10.5169/seals-40045

Nutzungsbedingungen
Die ETH-Bibliothek ist die Anbieterin der digitalisierten Zeitschriften auf E-Periodica. Sie besitzt keine Urheberrechte an den Zeitschriften und ist nicht verantwortlich für deren Inhalte. Die Rechte liegen in der Regel bei den Herausgebern beziehungsweise den externen Rechteinhabern. Das Veröffentlichen von Bildern in Print- und Online-Publikationen sowie auf Social Media-Kanälen oder Webseiten ist nur mit vorheriger Genehmigung der Rechteinhaber erlaubt. Mehr erfahren

Conditions d'utilisation
L'ETH Library est le fournisseur des revues numérisées. Elle ne détient aucun droit d'auteur sur les revues et n'est pas responsable de leur contenu. En règle générale, les droits sont détenus par les éditeurs ou les détenteurs de droits externes. La reproduction d'images dans des publications imprimées ou en ligne ainsi que sur des canaux de médias sociaux ou des sites web n'est autorisée qu'avec l'accord préalable des détenteurs des droits. En savoir plus

Terms of use
The ETH Library is the provider of the digitised journals. It does not own any copyrights to the journals and is not responsible for their content. The rights usually lie with the publishers or the external rights holders. Publishing images in print and online publications, as well as on social media channels or websites, is only permitted with the prior consent of the rights holders. Find out more

Download PDF: 09.07.2025

ETH-Bibliothek Zürich, E-Periodica, https://www.e-periodica.ch

Proof. Any s of the w_i's add to at least $\binom{s}{2}$
$\Leftrightarrow$ any $n-s$ of the w_i's add to at most $\binom{n}{2} - \binom{s}{2}$
$\Leftrightarrow$ any r of the w_i's add to at most $\binom{n}{2} - \binom{n-r}{2}$
$= [1 + 2 + \ldots + (n-1)] - [1 + 2 + \ldots + (n-r-1)]$
$= (n-1) + (n-2) + \ldots + (n-r)$.

V. W. Bryant, Department of Pure Mathematics,
Sheffield University

REFERENCES

1 V. W. Bryant and H. Perfect: Independence Theory in Combinatorics. Chapman and Hall, 1980.
2 D. R. Fulkerson: Upsets in Round Robin Tournaments, Can. J. Math. *17* (1965) 957–969.
3 L. Mirsky: Transversal Theory. Academic Press, 1971.

© 1987 Birkhäuser Verlag, Basel 0013-6018/87/060000-00$1.50+0.20/0

Didaktik und Elementarmathematik

Kombinatorik mit dem Computer: Partitionen und Frankaturen

1. Einleitung

In den verschiedensten Gebieten der Mathematik sieht man sich vor Probleme gestellt, die auf die Abzählung oder die Auflistung bestimmter Partitionen hinauslaufen. Als Partitionen bezeichnet man in der Kombinatorik additive Zerfällungen einer natürlichen Zahl n mit Summanden aus einer vorgegebenen Referenz-Menge, wobei die Reihenfolge der Summanden in einer solchen Figur belanglos ist. Die folgende Aufzählung zeigt die möglichen Partitionen der Zahl 20 über der Referenz-Menge {2, 3, 7}:

```
3+3+7+7
2+3+3+3+3+3
2+2+3+3+3+7
2+2+2+7+7
2+2+2+2+3+3+3+3

2+2+2+2+2+3+7
2+2+2+2+2+2+2+3+3
2+2+2+2+2+2+2+2+2+2

ANZAHL DER FIGUREN: 8
```

Die Summanden in den einzelnen Figuren sind hier nach zunehmenden Werten geordnet.
Kommt in einer Partition der Zahl n über der Referenz-Menge

$$\{a_1, a_2, \ldots, a_p\}$$

der Summand a_j genau x_j-mal vor, dann gilt

$$a_1 x_1 + a_2 x_2 + \ldots + a_p x_p = n. \tag{1.1}$$

Die *Menge der Partitionen* mit den genannten Parametern erweist sich damit als äquivalent mit der *Lösungsmenge der diophantischen Gleichung* (1.1), wenn nichtnegative ganze Werte x_j vorausgesetzt werden. Dieser Zusammenhang macht die Partitionen zugleich zu einem Gegenstand der Zahlentheorie.
Während das Auflisten von Partitionen bislang eine recht mühsame Aufgabe war, gibt es schon lange eingehende Untersuchungen zum Abzählen von Partitionen-Mengen. Erste systematische Untersuchungen über Partitionen finden sich bei L. Euler in dessen „Introductio in Analysin infinitorum" aus dem Jahre 1748 [1*]. Später hat J. J. Sylvester eine generelle Abzähltheorie für Partitionen aufgestellt [2*]. Grundsätzlich kann auf dieser Basis zu jeder Referenz-Menge für die Anzahl f_n der Partitionen von n eine Formel hergeleitet werden. Insbesondere hat Sylvester gezeigt, dass aus der Referenz-Menge $\{a_1, a_2, \ldots, a_p\}$ die formale Potenzreihe [3*]

$$\frac{1}{(1-x^{a_1})(1-x^{a_2})\ldots(1-x^{a_p})} = \sum_{n=0}^{\infty} f_n x^n \tag{1.2}$$

hervorgeht.
Die Sylvestersche Theorie ist leider mit dem Nachteil behaftet, dass sich die Bestimmung der Koeffizienten-Terme f_n mit zunehmendem Umfang der Referenz-Menge und wachsenden Werten der a_j schon bald als unpraktikabel erweist. So erhält man etwa für die Referenz-Menge $\{1, 2, 5\}$ die abzählende Potenzreihe

$$\frac{1}{(1-x)(1-x^2)(1-x^5)} = \sum_{n=0}^{\infty} f_n x^n.$$

Die Partialbruchzerlegung der linken Seite lautet mit $\omega = e^{i\frac{2\pi}{5}}$

$$\frac{1}{(1-x)(1-x^2)(1-x^5)} = \frac{1}{10}\frac{1}{(1-x)^3} + \frac{1}{4}\frac{1}{(1-x)^2} + \frac{13}{40}\frac{1}{1-x} + \frac{1}{8}\frac{1}{1+x}$$

$$+ \frac{1}{5}\frac{1}{(1-\omega-\omega^2+\omega^3)}\left(\frac{\omega^3}{(1-\omega x)} - \frac{\omega^2}{(1-\omega^2 x)} - \frac{\omega}{(1-\omega^3 x)} + \frac{1}{(1-\omega^4 x)}\right).$$

Unter Abstützung auf die für beliebiges $s \in N$ geltende Entwicklung

$$\frac{1}{(1-ax)^s} = \sum_{n=0}^{\infty} \binom{n+s-1}{s-1} a^n x^n$$

kann daraus die Abzähl-Formel

$$f_n = \left[\frac{(n+4)^2}{20} + \frac{1}{2}\right]$$

gewonnen werden [4*]. Dieses Beispiel lässt erkennen, dass die jeweils erforderliche Partialbruchzerlegung der Reziproken zu einem Polynom ein ernsthaftes Hindernis darstellt.
Im Umgang mit Partitionen hat in den letzten Jahren der Einsatz von Computern eine gewisse Verschiebung der Akzente gebracht. Während Auflistungen von umfangreichen Partitionen-Mengen erst mit Hilfe von Computern möglich geworden sind, scheinen andererseits Abzähl-Formeln zusehends an Bedeutung zu verlieren. Die sogenannte *konstruktive Kombinatorik* ist heute auch innerhalb des Problemfeldes der Partitionen deutlich zu erkennen. Diese Entwicklung nachzeichnend möchte die vorliegende Note je einen schnellen Algorithmus für das Abzählen und das Auflisten von Partitionen über beliebigen Referenz-Mengen vorstellen.

