PARTITIONS GENERATORS

Gideon Ehrlich

TR 74-199

February 1974

Department of Computer Science
Cornell University
Ithaca, New York 14850

PARTITIONS GENERATORS

Gideon Ehrlich
Cornell University

Abstract:

Several algorithms for generating partitions of positive numbers are given. First, an algorithm for generating all the partitions is given, then algorithms for generating of all partitions in which all terms are smaller than a given M greater than a given n, between n and m, and partitions of n to a given k. Each of the algorithms produces the whole sequence in a time proportional to the length of that sequence; moreover each new partition is produced in no more than a constant number of steps independent on the parameters of the algorithm.

PARTITIONS GENERATORS

Gideon Ehrlich
Cornell University

Introduction:

A partition of n is a non increasing sequence $P_1,P_2,\ldots,P_v$ such that $P_1+P_2+\ldots P_v = n$. $n,P_1,P_2,\ldots,P_n$, as any other number variables to appear in this paper, can get positive integer values only. A convenient form to represent a partition is the "exponential form".

$P = T_1^{M_1}T_2^{M_2}\ldots T_L^{M_L}$ means that P is a partition containing M_1 times the term T_1, M_2 times T_2, etc. It is assumed that the T_i's form decreasing sequences. The partition P will be represented by a triple (T,M,L) where T and M are vectors and L integer variable such that $T_1,T_2,\ldots,T_L$ form a decreasing sequenc and $\sum_{i=1}^{L} T_iM_i = n$. T as well as M, has at least L components. There is no importance to the values of T_i,M_i for $i > L$. For simplification we always assume that $T_0 = n+1$. The maxim.l value L can get is $\left[\sqrt{2n\frac{1}{4}} - \frac{1}{2}\right]$. This extreme case will occur when $P = L^1(L-1)^1\ldots2^1 1^1$.

$|P|$: If P is a partition of n then $|P| = n$.

Concatenation:

Let $P = a_1^{b_1} a_2^{b_2} \ldots a_L^{b_L}$ and $q = c_1^{d_1} c_2^{d_2} \ldots c_s^{d_s}$ if $a_L > c$, then $p*q = a_1^{b_1} a_2^{b_2} \ldots a_L^{b_L} c_1^{d_1} c_2^{d_2} \ldots c_s^{d_s}$ if $a_L = c_1$ then $p*q = a_1^{b_1} a_2^{b_2} \ldots a_L^{b_L+d_1} c_2^{d_2} \ldots c_s^{d_s}$ ($p*q$ is undefined for $a_L < c_1$).

We will use the concatenation operation only for q having $s < 4$, thus $p*q$ can be done in bound number of operations.

In some cases it may happen that the first "exponent" of q is zero, and in such a case the first factor has to be ignored (e.g., $6^5 3^1 * 2^0 1^8 = 6^5 3^1 1^8$, $6^5 3^1 * 2^0 = 6^5 3^1$).

The representation of a partition as described above will also be called the decreasing exponential form, to distinguish it from the increasing form which can be similarly defined. The increasing exponential form will be used in section 5 only. All lists of partitions will be ordered lexicographically namely if $P = (T,M,L)$, $P' = (T',M',L')$ then $P < P'$ (P precede P') if for the first i for which $T_i \neq T_i'$ or $M_i \neq M_2'$ it happened that $T_i < T_i'$ and if $T_i = T_i'$ then $M_i < M_i'$. (Obviously if $T_i = T_i'$ and $M_i = M_i'$ for $i = 1,\ldots,L$ then $L = L'$). first(n) and next(P) are the first partition of M and the immediate successor of P in the lexicographic ordered list.

As in [1] a partitions generator is called loopless iff there exists k such that for every n:

a) The initialization needs less than kn steps.

b) The halt description is received by less than k operations.

c) Every transition from P to next(P) requires less than

The partitions are generated for some aim. We are interested in some information related to each of the partitions. If in a loopless algorithm a new partition is only slightly different from the old one, there is a possibility that if in the output step we get information only on the change done in the partition, we will be able to update the information related to the old P in order to get the information related to next(P).

