Subject: Re: Vector comparison.
Posted by Craig Markwardt on Wed, 19 Nov 2003 23:48:07 GMT
View Forum Message <> Reply to Message

"hunter" <elhunter@rci.rutgers.edu> writes:

> Hello,
>
> These seems to be a fairly simple problem but I'm having difficulty coming
> up with an elegant solution.
>
> Let's say I have two vectors of type integer:
>
> A=[0,1,3,3,3,6,7,9,9]
> B=[3,7]
>
> I would like to design a function which returns the indices of all the
> elements of A which appear in B.

Now and again this question comes up. I trot out my own personal solution, which is CMSET_OP, which finds the intersection between two sets (appearing in both A and B):

```
  c = cmset_op(a, 'AND', b, /index)
```

There are no loops.

Good luck,
Craig

P.S. Please see
    http://cow.physics.wisc.edu/~craigm/idl/idl.html   (under array/set ops)

--
 -- --------------
Craig B. Markwardt, Ph.D.   EMAIL: craigmnet@REMOVEcow.physics.wisc.edu
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
 -- --------------

Subject: Re: Vector comparison.
Posted by Pavel Romashkin on Thu, 20 Nov 2003 00:09:31 GMT
View Forum Message <> Reply to Message

I would use Unique to get the non-repeating values. You can then compare
it with the continuous index of A to see if there are repating values.
Then, I'd use Value_locate function to match B against the unique
sub-array of A.

But Histogram would probably be faster.

Pavel

hunter wrote:
>
> Hello,
>
> These seems to be a fairly simple problem but I'm having difficulty coming
> up with an elegant solution.
>
> Let's say I have two vectors of type integer:
>
> A=[0,1,3,3,3,6,7,9,9]
> B=[3,7]
>
> I would like to design a function which returns the indices of all the
> elements of A which appear in B.
>
> i.e.
>
> C=get_match(A,B)
>
> should return
>
> C=[2,3,4,6]
>
> The simplest answer (I believe) is to loop through B and use the where
> command. I just wonder if there is a way to do this without useing the loop,
> as (in reality) the length of B may be very large.
>
> I suppose another possibility is to use the histogram command with
> reverse_indices set. But I think this would still require me to use a loop.
> Although it may be faster since I would only have to call histogram once.
> Any thoughts?
>
> Thanks,
> Eli

Subject: Re: Vector comparison.
Posted by R.Bauer on Thu, 20 Nov 2003 01:50:04 GMT
View Forum Message <> Reply to Message

hunter wrote:

> Hello,
>

> These seems to be a fairly simple problem but I'm having difficulty coming
> up with an elegant solution.
>
> Let's say I have two vectors of type integer:
>
> A=[0,1,3,3,3,6,7,9,9]
> B=[3,7]
>
Dear Eli

Here are some useful routines

http://www.fz-juelich.de/icg/icg-i/idl_icglib/idl_source/idl _html/dbase/download/a_and_b.tgz
http://www.fz-juelich.de/icg/icg-i/idl_icglib/idl_source/idl _html/dbase/download/a_or_b.tgz
http://www.fz-juelich.de/icg/icg-i/idl_icglib/idl_source/idl _html/dbase/download/a_not_b.tgz
http://www.fz-juelich.de/icg/icg-i/idl_icglib/idl_source/idl _html/dbase/download/a_xor_b.tgz
http://www.fz-juelich.de/icg/icg-i/idl_icglib/idl_source/idl _html/dbase/download/veklogik.tgz
http://www.fz-juelich.de/icg/icg-i/idl_icglib/idl_source/idl _html/dbase/download/indexlogic.tgz

```
print,a_and_b(a,b)
      3       7
print,a_or_b(a,b)
      0       1       3       6       7       9
print,a_not_b(a,b)
      0       1       6       9
print,a_xor_b(a,b)
      0       1       6       9
```

veklogic
This function applies different boolean operators to two vectors. It's a wrapper for the functions a_and_b, a_or_b, a_xor_b, a_not_b

indexlogic
This function applies different boolean operators to two index vectors. This routine is much faster then VekLogik but can only used with index vectors.

