Technical Report CS74009-R

ANALYSIS OF AN ENUMBERATION ALGORITHM FOR UNORDERED k-PARTITIONS OF n

by

Andy N. C. Kang**

and

C. K. Kang*

**Computer Science Department

Virginia Polytechnic Institute and State University

Blacksburg, Virginia 24061

*Dynamic Mechanics Department

National Tsing Hua University

Hsinchu, Taiwan

Key words and Phrases: combinatorics, partitions, enumeration, analysis of algorithms

Abstract

In many combinatorial problems, the need to compute the number of unrestricted partitions of n or to enumerate over the set of unrestricted partitions of n occurs frequently. Let two positive integers k and n be given, the set of unordered k-tuples $(a_1,a_2,\ldots a_k)$ of positive integers such that $a_1+a_2+\ldots+a_k=n$ is called the unordered k-partitions of n. In this paper, an algorithm to enumerate the set is presented and analyzed. The running time of the algorithm is shown to be linearly proportional to the number of elements in the set. The number of unordered k-partitions of n, $f_k(n)$, is given in a recurrence relation as a byproduct. With these results, to enumerate the unrestricted partitions of n, we need only to enumerate successfully the unordered 1-partition of n, 2-partitions of n,..., up to n-partitions of n. And p(n), the number of unrestricted partitions of n, is simply $\sum_{k=1}^{n} f_k(n)$.

2. Observations on the Set

Definition 1.

An unordered k-partitions of a positive integer n is the set of all unordered k-tuples $(a_1,a_2,\ldots,a_k)$ of positive integers such that $a_1+a_2\ldots+a_k = n$.

For example, the unordered 3-partitions of 8 is the set: {(1,1,6), (1,2,5), (1,3,4), (2,2,4), (2,3,3)}. In the definition above, "unordered" refers to the fact that the same combination of elements in a different order is not counted as a different partition. For instance, (1,1,6) and (1,6,1) are considered as one partition in the set of unordered 3-partitions of 8. Therefore, any enumeration of the set must not to enumerate the same combination of integers in different order. Our enumeration procedure is to enumerate the set of k tuples $(a_1,a_2,\ldots,a_k)$ in such a way that $a_1+a_2+\ldots a_k = n$ and $a_1 \leq a_2 \leq \ldots \leq a_k$. This will keep the enumeration of each partition at most once. The enumeration starts with the k-tuple (1,1,...,1,n-k+1) and then gradually increases the components of the k-tuple in lexicographic

order (i.e. row-major order), while in the process of enumeration, the two rules such that $a_1+a_2\ldots+a_k = n$ and $a_1\le a_2\le\ldots\le a_k$ are always kept valid before the k-tuple is produced.

3. Algorithm for Enumeration

Our algorithm to enumerate the set of unordered k-partitions of n is based on the following lemma.

Lemma.

Each k-tuple in the enumeration contains sufficient information to uniquely determine the next k-tuple in the sequence.

Proof.

Assume $(a_1,a_2,\ldots,a_k)$ is the current k-tuple under consideration. Thus, it has the properties: $\sum_{i=1}^{k} a_i = n$ and $a_1\le a_2\le\ldots\le a_k$. We will produce the next k-tuple in the sequence as follows: First see if $a_k > a_{k-1}+1$. If so, the next k-tuple $(a'_1,a'_2,\ldots,a'_k)$ is $(a_1,a_2,\ldots,a_{k-1}+1, a_k-1)$. This is a valid k-partition since $\sum_{i=1}^{k} a'_i = \sum_{i=1}^{k} a_i = n$ and $a_k > a_{k-1}+1$ implies that $a_k-1\ge a_{k-1}+1$, that is, $a'_k\ge a'_{k-1}$. Otherwise, we have $a_k \le a_{k-1}+1$. This means $a_k = a_{k-1}+1$, or $a_k = a_{k-1}$ since a_k must be greater than or equal to a_{k-1}. In this case, we will search for a largest integer j such that $1\le j\le k-2$ and $a_j<a_{j+1}$. If such j is found, then we consider the following three cases (notice that here we have $a_{j+1}=a_{j+2}=\ldots=a_{k-1}$):

