Subject: Re: how to sort data based on other sorted data
Posted by wlandsman on Wed, 09 Jan 2008 21:03:59 GMT
View Forum Message <> Reply to Message

On Jan 9, 2:24 pm, placebo <willie.mad...@gmail.com> wrote:
In order to keep this post short lets say the
> data file has 8 coordinates scrambled in the list, and I want to sort
> it first by the 'z' coordinate (3rd column) then by the 'y' coordinate
> (2nd column) then finally by the 'x' column. for example:

try Craig Markwardt's multisort

http://astrog.physics.wisc.edu/~craigm/idl/arrays.html#MULTI SORT

Subject: Re: how to sort data based on other sorted data
Posted by Brian Larsen on Wed, 09 Jan 2008 21:30:07 GMT
View Forum Message <> Reply to Message

On Jan 9, 4:03 pm, wlandsman <wlands...@gmail.com> wrote:
> On Jan 9, 2:24 pm, placebo <willie.mad...@gmail.com> wrote:
> In order to keep this post short lets say the
>
>> data file has 8 coordinates scrambled in the list, and I want to sort
>> it first by the 'z' coordinate (3rd column) then by the 'y' coordinate
>> (2nd column) then finally by the 'x' column. for example:
>
> try Craig Markwardt's multisort
>
> http://astrog.physics.wisc.edu/~craigm/idl/arrays.html#MULTI SORT

I would be interested in seeing a benchmark of the two methods, your
nested loop method and the oddness that is multisort. That is a heck
of a routine...

Cheers,

Brian

Brian Larsen
Boston University
Center for Space Physics

Subject: Re: how to sort data based on other sorted data
Posted by JMB on Thu, 10 Jan 2008 13:52:32 GMT

View Forum Message <> Reply to Message

> I have an input file containing coordinate data (three columns) with a
> couple thousand rows.

Hi Placebo!

What are the minimum and maximum values of your coordinate data?

The idea is to merge your array of coordinates in a single vector that can be sorted. The merging has to be done in a way that x,y,z info doesn't mix and can still be extracted after sorting. Like for the 16777216 colors that can be specified by their R,G,B components.

Example:
color=4700000
Blue=fix(color/2.^16)
Green=fix((color mod 2.^16)/2.^8)
Red=fix((color mod 2.^16) mod 2.^8)
Print,Blue,Green,Red

Jerome

Subject: Re: how to sort data based on other sorted data
Posted by placebo on Thu, 10 Jan 2008 18:23:49 GMT
View Forum Message <> Reply to Message

> What are the minimum and maximum values of your coordinate data?

The range for the data is small, nanometer sizes. My data files represent the coordinates for atom positions in a crystal lattice. The precision in the coordinates goes out to 15 or so decimal places, for ex: 4.271762465783982 nm.

> The idea is to merge your array of coordinates in a single vector that
> can be sorted.

> Example:
> color=4700000
> Blue=fix(color/2.^16)
> Green=fix((color mod 2.^16)/2.^8)
> Red=fix((color mod 2.^16) mod 2.^8)
> Print,Blue,Green,Red

I see how the example works with Integers, but I need to find what "color" would be for each row of data (floating point). I think the "MOD" function may pose a problem here, although I've not played

around with it yet.

Subject: Re: how to sort data based on other sorted data
Posted by JMB on Thu, 10 Jan 2008 18:47:45 GMT
View Forum Message <> Reply to Message

> The idea is to merge your array of coordinates in a single vector that
> can be sorted. The merging has to be done in a way that x,y,z info
> doesn't mix.

You can test the following method that seems to work for integers or
unsigned integers.
[-32768,+32768] or [0,+65535]
It doesn't work with floating points!:

```
;*************************************************

; a xyz matrix is generated for testing:
Rows=50000
xyz=dblarr(3,Rows)

xyz[0,*]=fix((randomu(S,Rows)-0.5)*2.^16)
xyz[1,*]=fix((randomu(S,Rows)-0.5)*2.^16)
xyz[2,*]=fix((randomu(S,Rows)-0.5)*2.^16)

; the method! quite short ;-)

c=xyz[2,*]*2.^32+xyz[1,*]*2.^16+xyz[0,*]
indices=sort(c)

end
;*************************************************
```

You can benchmark with multisort!

Cheers,

Jerome

Subject: Re: how to sort data based on other sorted data
Posted by JMB on Thu, 10 Jan 2008 18:52:12 GMT
View Forum Message <> Reply to Message

> The range for the data is small, nanometer sizes. My data files
> represent the coordinates for atom positions in a crystal lattice.

> The precision in the coordinates goes out to 15 or so decimal places,
> for ex: 4.271762465783982 nm.

but with this precision do you get equal values of z coordinates...
that you need then to sort respect to y and x coordinates?

Jerome

Subject: Re: how to sort data based on other sorted data
Posted by pgrigis on Thu, 10 Jan 2008 19:16:29 GMT
View Forum Message <> Reply to Message

On Jan 10, 1:23 pm, placebo <willie.mad...@gmail.com> wrote:
>> What are the minimum and maximum values of your coordinate data?
>
> The range for the data is small, nanometer sizes. My data files
> represent the coordinates for atom positions in a crystal lattice.
> The precision in the coordinates goes out to 15 or so decimal places,
> for ex: 4.271762465783982 nm.

