数理解析研究所講究録
第12巻 1966年 1-9

数値解析 Symposium報告　　1965．Nov．18．

整数の分割の生成

立教大学
理学部　一松　信

この問題は誤差解析の問題でなく，むしろアルゴリズムが主題であるが，最近この種の問題を手がけているのでここで報告する。

1. 問題の設定

正の整数Nを与えたとき，これを

$$N = 1\cdot k_1 + 2\cdot k_2 + \cdots + N\cdot k_N,\quad (k_1, \cdots, k_N \geqq 0) \tag{1}$$

というように分割し，このような分割をすべて作りだしたい。

この問題の由来は，ある物理学の問題で，合成函数のテイラー展開の係数を順次求めるために，(1)のような分割をして

$$N!/(k_1!\, k_2! \cdots k_N!) \tag{2}$$

を求める必要があった。当面の問題では手計算でも十分であったが，Nが大きくなると，すべての場合をつくすのが難しい，というので相談をうけた。ちょうど少し前から，他の必要で，順列・組合せなどを計算機で生成する問題を手がけていたので，計算機で実行することを考えてみたわけである。

上記の分割の総数 A_N は，Nの簡単な式では表わされないが，はじめの方はつぎのとおりである。

N	1	2	3	4	5	6	7	8	9	10	11	12	13
A_N	1	2	3	5	7	11	15	22	30	42	56	77	101

この数は $N!$ に比べてはるかに小さい。Nが大きいとき漸近的に次の関係が知られている（〔1〕）。

$$A_N \sim \frac{1}{4\sqrt{3N}} \exp\left\{\pi\left(\frac{2N}{3}\right)^{1/2}\right\} \qquad (3)$$

(1)のような分割ができれば，(2)の計算は問題がないから，ここではもっぱら(1)のような分割を順次作りだすことを考える。

2. 分割を作る方法(1)

問題を変形して，正の正数 $a_1, a_2, \cdots, a_k$ により

$$N = a_1 + a_2 + \cdots + a_k \qquad (k \leqq N) \qquad (4)$$

と分割することを考える。ここで $a_1 \leqq a_2 \leqq \cdots, \leqq a_k$ として一般性を失わない。これを個数kの小さい方から順に，kが同じものは $a_1\ a_2 \cdots a_k$ を辞書式に少さい方から順に作りだすことを考える。(4)のような分割ができれば，これを集計して(1)の形での $k_2, \cdots, k_N$ に換算するのは原理的にはなんでもない。

この生成は文献〔1〕にある方法をもとにして以下のようにやってみた。

第　一　法

```
begin integer N,I,K,M,L,S,W,LIM;
integer array A,B[1:13];
INPUT: READINTEGER(N); CRLF;
in N<=0 then begin HALT; go to INPUT end;
for I:=1 step 1 until N do A[I]:=0;
PRINTSTRING("n=  '); PRINTINTEGER(N); CRLF;
for K:=1 step 1 until N do
begin L:=K-1; W:=K+K-N;
LIM:=if W>0 then W else 0;
for I:=1 step 1 until L do A[I]:=1;
LOOP: S:=N;
for I:=1 step 1 until L do S:=S-A[I];
A[K]:=S;
INSATU: CRLF; for I:1 step 1 until K do
PRINTINTEGER(A[I]);
CRLF; for I:=1 step 1 until N do B[I]:=0;
for I:=1 step 1 until K do
```

```
begin W:=A[I]; B[W]:=B[W]+1 end;
for I:=1 step 1 until N do PRINTINTEGER(B[I]); comment
  end of INSATU;
M:=K;
CHECK: M:=M-1; if M>LIM then
begin W:=A[M]+1; if S>W then begin
for I:=M step 1 until L do A[I]:=W;
go to LOOP end;
go to CHECK
end
end;
CRLF; go to INPUT
end;
```

このプログラムのうち中央の縦線を引いた部分は印刷のプログラムであるが，ここに他のプログラムをはさんで，分割の決果 A[1],…, A[N] を利内することもできる。

