Subject: Adding x,y events to a 2d array (quickly)
Posted by oliver[1] on Thu, 07 Nov 2013 12:45:37 GMT
View Forum Message <> Reply to Message

Hi

This may be a much answered question, but searching for an answer has failed me.

I have 3 (very large) arrays giving x values, y values and energy values.

I wish to create two 2d arrays - one of total (summed) energy for a particular x,y value, and one of total counts per x,y value.

An example of what I tried is below:

```
x=[1,1,2]
y=[1,1,2]
e=[10,10,10]
```

To create the 'counts' value, i used the following:

```
counts=fltarr(5,5)

counts(x,y)++
```

This works. You end up with a value of 2 at position(1,1) and a value of 1 at position (2,2).

I hoped to get the 'total energy' value by doing the following:

```
totalenergy=fltarr(5,5)

totalenergy(x,y)+=e
```

However, this does not work. The final array only contains the last energy value added at each point.

Is there an IDL trick I'm missing that allows you to incrementally add values to an array quickly?

Thanks

Oliver

Subject: Re: Adding x,y events to a 2d array (quickly)
Posted by Russell Ryan on Thu, 07 Nov 2013 14:10:10 GMT
View Forum Message <> Reply to Message

I read your post several times, and I guess I'm not sure what you're after. That said, I've got a good hunch that you're going to need histogram and the reverse_indices output. If so, then you're actually gonna want hist_nd by JD Smith.

-Russell

Subject: Re: Adding x,y events to a 2d array (quickly)
Posted by Dick Jackson on Thu, 07 Nov 2013 19:27:00 GMT

View Forum Message <> Reply to Message

Oliver,

You have a good question, and I think this code illustrates it a little more
plainly, starting each time with an array of zero values:

```
counts=fltarr(3,3)
counts[[1,1,2],[1,1,2]] ++
Print, 'counts[[1,1,2],[1,1,2]] ++:'
Print, counts

counts=fltarr(3,3)
counts[[1,1,2],[1,1,2]] += 1
Print, 'counts[[1,1,2],[1,1,2]] += 1:'
Print, counts

counts=fltarr(3,3)
counts[[1,1,2],[1,1,2]] += [1,1,1]
Print, 'counts[[1,1,2],[1,1,2]] += [1,1,1]:'
Print, counts

counts=fltarr(3,3)
counts[[1,1,2],[1,1,2]] += [10,20,30]
Print, 'counts[[1,1,2],[1,1,2]] += [10,20,30]:'
Print, counts
```

The result of this is:

```
counts[[1,1,2],[1,1,2]] ++:
     0.000000     0.000000     0.000000
     0.000000      2.00000     0.000000
     0.000000     0.000000      1.00000
counts[[1,1,2],[1,1,2]] += 1:
     0.000000     0.000000     0.000000
     0.000000      1.00000     0.000000
     0.000000     0.000000      1.00000
counts[[1,1,2],[1,1,2]] += [1,1,1]:
     0.000000     0.000000     0.000000
     0.000000      1.00000     0.000000
     0.000000     0.000000      1.00000
counts[[1,1,2],[1,1,2]] += [10,20,30]:
     0.000000     0.000000     0.000000
     0.000000      20.0000     0.000000
     0.000000     0.000000      30.0000
```

It seems that ++ increments for each (x,y) pair as you expect. However, the +=
operation seems to be creating a set of result values by adding the set of
original values to the given scalar or vector, and then copying the results into
the array. In this way, when [1,1] is assigned values twice by this copying,
only the last value persists.

I seem to recall someone explaining this behaviour before, and thanks to
Russell, I realize one good way of getting *part* of what you (reasonably!) want
to do. If all of your 'e' values were equal, then you can find how many counts
of each (x,y) pair exist by using Hist_ND:
(http://tir.astro.utoledo.edu/idl/hist_nd.pro)

```
IDL> Print, Hist_ND(Transpose([[1,1,2],[1,1,2]]), 1, Min=0)
           0           0           0
           0           2           0
           0           0           1
```

But, in general, to add a varying set of 'e' values to those (x,y) locations...
I have to think a bit...

