Werk

Titel: Kanonische Codierungen von ...k.

Autor: Wymann-Böni, M.

Jahr: 1991

PURL: https://resolver.sub.uni-goettingen.de/purl?378850199_0046|log34

Kontakt/Contact

Digizeitschriften e.V.
SUB Göttingen
Platz der Göttinger Sieben 1
37073 Göttingen

info@digizeitschriften.de

(see [3, 12.35]; of course, such an equation holds when p, q, r are not integers, except that there may only be rational solutions h). However, h is also related to g by

$$\frac{64h}{g} = 12 - p - 2q - r + \frac{4}{p} + \frac{4}{r}$$

(see [3, 12.81]). In view of our independent calculations for g, this last equation yields an alternative way of finding h.

P. McMullen, University College London

REFERENCES

1 Coxeter H. S. M.: The densities of the regular polytopes. Proc. Camb. Phil. Soc. *27*, 201–211 (1931); *28*, 509–531 (1932).
2 Coxeter H. S. M.: The complete enumeration of the finite groups of the form $R_i^2 = (R_i R_j)^{k_{ij}} = 1$. J. London Math. Soc. *10*, 21–35 (1935).
3 Coxeter H. S. M.: Regular Polytopes (3rd ed.). Dover, New York (1973).
4 Coxeter H. S. M.: Star polytopes and the Schläfli function $f(\alpha, \beta, \gamma)$. Elem. Math. *44*, 25–36 (1989).
5 Grünbaum B.: Convex Polytopes. Wiley-Interscience, London-New York-Sydney (1967).
6 McMullen P.: Regular star-polytopes, and a theorem of Hess. Proc. London Math. Soc. (3) *18*, 577–596 (1968).
7 McMullen P.: Angle-sum relations for polyhedral sets. Mathematika *33*, 173–188 (1986).
8 Shephard G. C.: An elementary proof of Gram's theorem for convex polytopes. Canad. J. Math. *19*, 1214–1217 (1967).

© 1991 Birkhäuser Verlag, Basel 0013-6018/91/050121-10$1.50 + 0.20/0

Kanonische Codierungen von $\mathbb{N}^k$

Als erster brauchte Georg Cantor eine Codierung von $\mathbb{N}^2$, d. h. eine Bijektion zwischen $\mathbb{N}$ und $\mathbb{N}^2$. (Sie entspricht dem weiter unten angegebenen $\langle\ \rangle_2$.) Es gelang ihm, damit zu beweisen, dass unendliche Mengen nicht durch endliches Anwenden des kartesischen Produktes auf sich selbst vergrössert werden können, sondern eben immer dieselbe Mächtigkeit behalten (siehe [1] oder [2]). Heute ist sein Verfahren, das *Cantor*sche Diagonalisierungsverfahren, allgemein bekannt und findet sich in vielen Lehrbüchern, die sich mit dem Aufbau der Zahlensysteme befassen (z. B. [3], [4]). Andere Codierungen von abzählbaren Mengen treten vielerorts in der Logik und Berechenbarkeitstheorie auf, zum Beispiel als Gödelnummern, wenn es darum geht, sowohl die Zahlen selbst, als auch die Funktionen, die man auf sie anwenden kann, in einem einzigen Bereich darzustellen. Ein neueres Beispiel dafür liefern Modelle für den λ-Kalkül [5], in denen die ganze Berechenbarkeitstheorie behandelt werden kann. Dieses letzte Beispel ist insofern bemerkenswert, als es eines der wenigen ist, in denen die explizite Form einer Bijektion zwischen $\mathbb{N}^2$ und $\mathbb{N}$ überhaupt eine Rolle spielt. Für das Standard-Modell wählt man gerade $\langle\ \rangle_2$.

Unter einer *Codierung* von $\mathbb{N}^k$ verstehen wir nun jede bijektive Abbildung $\mathbb{N}^k \to \mathbb{N}$, mithin eine Abbildung, die jedem k-Tupel $(n_1, \ldots, n_k)$ von $\mathbb{N}^k$ in eindeutiger Art und Weise eine Zahl c zuordnet, die wir als *Code* von $(n_1, \ldots, n_k)$ bezeichnen.

