

Posted by [Jeremy Bailin](#) on Fri, 08 Mar 2013 23:51:30 GMT

On 3/8/13 4:24 PM, Gianquido Ciani wrote:

> Jeremy's method is twice as fast for large arrays. However, the two methods only give the same result for smallish arrays.

 $\geq$ $\geq$

```
> multisort, round(randomu(s,20,1e1)*10)
```

> : 0.00027489662 0.00026106834

 $\geq$
$$\geq 0$$

```
> : 0.00072312355 0.00039005280
```

 $\vee$ $\sim$

```
> PRO multisort. matrix
```

 $\vee$ $\vee$

1

```
> matrixshape = size(matrix, /dimen)
```

✓

```
> t1 = systime(1)
```

 $\sqrt{}$

✓

> ; this gives you the range of each column:

```
> matrixord = lonarr(matrixshape)
```

```
> for i=0l,matrixshape[0]-1 do matrixord[i,*] = ord(matrix[i,*])
```

```
> ordmax = max(matrixord, dimen=2)
```

 $\succ$

> ; what do you need to multiply by to get a unique range?

```
> column_multiply = [reverse(product(reverse(ordmax[1:*]+1), /int, /cumul)), 1]
```

$$>$$

```
> ; create a unique key and sort on it
```

```
> sortkey = total(matrixord * rebin(column_multiply, matrixshape, /sample), /int, 1)
```

```
> newmatrix = matrix[, sort(sortkey)]
```

```
> t2 = systime(1)
```

$$>$$
$$>$$

```
> newmatrix2 = matrix
```

```

> FOR i = matrixshape[0]-1, 0, -1 DO BEGIN
>   s = bsort(newmatrix2[i, *]);use bsort which maintains order of identical elements
>   newmatrix2 = newmatrix2[*, s]
> ENDFOR
> t3 = systime(1)
>
> print,array_equal( newmatrix, newmatrix2)
>
> print, t3-t2, t2-t1
>
>
> RETURN
> END
>

```

Bwahahah.... oh dear.

The problem is that in this case, column_multiply ends up being a LONG64 in order to fit the maximum value of the PRODUCT. But when you multiply:
 matrixord * rebin(column_multiply, matrixshape, /sample)
 the values are larger than the maximum LONG64... and wrap around to become negative numbers!

```

IDL> print, minmax(matrixord)
      0      10
IDL> print, minmax(column_multiply)
      1 5054470284992937710
IDL> print, minmax(matrixord * rebin(column_multiply, matrixshape, /sample))
-8337803503723676196 8596744417517336158
IDL> help, matrixord, column_multiply
MATRIXORD    LONG    = Array[20, 100]
COLUMN_MULTIPLY LONG64 = Array[20]

```

In principle there are 10^{20} possible rows, and it's trying to make sure it has a unique integer for each possibility.

I guess my advice is to cap it at 19 columns and maybe do it in chunks if there are more columns.

-Jeremy.