Subject: Finding strings values common to two (large!) arrays
Posted by rjp23 on Wed, 07 Oct 2015 14:13:53 GMT
View Forum Message <> Reply to Message

I have arrays of numerical IDs in string format.

I want to find all of the indices in Array A that contain a value that is present anywhere in Array B.

The arrays are both quite large (>1 million values) so a loop is out of the question and them being strings complicates it as well.

Any IDL Way tips?

Subject: Re: Finding strings values common to two (large!) arrays
Posted by Helder Marchetto on Wed, 07 Oct 2015 14:45:51 GMT
View Forum Message <> Reply to Message

On Wednesday, October 7, 2015 at 4:13:59 PM UTC+2, rj...@le.ac.uk wrote:
> I have arrays of numerical IDs in string format.
>
> I want to find all of the indices in Array A that contain a value that is present anywhere in Array B.
>
> The arrays are both quite large (>1 million values) so a loop is out of the question and them being strings complicates it as well.
>
> Any IDL Way tips?

Interesting... I guess that a set operation will do or in other words, you want to find (A) AND (B)
Did you look at David's page:
https://www.idlcoyote.com/tips/set_operations.html

There are some good tips, among which Craig's CMSET_OP which works also on strings (but does not return indices...).

I hope it helps.

Cheers,
Helder

Subject: Re: Finding strings values common to two (large!) arrays
Posted by wlandsman on Wed, 07 Oct 2015 15:09:26 GMT
View Forum Message <> Reply to Message

Two points to consider:

I second Helder's suggestions but have two additional points to consider:

1. Do your array A have duplicate values? And if so, do you want to find the indices of all the values, even if they are repeated? Then I would suggest using

http://idlastro.gsfc.nasa.gov/ftp/pro/misc/match2.pro

which will return every matching index even of duplicate values.

2. You say the arrays are "numerical IDs in string format". Are you able to convert these strings into numerical values? If so, the matching algorithms work faster for numerical arrays (especially integers) than for strings. I do suspect the speed difference is not important unless you have to do the matching many times.

--Wayne

On Wednesday, October 7, 2015 at 10:45:55 AM UTC-4, Helder wrote:
> On Wednesday, October 7, 2015 at 4:13:59 PM UTC+2, rj...@le.ac.uk wrote:
>> I have arrays of numerical IDs in string format.
>>
>> I want to find all of the indices in Array A that contain a value that is present anywhere in Array B.
>>
>> The arrays are both quite large (>1 million values) so a loop is out of the question and them being strings complicates it as well.
>>
>> Any IDL Way tips?
>
> Interesting... I guess that a set operation will do or in other words, you want to find (A) AND (B)
> Did you look at David's page:
> https://www.idlcoyote.com/tips/set_operations.html
>
> There are some good tips, among which Craig's CMSET_OP which works also on strings (but does not return indices...).
>
> I hope it helps.
>
> Cheers,
> Helder

Subject: Re: Finding strings values common to two (large!) arrays
Posted by rjp23 on Wed, 07 Oct 2015 15:21:27 GMT
View Forum Message <> Reply to Message

The IDs are of the form: 2009042300230430180019

I think that's too long to convert into a number (at least when I try to turn it into a long it ends up very different!)

CMSET_OP looks like it's what I need. Thanks both :-)

On Wednesday, October 7, 2015 at 4:09:29 PM UTC+1, wlandsman wrote:
> Two points to consider:
>
> I second Helder's suggestions but have two additional points to consider:
>
> 1. Do your array A have duplicate values? And if so, do you want to find the indices of all the values, even if they are repeated? Then I would suggest using
>
> http://idlastro.gsfc.nasa.gov/ftp/pro/misc/match2.pro
>
> which will return every matching index even of duplicate values.
>
> 2. You say the arrays are "numerical IDs in string format". Are you able to convert these strings into numerical values? If so, the matching algorithms work faster for numerical arrays (especially integers) than for strings. I do suspect the speed difference is not important unless you have to do the matching many times.
>
> --Wayne
>
> On Wednesday, October 7, 2015 at 10:45:55 AM UTC-4, Helder wrote:
>> On Wednesday, October 7, 2015 at 4:13:59 PM UTC+2, rj...@le.ac.uk wrote:
>>> I have arrays of numerical IDs in string format.
>>>
>>> I want to find all of the indices in Array A that contain a value that is present anywhere in Array B.
>>>
>>> The arrays are both quite large (>1 million values) so a loop is out of the question and them being strings complicates it as well.
>>>
>>> Any IDL Way tips?
>>
>> Interesting... I guess that a set operation will do or in other words, you want to find (A) AND (B)
>> Did you look at David's page:
>> https://www.idlcoyote.com/tips/set_operations.html
>>
>> There are some good tips, among which Craig's CMSET_OP which works also on strings (but does not return indices...).
>>
>> I hope it helps.
>>
>> Cheers,