Case 1. $a_k=a_{k-1}+1$. The next k-tuple $(a'_1,a'_2,\ldots a'_k)$ is defined as follows:

$$\left.\begin{aligned} a'_i &= a_i && \text{for } 1\le i<j \\ a'_i &= a_j+1 && \text{for } j\le i\le k-1 \\ a'_k &= \sum_{\ell=j}^{k} a_\ell - (a_j+1)(k-j) \end{aligned}\right\} \qquad (1)$$

This is a valid k-partition because:

(a) $$\sum_{i=1}^{k} a'_i = \sum_{i=1}^{j-1} a'_i + \sum_{i=j}^{k} a'_i = \sum_{i=1}^{j-1} a_i + (a_j+1)(k-j) + a'_k = \sum_{i=1}^{j-1} a_i + \sum_{i=j}^{k} a_i = n$$

(b) $$a'_k = a_j + \sum_{\ell=j+1}^{k-1} a_\ell + a_k - (a_j+1)(k-j)$$

$$= a_j + (a_{j+1})(k-1-j) + (a_{j+1}+1) - (a_j+1)(k-j) \quad (\text{Since } a_{j+1}=a_{j+2}=\ldots=a_{k-1} \text{ and } a_k = a_{k-1}+1 = a_{j+1}+1)$$

$$= a_j+1+(a_{j+1}-(a_j+1))(k-j)$$

$$= a'_{k-1} + (a_{j+1}-(a_j+1))(k-j) \geq a'_{k-1} \quad (\text{since } a_{j+1} > a_j)$$

Case 2. $a_k=a_{k-1}$ and $a_j+1 < a_{j+1}$. The next k-tuple $(a'_1,a'_2,\ldots,a'_k)$ is defined as the same as in equations (1) above. This is a valid k-partition because: $a'_k = \sum_{\ell=j}^{k} a_\ell - (a_j+1)(k-j) = a_j + \sum_{\ell=j+1}^{k} a_\ell - (a_j+1)(k-j) = a_j+(a_{j+1}-(a_j+1))(k-j)$

$$\geq a_j+(k-j) \quad (\text{since } a_j+1<a_{j+1} \text{ implies } a_{j+1}-(a_j+1) \geq 1)$$

$$\geq a'_{k-1} \quad (\text{since } j \leq k-2 \text{ and } a'_{k-1} = a_j+1)$$

Case 3. $a_k=a_{k-1}$ and $a_j+1 = a_{j+1}$. In this case we will search for another integer m such that $1 \leq m \leq j-1$ and $a_m<a_{m+1}$. If and when such m is found (notice now we have $a_m<a_{m+1}=\ldots=a_j$ and $a_j+1=a_{j+1}=\ldots=a_k$), we will define the next k-tuple $(a'_1,a'_2,\ldots,a'_k)$ as follows:

$a'_i = a_i$ for $1 \leq i < m$

$a'_i = a_m+1$ for $m \leq i \leq k-1$

$$a'_k = \sum_{\ell=m}^{k} a_\ell - (a_m+1)(k-m)$$

This is a valid k-partition because (a) $\sum_{i=1}^{k} a'_i = \sum_{i=1}^{k} a_i = n$ and (b)

$$a'_k = \sum_{\ell=m}^{k} a_\ell - (a_m+1)(k-m) = a_m + (j-m)(a_{m+1}) + (k-j)(a_{m+1}+1) - (a_m+1)(k-m)$$

$$= a_m + (k-j) + (a_{m+1} - (a_m+1))(k-m)$$

$$\geq a'_{k-1} . \quad (\text{since } a'_{k-1} = a_m+1 \text{ and } a_{m+1} > a_m)$$

This completes the proof of the lemma.

Based on the algorithm described above, a flowchart is given below. Labels on the arrows of the flowchart indicate the number of each path taken. Notice that in the flowchart the amount of flow into each node is equal to the amount of flow going out. (see following page for flowchart).