A small change in an exponential form may represent a change of many terms for others, but since all equal terms play the same role it is reasonable to assume that the updating will be dependent on the number of different terms changed. E.g., if the information we need is the mean and the standard deviation, we can update by a bound number of operations the information about the sum of squares, $\sum_i M_i T_i^2$, and the number of terms $\sum_i M_i$ from which the desired information can easily be computed. For computing P(n), the number of partitions of n, see [3], [4], for generating ordered partitions see [2].

1. Algorithm for generating all partitions of n.

The following lemma can be proven easily.

Lemma:

1. $\text{first}(n) = 1^n$.
2. $\text{next}(a^k) = (a+1)^1 *1^{a(k-1)-1}$, for $k > 1$.
3. $\text{next}(a^1)$ is undefined.
4. $\text{next}(b^k a^1) = (b+1)^1 *1^{b(k-1)+a-1}$.

5. Next $(T_1^{M_1}\ldots T_L^{M_L}) = T_1^{M_1}\ldots T_{i-1}^{M_{i-1}} * \text{next}(T_i^{M_i}\ldots T_L^{M_L})$ where i is any index for which $\text{next}(T_i^{n_i}\ldots T_L^{M_L})$ is defined. Using the above lemma we can get the following algorithms for producing the sequence of all partitions of n.

Algorithm 1.

1. Start with $P = 1^n \cdot [T_1 = 1, M_1 = n, L = 1]$.
2. OUTPUT.
3. If $T_1 = n$ then HALT.
4. If $M_L = 1$ then set $P \quad T_1^{M_1}\ldots T_{L-2}^{M_{L-2}} * (T_{L-1}+1)' * 1^{T_{L-1}(M_{L-1}-1)} + T_L$ and go to step 2.
5. Set $P \leftarrow T_1^{n_1}\ldots T_{L-1}^{M_{L-1}} * (T_L+1)' * 1^{T_L(M_L-1)-1}$ and goto step 2.

Producing a new partition needs two steps, each of which needs a bound number of operations only.

2. Partitions with upper bound

From the definition of the decrease exponential representation it follows that if, for some partition P, $T_1 \leq M$ then $T_i \leq M$ for all $0 < i \leq L$. The list of partitions generated by the algorithm No. 1 has lexicographic order thus once we get $T_1 > M$ for some P the same will be true for all the consequence partitions.

It follows that by changing step 2 in the algorithm No. 1 for:

2. a) If $T_1 > M$ then HALT
   b) OUTPUT

We will get an algorithm for the sequence of all unordered partitions of a positive integer n to unrestricted numbers of terms each of which is not greater than M.

3. Partition with lower bound: a loopless algorithm for the sequence of all unordered partitions of a positive integer n to unrestricted number of terms each of which is not smaller than $m \geq 1$. The whole sequence will be lexicographically ordered.

Lemma 1: first(n) is of the form $(m+k+1)^a(m+k)^b$. $a \geq 0$, $b \geq 1$

Proof: Assume P = first(n) = (T,M,L) is not of the above form thus $T_1 - T_L \geq 2$.

Let P' be a partition containing the same terms as P where one appearance of T_1 and one of T_L are changed for one appearance of $\left\lceil \frac{T_1+T_L}{2} \right\rceil$ and one of $\left\lfloor \frac{T_1+T_L}{2} \right\rfloor$, (the ceiling and floor of $\frac{T_1+T_2}{2}$ respectively), obviously P' precedes P. Since all the terms of P' are not smaller than M we get $P \neq \text{first}(n)$.

Lemma 2: Let $P = \text{first}(n) = (m+k+1)^a(m+k)^b$ then $a < m+k$.