>> Helder

Subject: Re: Finding strings values common to two (large!) arrays
Posted by Lajos Foldy on Wed, 07 Oct 2015 16:17:01 GMT
View Forum Message <> Reply to Message

On Wednesday, October 7, 2015 at 5:21:31 PM UTC+2, rj...@le.ac.uk wrote:
> The IDs are of the form: 2009042300230430180019
>
> I think that's too long to convert into a number (at least when I try to turn it into a long it ends up very different!)

If 20090423 means YYYYMMDD than you can recode it as yydoy (yy=YYYY-1950, doy=day of year) and it will fit into ulong64.

regards,
Lajos

Subject: Re: Finding strings values common to two (large!) arrays
Posted by rjp23 on Wed, 07 Oct 2015 19:57:19 GMT
View Forum Message <> Reply to Message

CMSET_OP looks to be working but I'm not 100% sure due to this comment in the header:

```
;   INDEX - if set, then return a list of indices instead of the array
;           values themselves.  The "slower" set operations are always
;           performed in this case.
;
;           The indices refer to the *combined* array [A,B].  To
;           clarify, in the following call: I = CMSET_OP(..., /INDEX);
;           returned values from 0 to NA-1 refer to A[I], and values
;           from NA to NA+NB-1 refer to B[I-NA].
```

When using the code like this, it is returning an array of indices that only seem to relate to the first array.

e.g. (massively simplied) A has 10 elements, B has 20 and the returned indices are an array of 7 values such as [0,1,2,5,7,8,9]

Would I not also expect indices for the elements in the second array (10-29) to also be returned going by the statement in the header?

On Wednesday, October 7, 2015 at 4:21:31 PM UTC+1, rj...@le.ac.uk wrote:
> The IDs are of the form: 200904230023043018 0019
>
> I think that's too long to convert into a number (at least when I try to turn it into a long it ends up very different!)
>
> CMSET_OP looks like it's what I need. Thanks both :-)
>
>
> On Wednesday, October 7, 2015 at 4:09:29 PM UTC+1, wlandsman wrote:
>> Two points to consider:
>>
>> I second Helder's suggestions but have two additional points to consider:
>>
>> 1. Do your array A have duplicate values? And if so, do you want to find the indices of all the values, even if they are repeated? Then I would suggest using
>>
>> http://idlastro.gsfc.nasa.gov/ftp/pro/misc/match2.pro
>>
>> which will return every matching index even of duplicate values.
>>
>> 2. You say the arrays are "numerical IDs in string format". Are you able to convert these strings into numerical values? If so, the matching algorithms work faster for numerical arrays (especially integers) than for strings. I do suspect the speed difference is not important unless you have to do the matching many times.
>>
>> --Wayne
>>
>> On Wednesday, October 7, 2015 at 10:45:55 AM UTC-4, Helder wrote:
>>> On Wednesday, October 7, 2015 at 4:13:59 PM UTC+2, rj...@le.ac.uk wrote:
>>>> I have arrays of numerical IDs in string format.
>>>>
>>>> I want to find all of the indices in Array A that contain a value that is present anywhere in Array B.
>>>>
>>>> The arrays are both quite large (>1 million values) so a loop is out of the question and them being strings complicates it as well.
>>>>
>>>> Any IDL Way tips?
>>>
>>> Interesting... I guess that a set operation will do or in other words, you want to find (A) AND (B)
>>> Did you look at David's page:
>>> https://www.idlcoyote.com/tips/set_operations.html
>>>
>>> There are some good tips, among which Craig's CMSET_OP which works also on strings (but does not return indices...).
>>>