2. Die Abzählung von Partitionen

Zunächst wollen wir uns mit der Abzählung der Partitionen über der Referenz-Menge

$$\{a_1, a_2, \ldots, a_p\}$$

im Lichte der Computer-Mathematik befassen. Insbesondere soll die endliche Anzahl-Folge

$$f_1, f_2, \ldots, f_n$$

mit einem Rechner bestimmt werden. Um zu einem entsprechenden Algorithmus zu gelangen, denken wir uns die vorgegebene Referenz-Menge über die Kette

$$\{a_1\} \to \{a_1, a_2\} \to \{a_1, a_2, a_3\} \to \cdots \to \{a_1, a_2, \ldots a_p\}$$

schrittweise aufgebaut. Zur k-elementigen Zwischenstufe $\{a_1, a_2, \ldots, a_k\}$ möge die Anzahl-Folge

$$f_0^{(k)}, f_1^{(k)}, f_2^{(k)}, \ldots, f_n^{(k)} \tag{2.1}$$

gehören, wobei $f_0^{(k)} := 1$ gesetzt ist. Diese Verlängerung der eigentlichen Anzahl-Folge nach links wird sich gleich als sehr zweckmässig erweisen.
Es ist jetzt

$$f_j = f_j^{(p)} \quad \text{für alle} \quad 0 \leqq j \leqq n.$$

Wird nun zur Referenz-Menge $\{a_1, a_2, \ldots, a_k\}$ das Element a_{k+1} adjungiert, dann gilt für alle $a_{k+1} \leqq j \leqq n$

$$f_j^{(k+1)} = f_j^{(k)} + f_{j-a_{k+1}}^{(k)} + f_{j-2a_{k+1}}^{(k)} + \ldots . \tag{2.2}$$

Die Summe ist jeweils so weit zu erstrecken, als der untere Index nicht-negativ ausfällt. Das Glied $f_{j-h a_{k+1}}^{(k)}$ auf der rechten Seite zählt die Partitionen der Zahl j über

der neuen Referenz-Menge, in denen der zuletzt adjungierte Summand a_{k+1} genau h-mal vorkommt.
Die Summe (2.2) kann mit dem Glied $f_0^{(k)}$ endigen, nämlich dann, wenn es zur Zahl j genau eine Partition mit lauter Summanden a_{k+1} gibt. Eine allenfalls vorhandene Figur dieser Art wird jetzt aufgrund der zuvor eingeführten Ergänzungswerte $f_0^{(k)} = 1$ korrekt mitgezählt.
Bei Verwendung der Programmier-Sprache BASIC können die mit dem Übergang von k zu $k+1$ anfallenden Summen (2.2) durch folgende FOR-NEXT-Schleife herbeigeführt werden:

```
FOR J = A(K + 1) TO N
   F(j) = F(J) + F(J − A(K + 1))                    (2.3)
NEXT J
```

Ist etwa $a_{k+1} = 5$ und $n = 20$, dann bewirkt (2.3) die Ersetzungen

$$\begin{aligned} f_5 &:= f_5 + f_0 \\ f_6 &:= f_6 + f_1 \\ f_7 &:= f_7 + f_2 \\ f_8 &:= f_8 + f_3 \\ f_9 &:= f_9 + f_4 \\ f_{10} &:= f_{10} + f_5 \\ &\vdots \\ f_{20} &:= f_{20} + f_{15} \end{aligned}$$

Da diese Ersetzungen in der notierten Reihenfolge realisiert werden, folgt daraus z. B.

$$f_{20}^{(k+1)} = f_{20}^{(k)} + f_{15}^{(k)} + f_{10}^{(k)} + f_5^{(k)} + f_0^{(k)}$$

und dies ist die Summe (2.2) für $j = 20$.
Es zeichnet sich jetzt die Möglichkeit ab, die Zahlen

$$f_0^{(p)}, f_1^{(p)}, \ldots, f_n^{(p)}$$

rekursiv zu berechnen. Wird die Referenz-Menge in DATA-Zeilen abgelegt, dann kann die massgebende Anzahl-Folge über das anschließend in SIMONS-BASIC [5*] notierte Rechner-Programm aufgebaut werden:

```
DIM F(N)
F(0) = 1
K = 1
LOOP
READ A
EXIT IF A > N
FOR J = A TO N
   F(J) = F(J) + F(J − A)
NEXT J
```

```
K = K + 1
EXIT IF K > P
END LOOP
  :
DATA P
DATA A(1), A(2), ..., A(P)
```

Als Start-Folge ist

$$f_0^{(0)} := 1,\ f_1^{(0)} := 0,\ f_2^{(0)} := 0, \ldots,\ f_n^{(0)} := 0$$

zu wählen; dazu sind die Anweisungen

```
DIM F(N): F(0) = 1
```

erforderlich.
Bei einem vorgegebenen maximalen Index n kommen nur jene Elemente der Referenz-Menge zum Tragen, die kleiner als n sind. Setzt man eine Anordnung der Referenz-Menge in den entsprechenden DATA-Zahlen nach aufsteigenden Werten voraus, dann kann man die LOOP-Schleife verlassen, sobald $a > n$ ist (EXIT IF A > N).
Im anschliessenden Rechner-Programm ist die Referenz-Menge

$$\{1, 2, 5, 10, 20, 50\}$$

enthalten (DATA-Zahlen 1850 und 1890).