For some more routine please have a look at
http://www.fz-juelich.de/icg/icg-i/idl_icglib/idl_lib_intro. html

regards

Reimar

> I would like to design a function which returns the indices of all the
> elements of A which appear in B.
>
> i.e.

>
> C=get_match(A,B)
>
> should return
>
> C=[2,3,4,6]
>
> The simplest answer (I believe) is to loop through B and use the where
> command. I just wonder if there is a way to do this without useing the
> loop, as (in reality) the length of B may be very large.
>
> I suppose another possibility is to use the histogram command with
> reverse_indices set. But I think this would still require me to use a
> loop. Although it may be faster since I would only have to call histogram
> once. Any thoughts?
>
> Thanks,
> Eli

--
Forschungszentrum Juelich
email: R.Bauer@fz-juelich.de
http://www.fz-juelich.de/icg/icg-i/
 == ======
a IDL library at ForschungsZentrum Juelich
 http://www.fz-juelich.de/icg/icg-i/idl_icglib/idl_lib_intro. html

---

Subject: Re: Vector comparison.
Posted by David Fanning on Thu, 20 Nov 2003 02:05:59 GMT
View Forum Message <> Reply to Message

Reimar Bauer writes:

> Here are some useful routines
>
> http://www.fz-juelich.de/icg/icg-i/idl_icglib/idl_source/idl _html/dbase/download/a_and_b.tgz
> http://www.fz-juelich.de/icg/icg-i/idl_icglib/idl_source/idl _html/dbase/download/a_or_b.tgz
> http://www.fz-juelich.de/icg/icg-i/idl_icglib/idl_source/idl _html/dbase/download/a_not_b.tgz
> http://www.fz-juelich.de/icg/icg-i/idl_icglib/idl_source/idl _html/dbase/download/a_xor_b.tgz
> http://www.fz-juelich.de/icg/icg-i/idl_icglib/idl_source/idl _html/dbase/download/veklogik.tgz
> http://www.fz-juelich.de/icg/icg-i/idl_icglib/idl_source/idl _html/dbase/download/indexlogic.tgz
>

```
> print,a_and_b(a,b)
>       3       7
> print,a_or_b(a,b)
>       0       1       3       6       7       9
> print,a_not_b(a,b)
>        0       1       6       9
>  print,a_xor_b(a,b)
>        0       1       6       9
```

Alas, Reimar, none of these gives the answer of [2,3,4,6]
that Mr. Hunter was looking for. :-)

Cheers,

David

P.S. I know we are all busy, but maybe we should make
it a new rule that we all have to take a deep breath and
a stretch, then read the question TWICE before
posting an answer. :-)

--
David W. Fanning, Ph.D.
Fanning Software Consulting, Inc.
Phone: 970-221-0438, E-mail: david@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: Vector comparison.
Posted by Craig Markwardt on Thu, 20 Nov 2003 08:23:55 GMT

View Forum Message <> Reply to Message

David Fanning <david@dfanning.com> writes:
> P.S. I know we are all busy, but maybe we should make
> it a new rule that we all have to take a deep breath and
> a stretch, then read the question TWICE before
> posting an answer. :-)

Heh, I am guilty of the same thing in my own reply. The original
poster's requirement of keeping duplicates is a bit harder. From my
experience, an efficient set-intersection routine is very likely going
to do a sort/unique, which removes duplicates.

If one knows that the intersection is going to be small, then a simple
post-processing loop, with a brute force search, will do the trick.

As in:
```
  c0 = cmset_op(a, 'AND', b, /index, count=ct)
  if ct EQ 0 then message, 'ERROR: No intersection found'

  c = -1L
  for i = 0 , n_elements(c0)-1 do begin & wh = where(a EQ a(c0(i)), ct) & $
```

```
 if ct GT 0 then c = [c, wh] & end
  c = c(1:*)
```

Craig

--
 -- --------------
Craig B. Markwardt, Ph.D.   EMAIL: craigmnet@REMOVEcow.physics.wisc.edu
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
 -- --------------

Subject: Re: Vector comparison.
Posted by R.Bauer on Thu, 20 Nov 2003 09:01:58 GMT
View Forum Message <> Reply to Message