Eine übliche Form, Codierungen von $\mathbb{N}^k$ nach $\mathbb{N}$ zu definieren, besteht darin, einmal eine Codierung $\langle\ \rangle_2$ für $\mathbb{N}^2$ explizite anzugeben

$$\langle n, m\rangle_2 := \frac{(m+n)(m+n+1)}{2} + n$$

und die Definition dann induktiv durch

$$\langle n_1, \ldots, n_{k+1}\rangle_{k+1} := \langle\langle n_1, \ldots, n_k\rangle_k, n_{k+1}\rangle_2$$

weiterzuführen.
Wir geben hier eine Familie von Codierungen an, die im Gegensatz zu obiger Variante, direkt aus der geometrischen Anschauung des Diagonalisierens gewonnen sind, und überraschende Eigenschaften, zum Beispiel einfache Darstellungen für die Projektionen $\mathbb{N}^{k+1} \to \mathbb{N}^k$, aufweisen. Des weiteren lassen sich die einzelnen Koordinaten n_i eines Punktes aus $\mathbb{N}^k$ sehr leicht aus seinem Code bestimmen.

Codierung

Im Gegensatz zur üblichen Methode aus einer Codierung der Ebene $\mathbb{N}^2$ durch wiederholtes Anwenden der Paarbildung höherdimensionale $\mathbb{N}^k$ zu codieren, gehen wir direkt von der geometrischen Idee der Diagonalisierung aus, allerdings auch, indem wir eine höherdimensionale Codierung auf eine niedrigerdimensionale zurückführen.
Um nun ausgehend von einer schon bekannten Codierung von $\mathbb{N}^k$ auch $\mathbb{N}^{k+1}$ zu codieren, unterteilen wir den Raum $\mathbb{N}^{k+1} = \{(n_1, \ldots, n_{k+1}) : n_i \in \mathbb{N}\}$ in Hyperebenen mit konstanter Koordinatensumme $s = n_1 + \ldots + n_{k+1}$, und ordnen diese Hyperebenen aufsteigend nach der Koordinatensumme an. Den Code eines Punktes $(n_1, \ldots, n_k, n_{k+1}) \in \mathbb{N}^{k+1}$ erhalten wir dann, indem wir die Anzahl aller Punkte, die in ursprungsnäheren Hyperebenen (mit entsprechend kleiner Koordinatensumme) liegen, zählen und den Code des Punktes $(n_1, \ldots, n_k) \in \mathbb{N}^k$ bezüglich der schon bekannten Codierung $\mathbb{N}^k$ dazu addieren. Die folgende Definition soll dieses Vorgehen formal erfassen.

Definition 1. *Wir definieren Funktionen* $\langle\ \rangle_k : \mathbb{N}^k \to \mathbb{N}$ *gemäss*

$$\langle n_1, \ldots, n_k\rangle_k = \sum_{i=1}^{k} \binom{\sum_{j=1}^{i} n_j + (i-1)}{i}. \tag{1}$$

Wir schauen an einem Beispiel, was diese Definition in der Dimension 3 liefert:

$$\langle 0, 0, m\rangle_3 = \binom{0}{1} + \binom{0}{2} + \binom{m}{3} = \frac{m(m+1)(m+2)}{6}$$

$$\langle m, 0, 0\rangle_3 = \binom{m}{1} + \binom{m+1}{2} + \binom{m+2}{3} = m + \frac{m(m+1)}{2} + \frac{m(m+1)(m+2)}{6}$$

$$\langle m, m, m\rangle_3 = \binom{m}{1} + \binom{2m+1}{2} + \binom{3m+2}{3} = m + \frac{2m(2m+1)}{2} + \frac{3m(3m+1)(3m+2)}{6}.$$

Wir betrachten einige Eigenschaften dieser Funktionen. Offenbar kann man die Funktionen $\langle\ \rangle_k$ darstellen mittels

$$\langle n_1, \ldots, n_k, n_{k+1}\rangle_{k+1} = \langle n_1, \ldots, n_k\rangle_k + \binom{n_1 + \ldots + n_{k+1} + k}{k+1}. \tag{2}$$

Mit den Abkürzungen $s_i = \sum_{j=1}^{i} n_j$, die wir künftig als *Partialsummen* bezeichnen, erhält man als äquivalente dritte Darstellungsform:

$$\langle n_1, \ldots, n_k\rangle_k = \langle s_1\rangle_1 + \langle 0, s_2\rangle_2 + \ldots + \langle 0, \ldots, 0, s_k\rangle_k . \tag{3}$$

In dieser Form wird klar, dass bei fester Koordinatensumme s für einen Punkt $(n_1, \ldots, n_k)$ gilt:

$$\langle n_1, \ldots, n_j+1, \ldots, n_l-1, \ldots, n_k\rangle_k > \langle n_1, \ldots, n_j, \ldots, n_l, \ldots, n_k\rangle_k . \tag{4}$$