Rechner-Programm 1

```
1000 REM ******************************
1010 REM *   ANZAHL DER PARTITIONEN   *
1020 REM * UEBER EINER REFERENZ-MENGE *
1030 REM *      [A1,A2,..... ,AP]     *
1040 REM *                            *
1050 REM *      A1<A2< ......  <AP    *
1060 REM ******************************
1070 :
1080 :
1090 REM INFORMATION
1100 PRINT CHR$(147)
1110 PRINT"DIE ELEMENTE DER"
1120 PRINT"REFERENZ-MENGE"
1130 PRINT"SIND IN DATA-ZEILEN"
1140 PRINT"ABZULEGEN !"
1150 PRINT
1160 :
1170 :
1180 REM EINGABE
1190 PRINT"BIS ZU WELCHEM INDEX N"
1200 PRINT"SOLL DIE FOLGE DER"
1210 PRINT"PARTITIONS-ZAHLEN"
1220 PRINT"BESTIMMT WERDEN ?"
1230 PRINT
1240 INPUT"INDEX N EINGEBEN";N
1250 TI$="000000"
1260 PRINT CHR$(147)
1270 :
1280 :
1290 REM LEGENDE
1300 PRINT SPC(2)"PARTITIONEN UEBER DER"
1310 PRINT SPC(2)"REFERENZ-MENGE"
1320 PRINT
1330 :
1340 :
1350 REM PARTITIONSZAHLEN BERECHNEN
1360 DIM F(N)
1370 READ P
1380 F(0)=1
1390 K=1: R$="["
1400 LOOP
1410 READ A
1420 A$=STR$(A): L=LEN(A$)
1430 EXIT IF A>N
1440 R$=R$+RIGHT$(A$,L-1)+" "
1450 : FOR J=A TO N
1460 : F(J)=F(J)+F(J-A)
1470 : NEXT J
1480 K=K+1
1490 EXIT IF K>P
1500 END LOOP
1510 :
```

```
1520 :
1530 REM REFERENZ-MENGE AUSDRUCKEN
1540 L=LEN(R$)
1550 R$=LEFT$(R$,L-1)+"]"
1560 PRINT SPC(5)R$
1570 PRINT
1580 :
1590 :
1600 REM TABELLE AUSDRUCKEN
1610 PRINT SPC(9)"INDEX";
1620 PRINT SPC(2)"PARTITIONSZAHL"
1630 PRINT
1640 EXEC OUTPUT
1650 PRINT
1660 PRINT"  RECHENZEIT: ";SPC(2)TI$
1670 END
1680 :
1690 :
1700 REM PROZEDUR OUTPUT
```

```
1710 PROC OUTPUT
1720 U=0
1730 FOR I=1 TO N
1740 U=U+1
1750 I$=STR$(I): L=LEN(I$)
1760 PRINT SPC(8-L)I$;
1770 F$=STR$(F(I)): L=LEN(F$)
1780 PRINT SPC(10-L)F$
1790 IF FRAC(U/5)=0 THEN PRINT
1800 NEXT I
1810 END PROC
1820 :
1830 :
1840 REM UMFANG DER REFERENZ-MENGE
1850 DATA 6
1860 :
1870 :
1880 REM ELEMENTE DER REFERENZ-MENGE
1890 DATA 1,2,5,10,20,50
```

Mit dem vorliegenden Programm kann das folgende *Geldwechsel-Problem* gelöst werden.

Aufgabe 1: In der Schweiz gibt es sogenannte Scheide-Münzen zu 1, 2, 5, 10, 20 und 50 Rappen. Auf wieviele Arten kann man mit solchen Geldstücken Beträge von 10, 20, ..., 100 Rappen zusammenstellen?

Hier sind zur genannten Referenz-Menge nur die Zahlen $f_{10}, f_{20}, \ldots, f_{100}$ gesucht. Man kann den Output auf diese Teilmenge beschränken, indem man die Zeile 1730 in der Ausgabe-Prozedur wie folgt ersetzt:

1730 FOR I = 10 TO N STEP 10.

Dem Drucker-Output kann entnommen werden, dass etwa der Betrag von 1 Franken auf $f_{100} = 4562$ Arten in Kleingeld ausbezahlt werden kann.

```
PARTITIONEN UEBER DER
REFERENZ-MENGE

   [1,2,5,10,20,50]

  INDEX  PARTITIONSZAHL

    10         11
    20         41
    30        103
    40        236
    50        451

    60        793
    70       1311
    80       2069
    90       3121
   100       4562

RECHENZEIT:  000008
```

```
PARTITIONEN UEBER DER
REFERENZ-MENGE

   [1,2,5,10,20,50,100,500,1000]

  INDEX  PARTITIONSZAHL

   500          582292
  1000       2103220619

RECHENZEIT:  000141
```

Bemerkenswert ist die effektive Rechenzeit von nur 8 Sekunden; man hat sich zu vergegenwärtigen, dass auch dem verkürzten Output eine 100-gliedrige Anzahl-Folge zugrunde liegt, die vollständig berechnet werden muss. Wir haben es hier offensichtlich mit einem sehr schnellen Algorithmus zu tun.
Der Übergang zu einer andern Referenz-Menge kann durch entsprechendes Überschreiben der DATA-Zeilen vorgenommen werden.

Aufgabe 2: Auf wieviele Arten kann man Beträge von 500 Franken und 1000 Franken mit Münzen zu 1, 2 und 5 Franken und Noten zu 10, 20, 50, 100, 500 und 1000 Franken zusammenstellen?

Die massgebende Referenz-Menge ist jetzt

$$\{1, 2, 5, 10, 20, 50, 100, 500, 1000\}.$$

Mit der modifizierten Anweisung

FOR I = 500 TO 1000 STEP 500

in der Ausgabe-Prozedur werden die beiden gesuchten Zahlen ausgedruckt.
Es sei darauf hingewiesen, dass mit dem präsentierten Rechner-Programm auch beim *Ausdrucken der Referenz-Menge* nur jene Elemente berücksichtigt werden, die noch in die vorgesehene Anzahl-Folge $f_1, f_2, \ldots, f_n$ eingehen.
Im Zentrum des dargelegten Abzähl-Algorithmus stehen die mit elementaren Überlegungen gewonnenen Beziehungen (2.2). Wir wollen diese abschliessend auch noch mit dem Abzählverfahren in Verbindung bringen, das mit Potenzreihen arbeitet. Verpackt man die unendliche Folge

$$f_0^{(k)}, f_1^{(k)}, \ldots, f_j^{(k)}, \ldots$$

in die Potenzreihe

$$\varphi_k(x) = \sum_{j=0}^{\infty} f_j^{(k)} x^j,$$

dann beinhalten die Beziehungen (2.2) die Rekursionsformel

$$\varphi_{k+1}(x) = (1 + x^{a_{k+1}} + x^{2a_{k+1}} + \ldots)\, \varphi_k(x) = \left(\frac{1}{1 - x^{a_{k+1}}}\right) \varphi_k(x).$$