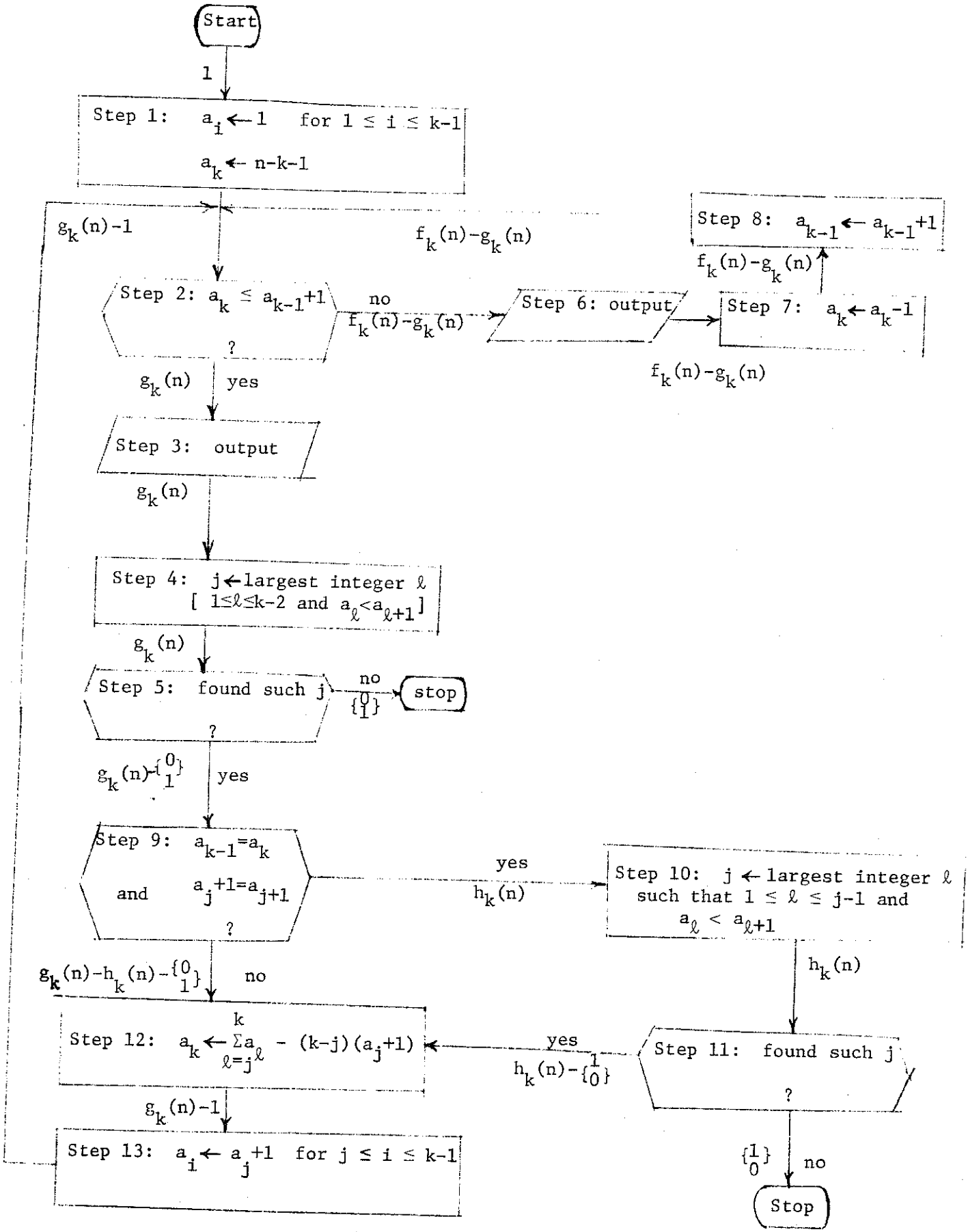
4. Running Time Considerations.

If we give the number of times each step is executed then we have enough information to determine the rumming time of the algorithm on any computer. Next, we are going to count the number of times each step is executed (cf. flowchart). We first make the following definitions:

Definition 2.