Proof: Otherwise $P' = (m+k+1)^{a-(m+k)}(m+k)^{b+(m+k+1)}$ precedes P.

Lemma 3: $(m+k+1)^a(m+k)^b = (m+k+1)^{a'}(m+k)^{b'}$ and $a < m+k$, and $a' < m+k$ implies that $a = a'$ and $b = b'$.

Proof: $(m+k+1)^{a-a'} = (m+k)^{b'-b}$ thus $a-a' = (m+k)[(b'-b)-(a-a')]$ the square bracket must be zero otherwise the difference between a and a' will be at least m+k while both are smaller than m+k hence $a = a'$ and thus $b = b'$.

Lemma 4: Let $n = dm + r$; $r < d$ then

$$first(n) = (m+k+1)^a (m+k)^b$$

where $k = \left\lfloor \frac{r}{d} \right\rfloor$, $a = r - kd$, $b = d-a$

Proof: it is easily seen that k,a,b are non-negative integers and that $(m+k+1)^a(m+k)^b$ is a partition of n. $a \leq r$ $m \leq m+k$ implies a $m+k$ as desired by lemma 2.

Let $P' = (m+k'+1)^{a'}(m+k')^{b'}$; $a' < m+k'$ be a partition of n. If $k'=k$ then $P' = P$ (lemma 3). If $k' < k$ then $(m+k+1)^a(m+k)^b = (m+k+1)^{a'}(m+k)^{b'}$ implies $a'+b' > a+b$, hence $a'+b' \geq d+1$. This implies

$$(m+k+1)^{a'}(m+k')^{b'} = m(a'+b')+(a'+b')k+a' \geq (d+1)m + [(a'+b')k+a']$$
$$> dm + r = n$$

Hence P' is not a partition of n.

Lemma 5: $next(m^u) = (m+k+1)^a(m+k)^b$

where $k = \left\lfloor \frac{m}{u-1} \right\rfloor$, $a = m-k(u-1)$, $b = u-1-a$

Proof: Lemmas similiar to 1,2,3,4 can be proved for the second partition of m^u where only small changes are needed. The proof of the quivalent lemma to lemma 4 must be started from the equality:

$$n = (u-1)m+m$$

Lemma 6: If the largest term of $next(m^u)$ is larger than T_{L-1} then the terms of $next(T^1_{L-1}m^u)$ are not smaller than T_{L-1}.

Proof: Let $P = T^1_{L-1}m^u$

Assume $P' = next(P) = (T'_1)^{M_i}(T'_2)^{M_i}\ldots(T_i)^{M_i}$ when $T'_{L'} < T_{L-1}$

if $T'_1 > T_{L-1}$ then $P'' = (T'_1)^{M'_1-1} * (T'_1-1) * (T'_2)^{M'_2} \ldots (T_i+1)' (T_i)^{M'_i-1}$

is a partition of $|T^1_{L-1}m^u|$ appears in the lexicographic list

after P, because $T_1'-1 \geq T_L$ and $T_i-1 > m$, and before P', and thus $P' \neq next(P)$. If $T_1' = T_{L-1}$ then

$$next(P) = P' = T_{L-1}*[(T_{L-1})^{M_i-1}(T_2')^{M_2'}\ldots T_i^{M_i}] = T_{L-1}*\bar{P}$$

$\bar{P}$ is a partition of $|m^u|$ and it is not identical with m^u (since $next(P) \neq P$) and its largest term is not larger than T_{L-1}. This contradicts the assumption on $next(m^u)$. If $T_1' < T_{L-1}$ then $P' = next(P)$ precedes P.

Lemma 7: If the largest term of $next(m^u)$ is larger than T_{L-1} then

$$next(T_{L-1}^1 m^u) = (m+k+1)^a(m+k)^b$$

where $k = \frac{T_{L-1}}{u}$, $a = T_{L-1} - ku$, $b = u - a$

(Note: The same is true also for $next(T_{L-1}^1 m^1) = (T_{L-1}+m)^1$).