>
>
> Alas, Reimar, none of these gives the answer of [2,3,4,6]
> that Mr. Hunter was looking for. :-)
>
> Cheers,
>
> David
>
> P.S. I know we are all busy, but maybe we should make
> it a new rule that we all have to take a deep breath and
> a stretch, then read the question TWICE before
> posting an answer. :-)
>

Oh, yes you are right, I missed to post the right thing. As I started
writing I thought I give a complete answer to all related topics.

```
IDL> A=[0,1,3,3,3,6,7,9,9]
IDL> B=[3,7]
IDL> print,which_indices(a,b)
           2           3           4           6
```

 http://www.fz-juelich.de/icg/icg-i/idl_icglib/idl_source/idl _html/dbase/download/which_indices.tgz

Reimar

--

Reimar Bauer

Institut fuer Stratosphaerische Chemie (ICG-I)
Forschungszentrum Juelich
email: R.Bauer@fz-juelich.de
--- -------
a IDL library at ForschungsZentrum Juelich
http://www.fz-juelich.de/icg/icg-i/idl_icglib/idl_lib_intro. html
== =======

Subject: Re: Vector comparison.
Posted by Chris Lee on Thu, 20 Nov 2003 10:09:23 GMT
View Forum Message <> Reply to Message

In article <3fbbc86c$1@rutgers.edu>, "hunter" <elhunter@rci.rutgers.edu>
wrote:

> Hello,
> These seems to be a fairly simple problem but I'm having difficulty
> coming up with an elegant solution.
> Let's say I have two vectors of type integer: A=[0,1,3,3,3,6,7,9,9]
> B=[3,7]
> I would like to design a function which returns the indices of all the
> elements of A which appear in B.
> i.e.
> C=get_match(A,B)
> should return
> C=[2,3,4,6]
> The simplest answer (I believe) is to loop through B and use the where
> command. I just wonder if there is a way to do this without useing the
> loop, as (in reality) the length of B may be very large. I suppose
> another possibility is to use the histogram command with reverse_indices
> set. But I think this would still require me to use a loop. Although it
> may be faster since I would only have to call histogram once. Any
> thoughts?
> Thanks,
> Eli
>

How much memory do you have.... (or, to put it another way, loops really
aren't that bad when n_elements(array) > big_number :)

anyway, the memory hogging non loopy answer.

```
a=[0,1,3,3,3,6,7,9,9]
b=[3,7]
```

....

```
function get_match,a,b

;needs some up-to-date version of IDL, 5.4 I think.

na=n_elements(a)
nb=n_elements(b)

;some checks go here to make sure the world won't explode, an exersice
;for the reader.

reb_a=[na,nb]
ref_a=[na,1]
reb_b=[na,nb]
ref_b=[1,nb]

a_temp=rebin(reform(a, ref_a),reb_a)
b_temp=rebin(reform(b, ref_b),reb_b)

c_temp=a_temp-b_temp
a_temp=0
b_temp=0

w=where(c_temp eq 0, count)

c_temp=0
;memory...

if( count gt 0) then c=w mod na
if( count eq 0) then c=-1

return, c
end
```

-----

if they are guaranteed to be integer (non-integer histogramming implies a
fuzzy match doesn't it?) and B is very big :-

```
hist=histogram(a,reverse_indices=r)

n=n_elements(b)

if(n eq 0) then c=-1

if(n ge 1) then c=r[r[b[0]-min(a)]:r[b[0]+1-min(a)]-1]
```

;does this work if b is not an array, something changed between 5.3 and
;5.5 but I forget.

```
if(n gt 1) then for i=1, n-1 do begin
c=[c,r[r[b[i]-min(a)]:r[b[i]+1-min(a)]-1]]
endfor
```

;this last bit should work....... This still has the loop of course but
It does grab all of the indices (unlike a UNIQ approach for example) and
uses less memory (O(na) instead of O(na*nb))

..

I think there must be a better way (a more IDL way), but inspiration
hasn't hit yet.

Chris.

