Subject: Re: Efficient comparison of arrays / Set operations
Posted by J.D. Smith on Mon, 01 Sep 1997 07:00:00 GMT
View Forum Message <> Reply to Message

Research Systems Inc. wrote:
>
> A somewhat belated reply to the numerous postings on finding the
> common elements of vectors:
>
>> Given vectors of the type...
>>
>> a = [1,2,3,4,5]
>> b = [3,4,5,6,7]
>>
>> What is the most efficient way to determine which values that occur in
>> a also occur in b (i.e., the values [3,4,5] occur in both a and b).
>>
>
> Below appear three IDL functions that operate on sets represented by
> arrays of positive integers. The SetIntersection(a,b) function
> returns the common elements, SetUnion(a,b) returns all unique elements
> in both arguments, and SetDifference(a,b) returns the elements
> (members) in a but not in b.
>
> It is faster than previously published functions, e.g. contain() and
> find_elements().
>
> Hope this helps,
>
> Research Systems, Inc.
> _____ _____
> ; Set operators. Union, Intersection, and Difference (i.e. return
> ; members of A that are not in B.)
> ;
> ; These functions operate on arrays of positive integers, which need
> ; not be sorted. Duplicate elements are ignored, as they have no
> ; effect on the result.
> ;
> ; The empty set is denoted by an array with the first element equal to
> ; -1.
> ;
> ; These functions will not be efficient on sparse sets with wide
> ; ranges, as they trade memory for efficiency. The HISTOGRAM function
> ; is used, which creates arrays of size equal to the range of the
> ; resulting set.

Very fast for non-sparse sets! But this last comment is important to

note. For sparse, large sets, other methods posted previously remain superior (given finite memory). I find contain() outperforms SetIntersection() on my 64 MB machine when the two vectors are roughly 1 in 25 sparse (i.e. when only 1 out of every group of 25 numbers, on average, exists in the vector). This will vary with your memory. A simple fix for negative numbers too would be useful.

JD