Cheers,
-Dick

Dick Jackson Software Consulting
Victoria, BC, Canada
www.d-jackson.com

oliver wrote, On 2013-11-07, 4:45am:
> Hi
>
> This may be a much answered question, but searching for an answer has failed me.
>
> I have 3 (very large) arrays giving x values, y values and energy values.
>
> I wish to create two 2d arrays - one of total (summed) energy for a particular x,y value, and one of total counts per x,y value.
>
> An example of what I tried is below:
>
> x=[1,1,2]
> y=[1,1,2]
> e=[10,10,10]

>
> To create the 'counts' value, i used the following:
>
> counts=fltarr(5,5)
>
> counts(x,y)++
>
> This works. You end up with a value of 2 at position(1,1) and a value of 1 at position (2,2).
>
> I hoped to get the 'total energy' value by doing the following:
>
> totalenergy=fltarr(5,5)
>
> totalenergy(x,y)+=e
>
> However, this does not work. The final array only contains the last energy value added at each point.
>
> Is there an IDL trick I'm missing that allows you to incrementally add values to an array quickly?
>
> Thanks
>
> Oliver
>

--

Cheers,
-Dick

Dick Jackson Software Consulting
Victoria, BC, Canada
www.d-jackson.com

Subject: Re: Adding x,y events to a 2d array (quickly)
Posted by Phillip Bitzer on Thu, 07 Nov 2013 20:16:48 GMT
View Forum Message <> Reply to Message

On Thursday, November 7, 2013 1:27:00 PM UTC-6, Dick Jackson wrote:

> I seem to recall someone explaining this behaviour before, and thanks to
>
> Russell, I realize one good way of getting *part* of what you (reasonably!) want
>
> to do. If all of your 'e' values were equal, then you can find how many counts
>

> of each (x,y) pair exist by using Hist_ND:
>
> (http://tir.astro.utoledo.edu/idl/hist_nd.pro)
>
> IDL> Print, Hist_ND(Transpose([[1,1,2],[1,1,2]]), 1, Min=0)
>
> But, in general, to add a varying set of 'e' values to those (x,y) locations...
>
> I have to think a bit...
>

I've got you covered....

Oliver, reverse indices are your friend here, as Russell alluded to. Get the two-dimensional histogram, slightly modified from Dick's version:

```
h = HIST_ND( [ TRANSPOSE(x), TRANSPOSE(y) ], 1, MIN=0, REVERSE_INDICES=ri )
```

Since you said you have large arrays, I transpose each individually, and then concatenate.

Now, go through the reverse indices:

```
totalE = FLTARR(SIZE(h, /DIM))
FOR i=0, N_ELEMENTS(h)-1 do if h[i] GT 0 THEN totalE[i]= TOTAL( e[ri[ri[i]:ri[i+1]-1]])

print, totalE
     0.00000      0.00000      0.00000
     0.00000      20.0000      0.00000
     0.00000      0.00000      10.0000
```

This is the basic idea. It can be sped up by only looping over the elements of h with non-zero counts (as opposed to "skipping" them as I did here).

Here's some highly recommended reading on histograms:
http://www.idlcoyote.com/tips/histogram_tutorial.html

Subject: Re: Adding x,y events to a 2d array (quickly)
Posted by Dick Jackson on Thu, 07 Nov 2013 22:20:15 GMT

View Forum Message <> Reply to Message

Phillip Bitzer wrote, On 2013-11-07, 12:16pm:
> On Thursday, November 7, 2013 1:27:00 PM UTC-6, Dick Jackson wrote:
>
>> I seem to recall someone explaining this behaviour before, and thanks to
>>
>> Russell, I realize one good way of getting *part* of what you (reasonably!) want
>>