Proof: Similiar to the proofs of lemmas 4 and 5.

Example: $m = 10$, $next(10^3) = 14^1 13^2$, $next(12^1 10^3) = 15^0 14^3 = 14^3$

Lemma 8: Let $P = T_1^{M_1}\ldots T_i^{M_i}\ldots T_L^{M_L}$

If a) $next(T_{i+1}^{M_{i+1}}\ldots T_L^{M_L})$ starts with a term larger than T_i, or $i=L$

b) $T_i^{M_i}\ldots T_L^{M_L} = (T_i+1)^1 * q$ where q is a partition all its terms are in the range $[m,T_i]$.

then $next(P) = P' = T_1^{M_1}\ldots T_{i-1}^{M_{i-1}} * (T_i+1)^1 * first(|q|)$

Proof: b implies that all the terms of $first(|q|)$ are smaller than (T_i+1).

Obviously $P' > P$. Let P" be a partition of n, such that $P' > P'' > P$ then the first i-1 "factors" of P" must be $T_1^{M_1} T_2^{M_2}\ldots T_{i-1}^{M_{i-1}}$. The next term can not be t_i+k since this will

prevent $P'' < P'$, it cannot be smaller than t_i since this implies $P'' < P'$; thus the next factor is t_i^x where $x > m_i$, as in the proof of lemma 6 this will contradict a.

It is easily seen that

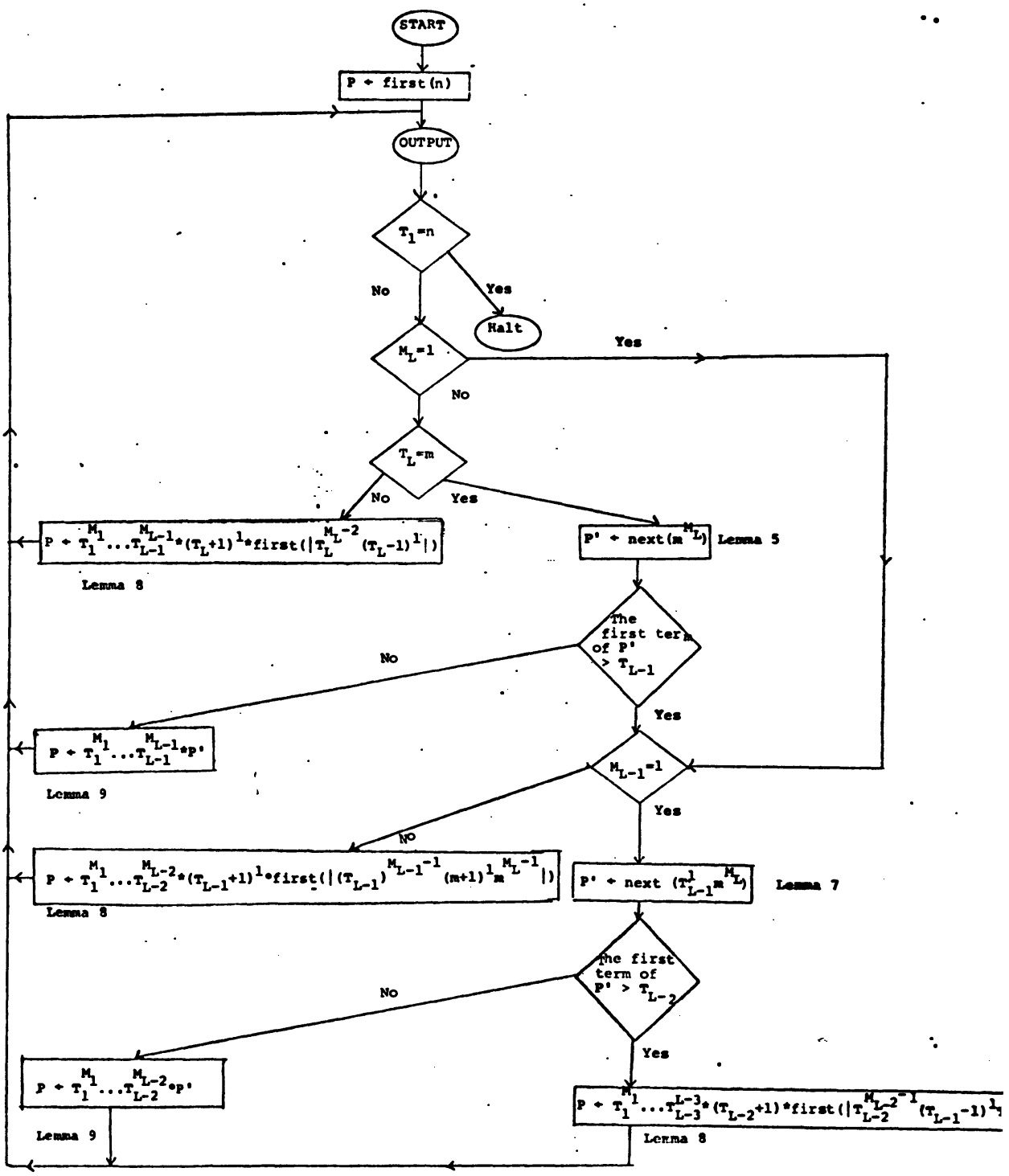
Lemma 9: $\text{next}(T_1^{M_1}\ldots T_L^{M_2}) = (T_1^{M_1}\ldots T_{i-1}^{M_{i-1}})*\text{next}(T_i^{M_i}\ldots T_i^{M_L})$