Wegen $f_0^{(0)} = 1$, $f_j^{(0)} = 0$ für $j > 0$ ist $\varphi_0(x) = 1$ und damit

$$\varphi_p(x) = \frac{1}{1 - x^{a_1}} \frac{1}{1 - x^{a_2}} \cdots \frac{1}{1 - x^{a_p}}$$

womit die Sylvestersche Reihe für die Partitionen über der Referenz-Menge $\{a_1, a_2, \ldots, a_p\}$ hergeleitet ist.
Unser Algorithmus ist ausgelegt auf die Berechnung der endlichen Folge

$$f_0^{(p)}, f_1^{(p)}, \ldots, f_n^{(p)}.$$

Diese Zahlen sind die Koeffizienten des sogenannten *Vorpolynoms vom Grad n* [6*]

$$\varphi_p^{(n)}(x) = \sum_{j=0}^{n} f_j^{(p)} x^j$$

der Potenzreihe $\varphi_p(x)$.
Der hier entwickelte Algorithmus kann übrigens leicht so modifiziert werden, dass die *Partitionen der Zahlen 1, 2, 3, ..., n mit genau s Summanden* aus der Referenz-Menge $\{a_1, a_2, \ldots, a_p\}$ abgezählt werden. Auch dies ist eine Problemstellung der abzählenden Kombinatorik, der man sich gelegentlich gegenübergestellt sieht. Da dieses neue Abzählproblem eine 2-parametrige Abzähl-Folge impliziert, ist zu Beginn ein entsprechendes 2-dimensionales Parameter-Feld zu reservieren. Der Kern des modifizierten Programmes lautet dann wie folgt:

```
DIM F(N, S)
F(0, 0) = 1
K = 1
LOOP
READ A
EXIT IF A > N
FOR H = 1 TO S
  FOR J = A TO N
  F(J, H) = F(J, H) + F(J − A, H − 1)
  NEXT J
NEXT H
K = K + 1
EXIT IF K > P
END LOOP
  ⋮
DATA P
DATA A(1), A(2), ..., A(P)
```

Anstelle von (2, 2) ist nämlich jetzt die Rekursionsformel

$$f_{(j,h)}^{(k+1)} = f_{(j,h)}^{(k)} + f_{(j-a_{k+1},h-1)}^{(k)} + f_{(j-2a_{k+1},h-2)}^{(k)} + \cdots \tag{2.4}$$

zuständig. Die Summe ist jeweils soweit zu erstrecken, als im Paar $(j - r.\, a_{k+1}, h - r)$ beide Komponenten nicht-negativ ausfallen.
Gezählt werden mit dem neuen Algorithmus zugleich die Lösungen des diophantischen Gleichungssystems

$$\begin{aligned} a_1 x_1 + a_2 x_2 + \cdots + a_p x_p &= j \\ x_1 + \quad x_2 + \cdots + \quad x_p &= s \end{aligned} \tag{2.5}$$

für $j = 1, 2, \ldots, n$.

3. Die Auflistung von Partitionen

Unser nächstes Ziel ist ein schneller Algorithmus für die Auflistung der Partitionen zur Zahl n über der Referenz-Menge $\{a_1, a_2, \ldots, a_p\}$ in der Form der Lösungsvektoren

$$(x_1, x_2, \ldots, x_p)$$

von (1.1). Wie in der Kombinatorik üblich, wollen wir fortan den Parameter n als Index der betreffenden Figuren bezeichnen.
Jede Auflistung einer Figuren-Menge setzt die Festlegung einer bestimmten Reihenfolge voraus. Der Auflist-Algorithmus besteht dann im wesentlichen in der Konstruktion des Nachfolgers zu einer beliebigen Figur in der Liste. Im vorliegenden Falle entscheiden wir uns für die *rückläufige lexikographische Anordnung* der Lösungsvektoren, weil diese mit einer relativ einfachen Nachfolger-Konstruktion einhergeht.
Beim gewählten Auflist-Konzept ergibt sich für die Referenz-Menge {2, 3, 5, 7} zum Index $n = 18$ der folgende Computer-Output:

```
PARTITIONEN VOM INDEX N= 18
UEBER DER REFERENZ-MENGE:

  [ 2 , 3 , 5 , 7 ]

FIGUREN:

  ( 9, 0, 0, 0)
  ( 6, 2, 0, 0)
  ( 3, 4, 0, 0)
  ( 0, 6, 0, 0)
  ( 5, 1, 1, 0)

  ( 2, 3, 1, 0)
  ( 4, 0, 2, 0)
  ( 1, 2, 2, 0)
  ( 0, 1, 3, 0)
  ( 4, 1, 0, 1)

  ( 1, 3, 0, 1)
  ( 3, 0, 1, 1)
  ( 0, 2, 1, 1)
  ( 2, 0, 0, 2)

ANZAHL FIGUREN: 14

RECHENZEIT:000006
```

Wir wollen daran die anschliessend dargelegte Nachfolger-Konstruktion exemplarisch einsichtig machen.
Um von einem bestimmten Lösungsvektor

$$(x_1, x_2, \ldots, x_p)$$

zum Nachfolger zu gelangen, suche man zunächst den kleinsten Komponenten-Index $j > 1$ auf mit der Eigenschaft, dass x_j unter alleiniger Berücksichtigung der gerade vorhandenen Werte von $x_{j+1}, x_{j+2}, \ldots, x_p$ um eine Einheit erhöht werden kann.

Ein solcher Sprung an der Stelle j ist genau dann möglich, wenn

$$a_j(x_j+1) \leqq n - \sum_{h=j+1}^{p} a_h x_h \tag{3.1}$$

ist. Die entsprechende Abfrage legt die Einführung des Rechen-Parameters

$$s = \sum_{h=j+1}^{p} a_h x_h$$

nahe, dessen Wert mit jeder Änderung von j nachzuführen ist. Sobald der Sprung-Index j feststeht, sind die Anweisungen

$$x_1 := (n-s)/a_1,\ x_2 := 0,\ x_3 := 0, \ldots, x_{j-1} := 0 \quad \text{und} \quad j := 1$$

vorzunehmen. Falls nun x_1 ganzzahlig ausfällt, ist die Nachfolger-Figur gefunden. Andernfalls benutze man den erhaltenen nicht-zulässigen Vektor als neue Start-Figur.
Beim vorangestellten Beispiel kann etwa im Lösungsvektor $(0, 1, 3, 0)$ erst die 4. Komponente um eine Einheit erhöht werden. Nach Ausführung dieses Sprunges gelangt man zum Vektor $(11/2, 0, 0, 1)$. Da dessen erste Komponente nicht ganzzahlig ist, hat man von diesem Vektor ausgehend den nächsten Sprung-Index aufzusuchen; dieser liegt bei $j = 2$. Die Ersetzung $x_2 := x_2 + 1$ ergibt nun den Vektor $(8/2, 1, 0, 1)$ und dies ist die nächste Figur in der Liste.
Die gegebene verbale Umschreibung des Auflist-Algorithmus kann nun sofort in ein Fluss-Diagramm umgesetzt werden (siehe Seite 166).

Da unserem Auflist-Algorithmus eine auf wenigen Rechenschritten beruhende Nachfolger-Konstruktion zugrunde liegt, sind relativ kurze Rechenzeiten zu erwarten. Das vorangestellte Rechenbeispiel bestätigt diese Vermutung.