>> to do. If all of your 'e' values were equal, then you can find how many counts
>>
>> of each (x,y) pair exist by using Hist_ND:
>>
>> (http://tir.astro.utoledo.edu/idl/hist_nd.pro)
>>
>> IDL> Print, Hist_ND(Transpose([[1,1,2],[1,1,2]]), 1, Min=0)
>>
>> But, in general, to add a varying set of 'e' values to those (x,y) locations...
>>
>> I have to think a bit...
>>
>
> I've got you covered....
>
> Oliver, reverse indices are your friend here, as Russell alluded to. Get the two-dimensional histogram, slightly modified from Dick's version:
>
> h = HIST_ND([TRANSPOSE(x), TRANSPOSE(y)], 1, MIN=0, REVERSE_INDICES=ri)
>
> Since you said you have large arrays, I transpose each individually, and then concatenate.
>
> Now, go through the reverse indices:
>
> totalE = FLTARR(SIZE(h, /DIM))
> FOR i=0, N_ELEMENTS(h)-1 do if h[i] GT 0 THEN totalE[i]= TOTAL(e[ri[ri[i]:ri[i+1]-1]])
>
> print, totalE
```
>       0.00000      0.00000      0.00000
>       0.00000      20.0000      0.00000
>       0.00000      0.00000      10.0000
```
>
> This is the basic idea. It can be sped up by only looping over the elements of h with non-zero counts (as opposed to "skipping" them as I did here).
>
> Here's some highly recommended reading on histograms: http://www.idlcoyote.com/tips/histogram_tutorial.html

Histograms and reverse-indices are amazingly powerful and the right way to go in
many tough problems, but I think Oliver is looking for a solution avoiding loops
(I am too!). If a loop solution were OK, the last block here would be more
direct, with no need for histograms:

```
x=[1,1,2]
y=[1,1,2]
e=[10,11,12]

counts=fltarr(3,3)
```

```
counts(x,y)++
Print, 'counts:'
Print, counts    ; Shows that three increments by 1 were done

totalenergy=fltarr(3,3)
totalenergy(x,y)+=e
Print, 'totalenergy:'
Print, totalenergy ; It appears that only two increments by 10 were done

totalenergy2=fltarr(3,3)
FOR i=0, N_Elements(x)-1 DO totalenergy2(x[i],y[i])+=e[i]
Print, 'totalenergy2:'
Print, totalenergy2 ; All three increments were done
```

... which gives us:

```
counts:
     0.000000     0.000000     0.000000
     0.000000      2.00000     0.000000
     0.000000     0.000000      1.00000
totalenergy:
     0.000000     0.000000     0.000000
     0.000000      11.0000     0.000000
     0.000000     0.000000      12.0000
totalenergy2:
     0.000000     0.000000     0.000000
     0.000000      21.0000     0.000000
     0.000000     0.000000      12.0000
```

Still looking for the "IDL way" (read: "ideal way") to do this...

--

Cheers,
-Dick

Dick Jackson Software Consulting
Victoria, BC, Canada
www.d-jackson.com

Subject: Re: Adding x,y events to a 2d array (quickly)
Posted by Michael Galloy on Thu, 07 Nov 2013 23:18:19 GMT
View Forum Message <> Reply to Message

On 11/7/13, 3:20 PM, Dick Jackson wrote:
> Phillip Bitzer wrote, On 2013-11-07, 12:16pm:
>> On Thursday, November 7, 2013 1:27:00 PM UTC-6, Dick Jackson wrote:

>>
>>> I seem to recall someone explaining this behaviour before, and thanks to
>>>
>>> Russell, I realize one good way of getting *part* of what you
>>> (reasonably!) want
>>>
>>> to do. If all of your 'e' values were equal, then you can find how
>>> many counts
>>>
>>> of each (x,y) pair exist by using Hist_ND:
>>>
>>> (http://tir.astro.utoledo.edu/idl/hist_nd.pro)
>>>
>>> IDL> Print, Hist_ND(Transpose([[1,1,2],[1,1,2]]), 1, Min=0)
>>>
>>> But, in general, to add a varying set of 'e' values to those (x,y)
>>> locations...
>>>
>>> I have to think a bit...
>>>
>>
>> I've got you covered....
>>
>> Oliver, reverse indices are your friend here, as Russell alluded to.
>> Get the two-dimensional histogram, slightly modified from Dick's version:
>>
>> h = HIST_ND([TRANSPOSE(x), TRANSPOSE(y)], 1, MIN=0,
>> REVERSE_INDICES=ri)
>>
>> Since you said you have large arrays, I transpose each individually,
>> and then concatenate.
>>
>> Now, go through the reverse indices:
>>
>> totalE = FLTARR(SIZE(h, /DIM))
>> FOR i=0, N_ELEMENTS(h)-1 do if h[i] GT 0 THEN totalE[i]= TOTAL(
>> e[ri[ri[i]:ri[i+1]-1]])
>>
>> print, totalE
```
>>       0.00000      0.00000      0.00000
>>       0.00000      20.0000      0.00000
>>       0.00000      0.00000      10.0000
```
>>
>> This is the basic idea. It can be sped up by only looping over the
>> elements of h with non-zero counts (as opposed to "skipping" them as I
>> did here).
>>
>> Here's some highly recommended reading on histograms:

```
>>  http://www.idlcoyote.com/tips/histogram_tutorial.html
>
>  Histograms and reverse-indices are amazingly powerful and the right way
>  to go in many tough problems, but I think Oliver is looking for a
>  solution avoiding loops (I am too!). If a loop solution were OK, the
>  last block here would be more direct, with no need for histograms:
>
>  x=[1,1,2]
>  y=[1,1,2]
>  e=[10,11,12]
>
>  counts=fltarr(3,3)
>  counts(x,y)++
>  Print, 'counts:'
>  Print, counts    ; Shows that three increments by 1 were done
>
>  totalenergy=fltarr(3,3)
>  totalenergy(x,y)+=e
>  Print, 'totalenergy:'
>  Print, totalenergy ; It appears that only two increments by 10 were done
>
>  totalenergy2=fltarr(3,3)
>  FOR i=0, N_Elements(x)-1 DO totalenergy2(x[i],y[i])+=e[i]
>  Print, 'totalenergy2:'
>  Print, totalenergy2 ; All three increments were done
>
>  ... which gives us:
>
>  counts:
>       0.000000     0.000000     0.000000
>       0.000000      2.00000     0.000000
>       0.000000     0.000000      1.00000
>  totalenergy:
>       0.000000     0.000000     0.000000
>       0.000000      11.0000     0.000000
>       0.000000     0.000000      12.0000
>  totalenergy2:
>       0.000000     0.000000     0.000000
>       0.000000      21.0000     0.000000
>       0.000000     0.000000      12.0000
>
>  Still looking for the "IDL way" (read: "ideal way") to do this...
>
```

Phillip's method loops over the bins in the histogram, so should be reasonable. My MG_HIST_ND does the same thing:

```
IDL> x = [1, 1, 2]
IDL> y = [1, 1, 2]
IDL> weights = [10., 11., 12.]
IDL>
IDL> h = mg_hist_nd([transpose(x), transpose(x)], weights=weights,
min=0, bin_size=1, unweighted=unweighted)
IDL> print, h
      0.00000      0.00000      0.00000
      0.00000      21.0000      0.00000
      0.00000      0.00000      12.0000
IDL> print, unweighted
           0           0           0
           0           2           0
           0           0           1
```

Get MG_HIST_ND on GitHub:

https://github.com/mgalloy/mglib/blob/master/src/analysis/mg _hist_nd.pro

Mike
--
Michael Galloy
www.michaelgalloy.com
Modern IDL: A Guide to IDL Programming (http://modernidl.idldev.com)
Research Mathematician
Tech-X Corporation

Subject: Re: Adding x,y events to a 2d array (quickly)
Posted by Dick Jackson on Fri, 08 Nov 2013 08:22:22 GMT

