Disk File Sorting*

THOMAS SCHICK
International Business Machines Corp., Endicott, N. Y.

Sorting techniques using an IBM 1401 with a random access storage device are evaluated.

Presently there are two methods used to sort a file of records. One of them is referred to as *record sorting*, and the other is called *tag sorting* or *key sorting*. Record sorting is a sorting technique in which the entire record is processed throughout the sort. Tag sorting on the other hand is a method whereby only the tag which consists of the control data and the address of the record is processed throughout most of the sort. Both are similar in that the number of times the control data of each record must be processed in sorting a given file is approximately the same. This holds true only if G, the number of records sorted internally, and the order of merge are held constant. Thus the advantage in using the tag sort is that the record being processed in the sort is usually smaller than it would be in a record sort. If the tag is equal in length to the record, then the tag sort can become a record sort. The major problem in using the tag approach is that an additional phase is required in which each record must be individually retrieved. This additional phase usually proves to be the most time consuming.

Tag Sort

The tag sort may be divided into three phases. Phase 1 performs the following functions. A record is read into memory, its control data is extracted, and a control word or tag is formed consisting of this control data and the address of the record. An internal sort is performed on each successive G records so processed. This is done for the entire file. It should be noted that the value of G may be larger in a tag sort than a record sort because the control word is usually a small fraction of the record. The output, then, of this phase is a set of strings each G in length.

Phase 2 consisting of several passes performs the function of merging these strings into one sequence. The tag sort may require fewer passes than the record sort because the strings developed in phase 1 may be longer. In the last pass of phase 2 the control data is dropped from the control word. The output of the phase is a string of addresses referring to records which when placed in the same order that the addresses appear will themselves be in order; i.e., the first address refers to that record in the file which contains the lowest control data, the second address refers to that record in the file which contains the second lowest control data, etc.

Phase 3, the final phase, retrieves the records and places them in the output area as desired. One manner by which this can be accomplished would be to seek each record as its address appears in the string of addresses. This requires a random seek for each record and therefore tends to be quite time-consuming. Another method is to reorder a set of the addresses so that this set can be retrieved relatively sequentially. The size of this set, call it C, controls the speed of the retrieval of the records. It is therefore of great importance that it be as large as possible. The size of C is determined by the area allowed for the rearranging of the records in memory. If the memory capacity is inadequate, a cylinder on the file could be set aside to function as additional memory.

One serious problem in the various phases but especially in phase 3 is the possible loss of time because of "rotational delay." This problem can exist in nearly all disk file operations. If one reads a block of records and then processes it, one cannot read the next sequential block immediately because the disk will have rotated past the desired point. It is necessary to wait until the disk has completed an entire revolution. This delay can be minimized by the following technique: if it is known that a certain amount of process time is required between reads or writes, one would read or write the next block at that point to where the disk has rotated after the processing has been completed. Thus, for example, on the 1401 with a 1311 attachment if one were to write a block of one sector and the process time were less than 2 msec, one would read or write on every third sector as follows:

$$0, 3, 6, 9, 12, 15, 18, 1, 4, 7, 10, 13, 16, 19, 2, 5, 8,$$

$$11, 17, 0, 3, \cdots$$

The time lost because of the rotational delay time is thus reduced by approximately 90 percent. Note in the above series that when the sector number exceeds 20 (the number of blocks on a track), 20 is subtracted from the last number and the series continues. It is important to note also that in this case the entire track will be fully packed.

The optimum intervals have the following restrictions. Depending on the requirements, they are prime numbers or multiples of prime numbers that do not contain any factors that are also a factor of the number of blocks on the track or cylinder. To insure that the resultant addresses refer to the first sector of the block, said number must be multiplied by the number of sectors in the block. For example, depending on the process time required when writing a sector at a time using the IBM 1401 with a 1311 attachment, the optimum intervals may be 3, 7 or 9. Different number schemes must be developed for other blocking configurations.

Record Sort

The record sort can be divided into two phases. The first phase performs an internal sort on strings of G records. The second phase performs a series of merge passes to

* Presented at an ACM Sort Symposium, November 29, 30, 1962.

order these strings. The internal sort is not dependent on the input-output device. Therefore, we will not discuss it here.

The speed of a record sort is a function of three major factors: (1) the number of access mechanisms available, (2) the maximum blocking factor possible (the maximum number of records contained in a block), and (3) the speed at which sequential operations can be performed. Clearly, these factors are interrelated.

To insure minimum seek times in the merging phase, there must be at least $M+1$ access mechanisms, where M is the order of merge. An approximation of the number of random seeks, S, required per block in each merging pass may be formulated as follows:

$$S = \frac{M + 1 - A + X}{M}$$

where M = order of merge

A = number of access mechanisms available

$$X = \begin{cases} M & \text{if } A = 1 \\ 1 & \text{if } 1 < A \leqq M \\ A - M - 1, & \text{otherwise.} \end{cases}$$

If there is only one access mechanism, there is need for a seek for nearly every read and write. If there are two access mechanisms, the output is sequential and therefore seek time for output is minimal. Only input requires a seek for nearly every read.

