Subject: Re: Array Questions
Posted by JD Smith on Fri, 13 Apr 2007 18:32:04 GMT
View Forum Message <> Reply to Message

On Fri, 13 Apr 2007 09:19:13 -0700, Eric Hudson wrote:

```
> Hi,
>
> I was wondering if anyone has any tricks for a couple of things I want
> to do with arrays.
>
> 1)  Remove columns & rows
> I have a 2D (NxN) array and want to remove column i and row j.  i,j
> might be 0, they might be N-1, they might be anything in between.  Is
> there something simpler than handling these 3 different cases
> separately and indexing [[ARR[*,0:j-1]],[ARR[*,j+1:N]]]...
```

Build the index list yourself. This operation will never be fast, since
it requires building an array of indices the size of the array, and a new
array to copy the original into. It's debatable whether this is any
better than a case statement:

```
IDL> a=bindgen(5,5)
IDL> colkill=0 & rowkill=3
IDL> cols=where(indgen(a[0]) ne colkill)
IDL> rows=where(indgen(a[1]) ne rowkill)
IDL> s=size(a,/DIMENSIONS)
IDL> print,a[rebin(cols,s-1),rebin(1#rows,s-1)]
```

Of course, if you need to perform such an operation again and again on
the same sized array, it's *always* preferable to build your own index
array(s) and reuse them, rather than have IDL build them anew each
time (which is all it does when it encounters *,a:b and other indexing
constructs).

```
> 2)  Complements of sets
> Often I end up with a set of element numbers and want to divide the
> array into two pieces, those in the set and those not in the set.
> Getting those in the set is trivial:  ARR[set].  Is there a quick way of
> generating the complement?  It's nice that where() will give you the
> complement, but I don't always get my sets from where.
```

General information about set intersections etc. are discussed in some
detail at:

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

It's probably more than you care to know about, since there are versions using HISTOGRAM (good for integers with fairly uniform spacing), array comparison (fastest for small sets), SORT (best for large, sparse, and/or non-integer data sets). I often use something called WHERE_NOT_ARRAY for just this, which uses array methods.

Yours is somewhat of a special case, since you know a priori that there are bounds on the maximum set range (total number of array elements), so you aren't as worried about unlimited memory use from some arbitrary set (like [0,100,100000,100000000000LL]). In this case, HISTOGRAM + WHERE will very quickly get you to the answer:

```
complement=where(histogram(set,MIN=0,MAX=n_elements(arr)-1) eq 0,cnt)
```

In fact, this is the same answer from the HISTOGRAM tutorial of the problem: "Remove some elements, listed in random order, from a vector."

```
>
>  3)  "in set"
>  This isn't really isn't an array question, but I thought of it writing
>  the previous question.  It's easy to use where to find elements of an
>  array that are equal to some value.  Is there a quick way to find
>  elements that are in a set of values, something like:
>
>  where(Arr in set)
```

WHERE_ARRAY is the "standard" for this, but again, there are many flavors (SORT/HISTOGRAM/ARRAY), summarized on the page linked above. Even though the SORT based method has some very nice advantages, it also is (unfortunately) sensitive to the implementation of SORT in IDL, which changes from architecture to architecture. For small sets, I usually end up using WHERE_ARRAY. I should probably roll this into a real uber set-slicing tool at some point.

JD