>>> I hope it helps.
>>>
>>> Cheers,
>>> Helder

Subject: Re: Finding strings values common to two (large!) arrays
Posted by Craig Markwardt on Wed, 07 Oct 2015 22:19:43 GMT
View Forum Message <> Reply to Message

On Wednesday, October 7, 2015 at 3:57:23 PM UTC-4, rj...@le.ac.uk wrote:
> CMSET_OP looks to be working but I'm not 100% sure due to this comment in the header:
>
> ; INDEX - if set, then return a list of indices instead of the array
> ; values themselves. The "slower" set operations are always
> ; performed in this case.
> ;
> ; The indices refer to the *combined* array [A,B]. To
> ; clarify, in the following call: I = CMSET_OP(..., /INDEX);
> ; returned values from 0 to NA-1 refer to A[I], and values
> ; from NA to NA+NB-1 refer to B[I-NA].
>
>
> When using the code like this, it is returning an array of indices that only seem to relate to the first array.
>
> e.g. (massively simplied) A has 10 elements, B has 20 and the returned indices are an array of 7 values such as [0,1,2,5,7,8,9]
>
> Would I not also expect indices for the elements in the second array (10-29) to also be returned going by the statement in the header?

If you are using 'AND', CMSET_OP() only returns indices to the first array. It doesn't ever return duplicates, so it doesn't need to tell you *again* the same values.

You might also want to check MATCH2 in the IDL astronomy library. I wrote that also, and it is designed to match large catalogs against each other and return the matching indices on both sides.

Craig

Subject: Re: Finding strings values common to two (large!) arrays
Posted by rjp23 on Thu, 08 Oct 2015 11:40:48 GMT
View Forum Message <> Reply to Message

Thanks Craig, that's very helpful.

Just from the note in the header I was expecting it to repeat the same values again in the second array but that fact that it shouldn't (and doesn't) means it's all fine and working perfectly for what I need.
Cheers

Rob

On Wednesday, October 7, 2015 at 11:19:50 PM UTC+1, Craig Markwardt wrote:
> On Wednesday, October 7, 2015 at 3:57:23 PM UTC-4, rj...@le.ac.uk wrote:
>> CMSET_OP looks to be working but I'm not 100% sure due to this comment in the header:
>>
>> ; INDEX - if set, then return a list of indices instead of the array
>> ; values themselves. The "slower" set operations are always
>> ; performed in this case.
>> ;
>> ; The indices refer to the *combined* array [A,B]. To
>> ; clarify, in the following call: I = CMSET_OP(..., /INDEX);
>> ; returned values from 0 to NA-1 refer to A[I], and values
>> ; from NA to NA+NB-1 refer to B[I-NA].
>>
>>
>> When using the code like this, it is returning an array of indices that only seem to relate to the first array.
>>
>> e.g. (massively simplied) A has 10 elements, B has 20 and the returned indices are an array of 7 values such as [0,1,2,5,7,8,9]
>>
>> Would I not also expect indices for the elements in the second array (10-29) to also be returned going by the statement in the header?
>
>
> If you are using 'AND', CMSET_OP() only returns indices to the first array. It doesn't ever return duplicates, so it doesn't need to tell you *again* the same values.
>
> You might also want to check MATCH2 in the IDL astronomy library. I wrote that also, and it is designed to match large catalogs against each other and return the matching indices on both sides.
>
> Craig

Subject: Re: Finding strings values common to two (large!) arrays
Posted by Jeremy Bailin on Wed, 28 Oct 2015 16:00:17 GMT
View Forum Message <> Reply to Message

On Wednesday, October 7, 2015 at 11:21:31 AM UTC-4, rj...@le.ac.uk wrote:

> The IDs are of the form: 2009042300230430180019
>
> I think that's too long to convert into a number (at least when I try to turn it into a long it ends up very different!)
>
> CMSET_OP looks like it's what I need. Thanks both :-)
>
>
> On Wednesday, October 7, 2015 at 4:09:29 PM UTC+1, wlandsman wrote:
>> Two points to consider:
>>
>> I second Helder's suggestions but have two additional points to consider:
>>
>> 1. Do your array A have duplicate values? And if so, do you want to find the indices of all the values, even if they are repeated? Then I would suggest using
>>
>> http://idlastro.gsfc.nasa.gov/ftp/pro/misc/match2.pro
>>
>> which will return every matching index even of duplicate values.
>>
>> 2. You say the arrays are "numerical IDs in string format". Are you able to convert these strings into numerical values? If so, the matching algorithms work faster for numerical arrays (especially integers) than for strings. I do suspect the speed difference is not important unless you have to do the matching many times.
>>
>> --Wayne
>>
>> On Wednesday, October 7, 2015 at 10:45:55 AM UTC-4, Helder wrote:
>>> On Wednesday, October 7, 2015 at 4:13:59 PM UTC+2, rj...@le.ac.uk wrote:
>>>> I have arrays of numerical IDs in string format.
>>>>
>>>> I want to find all of the indices in Array A that contain a value that is present anywhere in Array B.
>>>>
>>>> The arrays are both quite large (>1 million values) so a loop is out of the question and them being strings complicates it as well.
>>>>
>>>> Any IDL Way tips?
>>>
>>> Interesting... I guess that a set operation will do or in other words, you want to find (A) AND (B)
>>> Did you look at David's page:
>>> https://www.idlcoyote.com/tips/set_operations.html
>>>
>>> There are some good tips, among which Craig's CMSET_OP which works also on strings (but does not return indices...).
>>>
>>> I hope it helps.

>>>
>>> Cheers,
>>> Helder

If you expect duplicates, this is a good example of when you might want to use Value_Locate as a mapping function (see http://www.idlcoyote.com/code_tips/valuelocate.html). First create a list of unique IDs:

unique_IDs = ID[uniq(ID, sort(ID))]

Then use them to map IDs from array_A (and B, etc):

mapped_ID = value_locate(unique_IDs, array_A)

mapped_ID has converted the strings into a densely-packed set of integers that can be used efficiently in set operations.

-Jeremy.

Subject: Re: Finding strings values common to two (large!) arrays
Posted by Russell[1] on Wed, 28 Oct 2015 16:25:07 GMT
View Forum Message <> Reply to Message

On Wednesday, October 7, 2015 at 10:13:59 AM UTC-4, rj...@le.ac.uk wrote:
> I have arrays of numerical IDs in string format.
>
> I want to find all of the indices in Array A that contain a value that is present anywhere in Array B.
>
> The arrays are both quite large (>1 million values) so a loop is out of the question and them being strings complicates it as well.
>
> Any IDL Way tips?

AGGH! Jeremy beat me to it. I would think you want to map your strings onto the unique set of integers following Jeremy's post. Then perhaps you could use histogram on the mapped integers.

-Russell

Subject: Re: Finding strings values common to two (large!) arrays
Posted by Jeremy Bailin on Wed, 28 Oct 2015 16:40:20 GMT
View Forum Message <> Reply to Message

On Wednesday, October 28, 2015 at 12:25:10 PM UTC-4, rrya...@gmail.com wrote:
> On Wednesday, October 7, 2015 at 10:13:59 AM UTC-4, rj...@le.ac.uk wrote:

>> I have arrays of numerical IDs in string format.
>>
>> I want to find all of the indices in Array A that contain a value that is present anywhere in Array B.
>>
>> The arrays are both quite large (>1 million values) so a loop is out of the question and them being strings complicates it as well.
>>
>> Any IDL Way tips?
>
> AGGH! Jeremy beat me to it. I would think you want to map your strings onto the unique set of integers following Jeremy's post. Then perhaps you could use histogram on the mapped integers.
>
> -Russell

You know I'm not going to miss a chance to spread the gospel of value_locate! :)

-Jeremy.

Subject: Re: Finding strings values common to two (large!) arrays
Posted by Russell[1] on Wed, 28 Oct 2015 19:26:27 GMT
View Forum Message <> Reply to Message

Yeah, I learned that trick from your posts on Fanning's webpage. It's been a revelation. But i still don't like the structure of:

id = id[uniq(id,sort(id))]

i wish uniq just had a built-in flag to do this for me... Under what circumstance would I want to *NOT* sort? Seems like if they just built uniq to sort by default, you could probably optimize this at the compiler level --- though I'm hardly an expert...

