Subject: Re: matching lists
Posted by Dick Jackson on Wed, 06 Apr 2005 20:58:08 GMT
View Forum Message <> Reply to Message

Hi all,

This is a follow-up to a thread from March, 2000, where JD Smith had offered functions for returning indices of the values in one array that are found in a second array. Perhaps due to a change in what Sort() does with identical entries, the "ind_int_SORT" function gave erroneous results.

(I'm on IDL 6.1.1, Windows XP Pro, and I get this:
```
 IDL> print,sort([1,1])
           1           0
)
```

Here's a simple fix that lets it work no matter *what* order Sort() puts the indices of identical entries in. (even if it is inconsistent within a single call to Sort()!)

=====
```
;; Return the indices of values in a which exists anywhere in b
;; (one only for repeated values)
function ind_int_SORT, a, b
    flag=[replicate(0b,n_elements(a)),replicate(1b,n_elements(b) )]
   s=[a,b]
   srt=sort(s)
   s=s[srt] & flag=flag[srt]
   wh=where(s eq shift(s,-1) and flag ne shift(flag, -1),cnt)
   if cnt eq 0 then return, -1
   return,srt[wh+flag[wh]]
end
```
=====

The fix is just in the "return..." line, which had read:
```
   return,srt[wh]
```
... and which for me returned:

```
IDL> print,ind_int_SORT(['1','2'],['1','2'])
           2           3
```

Here we need to know whether an item in 'srt' came from 'a' or 'b', and we know that from 'flag'. So now, we get:

```
IDL> print,ind_int_SORT(['1','2'],['1','2'])
           0           1
```

Note that this doesn't guarantee that the result indices are sorted:

```
IDL> print,ind_int_SORT(['1','1','2','3'],['1','2','4'])
          1          0          2
```

Hmm, since JD's always did guarantee this (like Where() does it), here's one more amendment:

```
=====
;; Return the indices of values in a which exists anywhere in b
;; (one only for repeated values)
function ind_int_SORT, a, b
    flag=[replicate(0b,n_elements(a)),replicate(1b,n_elements(b) )]
   s=[a,b]
   srt=sort(s)
   s=s[srt] & flag=flag[srt]
   wh=where(s eq shift(s,-1) and flag ne shift(flag, -1),cnt)
   if cnt eq 0 then return, -1
   result=srt[wh+flag[wh]]
   return,result[sort(result)]
end
=====
```

Mark Fardal posted a version of this with a different calling sequence, here's the amended version of that (two changes, for a_ind and b_ind):

```
=====
pro listmatch, a, b, a_ind, b_ind
   flag=[replicate(0b,n_elements(a)),replicate(1b,n_elements(b) )]
  s=[a,b]
  srt=sort(s)
  s=s[srt] & flag=flag[srt]
  wh=where(s eq shift(s,-1) and flag ne shift(flag, -1),cnt)
  if cnt ne 0 then begin
    a_ind = srt[wh+flag[wh]]
    a_ind = a_ind[sort(a_ind)]
    b_ind = srt[wh+1-flag[wh]]-N_Elements(a)
    b_ind = b_ind[sort(b_ind)]
  endif else begin
    a_ind = -1
    b_ind = -1
    return
  endelse
end
=====
```

Thanks to JD and Mark for their original work!

Cheers,
--
-Dick

Dick Jackson / dick@d-jackson.com
D-Jackson Software Consulting / http://www.d-jackson.com
Calgary, Alberta, Canada / +1-403-242-7398 / Fax: 241-7392