where i is the maximal index for which $\text{next}(T_i^{M_i}\ldots T_L^{M_L})$ is defined and has first term not larger than T_{i-1}. (Every $j \leq i$ is good as well).

The following flow chart represents an algorithm for generating the lexicographic list of all unordered partitions of n to positive terms not smaller than m. To see that this is the case observe that the list generated by the algorithm started with the first partition. If a current P in the output block is not the last one than the flowcharts will lead us through one and only one block in which next(P) is generated and then immediately to the output block again. All the steps of the algorithm are well defined , hence it will not stop unless $T_1 = n$ namely when P is the last partition.

Each simple directed path in the flowchart contains the output block. Each block needs only a bound number of operations hence the total number of operations needed for generating a new partition is uniformly bounded, (not influenced by m or n), and the algorithm is thus a loopless one.

As to the "output", in that step we can output not the new partition but information about the changes that were done in the old partition. In practical cases the "output" step is the place for using the new partition for the purpose for which we generate it.


START
P ← first(n)
OUTPUT
T_1=n
No
Yes
Halt
M_L=1
Yes
No
T_L=m
No
Yes
$P \leftarrow T_1^{M_1} \ldots T_{L-1}^{M_{L-1}} * (T_L+1)^1 * first(|T_L^{M_L-2} (T_L-1)^1|)$
Lemma 8
P' ← next(m^{M_L}) Lemma 5
The first term of P' > T_{L-1}
No
Yes
$P \leftarrow T_1^{M_1} \ldots T_{L-1}^{M_{L-1}} * P'$
Lemma 9
M_{L-1}=1
Yes
No
$P \leftarrow T_1^{M_1} \ldots T_{L-2}^{M_{L-2}} * (T_{L-1}+1)^1 * first(|(T_{L-1})^{M_{L-1}-1} (m+1)^1 m^{M_L-1}|)$
Lemma 8
P' ← next ($T_{L-1}^1 m^{M_L}$) Lemma 7
The first term of P' > T_{L-2}
No
Yes
$P \leftarrow T_1^{M_1} \ldots T_{L-2}^{M_{L-2}} * P'$
Lemma 9
$P \leftarrow T_1^{M_1} \ldots T_{L-3}^{L-3} * (T_{L-2}+1) * first(|T_{L-2}^{M_{L-2}-1} (T_{L-1}-1)^1$
Lemma 8

4. Partitions with upper and lower bounds.

Algorithm 4: The same as algorithm 3 (the flowchart) but the output block is changed for:

a) if $T_1 > M$ then HALT.

b) OUTPUT.