Dadurch kann man den Bereich der Codes von Punkten der Hyperebene mit konstanter Koordinatensumme s angeben: Für alle Punkte mit Koordinatensumme s gilt

$$\langle 0, \ldots, 0, s\rangle_k \leq \langle n_1, \ldots, n_k\rangle_k \leq \langle s, 0, \ldots, 0\rangle_k . \tag{5}$$

Ebenso werden Punkte mit grösserer Koordinatensumme auf grössere Nummern abgebildet:

$$\langle s, 0, \ldots, 0\rangle_k + 1 = \langle 0, \ldots, 0, s+1\rangle_k . \tag{6}$$

$$\text{Ist} \quad s_k < s'_k, \quad \text{so ist} \quad \langle n_1, \ldots, n_k\rangle_k < \langle n'_1, \ldots, n'_k\rangle_k . \tag{7}$$

Mit diesen Eigenschaften fällt der Beweis des folgenden Satzes nicht schwer.

Satz 2. *Die Funktionen* $\langle\ \rangle_k : \mathbb{N}^k \to \mathbb{N}$ *gemäss Definition 1 bilden eine Familie von Codierungen für* $\mathbb{N}^k$.

Wir führen den Beweis mit Induktion über die Dimension k der Codierung:
Offensichtlich ist $\langle n_1\rangle_1 = \binom{n_1}{1} = n_1$, also ist $\langle\ \rangle_1$ bijektiv. Dies liefert uns den Induktionsanfang. Wir nehmen nun an, bis zur Dimension k seien alle Funktionen $\langle\ \rangle_i$ bijektiv und leiten daraus ab, dass dann auch $\langle\ \rangle_{k+1}$ bijektiv und somit eine Codierung im definierten Sinne ist.

Unter den obigen Voraussetzungen ist $\langle\ \rangle_{k+1}$ injektiv:
Sei $\langle n_1, \ldots, n_{k+1}\rangle_{k+1} = \langle n'_1, \ldots, n'_{k+1}\rangle_{k+1}$. Wegen (7) müssen die Koordinatensummen übereinstimmen: $s = s'$. Dann gilt wegen (2)

$$\begin{aligned}\langle n_1, \ldots, n_{k+1}\rangle_{k+1} &= \langle n_1, \ldots, n_k\rangle_k + \binom{s+(k-1)}{k} \\ &= \langle n'_1, \ldots, n'_k\rangle_k + \binom{s'+(k-1)}{k} = \langle n'_1, \ldots, n'_{k+1}\rangle_{k+1}\end{aligned}$$

also auch $\langle n_1, \ldots, n_k\rangle_k = \langle n'_1, \ldots, n'_k\rangle_k$ und aus der Induktionshypothese folgt $n_i = n'_i$ für $1 \leq i \leq k$. Zum Schluss ergibt

$$n_{k+1} = s - n_1 \ldots - n_k = s' - n'_1 - \ldots - n'_k = n'_{k+1}$$

die Injektivität von $\langle\ \rangle_{k+1}$.

$\langle\ \rangle_{k+1}$ ist auch surjektiv:

Eine Zählung der $k+1$-Tupel mit Koordinatensumme s ergibt $\binom{s+k}{k}$ Möglichkeiten. Da die Nummern dieser Punkte alle im Intervall $[\langle 0, \ldots, 0, s\rangle_{k+1}, \langle s, 0, \ldots, 0\rangle_{k+1}]$ liegen, das ebenfalls

$$\langle 0, \ldots, 0, s+1\rangle_{k+1} - \langle 0, \ldots, 0, s\rangle_{k+1} = \binom{s+k+1}{k+1} - \binom{s+k}{k+1} = \binom{s+k}{k}$$

Zahlen enthält und $\langle\ \rangle_{k+1}$ injektiv ist, muss es auch surjektiv sein. □

Decodierung

Wir betrachten nun ein Verfahren, um aus dem Code eines Punktes $(n_1, \ldots, n_k)$ wieder die einzelnen Koordinaten n_i zurückzugewinnen. Diese lassen sich aus dem Code auf (primitiv) rekursive Weise berechnen. Wir geben eine Beschreibung dieses Algorithmus in Pseudocode.