4. Die freien Partitionen

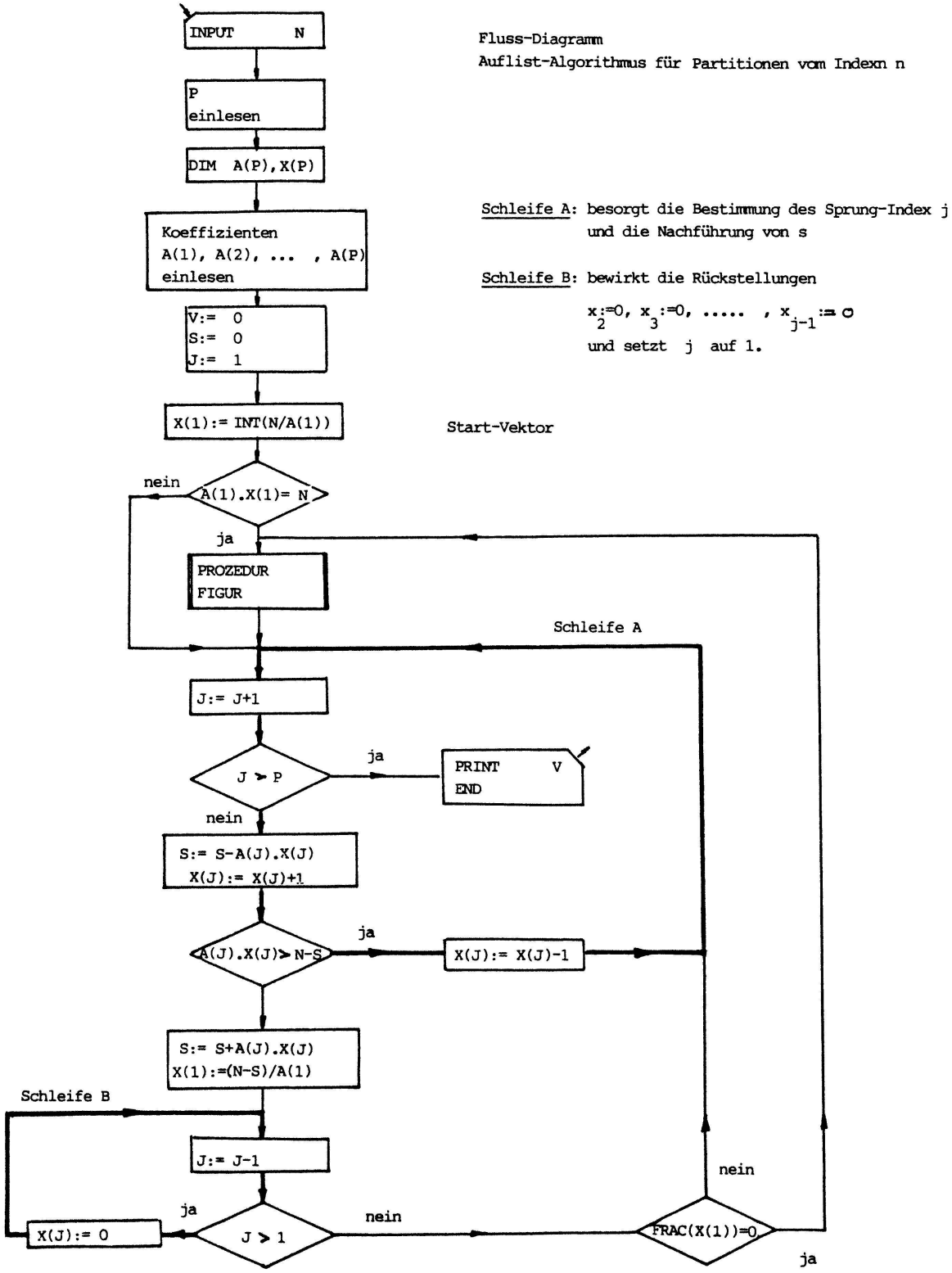
Bis jetzt wurden nur Partitionen über einer Referenz-Menge

$$R = \{a_1, a_2, \ldots, a_p\} \subset N$$

betrachtet, d.h. R war eine endliche Teilmenge von N. Wir lassen nun die Beschränkung auf Teilmengen von N fallen und setzen $R = N$. Man spricht dann von *freien Partitionen vom Index n.* Bei dieser Öffnung der Referenzmenge tritt an die Stelle von (1.1) die diophantische Gleichung

$$x_1 + 2x_2 + \ldots n\,x_n = n. \tag{4.1}$$

Fluss-Diagramm
Auflist-Algorithmus für Partitionen vom Index *n*



Fluss-Diagramm
Auflist-Algorithmus für Partitionen vom Indexn n

Schleife A: besorgt die Bestimmung des Sprung-Index j und die Nachführung von s

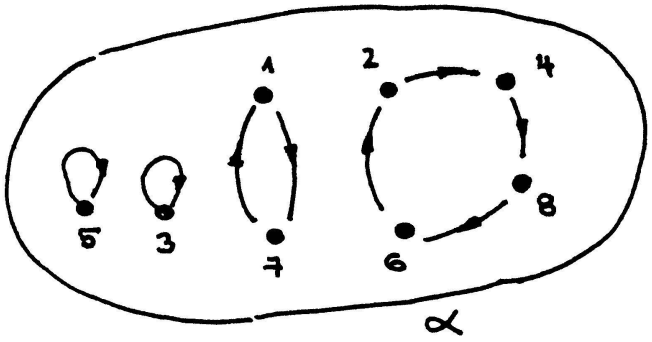
Schleife B: bewirkt die Rückstellungen

$x_2:=0,\ x_3:=0,\ \ldots\ ,\ x_{j-1}:=0$

und setzt j auf 1.

Grössere Summanden als n können nämlich bei einer additiven Zerfällung der Zahl n nicht auftreten, d.h. bei den freien Partitionen vom Index n kommen jeweils nur die Summanden aus $N_n = \{1, 2, \ldots, n\}$ zum Tragen.
Die Gleichung (4.1) ist in verschiedenen Gebieten der Mathematik anzutreffen. So beschreibt z.B. jeder Lösungsvektor von (4.1) eine mögliche Zyklen-Struktur in der Gruppe $\mathfrak{S}_n$ der Permutationen von n Objekten (symmetrische Gruppe vom Grad n).

Zyklen-Struktur (2, 1, 0, 1, 0, 0, 0, 0)



Figur 1.

