Subject: Re: quickly totaling sections of an array
Posted by Conor on Tue, 13 Feb 2007 18:34:18 GMT
View Forum Message <> Reply to Message

Thanks for the help. I'm starting to think that histogram can do a lot more than I realized. I'm going to have to learn the full uses of histogram and reverse_indicies, and then take a critical look at all my code again and see where I can tighten things up.

On Feb 13, 12:18 pm, "Brian Larsen" <balar...@gmail.com> wrote:
> While I almost hate to say it (as I have just learned how to use this
> recently) this could be a job for histogram and
reverse_indices.http://www.dfanning.com/tips/histogram_tutor ial.html
>
> It seems to me that one way to do this is to:
> - generate the mask [0,0,2,1,3,4,5,2,1,5,7,1,1,1] (same mask values
> stuff)
> - use histogram on the mask with binsize 1 which will then use put
> each mask value in its own bin, from there the reverse_indices thing
> is the trick.
>
> I have experimented with histogram() and found that if you are pulling
> one or two sets out of an array, then where() is faster, pulling three
> or so different sets out then where() and histogram() are comparable,
> and beyond that histogram() is significantly faster.
>
> In a previous post is a touch of code to make the reverse_indices part
> easier:http://groups.google.com/group/comp.lang.idl-pvwave/b rowse_thread/thr...
>
> but it uses idl pointer stuff which always confises the hell out of me
> (this isn't C after all) so I wrote this one, and the example in the
> doc header could be close to your problem, if I understand correctly.
>
> ;+
> ; NAME:
> ; reverse_indices
> ;
> ;
> ; PURPOSE:
> ; use histogram to pull out regions using the reverse_indices keyword
> ;
> ;
> ; INPUTS:
> ; histo - a histogram from the histogram() function
> ; ri - the histogram() reverse_indices keyword output
> ;
> ;
> ; OPTIONAL INPUTS:

```
>  ; bin - the bin you want the indices for
>  ;
>  ;
>  ; KEYWORD PARAMETERS:
>  ; (all must be specified to use value)
>  ; OMIN - omin keyword output from histogram()
>  ; OMAX - omax keyword output from histogram()
>  ; VALUE - specify a bin by value instead of number
>  ;
>  ;
>  ; OUTPUTS:
>  ; out - indices in that bin of a histogram
>  ;
>  ;
>  ; OPTIONAL OUTPUTS:
>  ; none
>  ;
>  ;
>  ; COMMON BLOCKS:
>  ; none
>  ;
>  ;
>  ; SIDE EFFECTS:
>  ; none
>  ;
>  ;
>  ; RESTRICTIONS:
>  ; none
>  ;
>  ;
>  ; EXAMPLE:
>  ; IDL>  data=fix(randomu(101,25)*12)
>  ; IDL>  h=histogram(data, OMIN=omin, OMAX=omax, REVERSE_INDICES=ri,
>  binsize=2, /nan)
>  ; IDL> print, data[sort(data)]
>  ;       0       0       0       1       1       1       1
>  2       3       3       4       4       4       4       4
>  ;       5       6       7       9      10      10      11      11
>  11      11
>  ; IDL>  print, data[reverse_indices(h, ri, 2)]
>  ;       5       4       4       4       4       4
>  ; IDL>  print, data[reverse_indices(h, ri, value=3.5 , OMIN=omin,
>  OMAX=omax)]
>  ;       2       3       3
>  ;
>  ;
>  ;
>  ; MODIFICATION HISTORY:
```

```
> ;
> ;       Mon Feb 12 15:14:46 2007, Brian Larsen
> ;       <lar...@ssel.montana.edu>
> ;
> ;               written and tested
> ;
> ;-
> FUNCTION reverse_indices, histo, ri, bin, OMIN=omin, OMAX=omax,
> VALUE=value
>
> ;;;;;;;;;;; error checking ;;;;;;;;;;;;;;;;;;;
> IF N_ELEMENTS(histo) EQ 0 THEN $
>   message, /ioerror, 'NULL histogram input'
>
> IF N_ELEMENTS(ri) EQ 0 THEN $
>   message, /ioerror, 'NULL REVERSE INDICES input'
>
> IF N_ELEMENTS(bin) EQ 0 AND N_ELEMENTS(value) EQ 0 THEN $
>   message, /ioerror, 'Must either specify bin or value'
>
> IF N_ELEMENTS(value) NE 0 AND (N_ELEMENTS(omax) EQ 0 OR
> N_ELEMENTS(omax) EQ 0) THEN $
>   message, /ioerror, 'Specifying value requires specifying omax and
> omin'
> ;;;;;;;;; things are ok ;;;;;;;;;;;;;;;;;;;;;
> IF N_ELEMENTS(value) NE 0 THEN BEGIN
>     ;; find which bin has 4 in it
>     binsize = (omin+omax)/(N_ELEMENTS(histo)-1)
>     bin = value/binsize
> ENDIF
> IF bin GE N_ELEMENTS(histo) THEN $
>   message, /ioerror, 'Bin out of range, [0,N_ELEMENTS(histo)-1]
> out = ri[ri[bin]:ri[bin+1]-1]
> RETURN, out
>
> END
>
> Brian
>
>  ------------------------------------------------------------ ---------
> Brian A. Larsen
> Dept. of Physics
> Space Science and Engineering Lab (SSEL)
> Montana State University - Bozeman
> Bozeman, MT 59717
>
> On Feb 13, 8:25 am, "Conor" <cmanc...@gmail.com> wrote:
>
```

>> Hmm... I found sample code here:http://www.dfanning.com/code_tips/
>> drizzling.html
>> that claims to do the job, but that doesn't work. The first problem
>> is that n_ind isn't defined. I set that to n_elements(h1)+1 and then
>> ran again. Then I got this:
>> % SPRSIN: Vector must have 6 elements: <FLOAT Array[3]>
>
>> Anyone happen to know what's going on here, or have a better
>> suggestion?
>
>> On Feb 13, 10:05 am, "Conor" <cmanc...@gmail.com> wrote:
>
>>> Hey Everyone,
>>> I'm essentially trying to add together separate sections of an
>>> array, and I need to do it in a very speedy fashion. Here's the
>>> breakdown in IDL. I would want to take an array like this:
>
>>> vals = [10, 15, 13, 12, 11, 14]
>
>>> and imagine I have a mask (which I can easily make) like this:
>
>>> mask = [0, 0, 0, 1, 1, 1]
>
>>> I would then want to add together everything with the same mask value
>>> and put it in a new array. So the result would be:
>
>>> sums = [38, 37]
>
>>> I'm generating images, and for each image I generate this will be done
>>> 1000 times, and there will be 1000 different mask values each time it
>>> is done. I'll be generating a couple hundred images, so I'll be
>>> running this task a couple hundred thounsand times - i.e. execution
>>> speed is very important. Any suggestions on how to speed things up
>>> would be highly appreciated.
>>> Thanks in advanced,
>>> Conor