Thus, as the number of access mechanisms increases, the seek time decreases. The above formula assumes a random file.

The second factor which influences the speed of the record sort is the blocking factor. The higher the blocking factor becomes, the less significant the rotational delay and seek time become per record. The formula below indicates the interrelation between seek time, rotational delay time and the blocking factor. It also indicates the importance of the rate of the transfer of data from the random access device to memory.

The average time required to read a record can be formulated as follows:

$$T = \frac{A + C \cdot B \cdot L + T}{B}$$

where $L \equiv$ number of characters contained within a record

$B \equiv$ the blocking factor

$A \equiv$ average rotational delay

$C \equiv$ character time

$T \equiv$ seek time (msec).

From the above formula it becomes clear that rotational delay and seek time are significant only if the blocking factor is small.

For example, on the 1401 with the 1311 attachment, the above formula would appear approximately as:

$$T = \frac{20 + .02B \cdot L + 150}{B} = \frac{170}{B} + .02L.$$

If $L = 100$ and $B = 10$, then $T = 17 + 2 = 19$ msec. If $L = 100$ and $B = 100$, then $T = 1.7 + 2 = 3.7$ msec.

Comparison of Tag Sort and Record Sort

To compare the tag and record approaches, the following observations should be made. The time required to read the input in phase 1 is equal for both approaches. Furthermore, the input-output time required in phase 2 of both sorts is directly related to the value Y, where:

$$Y = \frac{W}{L}$$

$W \equiv$ length of control word or key in the tag sort

$L \equiv$ record length.

This becomes evident when one realizes that in phase 2 of the tag sort, W is the size of the record processed and L is its counterpart in the record sort. It should be mentioned here that this holds true only if the block size, order of merge and G are held constant (G is actually equal for both sorts only if $W = L$).

Therefore, only the difference between phase 2 of the record sort and phases 2 and 3 of the tag sort need be compared. We have then the comparison:

$$F_2(1-Y) : F_3$$

where F_2 = input-output time for phase 2 of the record sort

F_3 = phase 3 time of the tag sort.

Clearly, if $Y = 1$, or the control word is equal in length to the record, then the record sort is far superior. However, if $Y = 0.1$, which might be an average case, the tag sort may be preferable.

It should be noted that the comparison is not nearly as straightforward as it appears. If $W = 10$, $L = 100$ and block length = 300, then for the record sort on the 1401-1311,

$$T_1 = \frac{20 + 6 + 150}{3} = 59 \text{ msec},$$

and for the tag sort on the 1401-1311,

$$T_2 = \frac{20 + 6 + 150}{30} = 5.9 \text{ msec}.$$

If $W = 10$, $L = 100$, and block length = 10,000, then as above for the record sort on the 1401-1311:

$$T_3 = \frac{20 + 200 + 150}{100} = 3.7 \text{ msec}$$

and for the tag sort on the 1401-1311:

$$T_4 = \frac{20 + 200 + 150}{1000} = .37 \text{ msec}.$$

Continued on page 339

```
12. THETA = (CZ − BZ) / (CY − BY)
    THETA = ATANF(TTHETA)
13. ALPHA = ATANF(TALPHA)
    BETA = ATANF(TBETA)
```

Write-out statements complete the program

The principal feature of the foregoing subroutines and their usefulness is that of revolving the configuration about an axis parallel to the x-, y-, or z-axis of the reference system. It is clear that rotation about an axis parallel to the z-axis will not alter the z-coordinate of points, nor about the x-axis alter the x-coordinates nor the y-axis the y-coordinates. The revolution of a configuration about any axis will change only the coordinates relative to the axes perpendicular to the axis of rotation.

The fifteen (15) subroutines comprise DESCRIPTRAN and are more than essential, but allow for variations in the programmer's choices. Together they facilitate the solution of three-dimensional problems with a digital computer and in a way analogous to that of descriptive geometry. Thus they constitute automated descriptive geometry.

SCHICK—continued from page 331

Thus as the block size becomes larger and the seek time and rotational delay time become less significant, the difference between the merge phase of the tag sort and record sort becomes much smaller.

$$T_1 - T_2 = 53.1 \text{ msec}, \qquad T_3 - T_4 = 3.33 \text{ msec}.$$

With a smaller blocking factor (occasioned by smaller core capacity) the difference between phases 2 of the record and tag sorts is substantial and offsets the time needed for phase 3 of the tag sort. In larger machines the blocking factor may be increased. Hence, the difference between phase 2 of the record and tag sort diminishes greatly while phase 3 does not decrease appreciably. For this reason more than any other, the record sort is faster than the tag sort on a larger machine.

Considerations of importance not discussed here are the order of merge and the processing time. In a large machine the order of merge can be increased considerably as long as the blocking factor does not decrease to a point where the seek time and rotational delay time become significant. Processing time consumes a greater percentage of the total time in a larger machine, whereas in a small machine, input-output time is the major factor. On a large machine, where the blocking factor is large, the process time becomes relatively more significant.

Thus it appears that on a machine with a small memory capacity the tag sort should prove to be more efficient. The record sort becomes increasingly more efficient as the machine size increases.