Subject: Re: Efficient comparison of arrays / Set operations
Posted by davidf on Fri, 29 Aug 1997 07:00:00 GMT
View Forum Message <> Reply to Message

Research Systems Inc. writes:

> 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.

Hey, way to go, RSI. Nice post! I, for one, think you guys
should post more often. Lord knows, we can often use the help. :-)

Cheers,

David

David Fanning, Ph.D.
Fanning Software Consulting
Customizable IDL Programming Courses
Phone: 970-221-0438 E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com

Subject: Re: Efficient comparison of arrays / Set operations
Posted by Research Systems Inc. on Fri, 29 Aug 1997 07:00:00 GMT
View Forum Message <> Reply to Message

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.

; For example:
;   a = [2,4,6,8]
;   b = [6,1,3,2]
; SetIntersection(a,b) = [ 2, 6]       ; Common elements
; SetUnion(a,b) = [ 1, 2, 3, 4, 6, 8]  ; Elements in either set
; SetDifference(a,b) = [ 4, 8]         ; Elements in A but not in B
; SetIntersection(a,[3,5,7]) = -1 = Null Set

FUNCTION SetUnion, a, b
if a[0] lt 0 then return, b    ;A union NULL = a
if b[0] lt 0 then return, a    ;B union NULL = b
return, where(histogram([a,b], OMIN = omin)) + omin ;Return combined set
end


FUNCTION SetIntersection, a, b
minab = min(a, MAX=maxa) > min(b, MAX=maxb) ;Only need intersection of ranges
maxab = maxa < maxb

  ;If either set is empty, or their ranges don't intersect: result = NULL.
if maxab lt minab or maxab lt 0 then return, -1
r = where((histogram(a, MIN=minab, MAX=maxab) ne 0) and  $
         (histogram(b, MIN=minab, MAX=maxab) ne 0), count)
if count eq 0 then return, -1 else return, r + minab
end
```

```
FUNCTION SetDifference, a, b  ; = a and (not b) = elements in A but not
in B
mina = min(a, MAX=maxa)
minb = min(b, MAX=maxb)
if (minb gt maxa) or (maxb lt mina) then return, a ;No intersection...
r = where((histogram(a, MIN=mina, MAX=maxa) ne 0) and $
        (histogram(b, MIN=mina, MAX=maxa) eq 0), count)
if count eq 0 then return, -1 else return, r + mina
end
```

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

Subject: Re: Efficient comparison of arrays / Set operations
Posted by davidf on Wed, 10 Sep 1997 07:00:00 GMT
View Forum Message <> Reply to Message

Saeid Zoonematkermani writes:

>> Research Systems Inc. writes:
>>
>>> 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.
>>
>
> Ooops! I was gone for a while and missed the posting of the above
> functions. Could some one repost the functions, or make them available at
> a web site some where. Thanks,

You can find them on my web page. The URL is:

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

Cheers,

David

David Fanning, Ph.D.
Fanning Software Consulting
Customizable IDL Programming Courses
Phone: 970-221-0438 E-Mail: davidf@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com

Subject: Re: Efficient comparison of arrays / Set operations
Posted by szoonem on Wed, 10 Sep 1997 07:00:00 GMT
View Forum Message <> Reply to Message

In article <MPG.e70ebcaf9a84bd79896f2@news.frii.com>, davidf@dfanning.com
(David Fanning) wrote:

> Research Systems Inc. writes:
>
>> 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.
>
> Hey, way to go, RSI. Nice post! I, for one, think you guys
> should post more often. Lord knows, we can often use the help. :-)
>
> Cheers,
>
> David
>

Ooops! I was gone for a while and missed the posting of the above
functions. Could some one repost the functions, or make them available at
a web site some where. Thanks,

- Saeid

Saeid Zoonematkermani	E-Mail: szoonem@astro.sunysb.edu
Dept. of Physics & Astronomy	Voice: (+1) (516) 632-8237
State University of New York	Fax: (+1) (516) 632-8742
Stony Brook, NY 11794-3800	

| Home Page --> http://ozone.ess.sunysb.edu/ |