Die Fig. 1 zeigt den Di-Graphen zu einer Permutation α aus $\mathfrak{S}_8$ mit der Zyklen-Struktur (2, 1, 0, 1, 0, 0, 0, 0), d.h. α zerfällt in 2 Zyklen der Länge 1 und je einen Zyklus der Längen 2 und 4.
Wir betrachten jetzt die Anzahl der Lösungen von (4.1) in nicht-negativen ganzen Zahlen mit p_n. Mit der Folge der *Partitionszahlen*

$$p_0, p_1, p_2, \ldots, p_n, \ldots \qquad (p_0 := 1)$$

werden aufgrund der letzten Bemerkungen zugleich die Zyklen-Strukturen in den Gruppen $\mathfrak{S}_n$ abgezählt. Andererseits weist diese Folge auch mancherlei Bezüge zur Zahlentheorie auf [7*].
Die beiden präsentierten Algorithmen für die Abzählung und für die Auflistung von Partitionen über endlichen Referenz-Mengen können problemlos auf die freien Partitionen übertragen werden.
Für das Abzählen der Figuren bis zu einem vorgegebenen Höchstindex n reicht N_n als Referenz-Menge aus. Diese kann jetzt zusammen mit der Konstruktion der Koeffizienten $f_j^{(k)}$ sukzessive hineingebracht werden, was die früheren DATA-Zeilen entbehrlich macht. Der Abschnitt des Rechner-Programms, der den mathematischen Kern des Algorithmus enthält, ist dann wie folgt zu modifizieren:

Rechner-Programm 2

```
1350 REM HAUPTPROGRAMM
1360 DIM P(N)
1370 P(0)=1: A=0
1380 LOOP
1390 A=A+1
1400 EXIT IF A>N
1410 : FOR J=A TO N
1420 : P(J)=P(J)+P(J-A)
1430 : NEXT J
1440 END LOOP
1450 EXEC OUTPUT
1460 PRINT
1470 PRINT RECHENZEIT: ;SPC(2)TI$
1480 END
```

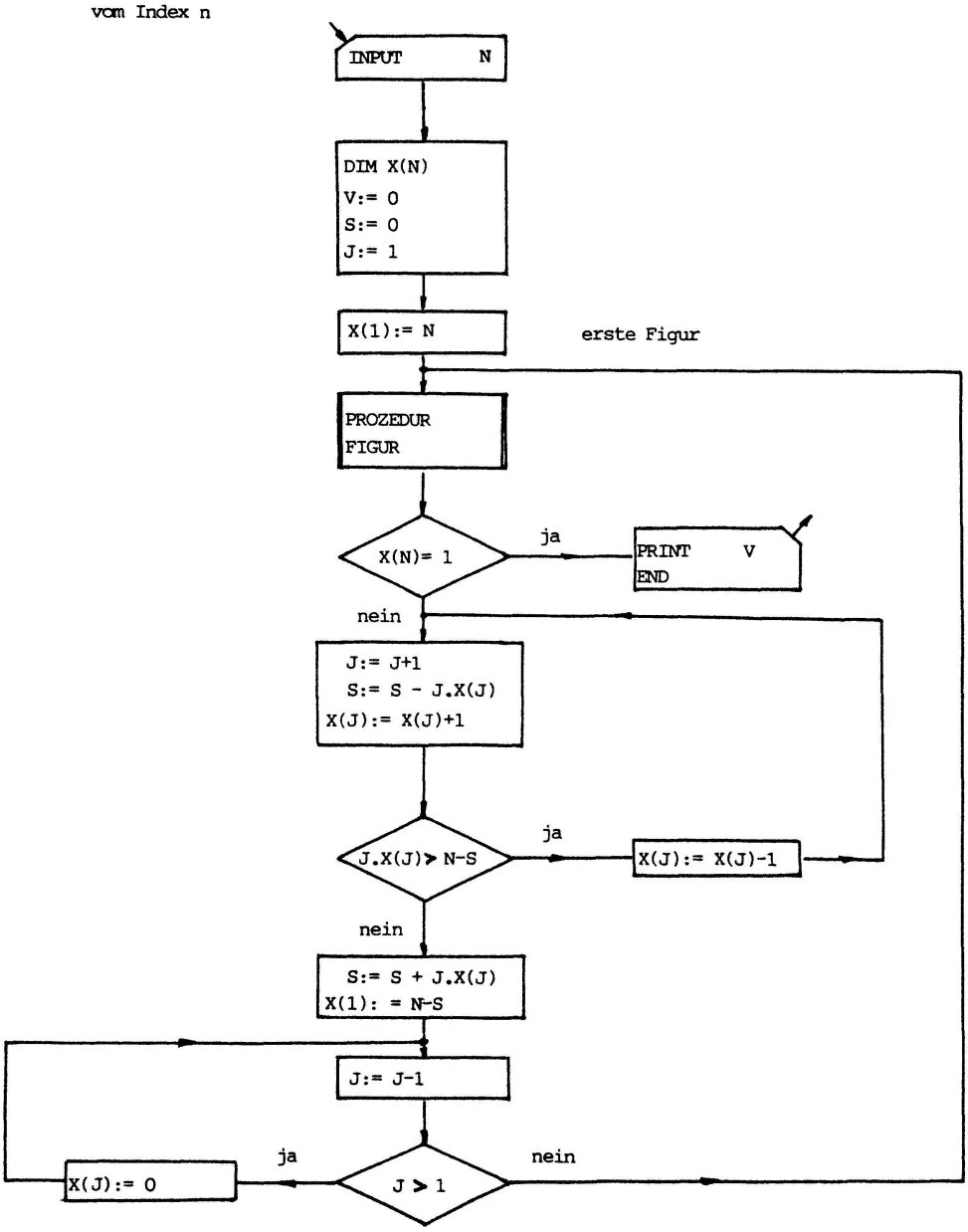
Darin sind die früheren Parameter f_j in Anpassung an die neue Bezeichnungsweise durch p_j ersetzt. Der anschliessende Rechner-Output zeigt eine Tafel der ersten 100 Partitionszahlen ($n = 100$). Bemerkenswert ist wiederum die kurze Rechenzeit von nur 1'20" (Rechner Commodore C 64).

```
ANZAHL DER
FREIEN PARTITIONEN
VOM INDEX N

INDEX  PARTITIONSZAHL

    1              1
    2              2
    3              3
    4              5
    5              7

    6             11
    7             15
    8             22
    9             30
   10             42

   11             56
   12             77
   13            101
   14            135
   15            176

   16            231
   17            297
   18            385
   19            490
   20            627

   21            792
   22           1002
   23           1255
   24           1575
   25           1958

   26           2436
   27           3010
   28           3718
   29           4565
   30           5604

   31           6842
   32           8349
   33          10143
   34          12310
   35          14883

   36          17977
   37          21637
   38          26015
   39          31185
   40          37338

   41          44583
   42          53174
   43          63261
   44          75175
   45          89134

   46         105558
   47         124754
   48         147273
   49         173525
   50         204226

   51         239943
   52         281589
   53         329931
   54         386155
   55         451276

   56         526823
   57         614154
   58         715220
   59         831820
   60         966467

   61        1121505
   62        1300156
   63        1505499
   64        1741630
   65        2012558

   66        2323520
   67        2679689
   68        3087735
   69        3554345
   70        4087968

   71        4697205
   72        5392783
   73        6185689
   74        7089500
   75        8118264

   76        9289091
   77       10619863
   78       12132164
   79       13848650
   80       15796476

   81       18004327
   82       20506255
   83       23338469
   84       26543660
   85       30167357

   86       34262962
   87       38887673
   88       44108109
   89       49995925
   90       56634173

   91       64112359
   92       72533807
   93       82010177
   94       92669720
   95      104651419

   96      118114304
   97      133239331
   98      150139134
   99      169229875
  100      190569292

RECHENZEIT:  000126
```