$f_k(n)$ is the number of all unordered k-partitions of n. $g_k(n)$ is the number of k-tuples $(a_1,a_2,\ldots,a_k)$ in the set of all unordered k-partitions of n such that $a_{k-1}=a_k$ or $a_{k-1}+1=a_k$. And $h_k(n)$ is the number of k-tuples $(a_1,a_2,\ldots,a_k)$ in the set of all unordered k-partitions of n such that there exists a j with the property that $1 \leq j \leq k-2$ and $a_j+1=a_{j+1}=a_{j+2}=\ldots=a_k$.

For example, 4-partitions of 9 has six elements: (1,1,1,6), (1,1,2,5), (1,1,3,4), (1,2,2,4), (1,2,3,3), (2,2,2,3), thus $f_4(9) = 6$. Among them


Start
1
Step 1: $a_i \leftarrow 1$ for $1 \le i \le k-1$
$a_k \leftarrow n-k-1$
$g_k(n)-1$
$f_k(n)-g_k(n)$
Step 8: $a_{k-1} \leftarrow a_{k-1}+1$
$f_k(n)-g_k(n)$
Step 2: $a_k \le a_{k-1}+1$?
no
$f_k(n)-g_k(n)$
Step 6: output
Step 7: $a_k \leftarrow a_k-1$
$f_k(n)-g_k(n)$
$g_k(n)$
yes
Step 3: output
$g_k(n)$
Step 4: $j \leftarrow$ largest integer ℓ [$1 \le \ell \le k-2$ and $a_\ell < a_{\ell+1}$]
$g_k(n)$
Step 5: found such j ?
no
$\{^0_1\}$
stop
$g_k(n)-\{^0_1\}$
yes
Step 9: $a_{k-1}=a_k$ and $a_j+1=a_{j+1}$?
yes
$h_k(n)$
Step 10: $j \leftarrow$ largest integer ℓ such that $1 \le \ell \le j-1$ and $a_\ell < a_{\ell+1}$
$h_k(n)$
$g_k(n)-h_k(n)-\{^0_1\}$
no
Step 12: $a_k \leftarrow \sum_{\ell=j}^{k} a_\ell - (k-j)(a_j+1)$
yes
$h_k(n)-\{^1_0\}$
Step 11: found such j ?
$g_k(n)-1$
Step 13: $a_i \leftarrow a_j+1$ for $j \le i \le k-1$
$\{^1_0\}$
no
Stop

(1,1,3,4), (1,2,3,3), and (2,2,2,3) has the property that $a_3+1 = a_4$ or $a_3 = a_4$, thus $g_4(9) = 3$. And we see that only (1,2,3,3) has the property that $a_2+1=a_3=a_4$, thus $h_4(9) = 1$. By observing the flowchart we have the following table of running time.

Step Number	Number of Times	Explanation
1	1	Initialize the k-tuple (1,1,...,n-k+1)
2	$f_k(n)$	There are $f_k(n)$ unordered k-partitions of n in the enumeration. All of them have to be tested in this step before output.
3,4,5	$g_k(n)$	There are only $g_k(n)$ k-tuples among $f_k(n)$ unordered k-partitions of n pass the test on Step 2 (by the definition of $g_k(n)$.)
6,7,8	$f_k(n)-g_k(n)$	The rest k-tuples among $f_k(n)$ which do not pass the test on Step 2.
9	$g_k(n)-\binom{0}{1}$	If the algorithm stops at Step 5, then Step 9 will be executed $g_k(n)-1$ times. Otherwise, it will be executed $g_k(n)$ times.
10,11	$h_k(n)$	The number of k-tuples pass Step 9 is exactly $h_k(n)$ (by the definition of $h_k(n)$).
12,13	$g_k(n) - 1$	If the algorithm stops at Step 5, then it will not stop at Step 11 and vise versa. Therefore, the total incoming flow to Step 12 is $g_k(n)-h_k(n)-\binom{0}{1}+h_k(n)-\binom{1}{0}= g_k(n)-1$.

