
Subject: Finding Common Elements in Two Arrays
Posted by [kucera](#) on Wed, 01 Jun 1994 20:45:00 GMT
[View Forum Message](#) <> [Reply to Message](#)

I'm looking for a quick way to compare two arrays in IDL, A and B,
and determine which elements of B are also in A,
so if:

A=[2,1,3,5,3,8,2,5]

B=[3,4,2,8,7,8]

I would get [0,2,3,5], because 3, 2, and 8 are in A as well as B.

I can do this with loops, but that takes too long for big arrays. Does anyone
have a way to do this using array functions or perhaps an external routine?

Terry Kucera

kucera@stars.gsfc.nasa.gov

Subject: Re: Finding Common Elements in Two Arrays
Posted by [dan](#) on Wed, 01 Jun 1994 23:23:48 GMT
[View Forum Message](#) <> [Reply to Message](#)

Subject: Finding Common Elements in Two Arrays
Newsgroup: comp.lang.idl-pwwave:

kucera@stars.gsfc.nasa.gov (Terry Kucera) writes :

> I'm looking for a quick way to compare two arrays in IDL, A and B,
> and determine which elements of B are also in A,
> so if:
> A=[2,1,3,5,3,8,2,5]
> B=[3,4,2,8,7,8]
> I would get [0,2,3,5], because 3, 2, and 8 are in A as well as B.

> I can do this with loops, but that takes too long for big arrays. Does anyone
> have a way to do this using array functions or perhaps an external routine?
> Terry Kucera
> kucera@stars.gsfc.nasa.gov

OK, basically what we have here is the problem of finding the intersection
of two sets of numbers. Assuming that the two sets contain non-negative
long or short integers, here is a slick (and tricky) way to do it :

IDL> A=[2,1,3,5,3,8,2,5]

IDL> B=[3,4,2,8,7,8]

IDL> ; PART 1. Find the numbers common to both sets.

```
IDL> set_length = Max([Max(A), Max(B)]) + 1L
IDL> set_1 = Bytarr(set_length)
IDL> set_2 = Bytarr(set_length)
IDL> set_1(A) = 1B
IDL> set_2(B) = 1B
IDL> common_num = Where(set_1 AND set_2)
IDL> Print, common_num
      2      3      8
```

IDL> ; PART 2. Find where the common numbers exist in array B.

```
IDL> index_arr1 = Replicate(1, N_Elements(B)) # common_num
IDL> index_arr2 = b # Replicate(1, N_Elements(common_num))
IDL> common_sub = Where(index_arr1 EQ index_arr2) MOD N_Elements(B)
IDL> Print, common_sub
      2      0      3      5
```

IDL> ; PART 3. Get ALL the common numbers in array B (including duplicates)

```
IDL> Print, B(common_sub(SORT(common_sub)))
      3      2      8      8
```

Try it, you'll like it !

 "If I don't read the net news, somebody else will."

```
 / \ \ ^ | \ | \ (Dan Carr)
 /   / \ | / | / (Research Systems)
 \   / \ | \ | \ (Boulder, Colorado)
 \___//   \ | \ | \ (dan@rsinc.com)
```

Subject: Re: Finding Common Elements in Two Arrays
 Posted by [stl](#) on Thu, 02 Jun 1994 06:26:38 GMT
[View Forum Message](#) <> [Reply to Message](#)

In article <1JUN199416454238@stars.gsfc.nasa.gov> kucera@stars.gsfc.nasa.gov (Terry Kucera) writes:

```
> I'm looking for a quick way to compare two arrays in IDL, A and B,
> and determine which elements of B are also in A,
> so if:
> A=[2,1,3,5,3,8,2,5]
> B=[3,4,2,8,7,8]
```

> I would get [0,2,3,5], because 3, 2, and 8 are in A as well as B.
>
> I can do this with loops, but that takes too long for big arrays. Does anyone
> have a way to do this using array functions or perhaps an external routine?

I just discovered (a few weeks ago) that this is a very usefull function! I was forced to use one loop (not sure if used one or two because you mention above 'loops'). If there is an array function I would LOVE to see it cause I need to do this LOTS of times, with arrays of up to 10,000 elements!

Here is what I did:

```
index = 0
for i = 0,n_elements(b)-1 do begin
  set = where(a eq b(i))
  if set(0) ne -1 then index = [index,set]
endfor
;get ride of the first 0!
if n_elements(index) gt 1 then index = index[1:*] $
else index = -1
```

for huge array, this is a bummer to have to do over and over! I am interested in an discussion and better solutions to this seamingly trivial but painfull little problem.

-stephen

--

```
Stephen C Strebel          /   SKI TO DIE
stl@maz.sma.ch            /   and
Swiss Meteorological Institute, Zuerich / LIVE TO TELL ABOUT IT
01 256 93 85              /   (and pray for snow)
```

Subject: Re: Finding Common Elements in Two Arrays

Posted by [landers](#) on Fri, 03 Jun 1994 13:02:43 GMT

[View Forum Message](#) <> [Reply to Message](#)

I'm gonna rebut my own post.... a bit.

In article <1994Jun2.223011.12904@mksol.dseg.ti.com>, landers@tsunami.dseg.ti.com (David Landers) writes:

```
> This is the first thing that popped into my head...
>
> Make 2-D arrays of A and B, like this:
>
> AA = A(*) # REPLICATE(1, N_ELEMENTS(B))
> BB = REPLICATE(1, N_ELEMENTS(A)) # B(*)
```

|>

May be better to do:

```
Na = N_ELEMENTS(a)
Nb = N_ELEMENTS(b)
L = LINDGEN(Na,Nb)
AA = A( L MOD Na )
BB = B( L / Na )
```

|> Now you can compare all combinations of one to the other:

|>

|> ALIKE = UNIQUE(AA(WHERE(AA EQ BB)))

|>

|> Then ALIKE contains a sorted list of the elements common to both arrays.