id = id[uniq(id)] ;would be nice or just

-R

On Wednesday, October 28, 2015 at 12:40:24 PM UTC-4, Jeremy Bailin wrote:
> On Wednesday, October 28, 2015 at 12:25:10 PM UTC-4, rrya...@gmail.com wrote:
>> On Wednesday, October 7, 2015 at 10:13:59 AM UTC-4, rj...@le.ac.uk wrote:
>>> I have arrays of numerical IDs in string format.
>>>
>>> I want to find all of the indices in Array A that contain a value that is present anywhere in Array B.
>>>
>>> The arrays are both quite large (>1 million values) so a loop is out of the question and them

being strings complicates it as well.
>>>
>>> Any IDL Way tips?
>>
>> AGGH! Jeremy beat me to it. I would think you want to map your strings onto the unique set of integers following Jeremy's post. Then perhaps you could use histogram on the mapped integers.
>>
>> -Russell
>
> You know I'm not going to miss a chance to spread the gospel of value_locate! :)
>
> -Jeremy.

Subject: Re: Finding strings values common to two (large!) arrays
Posted by Dick Jackson on Wed, 28 Oct 2015 21:48:09 GMT
View Forum Message <> Reply to Message

On Wednesday, 28 October 2015 12:26:30 UTC-7, rrya...@gmail.com wrote:
> Yeah, I learned that trick from your posts on Fanning's webpage. It's been a revelation. But i still don't like the structure of:
>
> id = id[uniq(id,sort(id))]
>
> i wish uniq just had a built-in flag to do this for me... Under what circumstance would I want to *NOT* sort? Seems like if they just built uniq to sort by default, you could probably optimize this at the compiler level --- though I'm hardly an expert...
>
> id = id[uniq(id)] ;would be nice or just
>
> -R

As of IDL 8.4, you can now do this:

```
IDL> a = [6, 2, 8, 3, 1, 8, 5, 3] ; (first digits of tau)
IDL> a.uniq()
      1       2       3       5       6       8
```

Reference: IDL_Variable::Uniq
http://www.exelisvis.com/docs/IDL_Variable.html#Uniq

Cheers,
-Dick

Dick Jackson Software Consulting Inc.
Victoria, BC, Canada --- http://www.d-jackson.com

Subject: Re: Finding strings values common to two (large!) arrays
Posted by Dick Jackson on Wed, 28 Oct 2015 21:53:30 GMT
View Forum Message <> Reply to Message

On Wednesday, 28 October 2015 14:48:13 UTC-7, Dick Jackson wrote:
> On Wednesday, 28 October 2015 12:26:30 UTC-7, rrya...@gmail.com wrote:
>> Yeah, I learned that trick from your posts on Fanning's webpage. It's been a revelation. But i still don't like the structure of:
>>
>> id = id[uniq(id,sort(id))]
>>
>> i wish uniq just had a built-in flag to do this for me... Under what circumstance would I want to *NOT* sort? Seems like if they just built uniq to sort by default, you could probably optimize this at the compiler level --- though I'm hardly an expert...
>>
>> id = id[uniq(id)] ;would be nice or just
>>
>> -R
>
> As of IDL 8.4, you can now do this:
>
> IDL> a = [6, 2, 8, 3, 1, 8, 5, 3] ; (first digits of tau)
> IDL> a.uniq()
> 1 2 3 5 6 8
>
> Reference: IDL_Variable::Uniq
> http://www.exelisvis.com/docs/IDL_Variable.html#Uniq

... and to put a finer point on it, this method assumes the elements need to be sorted. In the case where they are already sorted (where sorting time can thus be saved, which is the default with the regular uniq() function), you can use:
a.uniq(/no_sort)

Cheers,
-Dick

Dick Jackson Software Consulting Inc.
Victoria, BC, Canada --- http://www.d-jackson.com

Subject: Re: Finding strings values common to two (large!) arrays
Posted by Jeremy Bailin on Thu, 29 Oct 2015 18:17:22 GMT
View Forum Message <> Reply to Message