Assume that for a particular computer, each step takes one unit of time to execute (This is a rough estimation. It is quite obvious to assume the number of units of time to execute steps 3,4,6,10, 12, and 13 is a function of k). Then the running time,T(k,n), of the algorithm with imputs k and n is the summation of the total number of times executed by all steps. That is,

$$T(k,n) = 1+f_k(n)+3g_k(n)+3(f_k(n)-g_k(n))+(g_k(n)-\{{}^{0}_{1}\})+2h_k(n)+2(g_k(n)-1)-\{{}^{0}_{1}\}-\{{}^{1}_{0}\}*$$

$$= 4f_k(n)+3g_k(n)+2h_k(n)-\{{}^{2}_{3}\} \leq 7\ f_k(n)$$

The last inequality comes from the fact that $f_k(n)>g_k(n)>h_k(n)$. Thus, we deduce that the running time of the algorithm is linearly proportional to the number of items generated. For a closer estimation on T(k,n), we need to know $f_k(n)$, $g_k(n)$, and $h_k(n)$ explicitly.

Unfortunately, there are no simple analytic formulas for $f_k(n)$, $g_k(n)$, and $h_k(n)$. We are going to derive recurrence equations for them. Let us first prove a theorem.

Theorem.

In the enumeration of all $f_k(n)$ unordered k-partitions of n, there are exactly $f_{k-1}(n-1-(i-1)k)$ k-tuples with the first component to be the integer i.

Proof.

Let us write down the k-tuples in the enumeration of all unordered k-partitions of n:

---

* Two possible stop steps.

$(1,1,\ldots,1,n-k+1)$

$\cdot$

$\cdot$

$\cdot$

$(2,2,\ldots,2,n-k+1-(k-1)1)$

$\cdot$

$\cdot$

$\cdot$

$(3,3,\ldots,3,n-k+1-(k-1)2)$

$\cdot$

$\cdot$

$\cdot$

$(i,i,\ldots,i,n-k+1-(k-1)(i-1))$

$\cdot$

$\cdot$

$\cdot$

The list of k-tuples with the first component i (call it list A) start with $(i,i,\ldots,i,\ n-k+1-(k-1)(i-1))$ and are enumerated in lexicographic order. Let us make a second list (call it list B) by subtracting from each component of list A by i-1 and ignore the first component. Then list B is a list of k-1 tuples which starts with $(1,1,\ldots 1,n-k+1-(k-1)(i-1)-(i-1))$. By induction, we know there are $f_{k-1}(k-2+n-k+1-(k-1)(i-1)-(i-1)) = f_{k-1}(n-1-(i-1)k)$ items in list B. Since to each k-1 tuple in list B, there corresponds an k-tuple in list A and vise versa, this proves the theorem.

Corollary 1.

Let $f_k(n)$ be the number of all unordered k-partitions of n, then it is given by:

$f_1(n) = 1$, and for $k \geq 2$

$$f_k(n) = \sum_{i=1}^{[n/k]*} f_{k-1}(n-1-(i-1)k).$$

---

* $[x]$ is the largest integer less than or equal to x.

Proof.

$f_1(n) = 1$ is obvious, since only one 1-partition of n. In general, in the list of all unordered k-partitions of n the first component can be 1,2,..., i,... Where i must be less than or equal to [n/k]: If not, say i is [n/k] + 1 then the remaining k-1 components will sum up to n - [n/k] - 1. By the property that the remaining k-1 components are not less then [n/k] + 1, we deduce that the sum of the remaining k-1 components must not be less than $(k-1)([n/k]+1) = k\cdot[n/k]+k - [n/k] -1$, which is greater than n - [n/k] - 1. Contradicition. Since there are $f_{k-1}(n-1-(i-1)k)$ items with the first component i, and the total enumeration is a list with the first component to be one of the numbers in the set: {1, 2, ..., i,...,[n/k]}, we have

$$f_k(n) = \sum_{i=1}^{[n/k]} f_{k-1}(n-1-(i-1)k).$$

Corollary 2.

Let $g_k(n)$ be the number of k-tuples $(a_1,a_2,\ldots,a_k)$ in the set of all unordered k-partitions of n such that $a_{k-1}=a_k$ or $a_{k-1}+1 = a_k$, then it is given by:

$g_2(n) = 1$, and for $k \geq 3$

$$g_k(n) = \sum_{i=1}^{[n/k]} g_{k-1}(n-1-(i-1)k).$$

Proof.

notice that in the proof of the theorem list B generated from list A preserves the property that $a_{k-1}=a_k$ or $a_{k-1}+1 = a_k$, since list B is generated by subtracting each component by the same value. Thus, the recurrence equation holds as well for $g_k(n)$. For $g_2(n)$, we see that in the enumeration of 2-partitions of n: {(1,n-1), (2,n-2),...,([n/2],n-[n/2])} exactly only the

last item has the property that $a_1=a_2$ or $a_1+1=a_2$. Thus $g_2(n) = 1$.

