Subject: Re: Array intersections
Posted by Andy Loughe on Thu, 26 Feb 1998 08:00:00 GMT
View Forum Message <> Reply to Message

> What is the most efficient way (using IDL, of course) to return
> the index at which two arrays intersect? e.g.
> <snip>

I believe the response of David Fanning does not return the "index"
at which two arrays intersect, but the actual values themselves
(right?).
Here is one solution for what you have asked for...

```
FUNCTION where_array, A, B, IA_IN_B=iA_in_B

; Check for: correct number of parameters.
;                   that A and B have each only 1 dimension
;                   that A and B are defined
if (n_params() ne 2 or (size(A))(0) ne 1 or (size(B))(0) ne 1 $
   or n_elements(A) eq 0 or n_elements(B) eq 0) then begin
   message,'Inproper parameters',/Continue
   message,'Usage: result =
where_array(A,B,[IA_IN_B=ia_in_b]',/Continue
   return,-2
endif

; Parameters exist, let's make sure they are not structures.
if ((size(A))((size(A))(0)+1) eq 8 or $
   (size(B))((size(B))(0)+1) eq 8) then begin
    message,'Inproper parameters',/Continue
    message,'Parameters cannot be of type Structure',/Continue
    return,-2
endif

; Build two matrices to compare.
Na = n_elements(a)
Nb = n_elements(b)
l  = lindgen(Na,Nb)
AA = A(l mod Na)
BB = B(l / Na)

; Compare the two matrices we just created.
I  = where(AA eq BB)
Ia = i mod Na
Ib = i / na

; Normally (without keyword, return index of B that exist in A.
```

```
if keyword_set(iA_in_B) then index = Ia else index = Ib

; Make sure a valid value was found.
if Ia(0) eq -1 or Ib(0) eq -1 then index = -1

return,index
```

Subject: Re: Array intersections
Posted by davidf on Thu, 26 Feb 1998 08:00:00 GMT
View Forum Message <> Reply to Message

Robert Moss (mossrm@texaco.com) writes:

> I'm sure there were some clever responses to this same question
> just a few months ago, but for the life of me I cannot recall
> what they were. Here's the question:
>
> What is the most efficient way (using IDL, of course) to return
> the index at which two arrays intersect? e.g.
>
> a = [1B, 2B, 9B, 5B, 6B, 11B]
> b = [5B, 6B]
> idx = intersect(a, b)
> idx = 3
>
> This is a highly simplified example, but the point is that I want
> a function that will accept two array inputs (byte arrays in my specific
> case) and return to me the index in array a where the subset b starts.
>
> Refresh my memory, please.

You can find the clever responses on my web page at:

http://www.dfanning.com/tips/set_operations.html

Cheers,

David

David Fanning, Ph.D.
Fanning Software Consulting
E-Mail: davidf@dfanning.com
Phone: 970-221-0438
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: Array intersections
Posted by J.D. Smith on Tue, 03 Mar 1998 08:00:00 GMT
View Forum Message <> Reply to Message

Andy Loughe wrote:
>
>> What is the most efficient way (using IDL, of course) to return
>> the index at which two arrays intersect? e.g.
>> <snip>
>
> I believe the response of David Fanning does not return the "index"
> at which two arrays intersect, but the actual values themselves
> (right?).
> Here is one solution for what you have asked for...

I made these comments about this method last year:

> Check out the NASA library routine match(), which is array based. It uses a
> flag array and an index array, so the memory overhead is roughly 3 times the
> sum of the two arrays, but it's pretty fast. It's attached. Note that it takes
> vectors, so you've go to flatten your array upon input (with reform).
>

> Just make sure you don't try and use [where_array] with big arrays -- it's an n^2 >algorithm (versus the order n algorithms posted prior). E.g., to compare two >floating 128x128 arrays for overlapping values, the program would create 3 arrays, >each of which takes 1 GB! The routine match() is likely much more efficient for >doing intersections on big arrays. (Unless you have some serious RAM on your >machine).

JD

--
J.D. Smith |*| WORK: (607) 255-5842
Cornell University Dept. of Astronomy |*| (607) 255-4083
206 Space Sciences Bldg. |*| FAX: (607) 255-5875
Ithaca, NY 14853 |*|

Subject: Re: Array intersections
Posted by J.D. Smith on Tue, 10 Mar 1998 08:00:00 GMT
View Forum Message <> Reply to Message

David Foster wrote:
>
> J.D. Smith wrote:
>>
>> Andy Loughe wrote:
>>>

>>>> What is the most efficient way (using IDL, of course) to return
>>>> the index at which two arrays intersect? e.g.
>>>> <snip>
>>>
>>> I believe the response of David Fanning does not return the "index"
>>> at which two arrays intersect, but the actual values themselves
>>> (right?).
>>> Here is one solution for what you have asked for...
>>
>> I made these comments about this method last year:
>>
>>> Check out the NASA library routine match(), which is array based. It uses a
>>> flag array and an index array, so the memory overhead is roughly 3 times the
>>> sum of the two arrays, but it's pretty fast. It's attached. Note that it takes
>>> vectors, so you've go to flatten your array upon input (with reform).
>>>
>>
>>> Just make sure you don't try and use [where_array] with big arrays -- it's an n^2 >algorithm (versus the order n algorithms posted prior). E.g., to compare two >floating 128x128 arrays for overlapping values, the program would create 3 arrays, >each of which takes 1 GB! The routine match() is likely much more efficient for >doing intersections on big arrays. (Unless you have some serious RAM on your >machine).
>>
>> JD
>
> Some time ago someone from RSI posted these routines for doing
> array computations. I have found them to be very fast, and memory
> efficient as well. If you need a routine to return the VALUES of
> the intersection, you can download FIND_ELEMENTS.PRO at:
>
>   ftp://bial8.ucsd.edu   pub/software/idl/share/idl_share.tar.gz
>
> This routine is quite fast! It returns the values, not the indices.
>
> Enjoy!
>
> Here are the routines posted by RSI:
>
> ----------- SNIP --

It seems that set_intersection does indeed return the *values*, as the information says... If you need the *indices*, use could use something like find_elements, but this is just the same n^2 algorithm that I was warning against. So, I repeat, if you want to find the indices, and you have large data sets, you will be better off with a slower, but order n algorithm.

JD

--
J.D. Smith |*| WORK: (607) 255-5842
Cornell University Dept. of Astronomy |*| (607) 255-4083
206 Space Sciences Bldg. |*| FAX: (607) 255-5875
Ithaca, NY 14853 |*|