Subject: Re: Intersection of 2 sets--Beginner IDL question
Posted by John-David T. Smith on Wed, 17 Oct 2001 16:51:24 GMT
View Forum Message <> Reply to Message

Craig Markwardt wrote:
>
> John-David Smith <jdsmith@astro.cornell.edu> writes:
>
>> David Fanning wrote:
>>>
>>> Ted Cary (tedcary@yahoo.com) writes:
>>>
>>>> Is there any programming technique for finding the intersection of two sets
>>>> (arrays) of numbers without using WHERE in a loop to search the larger array
>>>> for every element in the smaller array? It seems like a very clumsy way to
>>>> find values shared by both arrays, especially with integer sets/arrays.
>>>
>>> How about the very tiny program, SetIntersection,
>>> which uses--what else--a Histogram! :-)
>>>
>>> http://www.dfanning.com/tips/set_operations.html
>>
>>
>> It's amazing how much recycled information flows through the newsgroup, if you
>> watch it long enough. I remember just like it was 4 years ago the detailed
>> discussions with which we whiled away our days, concerning value-based
>> intersection vs index-based intersection, order N vs. unknown order operations,
>> etc.
>>
> ...
>> In a classic example posed by Mark Fardal, you are matching up social security
>> numbers in two lists containing age and income. The set_intersection style
>> solution fails miserably here, and to a lesser degree for any arrays which are
>> somewhat sparse (where *somewhat* seems to be about 1 in 10, depending on lots
>> of factors).
>
> Hi JD--
>
> Thanks for beating me to the punch. The HISTOGRAM method is indeed
> very cool for a new learner, but it definitely starts to suck air (and
> memory) when the data sets become sparse.
>
> Long ago (1 year?) I tried to collect all the various algorithms that
> were being discussed, and some that weren't yet, to do set operations.
> CMSET_OP has the dreaded "CM" prefix, but it also knows how to do
> intersections, unions, and exclusive or's. It can do X AND NOT Y type
> intersections as well, in one self contained function.
>

> The syntax is:
>
> x_and_y = cmset_op(X, 'AND', y)
>
> It can return by value or index.

Ahah, a nice update since last I looked. I'm sure the exact break between histogram vs. sort is machine dependent, but your defaults seem logical.

There's one more thing I should point out in support of the much maligned ARRAY method, as exemplified in the where_array() routine originally by Dan Carr at RSI: it works for *any* IDL type.

In as much as comparisons like:

```
a=ptr_new('test') & b=a
print, b eq a
```

and

```
a=obj_new('myClass') & b=a
print, b eq a
```

work, you can do intersections on lists of pointers, lists of objects, etc., by using the array method. The underlying IDL operation which is data-type agnostic is simply array indexing, so in the context of the REFORM/REBIN tutorial, you can use the awkward "lindgen(n,m) mod m"-type method (of which where_array is a special case) to perform flexible operations on any type of array. Just beware of the N^2 performance.

I'm also not sure how sort is defined on pointer and object arrays... probably by heap variable number, in which case that one should work too.

JD