方法はつぎのとおりである。

1° (4)のk（プログラム中ではK）を定める。

2° まず $a_1 = \cdots = a_{k-1} = 1$ とおく。

3° $a_k = N - \sum_{i=1}^{k-1} a_i$ とする。

4° a_k をもとにし，a_{k-1}，a_{k-2}，…とみて，はじめて $a_m < a_k - 1$ となる所をみつける。もし

$$m \leqq 0 \quad \text{あるいは} \quad m \leqq 2k - N \tag{5}$$

となっても，このようなa_mがみつからなければ，そのkに対する分割は完了であるから，kを1つふやす。k=Nになっていれば完了である。

5° a_m を1ふやし，a_{m+1}，…，a_{k-1} をそれと同じ値におきかえて，3°にもどる。

(5)の所はつねにm=1になるまでためしてもよい。〔1〕にあるもとのプログラムはそうなっている。しかしkがNに近いときには，$m \leqq 2k - N$ になったらやめてよい。なぜなら，$a_1 = \cdots = a_m = 1$，$a_{m+1} = \cdots = a_k = 2$ ならば，もはやk個の分割は辞書式順序では終りであり，このとき

$$m + (k-m)2 = N \qquad \text{すなわち} \quad m = 2k - N$$

だからである。この限界として，はじめの方で

$$LIM = \max(0, 2k - N)$$

とおいてあり，$m > LIM$ の範囲で反復している。この操作を加えたことにより，かなり能率化されたはずである。

3. 分割を作る方法(2)

(1)の $k_1, \cdots, k_N$ を作るには，(4)のような分割を作ってから集計する方法では，じっさいにやってみると意外に時間がかかるので，直接法を考えた。

直接法では，$k_1, \cdots, k_N$ の値を配列 A[1], …, A[N] として用意し，A[K] ≠ 0 , A[K+1] = … = A[N] = 0 であるようなKをNから順次1まで下げてゆく。各Kに対しては，A[K], A[K−1], …, A[1]と並べた逆順の辞書式順序で作りだした。その手法は次のとおりである。

第　　2　　法

```
begin integer N,I,K,D,W,Q,J;
integer array A[1:13];
INPUT: READINTEGER(N); if N<=0 then HALT;
for K:=N step -1 until 1 do begin
for I:=1 step 1 until K-1 do A[I]:=0;
A[K]:=1; A[1]:=A[1]+N-k;
OUTPUT: CRLF; for I:=1 step 1 until K do PRINTINTEGER
  (A[1]);
D:=N-K*A[K];
if D=0 then go to END;
if A[D]=1 & D<K then go to END;
W:=A[1];
if W>1 then begin A[1]:=W-2;
A[2]:=A[2]+1; go to OUTPUT end;
I:=1;
LOOP 1: I:=I+1; Q:=A[I];
if Q=0 then go to LOOP1;
```

```
J:=I;
if W=1 or Q>1 then I:=W else begin
LOOP2: J:=J+1 ; Q:=A[J]; if Q=0 then go to LOOP2 ;
   A[I]:=0
end:
A[J+1]:=A[J+1]+1 ;  A[J]:=0 ;
A[1]:=(Q-1) *J+I-1 ;  go to OUTPUT;
END: A[K]:=0  end;
CRLF; CRLF; go to INPUT
end;
```

1° はじめ A[1]=…=A[K−1]=0 , A[K]=1 とおく。

2° A[1] をN−Kだけふやす。(K≧2のときには，A[1]=N−K としてもよいが，K=1のときをも考慮して，このようにした)

3° D=N−Kx[K] が0であるか，または，D<Kで，かつA[D]=1 のとき，つまり

0 0 …… 0 1 0 …… 0 S ,　　S × K + D = N

1番　　D番　　K番

となったら，Kに関するものは終りであるから，A[K] を払って0とし，Kを1つへらして1°にかえる。(当初うっかりD<Kを忘れたため，Nが偶数で K = N/2 のとき失敗した。)

4° A[1]≧2 なら，A[1] を2へらしA[2]を1ふやす。

5° そうでなければ，はじめてA[I] ≠ 0 であるIをさがす。

(a) もしA[1]=1なら，A[I+1] を1ふやし，A[I]を0とし，のこりをA[1]にくりこむ。したがって

$$A[I] = I \times A[I] - (I+1) + 1 = (A[I]-1) \times I + A_1 - 1 \qquad (6)$$

とすればよい。A_1 は以前のA[1]の値である。

(b) A[1] = 0でも，A[I]≧2なら，(a)と同じ操作でよい。このときには，

$$A[I] = I \times A[I] - (I+1) = (A[I]-1) \times I + A_1 - 1$$

だから，(6)の最右辺と同じ式でよい。

(c) A[1] = 0 , A[I]=1 なら，さらにI+1からはじめてA[J] ≧ 1 である所を

さがす。そして，A〔J+1〕を1ふやし，A〔I〕＝A〔J〕＝0とし，のこりをA〔1〕にくりこむ。けっきよく

$$A〔I〕=J\times A〔J〕-(J+1)+I=(A〔J〕-1)\times J+I-1 \qquad (7)$$

とすればよい。

以上のいずれかの操作の後3°にかえる。

(6)と(7)とは形式上同じ形の式なので，(6)の場合，$I\rightarrow J$，$A_1\rightarrow I$ とおきかえることにより，同じ式(7)ですませられる。別掲プログラムの下から 4行目の式はこのようにして一つにまとめたものである。（ここでQはA〔J〕を意味する。）