Algorithmus 3

```
1   function Decode (Dimension, Code) returns (n_1, n_2, ..., n_Dimension)
2       c := Code
3       for i := Dimension to 1 by -1 do
4           l := 0
5           while binom(l+(i-1)+1, i) <= c do
6               l := l+1
7           endwhile
8           s_i := l
9           c := c - binom(s_i+(i-1), i)
10      endfor

11      for i := Dimension to 2 by -1 do
12          n_i := s_i - s_{i-1}
13      endfor
14      n_1 := s_1
15      return (n_1, n_2, ..., n_Dimension)
16  endfunction Decode.
```

Das wesentliche geschieht in den Zeilen 3 bis 10. Hier werden sukzessive die Zahlen

$$\begin{aligned} s_k &= \max\{l: \langle 0,\ldots,0,l\rangle_k \le \langle n_1,\ldots,n_k\rangle_k\} \\ s_{k-1} &= \max\{l: \langle 0,\ldots,0,l\rangle_{k-1} \le \langle n_1,\ldots,n_k\rangle_k - \langle 0,\ldots,0,s_k\rangle_k\} \\ &\vdots \\ s_i &= \max\{l: \langle 0,\ldots,0,l\rangle_i \le \langle n_1,\ldots,n_k\rangle_k - \textstyle\sum_{j=i+1}^{k}\langle 0,\ldots,0,s_j\rangle_j\} \end{aligned}$$

berechnet. Ihre Bezeichnung soll darauf hinweisen, dass diese Maxima gerade mit den Partialsummen übereinstimmen. Wir können nämlich aus den Gleichungen (2) und (6) sowie der Ungleichung (5) die folgende Beziehung zusammensetzen:

$$\begin{aligned} \binom{s_k+k-1}{k} = \langle 0,\ldots,0,s_k\rangle_k &\le \langle n_1,\ldots,n_k\rangle_k \\ &\le \langle s_k,0,\ldots,0\rangle_k = \langle 0,\ldots,0,s_k+1\rangle_k - 1 = \binom{s_k+k}{k} - 1 . \end{aligned}$$

Man erhält also tatsächlich s_k als Maximum aller Zahlen, für die $\langle 0,\ldots,0,l\rangle_k = \binom{l+k-1}{k} \le \langle n_1,\ldots,n_k\rangle_k$ ist.
Die s_k werden der Reihe nach, beginnend mit k gleich der Dimension des codierten Raumes $\mathbb{N}^k$, behandelt. Gleichung (2) erlaubt uns, wenn wir ein s_i gefunden haben, die Decodierung mit dem neuen Code

$$\langle n_1,\ldots,n_{i-1}\rangle_{i-1} = \langle n_1,\ldots,n_i\rangle_i - \langle 0,\ldots,0,s_i\rangle_i$$

fortzusetzen. Dies geschieht im Algorithmus in Zeile 9.
Im zweiten Teil, von Zeilen 11 bis 14, wird dann noch das Gleichungssystem

$$\begin{aligned} s_1 &= n_1 \\ s_2 &= n_1 + n_2 \\ &\vdots \\ s_k &= n_1 + n_2 + \ldots + n_k \end{aligned}$$

nach den Koordinaten n_i aufgelöst.
Die folgende Tabelle stellt die Laufzeit [1] der obigen Decodierungsfunktion angewandt auf den Code n in der Dimension k dar.

[1] Die Laufzeiten wurden in allen Experimenten mit einem Modula-2-Programm, das fast wörtlich Algorithmus 3 entspricht, auf einem IBM PS/2 gemessen. Die Masseinheit ist eine Hundertstelsekunde.

	$n=100$	1000	10 000	100 000	1 000 000	10 000 000
$k=2$	0.19	0.41	1.20	4.61	14.41	43.96
3	0.19	0.32	0.57	1.28	3.15	6.56
4	0.21	0.30	0.57	1.06	1.78	3.81
5	0.27	0.38	0.57	0.82	1.39	2.03
6	0.32	0.40	0.60	1.06	1.45	2.27
7	0.40	0.49	0.76	1.15	1.45	2.03
8	0.45	0.65	0.76	1.12	1.44	1.86
9	0.54	0.70	1.01	1.14	1.72	2.79
10	0.62	0.65	0.87	1.33	1.97	2.33

