Subject: Re: subscript array question
Posted by eddie haskell on Thu, 11 Feb 1999 08:00:00 GMT
View Forum Message <> Reply to Message

```
>      I'm using IDL 5.0 and need to be able to use a subscript
>  array containing duplicate values like this:
>
>  array = intarr(5)
>  subs  = [0,2,4,4]
>  array[subs] = array[subs] + 1
>
>  and have the resulting values for array be:
>
>        1      0      1      0      2
```

How about something like this:

```
IDL> array = intarr(9)
IDL> subs = [2,3,4,2,4,4,7,5]
IDL> array[min(subs):max(subs)] = array[min(subs):max(subs)] +
histogram(subs)
IDL> print, array
       0       0       2       1       3       1       0       1       0
```

I checked it with arrays up to a size of findgen(100000) and it runs
without
any noticeable time delays. I have not, however, done any error
checking,
i.e., if subs contain elements outside of array, or any real checking of
any
sort for that matter. :-) HTH

cheers,
eddie

```
----- ---- --- ---  ---- --- --  --- --- --  -- - -  -  -
|\              A G Edward Haskell
|\    Center for Coastal Physical Oceanography
|\   Old Dominion University, Norfolk VA  23529
|\     Voice 757.683.4816     Fax 757.683.5550
|\          e-mail  haskell*ccpo.odu.edu
----- ---- --- ----  --- --- ---  --- -- --  -- - - -  -   -
```

Subject: Re: subscript array question
Posted by David Ritscher on Thu, 11 Feb 1999 08:00:00 GMT
View Forum Message <> Reply to Message

```
>  array = intarr(5)
```

> subs = [0,2,4,4]
> array[subs] = array[subs] + 1
>
> and have the resulting values for array be:
>
```
>        1      0      1      0      2
```
>
> Because of the way IDL manages memory for expression evaluation
> and assignments, what happens for the last two elements of the
> addition is that the original value of array[4] is used twice,
> rather than what I want, which is to use the current value of
> array[4] each time. I.e. IDL gives the resulting values for
> array to be:
>
```
>        1      0      1      0      1
```
>

With my version, IDL Version 5.1.1, I get the latter, not the former!
I suspect it is true with your version, as well. What IDL does is,
for the duplicate subscript, it does the operation twice, but since
'array' on the right hand of the expression is a copy of the original,
it goes and gets the same '0' twice, incremnts it by '1', and inserts
it into the same location twice.

If in your actual application you're having similar problems, the
uniq function might help you out:

```
array = intarr(5)
subs  = [0,2,4,4]

subs = subs(uniq(subs, sort(subs)))

print, subs
       0      2      4


array[subs] = array[subs] + 1
```

In this case, it gives the same result, as I explained, but in your
application, it might serve to solve your problem.

Good luck,

David Ritscher

--
Cardiac Rhythm Management Laboratory
Department of Medicine
University of Alabama at Birmingham
B168 Volker Hall - 1670 University Boulevard
Birmingham AL 35294-0019
Tel: (205) 975-2122   Fax: (205) 975-4720
Email: david.ritscher@bigfoot.com

Subject: Re: subscript array question
Posted by steinhh on Fri, 12 Feb 1999 08:00:00 GMT
View Forum Message <> Reply to Message

In article <7a0j1q$mvb$1@news.NERO.NET> bennetsc@ucs.orst.edu
(Scott Bennett) writes:

[..snip histogram solution, among other things..]

> That sure looks ingeniously devious to me. I had to try out all
> the pieces to see how it worked. :-)

I agree - almost sinister - a big contender for Hi-Tech Tip of the
year (and it's still just February!).

> However, I couldn't get my 2D
> case to perform well. I'm omitting here some non-essentials, but the
> routine originally had this in it:
>
[..]
>   ths[thsubs,ssubs] = ths[thsubs,ssubs] + llvol
[..]
>
> Written like that, it ran in ~15 seconds on my test data set, but gave
> values in ths that were often too small, as I originally posted.

[..loop version taking ~46 seconds omitted..]

[..hist_2d version taking 37 *minutes* omitted...]

What you ought to try instead is to calculate the one-dimensional
index values from the two-dimensional indices:

```
   subs = thsubs + ssubs * (size(ths))(1)
```

And then just plug it into the original scheme:

```
  ths[min(subs):max(subs)] = ths[min(subs):max(subs)] +histogram(subs)
```

On a general note, if "subs" covers the array very sparsely, the
histogram method is not necessarily faster than the loop version (as a
limiting case, consider a huge array, and you want to add 1 to the
first and last element only - the histogram is just as huge as the
array, and a lot of time will be spent adding zeros to the array!)