Subject: Re: Vector comparison.
Posted by Craig Markwardt on Thu, 20 Nov 2003 10:19:46 GMT
View Forum Message <> Reply to Message

Reimar Bauer <R.Bauer@fz-juelich.de> writes:
>>
>> Alas, Reimar, none of these gives the answer of [2,3,4,6]
>> that Mr. Hunter was looking for. :-)
>>
>> Cheers,
>>
>> David
>>
>> P.S. I know we are all busy, but maybe we should make
>> it a new rule that we all have to take a deep breath and
>> a stretch, then read the question TWICE before
>> posting an answer. :-)
>>
>
> Oh, yes you are right, I missed to post the right thing. As I started
> writing I thought I give a complete answer to all related topics.
>
>
>
> IDL> A=[0,1,3,3,3,6,7,9,9]
> IDL> B=[3,7]
> IDL> print,which_indices(a,b)

OK, now it's my turn to be nit-picky. The WHICH_INDICES routine uses
a brute force loop approach which was what the original poster was
trying to avoid.

I think that getting *all* matches without loops is a rather difficult
problem. The approach that I recommended separately involves a "fast"
intersection and then a "slow" rebuilding of a duplicates list. That
falls over too if the intersection result is large, but it should
always be about as fast as the brute force approach.

Craig

--
 -- --------------
Craig B. Markwardt, Ph.D.   EMAIL: craigmnet@REMOVEcow.physics.wisc.edu
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
 -- --------------

Subject: Re: Vector comparison.
Posted by R.Bauer on Thu, 20 Nov 2003 12:02:26 GMT

View Forum Message <> Reply to Message

Craig Markwardt wrote:
> Reimar Bauer <R.Bauer@fz-juelich.de> writes:
>
>>> Alas, Reimar, none of these gives the answer of [2,3,4,6]
>>> that Mr. Hunter was looking for. :-)
>>>
>>> Cheers,
>>>
>>> David
>>>
>>> P.S. I know we are all busy, but maybe we should make
>>> it a new rule that we all have to take a deep breath and
>>> a stretch, then read the question TWICE before
>>> posting an answer. :-)
>>>
>>
>> Oh, yes you are right, I missed to post the right thing. As I started
>> writing I thought I give a complete answer to all related topics.
>>
>>
>>
>> IDL> A=[0,1,3,3,3,6,7,9,9]
>> IDL> B=[3,7]
>> IDL> print,which_indices(a,b)
>

>
> OK, now it's my turn to be nit-picky. The WHICH_INDICES routine uses
> a brute force loop approach which was what the original poster was
> trying to avoid.
>

Dear Craig,

I have learned a new word. "nit-picky"

Yes this was not the best answer to the originally request. But he also
does not show a working routine. I believe it is better to show some
code as you did too. I did the postings because it is later easier to
find the complete knowledge by the google search in one thread.

I will try your method and will watch if some other experts could show a
different solution.
The best result is very important for our group too.

P.S. If we have all read it as carefully as mentioned what would we have
answered. Pavel will uses loops as well, we boths have not read
carefully enough, David hasn't replied.

Reimar

> I think that getting *all* matches without loops is a rather difficult
> problem. The approach that I recommended separately involves a "fast"
> intersection and then a "slow" rebuilding of a duplicates list. That
> falls over too if the intersection result is large, but it should
> always be about as fast as the brute force approach.
>
> Craig
>

--
Reimar Bauer

Institut fuer Stratosphaerische Chemie (ICG-I)
Forschungszentrum Juelich
email: R.Bauer@fz-juelich.de
-- -------
a IDL library at ForschungsZentrum Juelich
http://www.fz-juelich.de/icg/icg-i/idl_icglib/idl_lib_intro. html

== =======

Subject: Re: Vector comparison.
Posted by David Fanning on Thu, 20 Nov 2003 13:40:11 GMT
View Forum Message <> Reply to Message

Reimar Bauer writes:

> P.S. If we have all read it as carefully as mentioned what would we have
> answered. Pavel will uses loops as well, we boths have not read
> carefully enough, David hasn't replied.

No, David was interested in the *answer*. He didn't
have a solution of his own. :-)

Cheers,

David
--
David W. Fanning, Ph.D.
Fanning Software Consulting, Inc.
Phone: 970-221-0438, E-mail: david@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: Vector comparison.
Posted by hunter on Thu, 20 Nov 2003 14:49:15 GMT
View Forum Message <> Reply to Message

Hi,

Thanks, Chris. I came to a similar routine using histogram and a loop. It
may be more practical than the "memory hogging" answer, in my world anyway
:). Although, I'll keep both on hand. If I think of anything better, I'll be
sure to post it.

