Technical Notes

Collating Long Rows of a Character Matrix

This note presents an efficient APL function of general utility in sorting the rows of a character matrix M according to the collating sequence given in a character vector A. Of course, for a character matrix with short rows and a short collating sequence, the one-liner

```
I←⍋(2+⍴A)⊥⍉A⍳M
```

is perfectly suitable, yielding *M*[*I*;] as the collated matrix. This one-liner is the essence of the function SORT discussed by Charmonman (1) and Koegel (2). When long-rowed character matrices or long collating sequences are involved, more precisely, when (2+⍴*A*)⋆1↓⍴*M* is greater than the largest representable consecutive integer(2⋆56 *FOR APL*\360)then the one-liner is inadequate, since the least significant columns of M are effectively ignored. If fact, if (⌊/⍳0)<(2+⍴*A*)⋆1↓⍴*M* is true, then an attempt to execute the one-liner will generally result in a domain error for ⊥ .

The solution proposed by Charmonman (1) is relatively expensive in execution time, as shown by Koegel (2). Two solutions proposed by Koegel, SORTC and SORTG, are reasonably economical, and noticeably more elegant. Koegel notes that SORTC is applicable for collating vectors of any length, an advantage offset by the need to re-interpret a loop of code for every 9 columns of the input matrix. Comparison tests on the author's shared variable system show SORTC to require 1.5 times as much processing time as SORTG, notwithstanding Koegel's report of approximately equal time. The performance advantage of SORTG over SORTC on the author's system is presumably caused by the significantly smaller amount of interpretation required for the former function. Unfortunately, SORTG must be modified if the length of the collating vector changes.

The function COLLATE given below combines the speed of SORTG and the generality of SORTC with some other advantages.

```
          ∇ I←A COLLATE M;B;J
    [1]     J←(-J⌊⌊(B←2+⍴A)⍟2147483647)↑⍳J←1↓⍴M
    [2]     I←⍳1↑⍴M
    [3]     I←I[⍋B⊥⍉A⍳M[I;J]]
    [4]     →3××⍴J←(J≥⍳1)/J←J-⍴J
        ∇
```

In particular, the following points should be noted:

1. The function COLLATE simulates the one-liner in the extended domain including long-rowed matrices and long collating sequences. More precisely, COLLATE yields the vector I of indices such that the rows of *M*[*I*;] are in lexicographical order according to the order of characters in the character vector A. Characters of M not contained in A are all treated as if they were last in order, after all characters in A. (For example, the value of '' COLLATE M is ⍳1↑⍴*M* for any matrix M.) In summary, COLLATE acts as the one-liner would if large integers were represented with infinite precision. The index vector I can be used to sort an array of data associated with the rows of M, thus preserving the association.

2. The algorithm for COLLATE is to sort M by ⍴*J* columns at a time, beginning with the least significant columns. Line (1) computes ⍴*J* to be as large as possible, but small enough so that the results of base value in line (3) are always less than

2*31. (Using 2*31 rather than 2*56 avoids floating point values for ⊥ in APL/360, thereby noticeably improving speed of execution.) For sufficiently short rows and collating sequences the operation of COLLATE reduces essentially to that of the one-liner, in the sense that only one execution of each line is required.

3. The function COLLATE is more versatile than previously proposed functions:

a) Like the one-liner, COLLATE operates in either 0 or 1 origin. Both Charmonman's (1) and Koegel's (2) functions require 1 origin.

b) Like the one-liner, COLLATE handles empty arguments (0=ρ*A AND/OR* 0=×/ρ*M*) correctly. Both Charmonman's and Koegel's functions become suspended with index errors if 0=1↓ρ*M* is true, while COLLATE yields ι1↑ρ*M*.

4. The function COLLATE requires slightly less execution time than SORTG on the author's shared variable system, thus making it the fastest of the proposed functions.

REFERENCES

(1) Charmonman, S., "Numeric sort by binary search and insertion," APL QUOTE QUAD , 4 No. 3 (April 1973), pp. 19-21.

(2) Koegel, J. E., Short Note (technical), in APL QUOTE QUAD, 5, No. 1/3 (Spring 1974), pp. 35-37.

William B. Rubin
Poughkeepsie Laboratory
IBM Corp.
Poughkeepsie, NY