On Wednesday, October 28, 2015 at 5:53:32 PM UTC-4, Dick Jackson wrote:
> On Wednesday, 28 October 2015 14:48:13 UTC-7, Dick Jackson wrote:
>> On Wednesday, 28 October 2015 12:26:30 UTC-7, rrya...@gmail.com wrote:
>>> Yeah, I learned that trick from your posts on Fanning's webpage. It's been a revelation. But

i still don't like the structure of:
>>>
>>> id = id[uniq(id,sort(id))]
>>>
>>> i wish uniq just had a built-in flag to do this for me... Under what circumstance would I want to *NOT* sort? Seems like if they just built uniq to sort by default, you could probably optimize this at the compiler level --- though I'm hardly an expert...
>>>
>>> id = id[uniq(id)] ;would be nice or just
>>>
>>> -R
>>
>> As of IDL 8.4, you can now do this:
>>
>> IDL> a = [6, 2, 8, 3, 1, 8, 5, 3] ; (first digits of tau)
>> IDL> a.uniq()
>>   1   2   3   5   6   8
>>
>> Reference: IDL_Variable::Uniq
>> http://www.exelisvis.com/docs/IDL_Variable.html#Uniq
>
> ... and to put a finer point on it, this method assumes the elements need to be sorted. In the case where they are already sorted (where sorting time can thus be saved, which is the default with the regular uniq() function), you can use:
>   a.uniq(/no_sort)
>
> Cheers,
> -Dick
>
> Dick Jackson Software Consulting Inc.
> Victoria, BC, Canada --- http://www.d-jackson.com

Actually, there is a use case for running uniq intentionally without sorting, which I have used before: identifying duplications (similar to times you might use label_region).

IDL> q = [3,5,3,3,8,7,7]
IDL> q[uniq(q)]
      3   5   3   8   7

-Jeremy.

Subject: Re: Finding strings values common to two (large!) arrays
Posted by Dick Jackson on Thu, 29 Oct 2015 22:47:38 GMT
View Forum Message <> Reply to Message

On Thursday, 29 October 2015 11:17:26 UTC-7, Jeremy Bailin wrote:
> On Wednesday, October 28, 2015 at 5:53:32 PM UTC-4, Dick Jackson wrote:

>> On Wednesday, 28 October 2015 14:48:13 UTC-7, Dick Jackson wrote:
>>> On Wednesday, 28 October 2015 12:26:30 UTC-7, rrya...@gmail.com wrote:
>>>> Yeah, I learned that trick from your posts on Fanning's webpage. It's been a revelation. But i still don't like the structure of:
>>>>
>>>> id = id[uniq(id,sort(id))]
>>>>
>>>> i wish uniq just had a built-in flag to do this for me... Under what circumstance would I want to *NOT* sort? Seems like if they just built uniq to sort by default, you could probably optimize this at the compiler level --- though I'm hardly an expert...
>>>>
>>>> id = id[uniq(id)] ;would be nice or just
>>>>
>>>> -R
>>>
>>> As of IDL 8.4, you can now do this:
>>>
>>> IDL> a = [6, 2, 8, 3, 1, 8, 5, 3] ; (first digits of tau)
>>> IDL> a.uniq()
>>> 1 2 3 5 6 8
>>>
>>> Reference: IDL_Variable::Uniq
>>> http://www.exelisvis.com/docs/IDL_Variable.html#Uniq
>>
>> ... and to put a finer point on it, this method assumes the elements need to be sorted. In the case where they are already sorted (where sorting time can thus be saved, which is the default with the regular uniq() function), you can use:
>> a.uniq(/no_sort)
>>
>> Cheers,
>> -Dick
>>
>> Dick Jackson Software Consulting Inc.
>> Victoria, BC, Canada --- http://www.d-jackson.com
>
> Actually, there is a use case for running uniq intentionally without sorting, which I have used before: identifying duplications (similar to times you might use label_region).
>
> IDL> q = [3,5,3,3,8,7,7]
> IDL> q[uniq(q)]
> 3 5 3 8 7
>
> -Jeremy.

One thing that can come around to bite you is how Uniq() comes around:

IDL> q = [3,5,3,3,8,7,7,3] ; duplicates on each end...
IDL> q[uniq(q)] ; are considered adjacent with Uniq() unsorted...

```
      3      5      3      8      7
IDL> q.uniq(/no_sort)                  ; but not by IDL_Variable::Uniq(/NO_SORT)
      3      5      3      8      7      3
```

Cheers,
-Dick

Dick Jackson Software Consulting Inc.
Victoria, BC, Canada --- http://www.d-jackson.com