View Forum Message <> Reply to Message

Michael Galloy wrote, On 2013-11-07, 3:18pm:
> On 11/7/13, 3:20 PM, Dick Jackson wrote:
>> Phillip Bitzer wrote, On 2013-11-07, 12:16pm:
>>> On Thursday, November 7, 2013 1:27:00 PM UTC-6, Dick Jackson wrote:
>>>
>>>> I seem to recall someone explaining this behaviour before, and thanks to
>>>>
>>>> Russell, I realize one good way of getting *part* of what you
>>>> (reasonably!) want
>>>>
>>>> to do. If all of your 'e' values were equal, then you can find how
>>>> many counts
>>>>
>>>> of each (x,y) pair exist by using Hist_ND:
>>>>
>>>> (http://tir.astro.utoledo.edu/idl/hist_nd.pro)
>>>>

```
>>>>  IDL> Print, Hist_ND(Transpose([[1,1,2],[1,1,2]]), 1, Min=0)
>>>>
>>>>  But, in general, to add a varying set of 'e' values to those (x,y)
>>>>  locations...
>>>>
>>>>  I have to think a bit...
>>>>
>>>
>>>  I've got you covered....
>>>
>>>  Oliver, reverse indices are your friend here, as Russell alluded to.
>>>  Get the two-dimensional histogram, slightly modified from Dick's version:
>>>
>>>  h = HIST_ND( [ TRANSPOSE(x), TRANSPOSE(y) ], 1, MIN=0,
>>>  REVERSE_INDICES=ri )
>>>
>>>  Since you said you have large arrays, I transpose each individually,
>>>  and then concatenate.
>>>
>>>  Now, go through the reverse indices:
>>>
>>>  totalE = FLTARR(SIZE(h, /DIM))
>>>  FOR i=0, N_ELEMENTS(h)-1 do if h[i] GT 0 THEN totalE[i]= TOTAL(
>>>  e[ri[ri[i]:ri[i+1]-1]])
>>>
>>>  print, totalE
>>>        0.00000     0.00000     0.00000
>>>        0.00000     20.0000     0.00000
>>>        0.00000     0.00000     10.0000
>>>
>>>  This is the basic idea. It can be sped up by only looping over the
>>>  elements of h with non-zero counts (as opposed to "skipping" them as I
>>>  did here).
>>>
>>>  Here's some highly recommended reading on histograms:
>>>  http://www.idlcoyote.com/tips/histogram_tutorial.html
>>
>>  Histograms and reverse-indices are amazingly powerful and the right way
>>  to go in many tough problems, but I think Oliver is looking for a
>>  solution avoiding loops (I am too!). If a loop solution were OK, the
>>  last block here would be more direct, with no need for histograms:
>>
>>  x=[1,1,2]
>>  y=[1,1,2]
>>  e=[10,11,12]
>>
>>  counts=fltarr(3,3)
>>  counts(x,y)++
>>  Print, 'counts:'
>>  Print, counts    ; Shows that three increments by 1 were done
>>
>>  totalenergy=fltarr(3,3)
>>  totalenergy(x,y)+=e
>>  Print, 'totalenergy:'
>>  Print, totalenergy ; It appears that only two increments by 10 were done
>>
>>  totalenergy2=fltarr(3,3)
>>  FOR i=0, N_Elements(x)-1 DO totalenergy2(x[i],y[i])+=e[i]
>>  Print, 'totalenergy2:'
>>  Print, totalenergy2 ; All three increments were done
>>
>>  ... which gives us:
>>
>>  counts:
>>       0.000000     0.000000     0.000000
>>       0.000000      2.00000     0.000000
>>       0.000000     0.000000      1.00000
>>  totalenergy:
>>       0.000000     0.000000     0.000000
>>       0.000000      11.0000     0.000000
>>       0.000000     0.000000      12.0000
>>  totalenergy2:
>>       0.000000     0.000000     0.000000
>>       0.000000      21.0000     0.000000
>>       0.000000     0.000000      12.0000
>>
>>  Still looking for the "IDL way" (read: "ideal way") to do this...
>>
>
>  Phillip's method loops over the bins in the histogram, so should be reasonable.
>  My MG_HIST_ND does the same thing:
>
>  IDL> x = [1, 1, 2]
>  IDL> y = [1, 1, 2]
>  IDL> weights = [10., 11., 12.]
>  IDL>
>  IDL> h = mg_hist_nd([transpose(x), transpose(x)], weights=weights, min=0,
>  bin_size=1, unweighted=unweighted)
>  IDL> print, h
>       0.00000      0.00000      0.00000
>       0.00000      21.0000      0.00000
>       0.00000      0.00000      12.0000
>  IDL> print, unweighted
>            0           0           0
>            0           2           0
>            0           0           1
```