Man stellt ein recht unregelmässiges Laufzeitverhalten fest. Dies liegt daran, dass es vor allem von der Grösse $\sum_{i=1}^{n} s_i$ bestimmt ist, da dies die Anzahl Schritte ist, in denen die innerste Schleife des Algorithmus (Zeilen 5 bis 7) durchlaufen wird. Diese Grösse kann aber von einem Code n (bei gleicher Dimension) zum darauf folgenden stark variieren. Sie nimmt z. B. für $n=\langle 0, \ldots, 0, m\rangle_k$ den Wert m an und für $n-1=\langle m-1, 0, \ldots, 0\rangle_k$ den Wert $k(m-1)$. Im allgemeinen ist es so, dass der Aufwand bei gleichbleibender Dimension mit dem Code wächst, da grössere Codes auch grössere Koeffizientensummen haben. Dieses Wachstum ist aber aus dem oben erwähnten Grund nicht regelmässig.

Wir werden uns im folgenden bemühen, die Partialsummen s_i auf direktere Art und Weise zu bestimmen.
Für die Codierung von Paaren $\langle\ \rangle_2$ lassen sich die Koordinaten sehr direkt finden: Wir wissen aus (5), dass $\langle 0, n_1+n_2\rangle_2 \leq \langle n_1, n_2\rangle_2 \leq \langle n_1+n_2, 0\rangle_2$ ist. Also suchen wir eine Umkehrfunktion zu $x \mapsto \dfrac{x(x+1)}{2} = \langle 0, x\rangle_2$, was auf die Gleichung

$$x(x+1) = 2y \tag{8}$$

führt, welche in den natürlichen Zahlen die folgende Lösung hat:

$$x = \left[\frac{-1+\sqrt{1+8y}}{2}\right] \tag{9}$$

([ ] ist hier die Gauss'sche Klammer.) Also gilt

$$s = n_1 + n_2 = \left[\frac{-1+\sqrt{1+8\langle n_1, n_2\rangle_2}}{2}\right] \tag{10}$$

und wir erhalten n_1 und n_2 als

$$n_1 = \langle 0,s\rangle_2 - \langle n_1, n_2\rangle_2$$
$$n_2 = \langle n_1, n_2\rangle_2 - \langle s, 0\rangle_2 .$$

Um dies auf eine beliebige Dimension k auszudehnen, müssten wir nun zu jedem k die Umkehrfunktion von $x \mapsto \binom{x+(k-1)}{k}$ finden. Wir versuchen es mit einer Abschätzung: Mit (8) erhalten wir

$$x^2 < x(x+1) = 2y < (x+1)^2$$

und daraus ebenfalls die Umkehrfunktion (eingeschränkt auf natürliche Zahlen), und es gilt:

$$s = n_1 + n_2 = [\sqrt{2\langle n_1, n_2\rangle_2}]\,. \tag{11}$$

Diese Art der Abschätzung können wir nun verallgemeinern:

Lemma 4. *Es gilt*

$$\left(x + \frac{n-3}{2}\right)^n < x(x+1)\ldots(x+(n-1)) < \left(x + \frac{n-1}{2}\right)^n$$

für $n>1$. Dabei ist rechte Ungleichung für alle natürlichen Zahlen x gültig, die linke unter der Bedingung $x > \frac{1}{8}(n-1)^2$.

Wir führen einen induktiven Beweis durch:

Offensichtlich gelten die Ungleichungen

$$(x-\tfrac{1}{2})^2 < x(x+1) \qquad < (x+\tfrac{1}{2})^2$$
$$x^3 < x(x+1)(x+2) < (x+1)^3$$

für alle $x \in \mathbb{N}$. Dies liefert uns den Induktionsanfang. Wir nehmen nun an, die Aussage des Lemmas sei wahr für n, und erhalten

$$\left(x + \frac{n-1}{2}\right)^n < x(x+1)\ldots(x+(n-1))\,(x+n) < \left(x + \frac{n+1}{2}\right)^n$$

indem wir die Induktionshypothese und $x+1$ statt x verwenden. Weiter gilt dann

$$\left(x + \frac{n-1}{2}\right)^n x(x+n+1) < x(x+1)\ldots(x+n)(x+n+1) < \left(x + \frac{n+1}{2}\right)^n x(x+n+1).$$

Die gewünschte Ungleichung

$$\left(x + \frac{n-1}{2}\right)^{n+2} < x(x+1)\ldots(x+n+1) < \left(x + \frac{n+1}{2}\right)^{n+2}$$

gilt also, wenn nur

$$\left(x+\frac{n-1}{2}\right)^2 < x(x+n+1) < \left(x+\frac{n+1}{2}\right)^2.$$

Die rechte Ungleichung ist immer erfüllt und die linke gilt, sobald $x > \frac{1}{8}(n-1)^2$. □