5. Simple algorithm for partitions with lower bound.

The algorithm to be described here is a simple alternative algorithm for algorithm no. 3.

In this section we use the increasing exponential representation rather than the decreasing one.

$P = (T_1 M_1 L)$ preceding $P' = (T', M', L)$ in the first position they do not agree, either $T_i < T_i'$ or $T_i > T_i'$ and $M_i > M_i'$. Let $n \geq m$, define first(n,m) to be the first partition of n to terms not less than m.

It is easily seen that:

Lemma: Let $n = dm + r$; $r < m$ then

$$\text{first}(m,n) = m^{d-1} * (m+r)'$$

[For actual computation: $d = \lfloor \frac{n}{m} \rfloor$, $r = n - dm$; and the values of L and $T_1 \ldots T_2$, $M_1 \ldots M_2$ has to be computed]

It can be seen now that the following algorithm produces the desired sequence.

Algorithm 5:

1) Start with first(n,m).
2) Output.
3) If T_1=n then HALT.
4) If M_L>1 then $P \leftarrow T_1^{M_1} T_2^{M_2} \ldots * T_L^{M_L-2} * (2T_L)^1$
   goto step 2
5) $P \leftarrow T_1^{M_1} T_{L-2}^{M_{L-2}} \ldots * T_{L-1}^{(M_{L-1})-1} * \text{first}(T_{L-1}+T_L, T_{L-1}+1)$
   goto step 2.

Example: n = 12, m = 3

3^4, $3^2 6$, $3^1 4^1 5^1$, $3^1 0^1$, 4^3, $4^1 8^1$, $5^1 7^1$, 6^2, 12^1

6. An algorithm for partitions of n to k terms.

Given n and k $n \geq k > 0$. The algorithm described in this section will produce the lexicographic order list of all decreasing exponential descriptions of the partitions of n to k terms.

Definition: first(n,k) is the first partition of n to k terms.

Lemma: $\text{first}(n,k) = d+1)^r d^{k-r}$

where $d = \left\lfloor \frac{n}{k} \right\rfloor$, $r = n-dk$

Proof: Let P' be a partition having k terms such that P' < P. By writing P and P' explicitly we will get $P = (P_1 P_2 P_3 \ldots P_k)$, $P' = (P'_1, P'_2, \ldots, P'_k)$ where the two sequences are non-increasing

sequences $P_1 = P_2 \ldots = P_l = d+1$, $P = \ldots = P_k = d$. It follows from the definition of the lexicographic order that up to some place the terms of both sequences are the same and from that place on $P_i > P'_i$ which contradicts the assumption that P' is a partition of n.

The following lemma can be proved in a similar way.

Lemma: The last partition of n to k terms is $(n-k+1)1^{k-1}$.

