NAG Toolbox

nag_sort_realvec_sort (m01ca)

1 Purpose

nag_sort_realvec_sort (m01ca) rearranges a vector of double numbers into ascending or descending order.

2 Syntax

```
[rv, ifail] = nag_sort_realvec_sort(rv, m1, order, 'm2', m2)
[rv, ifail] = m01ca(rv, m1, order, 'm2', m2)
```

3 Description

nag_sort_realvec_sort (m01ca) is based on Singleton's implementation of the 'median-of-three' Quicksort algorithm (see Singleton (1969)), but with two additional modifications. First, small subfiles are sorted by an insertion sort on a separate final pass (see Sedgewick (1978)). Second, if a subfile is partitioned into two very unbalanced subfiles, the larger of them is flagged for special treatment: before it is partitioned, its end points are swapped with two random points within it; this makes the worst case behaviour extremely unlikely.

4 References

Sedgewick R (1978) Implementing Quicksort programs *Comm. ACM* **21** 847–857

Singleton R C (1969) An efficient algorithm for sorting with minimal storage: Algorithm 347 *Comm. ACM* **12** 185–187

5 Parameters

5.1 Compulsory Input Parameters

1: **rv**(**m2**) – REAL (KIND=nag_wp) array

Elements **m1** to **m2** of **rv** must contain double values to be sorted.

2: **m1** – INTEGER

The index of the first element of **rv** to be sorted.

Constraint: **m1** > 0.

3: **order** – CHARACTER(1)

If **order** = 'A', the values will be sorted into ascending (i.e., nondecreasing) order.

If **order** = 'D', into descending order.

Constraint: **order** = 'A' or 'D'.

5.2 Optional Input Parameters

1: **m2** – INTEGER

Default: the dimension of the array **rv**.

The index of the last element of **rv** to be sorted.

Constraint: **m2** $\geq$ **m1**.

5.3 Output Parameters

1: **rv**(**m2**) – REAL (KIND=nag_wp) array

These values are rearranged into sorted order.

2: **ifail** – INTEGER

ifail $= 0$ unless the function detects an error (see Section 5).

6 Error Indicators and Warnings

Errors or warnings detected by the function:

ifail $= 1$

On entry, **m2** < 1,
or **m1** < 1,
or **m1** $>$ **m2**.

ifail $= 2$

On entry, **order** is not `A' or `D'.

ifail $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

ifail $= -399$

Your licence key may have expired or may not have been installed correctly.

ifail $= -999$

Dynamic memory allocation failed.

7 Accuracy

Not applicable.

8 Further Comments

The average time taken by nag_sort_realvec_sort (m01ca) is approximately proportional to $n \times \log(n)$, where $n =$ **m2** $-$ **m1** $+ 1$. The worst case time is proportional to n^2 but this is extremely unlikely to occur.

9 Example

This example reads a list of double numbers and sorts them into ascending order.

9.1 Program Text

```
function m01ca_example

fprintf('m01ca example results\n\n');

rv = [1.3,     5.9,     4.1,     2.3,     0.5,     5.8,     1.3,     6.5, ...
      2.3,     0.5,     6.5,     9.9,     2.1,     1.1,     1.2,     8.6];
m1 = nag_int(1);
```

```
order = 'Ascending';

[rv, ifail] = m01ca(rv, m1, order);

fprintf('Sorted numbers:\n\n');
fprintf('%7.1f%7.1f%7.1f%7.1f%7.1f%7.1f%7.1f%7.1f\n',rv);
```

9.2 Program Results

```
     m01ca example results

Sorted numbers:

    0.5    0.5    1.1    1.2    1.3    1.3    2.1    2.3
    2.3    4.1    5.8    5.9    6.5    6.5    8.6    9.9
```