Regards,

Stein Vidar

Subject: Re: subscript array question
Posted by bennetsc on Fri, 12 Feb 1999 08:00:00 GMT
View Forum Message <> Reply to Message

In article <36C2B49A.204E@bigfoot.com>, David Ritscher wrote:

```
>>  array = intarr(5)
>>  subs  = [0,2,4,4]
>>  array[subs] = array[subs] + 1
>>
>>  and have the resulting values for array be:
>>
>>         1       0       1       0       2
>>
>>  Because of the way IDL manages memory for expression evaluation
>>  and assignments, what happens for the last two elements of the
>>  addition is that the original value of array[4] is used twice,
>>  rather than what I want, which is to use the current value of
>>  array[4] each time.  I.e. IDL gives the resulting values for
>>  array to be:
>>
>>         1       0       1       0       1
>>
>
>
>
> With my version, IDL Version 5.1.1, I get the latter, not the former!
```

Correct.

> I suspect it is true with your version, as well. What IDL does is,
> for the duplicate subscript, it does the operation twice, but since
> 'array' on the right hand of the expression is a copy of the original,
> it goes and gets the same '0' twice, incremnts it by '1', and inserts

> it into the same location twice.

Yes, that's what I figured was happening, too. For that reason,
I'd be very surprised if the behavior were to change from one version
of IDL to another.
>
> If in your actual application you're having similar problems, the
> uniq function might help you out:
>
> array = intarr(5)
> subs = [0,2,4,4]
>
> subs = subs(uniq(subs, sort(subs)))
>
> print, subs
>   0   2   4
>
>
> array[subs] = array[subs] + 1
>
> In this case, it gives the same result, as I explained, but in your
> application, it might serve to solve your problem.

Yes, it gives the same result, which is not what I want.

In article <36C2EFD5.8D76D36@no.spam.edu>,
eddie haskell <haskell@no.spam.edu> wrote:
>>   I'm using IDL 5.0 and need to be able to use a subscript
>> array containing duplicate values like this:
>>
>> array = intarr(5)
>> subs = [0,2,4,4]
>> array[subs] = array[subs] + 1
>>
>> and have the resulting values for array be:
>>
>>   1   0   1   0   2
>
> How about something like this:
> IDL> array = intarr(9)
> IDL> subs = [2,3,4,2,4,4,7,5]
> IDL> array[min(subs):max(subs)] = array[min(subs):max(subs)] +
> histogram(subs)
> IDL> print, array
>   0   0   2   1   3   1   0   1   0
>
> I checked it with arrays up to a size of findgen(100000) and it runs
> without

> any noticeable time delays. I have not, however, done any error
> checking,
> i.e., if subs contain elements outside of array, or any real checking of
> any
> sort for that matter. :-) HTH
>
    That sure looks ingeniously devious to me. I had to try out all
the pieces to see how it worked. :-) However, I couldn't get my 2D
case to perform well. I'm omitting here some non-essentials, but the
routine originally had this in it:

```
 llsubs = where(llthetaindex ne lmissing, llcnt)
 if llcnt gt 0 then begin
     llvol = latlonvol[j,k] ; get cell volume at this latitude
;    Add cell volumes to appropriate table entries
     thsubs = llthetaindex[llsubs]
     ssubs = llsindex[llsubs]
     ths[thsubs,ssubs] = ths[thsubs,ssubs] + llvol
 endif
```

Written like that, it ran in ~15 seconds on my test data set, but gave
values in ths that were often too small, as I originally posted. I no
longer have the number handy, but a "print,total(ths)" showed a result
that was only about 28-30% of the correct total. So I replaced the
last assignment statement with:

```
     for ll = 0, llcnt - 1 do    $
  ths[thsubs[ll],ssubs[ll]] = ths[thsubs[ll],ssubs[ll]] +$
   llvol
```

(Sorry about the terribly wide lines!) This takes ~46 seconds to run,
but does give the correct results. A "print,total(ths)" gives the
correct total of 1.32526e+18.
    After looking at your 1D example, I read the description in the
_IDL_Reference_Guide_ of hist_2d and tried replacing the for loop with:

```
     thmaxsub = max(thsubs)
     smaxsub = max(ssubs)
     ths[0:thmaxsub,0:smaxsub] = ths[0:thmaxsub,0:smaxsub] + $
      float(hist_2d(thsubs,ssubs)) * llvol
```

A "print,total(ths)" with this method also shows 1.32526e+18, which is
correct, but it took ~37 minutes 58 seconds to run! So I guess I'll
stick with the for loop for now. :-(
    Many thanks to both of you for your replies. Once again IDL has
provided me a "learning experience."

Scott Bennett, Comm. ASMELG, CFIAG
Dept. of Atmospheric Sciences
Oregon State University
Corvallis, Oregon 97331
*** **********
* Internet: sbennett@oce.orst.edu   *
-- ---------
* "The jury has a right to judge both the law as well as the fact in *
* controversy."--John Jay, First Chief Justice, U.S. Supreme Court   *
* in Georgia vs. Brailsford, 1794   *
*** **********