Thanks to everone forf their help.
Eli

"Christopher Lee" <cl@127.0.0.1> wrote in message
news:20031120.100922.1987231011.27332@buckley.atm.ox.ac.uk.. .
> In article <3fbbc86c$1@rutgers.edu>, "hunter" <elhunter@rci.rutgers.edu>
> wrote:

>
>
>> Hello,
>> These seems to be a fairly simple problem but I'm having difficulty
>> coming up with an elegant solution.
>> Let's say I have two vectors of type integer: A=[0,1,3,3,3,6,7,9,9]
>> B=[3,7]
>> I would like to design a function which returns the indices of all the
>> elements of A which appear in B.
>> i.e.
>> C=get_match(A,B)
>> should return
>> C=[2,3,4,6]
>> The simplest answer (I believe) is to loop through B and use the where
>> command. I just wonder if there is a way to do this without useing the
>> loop, as (in reality) the length of B may be very large. I suppose
>> another possibility is to use the histogram command with reverse_indices
>> set. But I think this would still require me to use a loop. Although it
>> may be faster since I would only have to call histogram once. Any
>> thoughts?
>> Thanks,
>> Eli
>>
>
> How much memory do you have.... (or, to put it another way, loops really
> aren't that bad when n_elements(array) > big_number :)
>
> anyway, the memory hogging non loopy answer.
>
> a=[0,1,3,3,3,6,7,9,9]
> b=[3,7]
>
> ....
>
> function get_match,a,b
>
> ;needs some up-to-date version of IDL, 5.4 I think.
>
> na=n_elements(a)
> nb=n_elements(b)
>
> ;some checks go here to make sure the world won't explode, an exersice
> ;for the reader.
>
> reb_a=[na,nb]
> ref_a=[na,1]
> reb_b=[na,nb]
> ref_b=[1,nb]

```
>
> a_temp=rebin(reform(a, ref_a),reb_a)
> b_temp=rebin(reform(b, ref_b),reb_b)
>
> c_temp=a_temp-b_temp
> a_temp=0
> b_temp=0
>
> w=where(c_temp eq 0, count)
>
> c_temp=0
> ;memory...
>
> if( count gt 0) then c=w mod na
> if( count eq 0) then c=-1
>
> return, c
> end
>
> -----
>
> if they are guaranteed to be integer (non-integer histogramming implies a
> fuzzy match doesn't it?) and B is very big :-
>
> hist=histogram(a,reverse_indices=r)
>
> n=n_elements(b)
>
> if(n eq 0) then c=-1
>
> if(n ge 1) then c=r[r[b[0]-min(a)]:r[b[0]+1-min(a)]-1]
>
> ;does this work if b is not an array, something changed between 5.3 and
> ;5.5 but I forget.
>
> if(n gt 1) then for i=1, n-1 do begin
> c=[c,r[r[b[i]-min(a)]:r[b[i]+1-min(a)]-1]]
> endfor
>
> ;this last bit should work....... This still has the loop of course but
> It does grab all of the indices (unlike a UNIQ approach for example) and
> uses less memory ( O(na) instead of O(na*nb) )
>
> ..
>
> I think there must be a better way (a more IDL way), but inspiration
> hasn't hit yet.
>
```

> Chris.

Subject: Re: Vector comparison.
Posted by tam on Thu, 20 Nov 2003 15:03:02 GMT
View Forum Message <> Reply to Message

David Fanning wrote:

> Reimar Bauer writes:
>
>
>> P.S. If we have all read it as carefully as mentioned what would we have
>> answered. Pavel will uses loops as well, we boths have not read
>> carefully enough, David hasn't replied.
>
>
> No, David was interested in the *answer*. He didn't
> have a solution of his own. :-)
>
> Cheers,
>
> David

