Subject: Re: How to get matching elements of array efficiently
Posted by penteado on Tue, 23 Feb 2010 20:36:53 GMT
View Forum Message <> Reply to Message

On Feb 23, 5:11 pm, JJ <j...@cornell.edu> wrote:
> I'm looking for an efficient (non-loop) solution to this problem. I
> have two arrays A, B. The elements of A are not necessarily unique,
> nor is A necessarily sorted. I'd like to find all the indices in A
> that match values that occur in B.
>
> For example, if A = [7,1,8,7,8], B = [7,8], the result should be
> [0,2,3,4].
>
> A and B can have many (order millions) of unique values, so I'd rather
> avoid a loop if I can.
>
> Is there any way to do this efficiently?

Are your values integers as in the example? If they are, and if there are no big gaps in B (the number of elements of B is not much smaller than max(B)-min(B)), histogram comes to mind:

h=histogram(A,binsize=1,min=min(b),max=max(b),reverse_indice s=ri)
if (max(h) gt 0L) then res=ri[n_elements(h)+1:*] else (deal with the case of none found in b)

Which would give the result you want in res, but ordered by bin (their order in B). In this example, res would be [0,3,2,4].

Subject: Re: How to get matching elements of array efficiently
Posted by wlandsman on Tue, 23 Feb 2010 21:06:02 GMT
View Forum Message <> Reply to Message

You might look at Craig Markwardt's program match2 in
http://idlastro.gsfc.nasa.gov/ftp/pro/misc/match2.pro

For your example, one would type
IDL> match2, a, b, suba, subb
IDL> print,where(suba GE 0)
0 2 3 4

You might be able to make match2.pro more efficient, because it also gives the index of a (not necessarily unique) value in B that matches a value in A, which is information you don't need. --Wayne

On Feb 23, 3:11 pm, JJ <j...@cornell.edu> wrote:

> Hi All,
>
> I'm looking for an efficient (non-loop) solution to this problem. I
> have two arrays A, B. The elements of A are not necessarily unique,
> nor is A necessarily sorted. I'd like to find all the indices in A
> that match values that occur in B.
>
> For example, if A = [7,1,8,7,8], B = [7,8], the result should be
> [0,2,3,4].
>
> A and B can have many (order millions) of unique values, so I'd rather
> avoid a loop if I can.
>
> Is there any way to do this efficiently?
>
> Thanks.
>
> -JJ

Subject: Re: How to get matching elements of array efficiently
Posted by rogass on Wed, 24 Feb 2010 08:54:45 GMT
View Forum Message <> Reply to Message

On 23 Feb., 21:11, JJ <j...@cornell.edu> wrote:
> Hi All,
>
> I'm looking for an efficient (non-loop) solution to this problem. I
> have two arrays A, B. The elements of A are not necessarily unique,
> nor is A necessarily sorted. I'd like to find all the indices in A
> that match values that occur in B.
>
> For example, if A = [7,1,8,7,8], B = [7,8], the result should be
> [0,2,3,4].
>
> A and B can have many (order millions) of unique values, so I'd rather
> avoid a loop if I can.
>
> Is there any way to do this efficiently?
>
> Thanks.
>
> -JJ

Hi,
I don't know the other solutions, but maybe this one will also work
for you :)

```
wh = where(total(((c=abs(rebin(transpose(a[*]),$
    ((nb=n_elements(b))),((na=n_elements(a))),/sample) - $
    rebin(b[*],nb,na,/sample)))) eq 0,1))
```

Cheers

CR

Subject: Re: How to get matching elements of array efficiently
Posted by Jeremy Bailin on Wed, 24 Feb 2010 14:46:40 GMT
View Forum Message <> Reply to Message

On Feb 23, 3:36 pm, pp <pp.pente...@gmail.com> wrote:
> On Feb 23, 5:11 pm, JJ <j...@cornell.edu> wrote:
>
>> I'm looking for an efficient (non-loop) solution to this problem. I
>> have two arrays A, B. The elements of A are not necessarily unique,
>> nor is A necessarily sorted. I'd like to find all the indices in A
>> that match values that occur in B.
>
>> For example, if A = [7,1,8,7,8], B = [7,8], the result should be
>> [0,2,3,4].
>
>> A and B can have many (order millions) of unique values, so I'd rather
>> avoid a loop if I can.
>
>> Is there any way to do this efficiently?
>
> Are your values integers as in the example? If they are, and if there
> are no big gaps in B (the number of elements of B is not much smaller
> than max(B)-min(B)), histogram comes to mind:
>
> h=histogram(A,binsize=1,min=min(b),max=max(b),reverse_indice s=ri)
> if (max(h) gt 0L) then res=ri[n_elements(h)+1:*] else (deal with the
> case of none found in b)
>
> Which would give the result you want in res, but ordered by bin (their
> order in B). In this example, res would be [0,3,2,4].

Of course, I'll have to pipe in here and say that if your values
aren't integers, or if there are large gaps in them, you can use UNIQ
and VALUE_LOCATE to map them into a set of consecutive integers, and
then use HIstOGRAM. :-)=

-Jeremy.

Subject: Re: How to get matching elements of array efficiently
Posted by JDS on Wed, 24 Feb 2010 22:26:22 GMT
View Forum Message <> Reply to Message

On Feb 23, 3:11 pm, JJ <j...@cornell.edu> wrote:
> Hi All,
>
> I'm looking for an efficient (non-loop) solution to this problem. I
> have two arrays A, B. The elements of A are not necessarily unique,
> nor is A necessarily sorted. I'd like to find all the indices in A
> that match values that occur in B.
>
> For example, if A = [7,1,8,7,8], B = [7,8], the result should be
> [0,2,3,4].
>
> A and B can have many (order millions) of unique values, so I'd rather
> avoid a loop if I can.
>
> Is there any way to do this efficiently?

http://www.dfanning.com/tips/set_operations.html

Subject: Re: How to get matching elements of array efficiently
Posted by JJ on Mon, 01 Mar 2010 18:58:31 GMT
View Forum Message <> Reply to Message

Thanks for the solutions everyone. I didn't know that Craig Markwardt
had written a solution. I'll probably fetch that. In the mean time,
I was able to use the histogram with reverse_indices method.

JD, I'm not sure that the code you pointed me at would work for my
case, as I need the indices of all matches, even if they are
duplicates.

-JJ

Subject: Re: How to get matching elements of array efficiently
Posted by JJ on Mon, 01 Mar 2010 19:18:31 GMT
View Forum Message <> Reply to Message

>
> Are your values integers as in the example? If they are, and if there
> are no big gaps in B (the number of elements of B is not much smaller
> than max(B)-min(B)), histogram comes to mind:
>
> h=histogram(A,binsize=1,min=min(b),max=max(b),reverse_indice s=ri)

> if (max(h) gt 0L) then res=ri[n_elements(h)+1:*] else (deal with the
> case of none found in b)
>
> Which would give the result you want in res, but ordered by bin (their
> order in B). In this example, res would be [0,3,2,4].

What I actually used was something akin to this:

```
h = histogram(a,min=0,max=(max(a) > max(b)),reverse_indices=ri)
w = where(h[b] gt 0)
res = ri[[ri[b[w]],ri[b[w]+1]-1]]
```

-JJ