It seems like you are getting dangerously close to the Planck
scale there, so don't be surprised if your results get a little
fuzzy... ;-)

Cheers,
Paolo

>
>> The idea is to merge your array of coordinates in a single vector that
>> can be sorted.
>> Example:
>> color=4700000
>> Blue=fix(color/2.^16)
>> Green=fix((color mod 2.^16)/2.^8)
>> Red=fix((color mod 2.^16) mod 2.^8)
>> Print,Blue,Green,Red
>
> I see how the example works with Integers, but I need to find what
> "color" would be for each row of data (floating point). I think the
> "MOD" function may pose a problem here, although I've not played
> around with it yet.

Subject: Re: how to sort data based on other sorted data
Posted by placebo on Thu, 10 Jan 2008 20:51:42 GMT

> try Craig Markwardt's multisort
>
> http://astrog.physics.wisc.edu/~craigm/idl/arrays.html#MULTI SORT

Brian,

The multisort method works quite well.

I compared my "nested FOR loop method" to the MULTISORT method and here are some comments:

As it stands now, my FOR loop procedure works with 3 columns, no more no less. The MULTISORT method can take 1 to 9 columns. MULTISORT is more robust, flexible, and user friendly. So, MULTISORT wins in this category.

As far as runtime is concerned, I used SYSTIME at the beginning and end of each routine to benchmark. I ran the program a couple of times for each sorting routine. The benchmarks are listed below:

FOR loop runtimes (seconds):
0.078000069
0.077999830
0.062999964
0.078000069

MULTISORT runtimes (seconds):
0.094000101
0.092999935
0.108999970
0.094000101

As you can see, the FOR loop method is "slightly" faster than the MULTISORT method.
Both routines sorted the same data set containing 931 lines of x,y,z coordinates.
I performed the calculation on my HP WinXP (SP2) laptop with an Intel T2050 and 1 gig ram.
As far as I am concerned, the robustness of MULTISORT definitely outweighs the few milliseconds of lost time.

It would be interesting, however, to compare the benchmarks of larger files and also files with more columns. My guess is MULTISORT would eventually outperform my FOR loop if calculations involved 4 or more columns, since each column requires you to nest an additional FOR loop in my code.

If anyone is interested in pursuing this task, I'd be happy to post my
FOR loop for you guys to play with.

Thanks wlandsman for the MULTISORT tip!

Subject: Re: how to sort data based on other sorted data
Posted by Tom McGlynn on Fri, 11 Jan 2008 17:53:21 GMT
View Forum Message <> Reply to Message

On Jan 10, 3:51 pm, placebo <willie.mad...@gmail.com> wrote:
>> try Craig Markwardt's multisort
>
>> http://astrog.physics.wisc.edu/~craigm/idl/arrays.html#MULTI SORT
>
> Brian,
>
> The multisort method works quite well.
>
...

Multisort works fine and knowing Craig will work very robustly
but I don't think you need to have any limit on
the number of columns. Below is a routine that should
be able to handle an arbitrary number of columns and rows...
I tried it on a 500000 row, 5 column structure -- with each field
a random int between 0 and 29. It ran a bit faster than multisort
(38 s versus 56 s) but gave identical results.
Accommodating different directions for sorting
the different columns should be straightforward. You just need to
invert the index at the appropriate point.

In this case I've had the user organize the input as a structure
where the columns are the sort fields, but they could
just as easily be separate arguments as in multisort. I believe
the columns can be any sortable type.

One thing it does is check if it needs to sort by the next column
or if the sort order is fully determined by the columns already
processed.

The bit I like is the second call to ndistinct which collapses
the maximum values that the key (fullIndex) can have from nrow^2 back
to nrow.
Without something like this the algorithm would either need to
use a big string to accumulate the index (I think that's what
multisort
does) or suffer exponential growth in the indices requiring

limiting nrow^ncol to < 2^64. In principle I think you can sort up
to 2^31.5 rows with this algorithm which is probably enough for most
of us.

Note that I've just put this together this morning, so I wouldn't
be surprised if I've missed some edge cases (e.g., I believe it will
fail with arrays of length 1).

Regards,
Tom McGlynn

```
; This function takes an array and
; returns the number of transitions
; (i.e., x[i] ne x[i-1]) before the current
; element.  It returns a array one shorter
; than the input.  If the original array
; is sorted it returns the number of distinct
; entries before the current entry.

function ndistinct, x
    n = n_elements(x)
    chng = long(x[0:n-2] ne x[1:n-1])
    return, long(total(chng, /cumulative))
end

;+ Main routine
;  Usage: sortIndex = bigsort(userStructure)
;-
function bigsort, x

    nrow = n_elements(x)
    ncol = n_tags(x)

    longlong  = nrow gt 40000 ; rougly sqrt(2^31)
    fullIndex = lonarr(nrow)

    fullIndex[*] = 0

    if (longlong) then begin
        fullIndex = long64(fullIndex)
    endif
```

```
   for i=0, ncol-1 do begin

       ; Get the column order for a column.
       index  = sort(x.(i))

 ; Now see if everything is distinguished.
 cum = ndistinct(x[index].(i))

 ; This creates the index for all columns so far
 if i gt 0 then begin
    fullIndex  = fullIndex*nrow
 endif
 fullIndex[index[1:*]] = fullIndex[index[1:*]] + cum

 ; Sort by the index so far.
       index = sort(fullIndex)
 cum   = ndistinct(fullIndex[index])
 fullIndex[index[0]]        = 0
 fullIndex[index[1:nrow-1]] = cum

 if fullIndex[index[nrow-1]] eq nrow-1 then begin
          ; All rows are distinct, so we're done.
   return, index
 endif
    endfor

    return, index
end
```