Subject: Re: Finding a value in a array efficiently
Posted by David Fanning on Thu, 13 Apr 2006 20:49:44 GMT
View Forum Message <> Reply to Message

Wayne Landsman writes:

> So what we need is a new ARRAY_OR(x,y) function which returns 1 as
> soon as it finds any match between X and Y. Or am I missing another
> method?

A long shot, but if the array is monotonic, you could
try VALUE_LOCATE. Then you have to only test the value
you are close to. Sorting the array first would probably
take this outside the "efficient" category.

Cheers,

DAvid

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: Finding a value in a array efficiently
Posted by JD Smith on Thu, 13 Apr 2006 22:39:59 GMT
View Forum Message <> Reply to Message

On Thu, 13 Apr 2006 13:31:42 -0700, Wayne Landsman wrote:

> I was asked an apparently simple question -- what is the most efficient
> way in IDL to determine if a particular scalar value is in an array.
[..]
> A more efficent (if less transparent) method might be
>
> found = 1 - array_equal(array EQ value, 0)
>
> where (array EQ value) will contain all zeros if there is no match.
> The ARRAY_EQUAL(x,0) function returns a value of zero as soon as it
> finds a non-zero value in x. So we are part way there but (array EQ
> value) still requires testing every value of "array".
>
> So what we need is a new ARRAY_OR(x,y) function which returns 1 as
> soon as it finds any match between X and Y. Or am I missing another
> method? --Wayne

Just negate your ARRAY_EQUAL. I use this all the time:

 in_array=~array_equal(array NE value,1b)

In English that's "is it not everywhere not equal to the value". As soon as it comes across (array NE value)=0 (i.e. a place where it's "not not equal") it will return.

JD

Subject: Re: Finding a value in a array efficiently
Posted by JD Smith on Thu, 13 Apr 2006 23:00:45 GMT
View Forum Message <> Reply to Message

On Thu, 13 Apr 2006 13:31:42 -0700, Wayne Landsman wrote:

> I was asked an apparently simple question -- what is the most efficient
> way in IDL to determine if a particular scalar value is in an array.
>   The loop method is straightforward:
>
> found = 0
> for i=0,n-1 do begin
>   if array[i] EQ value then begin
>   found = 1
>   goto, DONE
>   endif
> endfor
> DONE:
>
> This ugly code will end the loop as soon as a match is found.
> A first attempt at vectorized code might be

As an aside, do you avoid using BREAK instead of goto for compatibility with old IDL versions?

JD

Subject: Re: Finding a value in a array efficiently
Posted by Wayne Landsman on Fri, 14 Apr 2006 01:58:29 GMT
View Forum Message <> Reply to Message

"JD Smith" <jdsmith@as.arizona.edu> wrote in message news:pan.2006.04.13.23.00.45.990566@as.arizona.edu...
> As an aside, do you avoid using BREAK instead of goto for compatibility
> with old IDL versions?

>

Actually, today I decided to set the minimum IDL version for my code to V5.4
instead of V5.3. V5.4 is when both BREAK and ARRAY_EQUAL() were
introduced. So after first updating the the GOTOs to BREAK, I started
to look if the code could be more efficient using ARRAY_EQUAL(). (I
didn't immediately realize, though, that BREAK could be safely placed
inside of an inner ENDIF clause, and it would still break out of the FOR
loop.)

>> found = 1 - array_equal(array EQ value, 0)

> Just negate your ARRAY_EQUAL. I use this all the time:
>
 >in_array=~array_equal(array NE value,1b)

This is still not quite optimal since the term (array NE value) is always
evaluated for the entire array. That is why I was looking for a function,
say, ARRAY_OR for which one could write

in_array = array_or(array,value)

which would return 1 as soon as it found an element of array equal to value,
and not perform any operations on the remaining elements of the array.

--Wayne

Subject: Re: Finding a value in a array efficiently
Posted by JD Smith on Fri, 14 Apr 2006 18:05:02 GMT
View Forum Message <> Reply to Message

On Fri, 14 Apr 2006 01:58:29 +0000, Wayne Landsman wrote:

>
> "JD Smith" <jdsmith@as.arizona.edu> wrote in message
> news:pan.2006.04.13.23.00.45.990566@as.arizona.edu...
>> As an aside, do you avoid using BREAK instead of goto for compatibility
>> with old IDL versions?
>>
>
> Actually, today I decided to set the minimum IDL version for my code to V5.4
> instead of V5.3. V5.4 is when both BREAK and ARRAY_EQUAL() were
> introduced. So after first updating the the GOTOs to BREAK, I started
> to look if the code could be more efficient using ARRAY_EQUAL(). (I
> didn't immediately realize, though, that BREAK could be safely placed
> inside of an inner ENDIF clause, and it would still break out of the FOR
> loop.)

Yep, it works much like C's break.

>>> found = 1 - array_equal(array EQ value, 0)
>
>> Just negate your ARRAY_EQUAL. I use this all the time:
>>
>> in_array=~array_equal(array NE value,1b)
>
> This is still not quite optimal since the term (array NE value) is always
> evaluated for the entire array.   That is why I was looking for a function,
> say, ARRAY_OR for which one could write
>
> in_array = array_or(array,value)
>
> which would return 1 as soon as it found an element of array equal to value,
> and not perform any operations on the remaining elements of the array.

I decided to check, and at least for my machine, evaluating "array NE value" is sufficiently slower than "array EQ value" (about 1.8x, not sure why), that the hypothetical speedup never materializes. In fact, now that I think about it a little more (and contrary to what you said), I see your method *does* stop as soon as it finds a match as well (and the timing confirms that). Both do require doing a boolean operation on the entire array first though. So, I think:

 ~array_equal(array EQ value,0b)

is your best bet.

I think you want an ARRAY_NE. I'm sure RSI would be happy to add that, but then you'd need to require IDL 7.0 ;).

JD