Storage Allocation in a Certain Iterative Process

JOHN ABRAMOWICH
Wayne State University, Detroit, Michigan*

A method of core storage allocation in a certain iterative process is described and estimates of the machine time required are given. The method is applicable to iterative processes in which input data items once chosen are never again needed. In this method the input data is continuously relocated and the space made available apportioned to the output tables when an overflow occurs. Some important special cases are considered in which considerable simplification occurs.

It sometimes happens that processing a table of data (input) consists of consecutive passes through the table, in each of which certain items (one or more contiguous words) are chosen for processing according to some rule and that thereafter the items become "obsolete," i.e., are never again chosen. For each item chosen, one or more output data items are generated and are retained (at least temporarily) in core storage.

Should it happen that there is insufficient core storage space available to hold both the input and output data tables but sufficient space for the output data alone, then instead of using peripheral storage (which may be undesireable or even impossible) we can make available the space occupied by the obsolete data items. This is done by continuous relocation of the nonobsolete input data in the course of a pass, so that before the end of a pass a gap occurs between the nonobsolete data and the data yet to be examined in the pass. At the end of a pass the input table is shortened by the number of items chosen and the space made available is used for the storage of output.

To accomplish this, we arrange the input and output tables in a contiguous fashion and whenever an overflow of the output tables occurs (if before the end of a pass we first close the gap in the input table by relocating the items yet to be examined in that pass) we relocate the output tables, apportioning to each some of the space made available. All address modifications are, of course, done by the machine.

This process is quite inexpensive timewise and can result in a saving of machine time even in the case where there is no shortage of core storage space. To show this, we compute the upper and lower bounds for the time used in moving the tables in core storage.

Let A be the total available core storage space (for both input and output); L_0, the length of the input table; $L_1, \cdots, L_k$, the lengths of the k output tables. Let $L = L_1 + \cdots + L_k$, and necessarily $L \leqq A$. If $A - L_0 = I \geqq L$, there is no shortage of core storage and we assume the contrary. Let us assume that for each word (unit of core storage) of input that is chosen (made obsolete) $n_1, n_2, \cdots, n_k$ words are added to the output tables $L_1, L_2, \cdots, L_k$, respectively. We shall assume that $n_1, n_2, \cdots, n_k$ are reasonably constant throughout the processing and hereafter it is assumed that these are the constant averages for the process. Initially, we apportion the space I for the tables $L_1, L_2, \cdots, L_k$ in the ratio $n_1{:}n_2{:}\cdots{:}n_k$, respectively, and in the analysis that follows it is assumed that the output tables maintain this constant ratio as they grow. Clearly, we must assume that I is reasonably greater than $n = n_1 + \cdots + n_k$.

When the output tables overflow, the gap in the input table (if we are not at the end of a pass) is closed, the output tables are relocated and the space made available is apportioned to each again in the ratio $n_1{:}n_2{:}\cdots{:}n_k$.

First we compute the number of times N that an overflow of the output tables can occur. Assuming that $n > 1$, the first time that an overflow occurs, the total length of the output tables is I; I/n words have been removed from the input table and the space made available for the output tables. The second time an overflow occurs, $I/n + I/n^2$ words have been removed from the input table, etc. The maximum number of times that the output tables overflow is the smallest integer N such that

$$I/n + I/n^2 + \cdots + I/n^N \geqq L.$$

Summing and using the relations $L_0 n = L$ and $I = A - L_0$, we obtain $I/n^{N+1} \geqq (A - L)/(A - L_0)$.

Thus N is the smallest integer such that

$$N \geqq \log \frac{A - L_0}{A - L} / \log n - 1.$$

For $n = 1$ we distinguish two cases, $k = 1$ and $k \neq 1$. For $k = 1$ the output table will not overflow if no more than I words are removed from the input table in a pass. In this case we define N as the number of times that an overflow could occur. In the case $k \neq 1$, overflows shall

* Department of Mathematics

actually occur. In either case it is not difficult to see that N is the smallest integer such that

$$N \geqq \frac{L_0}{A - L_0} - 1.$$

To write N in a more suitable form, let us define:

$p_0 = L_0/A$ = fraction of total available core storage required for the input data,

$p_1 = L/A$ = fraction of total available core storage required for the input data,

$q_0 = 1 - p_0$ = fraction of total core storage initially empty,

$q_1 = 1 - p_1$ = fraction of total core storage finally empty.

Then $n = p_1/p_0$ and we can write

$$N \geqq \begin{cases} \dfrac{\log (q_0/q_1)}{\log (p_1/p_0)} - 1, & n = p_1/p_0 > 1, \\ \dfrac{p - q}{q}, & p_0 = p_1 = p. \end{cases}$$

Applying L'Hospital's rule, we can easily see that the second of these expressions is the limit as $p_1 \to p_0$ of the first.