Also kann man in den vielen Fällen die Partialsummen eines Codes $\langle n_1, \ldots, n_k\rangle_k$ wie folgt abschätzen:

Satz 5. *Für alle Partialsummen* s_i $(1 \leq i \leq k)$ *des Codes eines Punktes* $(n_1, \ldots, n_k)$ *gilt:*

$$s_i - 1 < \sqrt[i]{i!\,\langle n_1, \ldots, n_i\rangle_i} - \frac{i-1}{2} \tag{12}$$

$$\sqrt[i]{i!\,\langle n_1, \ldots, n_i\rangle_i} - \frac{i-1}{2} < s_i + 1 \quad \textit{falls} \quad s_i > \tfrac{1}{8}(i-1)^2. \tag{13}$$

Zum Beweis bemerken wir, dass

$$\langle 0, \ldots, s_i\rangle_i = \frac{1}{i!}\, s_i(s_i+1)\ldots(s_i+i-1)$$

ist. Nach Ungleichung (5) und dem obigen Lemma gelten also die Abschätzungen

$$i!\,\langle n_1, \ldots, n_i\rangle_i < i!\,\langle 0, \ldots, s_i+1\rangle_i = s_i(s_i+1)\ldots(s_i+i-1) < \left(s_i+1+\frac{i-1}{2}\right)^i$$

und unter der Nebenbedingung $s_i > \frac{1}{8}(i-1)^2$ auch

$$\left(s_i-1+\frac{i-1}{2}\right)^i < i!\,\langle 0, \ldots, 0, s_i\rangle_i \leq i!\,\langle n_1, \ldots, n_i\rangle_i.$$

Durch Ziehen der i-ten Wurzel folgt der Satz. □

Nachdem die s_i bekannt sind, können aus ihnen wieder die Koordinaten n_i berechnet werden. Wir realisieren dieses Abschätzungsverfahren in einer Variante des Algorithmus 3:

Algorithmus 6

```
1   function Decode 2 (Dimension, Code) returns (n_1, n_2, ..., n_Dimension)
2       c := Code
```

3 **for** $i := Dimension$ **to** 1 **by** -1 **do**

4 $l := \left[\sqrt[i]{i! \cdot c} - \frac{i+1}{2}\right]$

5 **while** $\binom{l+(i-1)+1}{i} \leq c$ **do**

6 $l := l+1$

7 **endwhile**

8 $s_i := l$

9 $c := c - \binom{s_i+(i-1)}{i}$

10 **endfor**

11 **for** $i := Dimension$ **to** 2 **by** -1 **do**

12 $n_i := s_i - s_{i-1}$

13 **endfor**

14 $n_1 := s_1$

15 **return** $(n_1, n_2, \ldots, n_{\text{Dimension}})$

16 **endfunction** *Decode 2.*

Dabei wurde l in Zeile 3 mit unserem Schätzwert initialisiert. Wegen Satz 5 wissen wir, dass die kritische Schleife (Zeilen 5 bis 7) nur in Ausnahmefällen mehr als zweimal durchlaufen wird, nämlich wenn der Wert von s_i kleiner ist als $\frac{1}{8}(i-1)^2$.
Hier die Resultate eines Experimentes für den verbesserten Algorithmus 6:

	$n=100$	1000	10 000	100 000	1 000 000	10 000 000
$k=2$	0.15	0.10	0.13	0.16	0.15	0.16
3	0.21	0.26	0.27	0.21	0.26	0.27
4	0.37	0.38	0.37	0.37	0.38	0.37
5	0.54	0.52	0.54	0.52	0.51	0.51
6	0.68	0.70	0.71	0.71	0.73	0.73
7	0.90	0.90	1.01	0.93	0.92	0.93
8	1.14	1.14	1.15	1.17	1.18	1.20
9	1.36	1.42	1.42	1.44	1.42	1.41
10	1.67	1.67	1.72	1.72	1.72	1.75

Da der Berechnungsaufwand einer einzelnen Partialsumme s_i, dank der meist gut zutreffenden Schätzung – falls die Nebenbedingung des Satzes erfüllt ist, kommen nur zwei Zahlen für s_i in Frage – praktisch konstant ist, hängt er praktisch nicht mehr vom Code ab. Das Wachstum des Aufwandes in der Dimension war in diesem Experiment nur wenig stärker als linear.