18.3 Sorting Partially Ordered Data of Arbitrary Structure in Memory

A. Purpose
Sort data having an organization or structure not supported by one of the subprograms in Chapter 18.1, for example, data having more than one key to determine the sorted order. This subprogram has similar functionality to GSORTP of Chapter 18.2 and is more efficient when the data are initially partially ordered, or when the ordering criterion is expensive to determine.

B. Usage
B.1 Program Prototype
INTEGER N, L(≥N), L1, COMPAR EXTERNAL COMPAR
Assign values to N and data elements indexed by 1 through N. Require N ≥ 1.

CALL INSORT (COMPAR, N, L, L1)
Following the call to INSORT the contents of L(1) through L(N) contain a linked list that defines the sorted order of the data. L1 is the index of the first record of the sorted sequence. Let I = L(J). If I = 0 record J is the last record in the sorted sequence, else record I is the immediate successor of record J in the sorted sequence.

B.2 Argument Definitions
COMPAR [in] An INTEGER FUNCTION subprogram that defines the relative order of elements of the data. COMPAR is invoked as COMPAR(I, J), and is expected to return −1 (or any negative integer) if the Ith element of the original data is to precede the Jth element in the sorted sequence, +1 (or any positive integer) if the Ith element is to follow the Jth element, and zero if the order is immaterial. INSORT does not have access to the data. It is the caller’s responsibility to make the data known to COMPAR. Since COMPAR is a dummy procedure, it may have any name. Its name must appear in an EXTERNAL statement in the calling program unit.

N [in] The upper bound of the indices to be presented to COMPAR.
L() [out] An array to contain the definition of the sorted sequence. L(1:N) are set so that the immediate successor of the Jth record of the sorted sequence is L(J) if the Jth record is not the last record in the sorted sequence, else L(J) is zero.
L1 [out] The index of the first record of the sorted sequence.

B.3 Converting the Linked List in L() to a Permutation Vector
The linked list produced by INSORT (or by INSRTX, see Chapter 18-04) in the array L() may be converted to a permutation vector by

CALL PVEC (L, L1)
where L() and L1 are as above. Upon return from PVEC, L() is a permutation vector, as described for the argument IP() of GSORTP (Chapter 18.2).

C. Examples and Remarks
The program DRINSORT illustrates the use of INSORT to sort 1000 randomly generated real numbers. The output should consist of the single line

INSORT succeeded

Stability
A sorting method is said to be stable if the original relative order of equal elements is preserved. This subroutine uses a merge sort algorithm, which is not inherently stable. To impose stability, return COMPAR = I − J if the Ith and Jth elements are equal.

D. Functional Description
The INSORT subprogram uses an opportunistic merge sort algorithm, as described by Sedgewick [1], with a modification suggested by Power [2]. In the basic opportunistic merge sort algorithm, the first step consists of detecting either ascending or descending sequences of initially ordered data. In the second step, these sequences are merged in pairs to form half as many sequences, each approximately twice as long as the original sequences (descending sequences are considered in reverse order). The second step is repeated until only one sequence remains. The Power modification consists of putting each sequence into a “bucket” indexed by the base-2 logarithm of its length. When a third sequence is to be put into a bucket, the two longest sequences are merged and put into the next bucket. If this would require putting three sequences into the next bucket, the process is repeated. Finally, sequences remaining in the buckets after the initial order-detecting stage are merged, starting with the smallest sequences and proceeding to the largest, to produce a single sequence.
References


E. Error Procedures and Restrictions

INSORT neither detects nor reports any erroneous conditions.

Limitations on the size of array that can be sorted are imposed by the amount of memory available to hold the array, and the length of an internal array to hold the “buckets” used in the Power modification. The number of buckets is given by a Fortran PARAMETER, currently set to 32. This permits sorting at least 4,294,467,295 records.

F. Supporting Information

The source language is Fortran 77.

<table>
<thead>
<tr>
<th>Entry</th>
<th>Required Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>INSORT</td>
<td>INSORT</td>
</tr>
<tr>
<td>PVEC</td>
<td>PVEC</td>
</tr>
</tbody>
</table>


---

DRINSORT

```fortran
! 1990–02–09 DRINSORT Snyder Initial code.
!
! Test driver for INSORT.
!
! Construct an array of 1000 random numbers using SRANUA.
! Sort it using INSORT.
! Check whether it is in order.
!
logical OK
integer COMPAR,L(1:1000),L1,LS,NCOMP
external COMPAR
real PREV,R(1:1000)
common /RCOM/ NCOMP,R
!
! Generate 1000 random numbers
call sranua(r,1000)
!
! Sort them using INSORT.
ncomp=0
call insort(compar,1000,1,L1)
ls = 11
!
! Check the order.
ok=.TRUE.
prev=-1.0
10 if (L1.ne.0) then
   if (r(L1).lt.prev) ok=.FALSE.
   prev=r(L1)
   L1=L(L1)
   go to 10
end if
!
! Convert l to a permutation vector.
call pvec(L,LS)
!
! Check the order again.
do 20 L1=2,1000
   if (r(L1).lt.r(L1-1)) ok=.FALSE.
20 continue
!
! Print the results.
if (ok) then
   print '( ''INSORT succeeded using '' ,i6 ,'' compares '')', ncomp
else
   print *,'INSORT failed'
```

18.3–2 Sorting Partially Ordered Data of Arbitrary Structure in Memory July 11, 2015
end if

end

integer function COMPARI, J)
end

integer function COMPARI, J)

Determine the relative order of \( R(I) \) and \( R(J) \), where \( R \) is in
the common block /R/COM/. Return \(-1\) if \( R(I) \) should precede \( R(J) \)
in the sorted order, \(+1\) if \( R(I) \) should follow \( R(J) \), and \(0\)
otherwise.

integer I, J, NCOMP
real R(1:1000)
common /R/COM/ NCOMP, R
ncomp=ncomp+1
if (R(I)-R(J)) 10,20,30
10 compar=-1
return
20 compar=0
return
30 compar=+1
return

end