>
> Get MG_HIST_ND on GitHub:
>
>   https://github.com/mgalloy/mglib/blob/master/src/analysis/mg _hist_nd.pro
>
> Mike

Mike,

That's an amazing routine (thank you!), and the Weights option provides exactly the functionality and result Oliver is looking for. However, in most of my test cases it seems to be less efficient than the simple loop in time or space. (it's good when the 2-D counts array is small and you don't mind using lots of memory) Here's my test, with no idea if it covers similar scale to Oliver's application!

```
PRO IncrementTest

FOREACH size, [100, 1000, 10000] DO $ ; Width, height of (square) counts array
   FOREACH nPts, [1E6, 1E7, 4E7] DO BEGIN ; Number of points to create

   x = Long(RandomU(42L, nPts) * size)
   y = Long(RandomU(56L, nPts) * size)
   e = Long(RandomU(98L, nPts) * 10) + 1 ; Random from 1-10

   Print
   Print
   Help, size, nPts

   Print
   Print, 'MG_Hist_ND method'
   m0 = Memory(/Current)
   Tic
   countsMGhistND = mg_hist_nd([transpose(x), transpose(y)], weights=e, $
                           min=0, bin_size=1) ; , unweighted=unweighted)
   Toc
   Print, (Memory(/Highwater)-m0)/(1024.^2), ' MB used'

   Print
   Print, 'Loop method'
   m0 = Memory(/Current)
   Tic
   countsLoop = LonArr(size, size) ; FltArr(size, size)
   FOR i=0, N_Elements(x)-1 DO countsLoop(x[i],y[i]) += e[i]
   Toc
   Print, (Memory(/Highwater)-m0)/(1024.^2), ' MB used'

   Print
   Print, 'Results are ' + $
     (Array_Equal(countsLoop, countsMGhistND) ? '' : 'not ') + 'equal!'

ENDFOREACH

END
```

