Subject: Re: The IDL way: Find last non-zero value
Posted by Chris[6] on Mon, 24 Aug 2009 00:32:18 GMT
View Forum Message <> Reply to Message

On Aug 24, 2:07 am, Eric Hudson <ehud...@mit.edu> wrote:
> Hi,
>
> I have a 2D array that looks something like:
> x x 0 x x 0 0 x 0 0 0 0 0
> x 0 x x x 0 x x x x 0 0 0
> x 0 0 0 0 0 0 0 0 0 0 0 0
> x x x x x 0 0 0 0 0 0 0 0
>
> where x is some non-zero (positive definite) value. You'll notice
> that each row ends with a string of zeros.
> What I'd like to know is the 'IDL way' of returning a vector of the
> location (column) of the last non-zero elements in each row. So in
> this case, [7,9,0,4]
>
> It's straight forward to program with loops, but I figure there must
> be a clever way. I thought that maybe reversing it and doing a
> cumulative total might be a start, but then I can't convince myself
> that that is really going to be faster than doing a loop.
>
> For a sense of scale, the real array is something like 200 x 160000
>
> Thanks,
> Eric

hmmmm.....

```
sz = size(array)
ncol = sz[1]
nrow = sz[2]
nonzero = where(array ne 0)
ind = array_indices(array, nonzero)
sorted = sort(ind[0, *])
result = fltarr(nrow) - 1
result[ind[1, sorted]] = ind[0, sorted]
```

kind of hacky, but heres the idea:

find all of the nonzero elements, and then use array_indices to give
their row/column numbers. Then, find sorting of the array_indices
array that puts the column indices in ascending order. Next, make a
result array with the correct size, and copy, _in this sorted order_,
the columns from the array_indices array into the representative rows
of the result vector. This way, low column indices will get

overwritten by higher column values during the copy.

My test on a small array:

```
IDL> print, array
      1       2       0       0       0
      0       0       0       0       1
      1       1       3       1       0
      0       0       0       0       0
IDL> print, result
     1.00000      4.00000      3.00000     -1.00000
```

Of course, I'm not convinced that this is easier to read or faster
than using a loop....

Chris