How about the following, which handles duplicates in both arrays
and uses no loops.

```
; Find all of the indices of the elements
; in the first array that are matched in the second
; array
function match, a, b

    m = n_elements(a)
    n = n_elements(b)


    ; Create an array with each possible pairing of the first
    ; and last elements

    cmp = replicate(a[0], 2, m*n)

    ind  = lindgen(m*n)
    ind0 = ind mod m
    ind1 = ind  /  m

    ; Just lay out the 'a' array multiple times.
    cmp[0,*] = a[ind0]
    ; Repeat each element of the 'b' array m times so that
    ; we get each a element paired with each b element.
    cmp[1,*] = b[ind1]

    ; Now find all of the matches.
    w = where(cmp[0,*] eq cmp[1,*])

    if (w[0] eq -1) then begin
       return, w
    endif else begin

 ; Handle multiples in the 'b' set.
 ; If the elements in 'b' are guranteed
 ; to be unique then we can just return 'w mod m'

 h = histogram(w mod m, min=0)
 return, where(h ne 0)
    endelse
end
```

Have I missed something here... I think this would be reasonably efficient. Probably don't want to have separate ind, ind0, and ind1 arrays, but I thought that might show the algorithm more clearly.

```
     cmp[*,*] = [[a[lindgen(m*n) mod m][b[lindgen(m*n)/m]]
```

is not the kind of thing I can follow!

Regards,
Tom McGlynn

Subject: Re: Vector comparison.
Posted by R.Bauer on Thu, 20 Nov 2003 15:11:22 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
> Reimar Bauer writes:
>
>
>> P.S. If we have all read it as carefully as mentioned what would we have
>> answered. Pavel will uses loops as well, we boths have not read
>> carefully enough, David hasn't replied.
>
>
> No, David was interested in the *answer*. He didn't

> have a solution of his own. :-)
>
> Cheers,
>
> David

ok,ok,ok I am better quiet

Reimar

--
Reimar Bauer

Institut fuer Stratosphaerische Chemie (ICG-I)
Forschungszentrum Juelich
email: R.Bauer@fz-juelich.de
-- -------
a IDL library at ForschungsZentrum Juelich
http://www.fz-juelich.de/icg/icg-i/idl_icglib/idl_lib_intro. html
== =======

Subject: Re: Vector comparison.
Posted by Chris[1] on Thu, 20 Nov 2003 16:08:14 GMT
View Forum Message <> Reply to Message

Darnit! I was so proud of my answer... only to find that Chris Lee already proposed the same method (the "memory-hogging" one), albeit using more lines and this "if count eq 0 then ..." after WHERE, which always annoys me for some stupid reason. It can be avoided, which gives me an excuse to post my suggestion anyway:

```
function Get_Match, a, b
  na = n_elements(a) & nb = n_elements(b)
  return, where(total((rebin(a,na,nb) - rebin(transpose(b), na, nb)) eq 0, 2))
end
```

Cheers,

Timm

Today at 10:09 -0000, Christopher Lee wrote:

> [...]
>
> function get_match,a,b
>

```
> ;needs some up-to-date version of IDL, 5.4 I think.
>
> na=n_elements(a)
> nb=n_elements(b)
>
> ;some checks go here to make sure the world won't explode, an exersice
> ;for the reader.
>
> reb_a=[na,nb]
> ref_a=[na,1]
> reb_b=[na,nb]
> ref_b=[1,nb]
>
> a_temp=rebin(reform(a, ref_a),reb_a)
> b_temp=rebin(reform(b, ref_b),reb_b)
>
> c_temp=a_temp-b_temp
> a_temp=0
> b_temp=0
>
> w=where(c_temp eq 0, count)
>
> c_temp=0
> ;memory...
>
> if( count gt 0) then c=w mod na
> if( count eq 0) then c=-1
>
> return, c
> end
>
> [...]
```

--
Timm Weitkamp <http://people.web.psi.ch/weitkamp>

Subject: Re: Vector comparison.
Posted by David Fanning on Thu, 20 Nov 2003 16:08:50 GMT
View Forum Message <> Reply to Message

Reimar Bauer writes:

> ok,ok,ok I am better quiet

I was thinking, Reimar, that you were probably going for
my world record in number of newsgroup posts in a single
day. It ain't a gonna happen! :-)