... and results, with "better" values labeled with "*****" :

```
IDL> incrementtest
% Compiled module: INCREMENTTEST.


SIZE           INT       =      100
NPTS            FLOAT     =  1.00000e+006

MG_Hist_ND method
% Time elapsed: 0.23399997 seconds. *****
      19.0744 MB used

Loop method
% Time elapsed: 0.35899997 seconds.
    0.0382977 MB used              *****

Results are equal!


SIZE           INT       =      100
NPTS            FLOAT     =  1.00000e+007

MG_Hist_ND method
% Time elapsed: 2.3920002 seconds. *****
      190.736 MB used

Loop method
% Time elapsed: 3.4849999 seconds.
    0.0382271 MB used             *****

Results are equal!


SIZE           INT       =      100
NPTS            FLOAT     =  4.00000e+007

MG_Hist_ND method
% Time elapsed: 10.017000 seconds. *****
```

```
       762.940 MB used

Loop method
% Time elapsed: 14.156000 seconds.
     0.0382271 MB used              *****

Results are equal!


SIZE            INT       =     1000
NPTS             FLOAT     =  1.00000e+006

MG_Hist_ND method
% Time elapsed: 1.3910000 seconds.
       26.7042 MB used

Loop method
% Time elapsed: 0.51600003 seconds. *****
       3.81478 MB used              *****

Results are equal!


SIZE            INT       =     1000
NPTS             FLOAT     =  1.00000e+007

MG_Hist_ND method
% Time elapsed: 5.6570001 seconds.
       190.736 MB used

Loop method
% Time elapsed: 4.1250000 seconds. *****
       3.81478 MB used             *****

Results are equal!


SIZE            INT       =     1000
NPTS             FLOAT     =  4.00000e+007

MG_Hist_ND method
% Time elapsed: 16.332000 seconds.
       762.940 MB used

Loop method
% Time elapsed: 16.274000 seconds. *****
       3.81478 MB used             *****
```

```
Results are equal!


SIZE          INT      =    10000
NPTS           FLOAT     =  1.00000e+006

MG_Hist_ND method
% Time elapsed: 18.353000 seconds.
     1159.67 MB used

Loop method
% Time elapsed: 1.5000000 seconds. *****
     381.470 MB used             *****

Results are equal!


SIZE          INT      =    10000
NPTS           FLOAT     =  1.00000e+007

MG_Hist_ND method
% Unable to allocate memory: to make array.
  Not enough space
```

Oliver, I hope this helps!

--

Cheers,
-Dick

Dick Jackson Software Consulting
Victoria, BC, Canada
www.d-jackson.com

Subject: Re: Adding x,y events to a 2d array (quickly)
Posted by oliver[1] on Fri, 08 Nov 2013 09:19:11 GMT
View Forum Message <> Reply to Message

Thanks for all the replies - need time to digest last one, thanks!

Dick was right in that I was trying to avoid loops.

I get the required result using

foreach element,EnergyArray,index DO Image(XArray(index),YArray(index))+=element

but it takes ~30 seconds for the array size I am using.

Image(XArray,YArray)+=EnergyArray

takes only 1 second to run, but doesn't give the result I expected.

As I said, I haven't fully gone through Dick's last message - I just wanted to say thanks for efforts!

Oliver

Subject: Re: Adding x,y events to a 2d array (quickly)
Posted by oliver[1] on Fri, 08 Nov 2013 14:18:26 GMT
View Forum Message <> Reply to Message

Hi again - thanks for all replies especially Dick's last with all the timings, which allows me to sheepishly admit that I solved the speed problem, but not how I expected to!

Looking at the timings, which I could match in the test program but not in my main data program, it turns out that the single loop over the array contents was taking much longer as the array itself was buried in a structure.

Creating a temporary array and looping over that increased the speed from ~30 seconds to ~2 seconds

The red herring was that using the non looping method, the fact that it was in a structure hadn't affected the speed...

(Although I stand by original message that the += operator doesn't work as you might expect with arrays!)

Thanks again

Oliver

Subject: Re: Adding x,y events to a 2d array (quickly)
Posted by Dick Jackson on Fri, 08 Nov 2013 16:11:43 GMT
View Forum Message <> Reply to Message

Oliver wrote, On 2013-11-08, 6:18am:
> Hi again - thanks for all replies especially Dick's last with all the timings, which allows me to sheepishly admit that I solved the speed problem, but not how I expected to!
>
> Looking at the timings, which I could match in the test program but not in my main data

program, it turns out that the single loop over the array contents was taking much longer as the array itself was buried in a structure.
>
> Creating a temporary array and looping over that increased the speed from ~30 seconds to ~2 seconds
>
> The red herring was that using the non looping method, the fact that it was in a structure hadn't affected the speed...
>
> (Although I stand by original message that the += operator doesn't work as you might expect with arrays!)
>
> Thanks again
>
> Oliver

You're most welcome. It is indeed counterintuitive that
    array[indicesWithDuplicates] ++

is not equivalent to
    array[indicesWithDuplicates] += 1

--

Cheers,
-Dick

Dick Jackson Software Consulting
Victoria, BC, Canada
www.d-jackson.com