
Subject: Re: The intersection of 2 arrays

Posted by [J.D. Smith](#) on Tue, 04 Mar 1997 08:00:00 GMT

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

David Foster wrote:

> PS>

>

> I've written a FIND_ELEMENTS.PRO that returns subscripts of
> elements in one array that are found in another array, like:

>

> ind = Find_Elements(Array, Tofind [, Adjust_array])

>

> This searches Array for values in Tofind, returning
> the subscripts of Array for those values found. The optional
> argument Adjust_array contains the same number of elements as
> Tofind, and is a way of allowing you to adjust the elements
> of Array (the values of Adjust_array and Tofind correspond).
> We use this for finding and adjusting Regions-of-Interest in
> images which have a limited number of discrete values. It
> uses a loop to search Array for each value of Tofind, so
> it's not real fast for large search arrays. [If anyone has
> an array-oriented method I'd love to see it!!]

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 got to flatten your array upon input (with reform).

I'd be interested to see the time comparisons for, say 2 128^2-element vectors of random integers on the interval [0-2000], or some equivalent big array test, because I've got some non-optimized routines that I've been trying to convince myself to rewrite.

JD

```
pro match, a, b, suba, subb, COUNT = count
;+
; NAME:
;   MATCH
; PURPOSE:
;   Routine to match values in two vectors.
;
; CALLING SEQUENCE:
;   match, a, b, suba, subb, [ COUNT = ]
;
; INPUTS:
;   a,b - two vectors to match elements
```

```

;
;
; OUTPUTS:
;   suba - subscripts of elements in vector a with a match
;         in vector b
;   subb - subscripts of the positions of the elements in
;         vector b with matches in vector a.
;
;   suba and subb are ordered such that a(suba) equals b(subb)
;
; OPTIONAL KEYWORD OUTPUT:
;   COUNT - set to the number of matches, integer scalar
;
; SIDE EFFECTS:
;   !ERR is set to the number of matches, can be used instead of COUNT
;
; RESTRICTIONS:
;   a and b should not have duplicate values within them.
;   You can use rem_dup function to remove duplicate values
;   in a vector
;
; EXAMPLE:
;   If a = [3,5,7,9,11] & b = [5,6,7,8,9,10]
;   then
;       IDL> match, a, b, suba, subb, COUNT = count
;
;   will give suba = [1,2,3], subb = [0,2,4], COUNT = 3
;   and    suba(a) = subb(b) = [5,7,9]
;
; HISTORY:
;   D. Lindler Mar. 1986.
;   Fixed "indgen" call for very large arrays W. Landsman Sep 1991
;   Added COUNT keyword W. Landsman Sep. 1992
;   Fixed case where single element array supplied W. Landsman Aug 95
;-
;-----
On_error,2

if N_params() LT 3 then begin
  print,'Syntax - match, a, b, suba, subb, [ COUNT = ]'
  print,'  a,b -- input vectors for which to match elements'
  print,'  suba,subb -- output subscript vectors of matched elements'
  return
endif

na = N_elements(a)      ;number of elements in a
nb = N_elements(b)      ;number of elements in b

; Check for a single element array

```

```

if (na EQ 1) or (nb EQ 1) then begin
  if (nb GT 1) then begin
    subb = where(b EQ a(0), nw)
    if (nw GT 0) then suba = replicate(0,nw) else suba = [-1]
  endif else begin
    suba = where(a EQ b(0), nw)
    if (nw GT 0) then subb = replicate(0,nw) else subb = [-1]
  endelse
  count = nw
  return
endif

c = [ a, b ]           ;combined list of a and b
ind = [ lindgen(na), lindgen(nb) ]   ;combined list of indices
vec = [ bytarr(na), replicate(1b,nb) ] ;flag of which vector in combined
                                     ;list  0 - a  1 - b

; sort combined list

sub = sort(c)
c = c(sub)
ind = ind(sub)
vec = vec(sub)

; find duplicates in sorted combined list

n = na + nb           ;total elements in c
firstdup = where( (c EQ shift(c,-1)) and (vec NE shift(vec,-1)), Count )

if Count EQ 0 then begin           ;any found?
  suba = lonarr(1)-1
  subb = lonarr(1)-1
  return
end

dup = lonarr( Count*2 )           ;both duplicate values
even = lindgen( N_elements(firstdup))*2   ;Changed to LINDGEN 6-Sep-1991
dup(even) = firstdup
dup(even+1) = firstdup+1
ind = ind(dup)                   ;indices of duplicates
vec = vec(dup)                   ;vector id of duplicates
suba = ind( where( vec EQ 0 ) )   ;a subscripts
subb = ind( where( vec EQ 1 ) )   ;b subscripts

return
end

```