Cheers,

David

--
David W. Fanning, Ph.D.
Fanning Software Consulting, Inc.
Phone: 970-221-0438, E-mail: david@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: Vector comparison.
Posted by David Fanning on Thu, 20 Nov 2003 16:52:11 GMT
View Forum Message <> Reply to Message

Timm Weitkamp writes:

```
> Darnit! I was so proud of my answer... only to find that Chris Lee already
> proposed the same method (the "memory-hogging" one), albeit using more
> lines and this "if count eq 0 then ..." after WHERE, which always annoys
> me for some stupid reason. It can be avoided, which gives me an excuse to
> post my suggestion anyway:
>
> function Get_Match, a, b
>   na = n_elements(a) & nb = n_elements(b)
>   return, where(total((rebin(a,na,nb) - rebin(transpose(b), na, nb)) eq 0, 2))
> end
```

Now there you go! That's a solution I can
steal for my web page. :-)

Cheers,

David
--
David W. Fanning, Ph.D.
Fanning Software Consulting, Inc.
Phone: 970-221-0438, E-mail: david@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: Vector comparison.
Posted by tam on Thu, 20 Nov 2003 17:24:15 GMT
View Forum Message <> Reply to Message

Timm Weitkamp wrote:
> Darnit! I was so proud of my answer... only to find that Chris Lee already
> proposed the same method (the "memory-hogging" one), albeit using more
> lines and this "if count eq 0 then ..." after WHERE, which always annoys
> me for some stupid reason. It can be avoided, which gives me an excuse to
> post my suggestion anyway:
>
```
> function Get_Match, a, b
>   na = n_elements(a) & nb = n_elements(b)
>   return, where(total((rebin(a,na,nb) - rebin(transpose(b), na, nb)) eq 0, 2))
> end
```
>
> Cheers,
>
> Timm

Hey, even I can understand this one! The only problem I can see
is that it can't handle arrays of strings... It's easy enough
to change the "a-b eq 0" to just "a eq b", but I don't see
how to do the rebin functionality on a string array. Is there
some other way to expand a string vector into a matrix (without
using something like the explicit pointer vector I had used).

Congratulations,

 Tom

Subject: Re: Vector comparison.
Posted by Craig Markwardt on Thu, 20 Nov 2003 17:35:48 GMT
View Forum Message <> Reply to Message

David Fanning <david@dfanning.com> writes:
> Timm Weitkamp writes:
>
>> Darnit! I was so proud of my answer... only to find that Chris Lee already
>> proposed the same method (the "memory-hogging" one), albeit using more
>> lines and this "if count eq 0 then ..." after WHERE, which always annoys
>> me for some stupid reason. It can be avoided, which gives me an excuse to
>> post my suggestion anyway:
>>
```
>> function Get_Match, a, b
>>   na = n_elements(a) & nb = n_elements(b)
>>   return, where(total((rebin(a,na,nb) - rebin(transpose(b), na, nb)) eq 0, 2))
>> end
```
>
> Now there you go! That's a solution I can
> steal for my web page. :-)

I think this is the same solution as Tom McGlynn. This is a classic
solution by brute force *memory* hogging instead of compute hogging.

If the lists are really large, then the arrays will be
n_elements(a)*n_elements(b) in size, which can be pretty big. For
example if each list had 10000 elements, then those 100 million
element arrays would start to suck a large amount of virtual memory.

Craig

--
 -- --------------
Craig B. Markwardt, Ph.D.   EMAIL: craigmnet@REMOVEcow.physics.wisc.edu
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
 -- --------------

Subject: Re: Vector comparison.
Posted by David Fanning on Thu, 20 Nov 2003 17:55:08 GMT
View Forum Message <> Reply to Message

Craig Markwardt writes:

>> Now there you go! That's a solution I can
>> steal for my web page. :-)
>
> I think this is the same solution as Tom McGlynn. This is a classic
> solution by brute force *memory* hogging instead of compute hogging.
>
> If the lists are really large, then the arrays will be
> n_elements(a)*n_elements(b) in size, which can be pretty big. For
> example if each list had 10000 elements, then those 100 million
> element arrays would start to suck a large amount of virtual memory.

I limit all arrays on my web page to less than 100
elements. Most people don't need all that complicated
detail. :-)

Cheers,

David

P.S. 10000 elements!? Astonomers!