Im Auflist-Algorithmus für die Figuren vom Index n können im Falle der freien Partitionen verschiedene Vereinfachungen vorgenommen werden. Zum einen sind jetzt die erste und die letzte Figur in der Liste bekannt: Am Anfang der Liste steht

nämlich der Lösungsvektor $(n, 0, 0, \ldots, 0, 0)$ und Schluss-Figur ist jeweils der Lösungsvektor $(0, 0, 0, \ldots, 0, 1)$. Andererseits ist der berechnete Wert von x_1 wegen der besonderen Struktur der Gleichung (4.1) immer ganzzahlig, so dass die entsprechende Abfrage auf Ganzzahligkeit entfallen kann. Der aufgrund dieser Überlegungen verkürzte Auflist-Algorithmus für die freien Partitionen vom Index n ist im anschliessenden Fluss-Diagramm festgehalten.

Fluss-Diagramm
Algorithmus für die Auflistung der freien Partitionen vom Index n




```
FREIE PARTITIONEN
VOM INDEX N

N= 10

FIGUREN:

(10. 0. 0. 0. 0. 0. 0. 0. 0. 0)
( 8. 1. 0. 0. 0. 0. 0. 0. 0. 0)
( 6. 2. 0. 0. 0. 0. 0. 0. 0. 0)
( 4. 3. 0. 0. 0. 0. 0. 0. 0. 0)
( 2. 4. 0. 0. 0. 0. 0. 0. 0. 0)

( 0. 5. 0. 0. 0. 0. 0. 0. 0. 0)
( 7. 0. 1. 0. 0. 0. 0. 0. 0. 0)
( 5. 1. 1. 0. 0. 0. 0. 0. 0. 0)
( 3. 2. 1. 0. 0. 0. 0. 0. 0. 0)
( 1. 3. 1. 0. 0. 0. 0. 0. 0. 0)

( 4. 0. 2. 0. 0. 0. 0. 0. 0. 0)
( 2. 1. 2. 0. 0. 0. 0. 0. 0. 0)
( 0. 2. 2. 0. 0. 0. 0. 0. 0. 0)
( 1. 0. 3. 0. 0. 0. 0. 0. 0. 0)
( 6. 0. 0. 1. 0. 0. 0. 0. 0. 0)

( 4. 1. 0. 1. 0. 0. 0. 0. 0. 0)
( 2. 2. 0. 1. 0. 0. 0. 0. 0. 0)
( 0. 3. 0. 1. 0. 0. 0. 0. 0. 0)
( 3. 0. 1. 1. 0. 0. 0. 0. 0. 0)
( 1. 1. 1. 1. 0. 0. 0. 0. 0. 0)

( 0. 0. 2. 1. 0. 0. 0. 0. 0. 0)
( 2. 0. 0. 2. 0. 0. 0. 0. 0. 0)
( 0. 1. 0. 2. 0. 0. 0. 0. 0. 0)
( 5. 0. 0. 0. 1. 0. 0. 0. 0. 0)
( 3. 1. 0. 0. 1. 0. 0. 0. 0. 0)

( 1. 2. 0. 0. 1. 0. 0. 0. 0. 0)
( 2. 0. 1. 0. 1. 0. 0. 0. 0. 0)
( 0. 1. 1. 0. 1. 0. 0. 0. 0. 0)
( 1. 0. 0. 1. 1. 0. 0. 0. 0. 0)
( 0. 0. 0. 0. 2. 0. 0. 0. 0. 0)

( 4. 0. 0. 0. 0. 1. 0. 0. 0. 0)
( 2. 1. 0. 0. 0. 1. 0. 0. 0. 0)
( 0. 2. 0. 0. 0. 1. 0. 0. 0. 0)
( 1. 0. 1. 0. 0. 1. 0. 0. 0. 0)
( 0. 0. 0. 1. 0. 1. 0. 0. 0. 0)

( 3. 0. 0. 0. 0. 0. 1. 0. 0. 0)
( 1. 1. 0. 0. 0. 0. 1. 0. 0. 0)
( 0. 0. 1. 0. 0. 0. 1. 0. 0. 0)
( 2. 0. 0. 0. 0. 0. 0. 1. 0. 0)
( 0. 1. 0. 0. 0. 0. 0. 1. 0. 0)

( 1. 0. 0. 0. 0. 0. 0. 0. 1. 0)
( 0. 0. 0. 0. 0. 0. 0. 0. 0. 1)

ANZAHL FIGUREN: 42

RECHENZEIT: 000020
```

5. Frankatur-Probleme

Mit den beiden Primär-Algorithmen können auch noch gewisse Fragen aus der Kombinatorik geklärt werden, die auf den ersten Blick nicht mehr zum Problemfeld der Partitionen zu gehören scheinen. Als Aufhänger für die folgenden Schlussbetrachtungen diene die

Aufgabe 3: Die schweizerische Postverwaltung hat zu einem gewissen Zeitpunkt 5 verschiedene 10-Rappen-Marken und 3 verschiedene 20-Rappen-Marken im Umlauf. Auf wieviele und auf welche Arten kann man damit ein Porto im Betrage von 50 Rappen zusammenstellen?
Hier geht es offenbar um die Lösungsmenge der diophantischen Gleichung

$$10x_1 + 10x_2 + 10x_3 + 10x_4 + 10x_5 + 20x_6 + 20x_7 + 20x_8 = 50,$$

die sofort durch die äquivalente Gleichung

$$x_1 + x_2 + x_3 + x_4 + x_5 + 2x_6 + 2x_7 + 2x_8 = 5$$

ersetzt werden kann.
Da die Frankatur-Werte 10 und 20 in mehreren Marken-Typen zur Verfügung stehen, hat man es auf Anhieb nicht mit Partitionen im bisherigen Sinne zu tun. Man kann aber die vorliegenden kombinatorischen Figuren dennoch als Partitionen interpretieren, wenn man die den verschiedenen Marken-Typen entsprechenden Summanden mit einem spezifischen Etikett versieht. Dazu kann man z. B. die Farben der einzelnen Briefmarken verwenden. Auf diese Weise lässt sich eine Referenz-Menge definieren, die etwa aus je einer Zahl 10 mit dem Etikett rot, blau, gelb, grün und violett und aus je einer Zahl 20 mit dem Etikett rot, blau und gelb besteht. Das bisherige Figuren-Modell wird damit auch auf die neue Situation anwendbar.
In Anlehnung an die Einkleidung in der Aufgabe 3 spricht man bei Vorliegen einer Referenz-Menge mit „verschieden gefärbten" gleichwertigen Elementen von einem *Frankatur-Problem.*
Der folgende Drucker-Output (linker Teil) zeigt die zur diophantischen Gleichung