Corollary 3.

Let $h_k(n)$ be the number of k-tuples $(a_1,a_2,\ldots,a_k)$ in the set of all unordered k-partitions of n such that there exists a j with the property that $1 \le j \le k-2$ and $a_j+1 = a_{j+1}=a_{j+2}=\ldots a_k$, then it is given by:

$$h_3(n) = \begin{cases} 1 & \text{if there exists an integer } \ell \text{ such that } 1 \le \ell \text{ and } n = 2+3\ell, \\ 0 & \text{otherwise.} \end{cases}$$

and for $k \ge 4$,

$$h_k(n) = \begin{cases} 1 + \sum_{i=1}^{[n/k]} h_{k-1}(n-1-(i-1)k) & \text{if there exists an integer } \ell \text{ such that } 1 \le \ell \text{ and } n = k-1+k\ell, \\ \sum_{i=1}^{[n/k]} h_{k-1}(n-1-(i-1)k) & \text{otherwise} \end{cases}$$

Proof.

For $h_3(n)$, we see that there is at most one 3-tuple in the list of all unordered 3-partitions of n with the property that $a_1+1=a_2=a_3$. That is, when $n = \ell + (\ell+1) + (\ell+1) = 2+3\ell$ for some ℓ. The 3-tuple is $(\ell,\ell+1,\ell+1)$.

Let us consider $h_k(n)$: Similarly, we see that in the proof of the theorem, list B generated from list A preserves the property that "an integer j exists such that $j \le k-2$ and $a_j+1=a_{j+1}=\ldots=a_k$" for all k-tuples in list A except possibly the last k-tuple of the form $(\ell,\ell+1,\ldots,\ell+1)$. Since then the corresponding k-1 tuple in the list B is $(2,2,\ldots,2)$, which does not contribute to $h_{k-1}(n-1-(\ell-1)k)$. In this case n is $k(\ell+1)-1 = k\ell+k-1$. Thus, only when n is $k\ell+k-1$ for some $\ell \ge 1$, we need to add 1 to compensate the above mentioned item. Otherwise, to every k-tuple in list A that contributes to $h_k(n)$, there corresponds a k-1 tuple in list B that

contributes to $h_{k-1}(n-1-(i-1)k)$ and vise versa. Therefore, the recurrence equation for $h_k(n)$ holds as asserted.

From the recurrence equations for $f_k(n)$, $g_k(n)$ and $h_k(n)$, we see that $f_k(n) \simeq C_1(k)\ n^{k-1}$*, $g_k(n) \simeq C_2(k)n^{k-2}$, and $h_k(n) \simeq C_3(k)n^{k-3}$. Thus, for fixed k, $f_k(n) \gg g_k(n) \gg h_k(n)$ for all but finite numbers of n. Therefore, for large n we can see that most of the running time is spending on Step 6,7, and 8 (cf. flowchart), which are straight forward output and simple updating steps.

6. References.

[1] Ewell, J. A.,"Partition Recurrences", Journal of Combinatorial Theory, Series A, 14, 1973.

[2] Hall, M., "Combinatorial Theory", Blaisdell Pub. Co., 1967.

[3] Hardy, G.H. and Wright, E.W., "An Introduction to the Theory of Numbers", Oxford & Clarendon Press, 1964.

[4] Kang, A. and Kang, C.K., "A Formula for the Partition Function", AMS Notices, Vol. 20, No. 7, 1973.

[5] Knuth, D., "The art of Computer Programming", Addison Wesley, 1968.

[6] Ramanujan, S., "Collected Papers", Chelsea Pub. Co., 1962.

---

* See [2], where $f_k(n) \simeq \dfrac{n^{k-1}}{(k-1)!\ k!}$