Yep. The advantage to the above over the matrix multiply (#) method is that it may be faster, and it will work with strings and structures (although EQ doesn't work with structs).

|> This could be more efficient, since you really only need a triangular array to
|> do this (you get 2 of each match, hence the UNIQUE).

What did I say? This is total nonsense (the triangular array bit). Just ignore this.

|>

|> --

|> Dave

Subject: Re: Finding Common Elements in Two Arrays

Posted by [landers](#) on Fri, 03 Jun 1994 13:10:50 GMT

[View Forum Message](#) <> [Reply to Message](#)

It's me again...

Now I noticed you want the indices rather than the values.... So

In article <1994Jun3.130243.3453@mksol.dseg.ti.com>, landers@tsunami.dseg.ti.com (David Landers) writes:

|>

|> Na = N_ELEMENTS(a)

|> Nb = N_ELEMENTS(b)

|> L = LINDGEN(Na,Nb)

```
|> AA = A( L MOD Na )
|> BB = B( L / Na )
```

Then...

```
I = AA EQ BB
Ia = UNIQUE( I MOD Na )
Ib = UNIQUE( I / Na )
```

Now Ia has the indices of A that are in B, and Ib has the indices of B that are in A. Leave off the UNIQUE if you care that an element matches multiple times.

Subject: Re: Finding Common Elements in Two Arrays
Posted by [stl](#) on Mon, 06 Jun 1994 08:09:03 GMT
[View Forum Message](#) <> [Reply to Message](#)

Good morning,

My program, where_array() that I posted on friday has a few bugs. I will modify it with some of the ideas from below.

NOTE: I beleive the below posting has an error in it:

In article <1994Jun3.131050.3790@mksol.dseg.ti.com> landers@tsunami.dseg.ti.com writes:
> It's me again...

>

> Now I noticed you want the indices rather than the values.... So

>

>

> In article <1994Jun3.130243.3453@mksol.dseg.ti.com>, landers@tsunami.dseg.ti.com (David Landers) writes:

>

> |>

> |> Na = N_ELEMENTS(a)

> |> Nb = N_ELEMENTS(b)

> |> L = LINDGEN(Na,Nb)

> |> AA = A(L MOD Na)

> |> BB = B(L / Na)

>

> Then...

>

> I = AA EQ BB

> Ia = UNIQUE(I MOD Na)
 ^^^

> Ib = UNIQUE(I / Na)
 ^^^

>
> Now Ia has the indices of A that are in B, and Ib has the indices of B that
> are in A. Leave off the UNIQUE if you care that an element matches multiple
> times.

I believe the above two lines with the unique(i ...) should be as follows:

```
Ia = unique(where(i) mod na)
Ib = unique(where(i) / na)
```

then clearly the common elements are just a(ia) and b(ib)

I will post a more stable routine with these new ideas. This is indeed better because it should handle string arrays, i too believe it to be faster.

-stephen Strebel

--
Stephen C Strebel / SKI TO DIE
stl@maz.sma.ch / and
Swiss Meteorological Institute, Zuerich / LIVE TO TELL ABOUT IT
01 256 93 85 / (and pray for snow)

Subject: Re: Finding common elements in two arrays

Posted by [andy](#) on Tue, 26 Jul 1994 16:39:57 GMT

[View Forum Message](#) <> [Reply to Message](#)

In article <CtJyp1.77x@ngdc.noaa.gov>, greg@farpoint.ngdc.noaa.gov (Greg Ushomirskiy) writes:

> A while ago there has been a discussion on what is the fastest way to find common
> elements in a pair of arrays. Of course, then I didn't listen, and now I need to
> do just that -- find common values in two arrays of longs. Since the articles
> describing the solution already expired, can someone post a summary?

>
> Thanks...

>
>
> --

> Greg Ushomirskiy #include <std_disclaimer.h>
> greg@farpoint.ngdc.noaa.gov
> National Geophysical Data Center
> NOAA, US. Department of Commerce

Ahhh! Those were the days. What a good discussion.
I don't know if this is the *BEST* solution, but it
is one that I found useful. I added some comments
but did not change the workings.

```
; General purpose routine to return the common elements of two vectors.  
; Uses USERLIB routine UNIQ to obtain unique elements of a,b.  
; Written by M. J. Dutch June-1994.  
; Centre de Recherches en Physique des Plasmas, EPFL, Switzerland
```

```
function same, a, b
```

```
if (n_params() lt 1) then message, ' Usage... result = same(a, b)'
```

```
ab = [a(uniq(a,sort(a))), b(uniq(b,sort(b)))] ;Combine unique elements of a,b  
ab = ab(sort(ab)) ;Sort the combined elements  
nab = n_elements(ab)
```

```
diff = ab(1:nab-1) - ab(0:nab-2)  
ind = where(diff eq 0) ;Find repeated elements
```

```
if (ind(0) eq -1) then return, -999  
if (ind(0) ne -1) then return, ab(ind)
```

```
end
```

```
--
```

```
,__o Andrew F. Loughe (Mail Code 971)  
-\"_<, NASA Goddard Space Flight Center phone: (301) 286-5899  
(*)/'(*) Greenbelt, MD 20771 email: andy.loughe@gsfc.nasa.gov
```