4. 計 算 結 果

以上のプログラムはともにHIPAC101用 JUSE ALGOLで書いてあり，そのまま計算機にかけて計算できた。このプログラムでarray の大きさを18にしたのは，1行に印刷できる数の制限のためである。文法上は，任意の大きさのarrayでこのプログラムが使用できる（別掲結果A）。

しかし，立教大学数学教室のHIPAC101ではあまりに時間がかかりすぎた。例をあげると，N＝9の場合，第1法で印刷をこめて約20分，第2法で約10分を要した。これは計算機そのものおよび印刷がおそいだけでなく，ALGOLをそのままコンパイルしたため，とくに添字付変数の処理にいちぢるしく時間がかかるせいである。

そこで東京大学理学部数学教室のTOSBAC3300により，上記のプログラムをアセンブラ言語（筆者の作ったMINITAP-5）に書き下して実行させてみた。この方はarrayの大きさNは一応無制限（じっさいには50まで）とした。第1法でははじめ集計をはぶいて分割そのものだけを印刷させた（別掲結果B）。このほうも印刷が電動タイプであるために，制約をうけたが，ほとんど大半は印刷時間ばかりで，たとえばN＝10のとき2分半，N＝15のとき10分程度でできた。ただし第1法で集計をも含めると，この倍くらいの時間を要する。ラインプリンタによれば，N＝15でも30秒以下ですむはずである。なお一つの分割から次の分割を作りだすまでの所要時間そのものは，TOSBAC3300の場合，プログラムのステップ数から50 ms 以下と推定される。

A. 第2法による計算結果 (2)

1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	1	
2	0	0	0	0	0	1		
0	1	0	0	0	0	1		
3	0	0	0	0	1			
1	1	0	0	0	1			
0	0	1	0	0	1			
4	0	0	0	1				
2	1	0	0	1				
0	2	0	0	1				
1	0	1	0	1				
0	0	0	1	1				
5	0	0	1					
3	1	0	1					
1	2	0	1					
2	0	1	1					
0	1	1	1					
1	0	0	2					
6	0	1						
4	1	1						
2	2	1						
0	3	1						
3	0	2						
1	1	2						
0	0	3						
7	1							
5	2							
3	3							
1	4							
9								

B． 第１法による計算結果

```
1

2
1  1

3
1  2
1  1  1

4
1  3
2  2
1  1  2
1  1  1  1

5
1  4
2  3
1  1  3
1  2  2
1  1  1  2
1  1  1  1  1

6
1  5
2  4
3  3
1  1  4
1  2  3
2  2  2
1  1  1  3
1  1  2  2
1  1  1  1  2
1  1  1  1  1  1

7
1  6
2  5
3  4
1  1  5
1  2  4
1  3  3
2  2  3
1  1  1  4
1  1  2  3
1  2  2  2
1  1  1  1  3
1  1  1  2  2
1  1  1  1  1  2
1  1  1  1  1  1  1
```

C． 第２法による計算結果

```
1
1

1  2
0  1
2

1  2  3
0  0  1
1  1
3

1  2  3  4
0  0  0  1
1  0  1
2  1
0  2
4

1  2  3  4  5
0  0  0  0  1
1  0  0  1
2  0  1
0  1  1
3  1
1  2
5

1  2  3  4  5  6
0  0  0  0  0  1
1  0  0  0  1
2  0  0  1
0  1  0  1
3  0  1
1  1  1
0  0  2
4  1
2  2
0  3
6
```

5. むすび

この問題は単なる演習問題にすぎないが，仕事の余暇に，だいたい1週1回計算機を使用して，完成するまで約1ケ月を費した。このような問題を手がけて次のことを感じた。これは数値解析の問題でなく，計算機の設置や教育の問題であるが，参考までにしるしておく。

1° この種の組合せ問題は，アルゴリズム解析の実例および初歩のプログラミングの演習問題として好適と思われる。それは問題そのものはいたって単純であるし，参考書は少ないし，debug が案外に大変であるし，さらに工夫次第でいろいろ改良ができるからである。

2° コンパイラを作るにあたっては，とくに添え字付変数の処理をできるだけ能率化しないと実用になりにくい（とくに小型機の場合）。

3° 小形機に対しても，高速度の印刷機があると能率が格段にちがうことが多い。

<参> 〔1〕 D.H. Lehmer, The machine tools of combinations; Beckenbach編, Applied combinational mathematics, John Weley and Sons, 1964; §1.8, pp.25–26.

〔2〕 川端 親雄・一松 信， 整数の分割のプログラム，情報処理，プログラムのページに発表予定。