--
David W. Fanning, Ph.D.
Fanning Software Consulting, Inc.

Phone: 970-221-0438, E-mail: david@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: Vector comparison.
Posted by tam on Thu, 20 Nov 2003 18:55:18 GMT
View Forum Message <> Reply to Message

Craig Markwardt wrote:

```
>  David  Fanning <david@dfanning.com> writes:
>
>> Timm Weitkamp writes:
>>
>>
>>> Darnit! I was so proud of my answer... only to find that Chris Lee already
>>> proposed the same method (the "memory-hogging" one), albeit using more
>>> lines and this "if count eq 0 then ..." after WHERE, which always annoys
>>> me for some stupid reason. It can be avoided, which gives me an excuse to
>>> post my suggestion anyway:
>>>
>>> function Get_Match, a, b
>>>   na = n_elements(a) & nb = n_elements(b)
>>>   return, where(total((rebin(a,na,nb) - rebin(transpose(b), na, nb)) eq 0, 2))
>>> end
>>
>> Now there you go! That's a solution I can
>> steal for my web page. :-)
>
>
>  I think this is the same solution as Tom McGlynn.  This is a classic
>  solution by brute force *memory* hogging instead of compute hogging.
>
>  If the lists are really large, then the arrays will be
>  n_elements(a)*n_elements(b) in size, which can be pretty big.  For
>  example if each list had 10000 elements, then those 100 million
>  element arrays would start to suck a large amount of virtual memory.
>
>  Craig
>
```

If the range of elements in b is not too large, then I suppose
something like:

```
  mn = min(a, max=mx)-1
  mx = mx+1
  h  = histogram(a,min=mn,max=mx)
```

```
   w  = where(h[a-mn] ne 0)
```

would do the trick.... We make the histogram one
to large on both sides so that we can use IDL's quirk
that going beyond the edge of an array is treated as
the last (or first) element. So long as the range in
b is smaller than the product of the number of elements
in the two arrays, this may be faster.

But it could only work on integers.
Tom

P.S. Note that I'm doing my best in all my solutions to ensure
that the histogram function is called.

Subject: Re: Vector comparison.
Posted by JD Smith on Thu, 20 Nov 2003 19:38:51 GMT
View Forum Message <> Reply to Message

On Thu, 20 Nov 2003 10:35:48 -0700, Craig Markwardt wrote:

```
>  David  Fanning <david@dfanning.com> writes:
>>  Timm Weitkamp writes:
>>
>>>  Darnit! I was so proud of my answer... only to find that Chris Lee
>>>  already proposed the same method (the "memory-hogging" one), albeit
>>>  using more lines and this "if count eq 0 then ..." after WHERE, which
>>>  always annoys me for some stupid reason. It can be avoided, which
>>>  gives me an excuse to post my suggestion anyway:
>>>
>>>  function Get_Match, a, b
>>>    na = n_elements(a) & nb = n_elements(b) return,
>>>    where(total((rebin(a,na,nb) - rebin(transpose(b), na, nb)) eq 0,
>>>    2))
>>>  end
>>
>>  Now there you go! That's a solution I can steal for my web page. :-)
>
>  I think this is the same solution as Tom McGlynn.  This is a classic
>  solution by brute force *memory* hogging instead of compute hogging.
>
>  If the lists are really large, then the arrays will be
>  n_elements(a)*n_elements(b) in size, which can be pretty big.  For
>  example if each list had 10000 elements, then those 100 million element
>  arrays would start to suck a large amount of virtual memory.
```

It's amazing how often this one comes up. I recall we hashed through this ad naseum several times over the last couple of years.

In this post I recounted some earlier info on the problem, and gave three comparable solutions, contrasting them by speed and features:

http://groups.google.com/groups?selm=38CBF8B6.5BF0AB50%40ast ro.cornell.edu

This includes the comment:

1. ARRAY is almost always slowest due to the large memory requirement. Only for very small input vectors (10 elements or so), will ARRAY beat SORT. It is, however, the only method which correctly identifies repeated entries (not used in this test).

See this whole thread for more info. I've yet to find a non-loop, non-array (i.e. memory hog) method of doing this with repeated indices.

JD