To obtain the time required, we first compute the total number M_1 of words moved in relocating the output data tables. It is not difficult to see that

$$M_1 = NI + (N - 1)I/n + \cdots + I/n^{N-1}, \quad n = 1,$$
$$= NI + (N - 1)I + (N - 1)I + \cdots + I, \quad n = 1, \quad k \neq 1.$$

In the case $n = 1$ and $k = 1$, the output table is never relocated.

The total number M_2 of words moved in relocating the input data is

$$M_2 = C(L_0 - I/n) + \cdots + C(L_0 - I/n - \cdots - I/n^N), \quad n \neq 1,$$
$$= C(L_0 - I) + \cdots + C(L_0 - NI), \quad n = 1,$$

where $0 \leqq C \leqq 1$ ($C = 0$ in the case that each time an overflow of the output tables occurred, we were at the end of a pass through the input table; $C = 1$, near the beginning). We obtain for the sum $M = M_1 + M_2$,

$$M = CNL_0 + I(1 - C/n)(N + (N - 1)/n + \cdots + 1/n^{N-1}), \quad n \neq 1,$$
$$= IN(N + 1)/2 + CN(L_0 - (N + 1)I/2), \quad n = 1, \quad k \neq 1.$$

If we sum the second term in the first of the above formulas for M, we easily see that the second of these is the limit of the first as $n \to 1$. Indeed,

$$N + (N - 1)/n + \cdots + 1/n^{N-1} = \sum_{k=1}^{N} (1 - 1/n^k)/(1 - 1/n),$$

and the assertion easily follows by applying L'Hospital's rule. The maximum and minimum values of M correspond to $C = 1$ and $C = 0$, respectively; the average value of M corresponds to the average value $C = \frac{1}{2}$.

It is convenient to consider $m = M/A$, the number of relocations per word of total available core storage. Let us define N_1 and N_2 as the least integral upper bounds of $(\log (1 - p_0)/(1 - p_1))/\log p_1/p_0$ and $p/(1 - p)$ ($p_0 = p_1 = p$), respectively.

Let us compute the maximum and average values of m corresponding to $C = 1$ and $C = \frac{1}{2}$, respectively. Assuming that $1/n^{N+1}$ is approximately equal to $(1 - p_1)/(1 - p_0)$, we obtain after a somewhat lengthy calculation:

$$\max m = \begin{cases} N_1 - (p_0 + p_1), & n \neq 1, \\ N_2 p - p, & n = 1, \quad k \neq 1; \end{cases}$$

$$\text{av } m = \begin{cases} \frac{1}{2}N_1(1 + p_1(1 - p_0)/(p_1 - p_0)) \\ \qquad - \frac{1}{2}(p_0 + p_1 + p_1^2/(p_1 - p_0)), & n \neq 1, \\ \frac{1}{4}N_2^2(1 - p) + \frac{1}{4}N_2(3p - 1) - \frac{1}{2}p, \\ \qquad n = 1, \quad k \neq 1. \end{cases}$$

The case $k = 1$ and $n = 1$ is somewhat trivial and we shall not discuss it in detail. In this case, $M = M_2$ and m can be easily found.

The preceding formulas for max m and av m are not too accurate for n approximately equal to unity, and in this case they should be replaced by the exact formulas:

$$\max m = N - \frac{(1 - p_0)}{p_1/p_0 - 1}\left(1 - \left(\frac{p_0}{p_1}\right)^N\right), \quad n \neq 1,$$

$$\text{av } m = \frac{1}{2}N\left(1 + \frac{p_1(1 - p_0)}{(p_1/p_0 - 1)p_0}\right) - \frac{(1 - p_0)(p_1/p_0 - \frac{1}{2})}{(p_1/p_0 - 1)^2}\left(1 - \left(\frac{p_0}{p_1}\right)^N\right), \quad n \neq 1,$$

where N is the number of times that the output tables have to be relocated. The preceding formulas for max m and av m and $n \neq 1$ are exact.

The value of the constant C deserves comment. The value of C depends principally upon the rate at which words are removed from the input table. Indeed, if we assume that the rate of removal is constant and that sI/n words are removed in the first pass through the input table, then assuming that s is an integer and that sI/n is an integral multiple of L_0, the value of C is easily seen to be $\frac{1}{2}(1 - 1/s)$. Thus if $s = 1$, $C = 0$, and in general one would probably be working with a value of C that is less than $\frac{1}{2}$.

Two time-saving variations of the basic procedure can often be employed:

(A) Suppose the exact lengths of the output tables $L_1, \cdots, L_k$, is known beforehand. Then if I is large enough we can set the first k_1 output tables to their full lengths and will need to relocate only the remaining $k - k_1$ tables. This decreases M_1.

(B) Suppose as in (A) that the lengths of the output tables are known but I is not large enough to set any of the output tables to their full lengths initially. Then if considerable space is made available in the first few passes through the input table, some of the output tables can then be set to their full lengths. This again decreases M_1.