$$x_1 + x_2 + x_3 + x_4 + x_5 + 2x_6 + 2x_7 + 2x_8 = n$$

```
PARTITIONEN UEBER DER
REFERENZ-MENGE

   [1,1,1,1,1,2,2,2]

  INDEX  PARTITIONSZAHL

     0          1
     1          5
     2         18
     3         59
     4        121

     5        261
     6        529
     7        969
     8       1719
     9       2906

    10       4832
    11       7909
    12      11286
    13      16039
    14      24552

    15      35112
    16      49335
    17      68211
    18      92359
    19     124502

    20     166023

RECHENZEIT:  000003
```

```
PARTITIONEN UEBER DER
REFERENZ-MENGE

   [1,1,1,1]

  INDEX  PARTITIONSZAHL

     0          1
     1          4
     2         10
     3         20
     4         35

     5         56
     6         84
     7        120
     8        165
     9        220

    10        286
    11        364
    12        455
    13        560
    14        680

    15        816
    16        969
    17       1140
    18       1330
    19       1540

    20       1771

RECHENZEIT:  000002
```

gehörende Anzahl-Folge bis zum Höchst-Index $n = 20$. Insbesondere kann man daraus entnehmen, dass $f_5 = 261$ ist; es gibt also 261 verschiedene Frankaturen, die den Forderungen der Aufgabe 3 genügen. Mit dem Auflist-Algorithmus gemäss Abschnitt 3 könnten diese 261 Figuren problemlos aufgezählt werden (siehe Seite 170).

Auch die spezielle diophantische Gleichung

$$x_1 + x_2 + x_3 + \ldots + x_p = n, \qquad x_i \in N \cup \{0\} \tag{5.1}$$

kann mit unsern beiden Algorithmen erschlossen werden. In diesem Falle lassen sich die Lösungen auch mit ganz elementaren Überlegungen abzählen.
Steht etwa die Gleichung

$$x_1 + x_2 + x_3 + x_4 = 10$$

zur Diskussion, dann betrachte man die sämtlichen Wörter aus 10 Zeichen 1 und 3 Zeichen /. Jedes dieser Wörter repräsentiert offenbar gerade eine Lösung unserer Gleichung, was sofort aus den folgenden Beispielen abgelesen werden kann:

$$\underbrace{1\ 1}_{2}\ /\ \underbrace{1\ 1}_{2}\ /\ /\ \underbrace{1\ 1\ 1\ 1\ 1\ 1}_{6} \quad \rightarrow \quad x_1 = 2, \quad x_2 = 2, \quad x_3 = 0, \quad x_4 = 6$$

$$/\ \underbrace{1\ 1\ 1}_{3}\ /\ \underbrace{1\ 1\ 1\ 1\ 1}_{5}\ /\ \underbrace{1\ 1}_{2} \quad \rightarrow \quad x_1 = 0, \quad x_2 = 3, \quad x_3 = 5, \quad x_4 = 2$$

$$\underbrace{1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1}_{10}\ /\ /\ /\ \underbrace{0\ 0\ 0}_{} \quad \rightarrow \quad x_1 = 10, \quad x_2 = 0, \quad x_3 = 0, \quad x_4 = 0$$

Bei der allgemeinen Gleichung (5.1) wird man auf Wörter der Länge $n + p - 1$ geführt, die aus genau n Zeichen 1 und $p - 1$ Zeichen / bestehen. Die Anzahl derartiger Wörter ist

$$f_n = \binom{n + p - 1}{p - 1}. \tag{5.2}$$

Mit unserem Abzähl-Algorithmus kann also insbesondere auch die Folge der *Binomial-Koeffizienten mit fester Unterzahl* erzeugt werden. Bemerkenswert ist, dass man dabei ohne irgendwelchen formalen Zusammenhang über Binomialkoeffizienten auskommt. Der letzte Drucker-Output zeigt einen Teil der Folge (5.2) für $p = 4$.
Die Lösungen der diophantischen Gleichung (5.1) beinhalten *additive Zerfällungen der Zahl n in genau p Summanden* über der Menge der nicht-negativen ganzen Zahlen,

wobei jetzt deren *Reihenfolge mitberücksichtigt* wird. So sind jetzt etwa $2+3+0+5$ und $3+5+0+2$ zwei verschiedene Zerfällungen der Zahl 10. Additive Zerfällungen unter Berücksichtigung der Summanden-Reihenfolge werden in der Kombinatorik als *Kompositionen* bezeichnet.

M. Jeger, Mathematik-Departement ETH Zürich

LITERATURVERZEICHNIS

[1] G. E. Andrews: The Theory of Partitions. Encyclopedia of Mathematics and its applications, Vol. 2. London-Amsterdam-Don Mills-Sydney-Tokyo 1976.
[2] L. Comtet: Advanced Combinatorics. Dordrecht 1974.
[3] G. H. Hardy and E. M. Wright: An Introduction to the Theory of Numbers. New York 1960[4].
[4] M. Jeger: Einführung in die Kombinatorik, Bd. 2. Stuttgart 1976.
[5] D. E. Knuth: The Art of Computer Programming, Vol. 1 (Fundamental Algorithms). Menlo Park-London-Amsterdam-Don Mills-Sydney 1968.
[6] E. Netto: Lehrbuch der Combinatorik. Leipzig 1901.
[7] A. Nijenhuis and H. S. Wilf: Combinatorial Algorithms. New York 1975.
[8] D. Stanton and D. White: Constructive Combinatorics. New York-Berlin-Heidelberg-Tokyo 1986.

ANMERKUNGEN

[1*] Vgl. L. Euler: Opera Omnia I,8. Leipzig-Berlin 1922.
[2*] Vgl. [6], p. 147 ff.
[3*] Vgl. etwa [4], p. 11 ff.
[4*] Vgl. [4], p. 104/105.
[5*] SIMONS-BASIC ist eine auf den Rechner Commodore C-64 zugeschnittene BASIC-Erweiterung, die ein strukturiertes Programmieren erlaubt. Diese BASIC-Version bietet zudem direkte Befehle für die Graphik-Programmierung an.
[6*] Vgl. [4], p. 14 ff.
[7*] Vgl. [4], p. 119–122 oder auch [3].

© 1987 Birkhäuser Verlag, Basel 0013-6018/87/060000-00 $1.50+0.20/0

Aufgaben

Aufgabe 950. Man bestimme den kleinsten Wert, den der Durchmesser (d. i. der maximale Abstand von zwei Punkten) einer ebenen nichtkollinearen Menge von vier Punkten mit paarweise ganzzahligen Abständen annehmen kann.

H. Harborth, Braunschweig, BRD

Lösung: Der gesuchte kleinste Wert ist 4. Tatsächlich haben beide Diagonalen des gleichschenkligen Trapezes mit Länge 3 bzw. 4 der parallelen Seiten sowie Schenkellänge 2 die Länge 4, womit der Durchmesser dieses Trapezes gleich 4 ist.