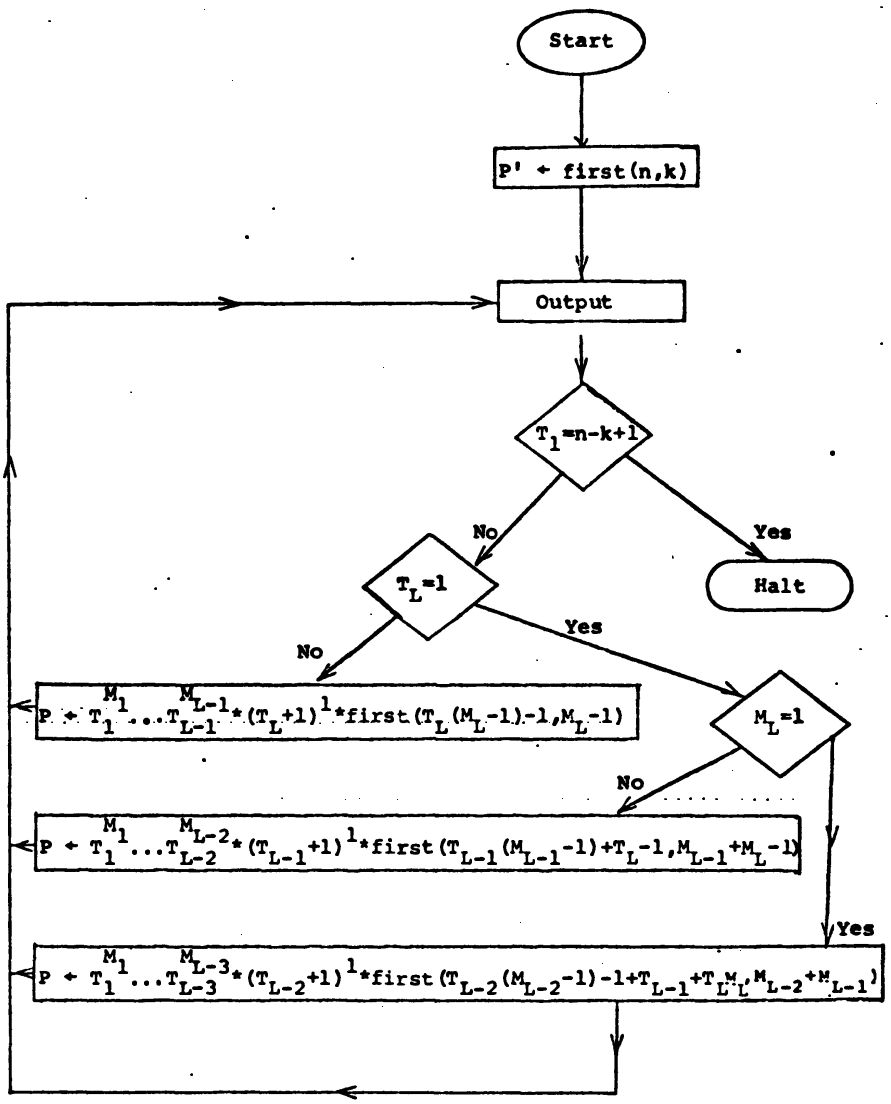
It follows that the last partition can be specified by: $T_1 = n-k+1$. Using the above lemma and ideas already used in previous algorithms we will get the following flowchart for the algorithm generates the desired sequence (to be called Algorithm No. 6.)

Note 1: If one wants all k terms to be smaller than some M then he can add the same condition as in algorithm No. 2.

Note 2: Given n,k_1,k_2 such that $n \geq k_2 \geq k_1 > 0$, by executing Algorithm no. 4 first for $k = k_1, k_1+1, \ldots, k_2$ we will get a sequence of all partitions of n to k terms when $k_1 \leq k \leq k_2$. The initialization of Algorithm No. 6 needs bound number of operations and thus our algorithm is still loopless.

Algorithm 6:

Partition of n to exactly k terms



References

[1] Ehrlich, G., "Loopless Algorithms for Generating Permutations Combinations and other Combinatorial Configurations", JACM 20 (1973), 500-513.

[2] _____, "Four Combinatorial Algorithms", CACM 16 (Nov 73) 690-691.

[3] Ewell, J.A., "Partition Recurrences", Journal of Combinatorial Theory, Series A 14 (Jan 73), 125-127.

[4] Overholt, K.J., "Partition Functions (modulo d)" Bit 9 (1969), 83-86.

[5] Stockmal, F., "Generating of Partitions in Part-count Form", CACM 5 (June 1962), 344.

[6] _____, "Generating of Partitions with Constraints", CACM 5 (August 1962), 434.