These variations are particularly useful if all but one of the output tables can be set to their full lengths. This could decrease the value of the constant C in the subsequent processing.

We have tacitly assumed throughout that index registers are at our disposal for the relocation of the data and the location of the output tables. If this were not so, time would be lost.

The advantages or disadvantages of continuous relocation of the input data also depend on the relative price of moving items and examining them.

The time T required for the relocation of the output and input tables is given by $T = MT_0$, where T_0 is the time required to move one word. For a 1 address fixed-word length machine T_0 would usually be 4 machine cycles.

It may be remarked here that continuous relocation of the input data reduces the number of items that are examined. Indeed, assuming that P passes are made through the input table and that the number of items processed in each pass is the same, a total of L_0P items would be examined without continuous relocation as against a total of $L_0(P + 1)/2$ items with continuous relocation.

For example, taking $A = 25{,}000$, $p_0 = 0.48$, $p_1 = .96$, we find $2.21 \leqq m \leqq 2.54$ and av $m = 2.38$. Thus $T_{\max} = 2.54 \times 2.5 \times 10^4 T_0$. We find $M_2 = .17 \times 25{,}000$ and continuous relocation of the input data would break even after two passes through the input data.

It must be pointed out that N was computed on the assumption that the output tables grow at a uniform rate. If this is not the case, the value of N can be considerably larger. In some cases this can be overcome by using one of the variations of the basic procedure described above. For example, if only one of the tables does not grow uniformly and we can use variation (A), then N would remain unchanged.

A short table of values of av m as a function of p_0 and p_1 is given below.

p_1 / p_0	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2
0.9	5.35	3.03	2.27	1.94	1.05	0.91	0.41	0.32
0.8	4.35	1.80	1.77	1.02	0.44	0.40	0.33	
0.7	2.84	1.98	1.15	0.48	0.43	0.39		
0.6	2.17	1.29	0.53	0.50	0.46			
0.5	1.42	0.69	0.57	0.54				
0.4	1.54	0.65	0.62					
0.3	0.74	0.72						
0.2	0.81							

RECEIVED OCTOBER, 1966; REVISED JANUARY, 1967

LETTERS—cont'd from p. 342

The Checkless Society: Individual Authorization of Payment

EDITOR:

Recent literature on automated banking systems has proposed the "Checkless Society," the automatic crediting of wages and debiting of bills from an individual's bank account, with an automatic loan to cover any deficit. Details concerning remote terminals in retail establishments vary. One area of the system that should be emphasized is the ability of making individual authorization for payment of each bill. There are three significant reasons for delaying the payment of a bill (an action possible only when each payment is individually authorized): to obtain correction of an erroneous bill, to save paying interest, and to wait until an advanced billing falls due.

By delaying payment of an incorrect bill, a consumer may get the bill corrected. If, on the other hand, the consumer pays an incorrect bill, he may have difficulty in securing a correction. In the extreme case, the party to whom funds are due must substantiate his claim in court; the party against whom the claim is made has the defendant's position of receiving the benefit of doubt.

Time purchases often provide that no interest will be charged if the full purchase price is paid within ninety days. Saving this interest is conditional upon the bill being paid when the consumer authorizes payment, not when the bill is submitted by the merchant. Otherwise, payment could be automatically made, resulting in the customer being charged interest for ninety extra days for an automatic loan. Included within this situation is the payment for professional services. Often, doctors will allow their patients to pay their bills in installments rather than require them to go into debt to pay their bills when due.

When stores bill customers for merchandise ordered but not delivered, payment is not required until the customer takes delivery. This situation differs from those described in the previous paragraph; here the bills are not yet due. Rather than pay bills before they fall due, the consumer should have use of his funds and not put himself into a position where he might have to lose interest on his deposits (or pay interest on an unnecessary automatic loan).

Another reason to defer payment of statements is of lesser importance: taxes. It is sometimes advantageous for a person to delay paying a bill for a deductible expense. If, for example, a child marries and the parent thus loses a $600 dependent's deduction, the parent might reduce his taxes by paying a doctor's bill in January rather than in the preceding December.

The position of business and industry in a checkless society has not been examined in the light of making payments other than payroll. Accounting departments verify invoices and hold up incorrect ones before making payment. Invoices with discount for immediate payment are often paid quicker than those not offering such terms, thus giving the company the best use of its internal funds and reducing the need to borrow. Retail merchants, under the checkless society, will not have the use of sales receipts to the extent which has been claimed; the wholesaler will be removing funds from the merchant's account with the same ease that the merchant removes funds from the consumer's account.

A check is a written authorization to transfer funds. In an automated banking system, the slip of paper representing a check may become obsolete and unused; but to facilitate the procedure of authorizing the payment of bills only after examining each one of them, the use of such authorizations will be desired by the users of the system. The form may change, but the concept will be preserved; an automated banking system will not be a checkless system.

DAVID ROSS
Computer Applications Inc.
Los Angeles, California 90024