Subject: Search routines
Posted by bowman on Fri, 18 Sep 1998 07:00:00 GMT
View Forum Message <> Reply to Message

IDL has a pretty good SORT routine, but no SEARCH routine that I have been able to find (that is, a procedure to find the index of the closest/first match in an ordered list). Once again, this can be done with loops, but such an implementation would almost certainly be much slower than a built-in function. Since searching and sorting are such basic operations, does anyone know why there is no SEARCH in IDL?

Ken Bowman

--

Dr. Kenneth P. Bowman, Professor 409-862-4060
Department of Meteorology 409-862-4466 fax
Texas A&M University bowmanATcsrp.tamu.edu
College Station, TX 77843-3150 Replace AT with @

Subject: Re: Search routines
Posted by Martin Schultz on Mon, 21 Sep 1998 07:00:00 GMT
View Forum Message <> Reply to Message

Kenneth P. Bowman wrote:
>

>> Kenneth P. Bowman wrote:
>>> IDL has a pretty good SORT routine, but no SEARCH routine [...]

Well, how about the one attached? I tested it with a 10000 element array, and it is in fact about a factor of ten faster than MIN or WHERE.

Enjoy,
Martin.

---------- -------
Dr. Martin Schultz
Department for Earth&Planetary Sciences, Harvard University
109 Pierce Hall, 29 Oxford St., Cambridge, MA-02138, USA

phone: (617)-496-8318
fax : (617)-495-4551

e-mail: mgs@io.harvard.edu
Internet-homepage: http://www-as.harvard.edu/people/staff/mgs/
---------- -------

```
;------------------------------------------------------------ --
;+
; NAME:
;       SEARCH (function)
;
; PURPOSE:
;       Perform a binary search for the data point closest
;       to a given value. Data must be sorted.
;
; CATEGORY:
;       Math
;
; CALLING SEQUENCE:
;       index = SEARCH( DATA, VALUE )
;
; INPUTS:
;       DATA -> a sorted data vector
;
;       VALUE -> the value to look for
;
; KEYWORD PARAMETERS:
;       none.
;
; OUTPUTS:
;       The function returns the index of the nearest data
;       point.
;
; SUBROUTINES:
;
; REQUIREMENTS:
;
; NOTES:
;       This routine is much faster than WHERE or MIN for
;       large arrays. It was written in response to a newsgroup
;       request by K.P. Bowman.
;
; EXAMPLE:
;       test = findgen(10000)
;       print,search(test,532.3)
;       ; prints 532
;
; MODIFICATION HISTORY:
;       mgs, 21 Sep 1998: VERSION 1.00
;
;-
; Copyright (C) 1998, Martin Schultz, Harvard University
; This software is provided as is without any warranty
```

```
; whatsoever. It may be freely used, copied or distributed
; for non-commercial purposes. This copyright notice must be
; kept with any copy of this software. If this software shall
; be used commercially or sold as part of a larger package,
; please contact the author to arrange payment.
; Bugs and comments should be directed to mgs@io.harvard.edu
; with subject "IDL routine search"
 ;--------------------------------------------------------- --


function search,data,value


    ; search first occurence of value in data set
    ; data must be sorted

    ; simple error checking on data and value
    if (n_elements(value) eq 0) then begin
       message,'Must supply sorted data array and value),/CONT
       return
    endif

    ndat = n_elements(data)

    try = fix(0.5*ndat)
    step = 0.5*try

    ; find index of nearest points
    while (step gt 1) do begin
       if (data[try] gt value) then $
          try = try-step $
       else   $
          try = try+step
       step = fix(0.5*(step+1))
    endwhile

    ; now get the data point closest to value
    ; can only be one out of three (try-1, try, try+1)
    dummy = min( abs(value-data[try-1:try+1]), location )

    return,try+location-1

end
```

File Attachments

1) search.pro, downloaded 131 times

Subject: Re: Search routines
Posted by R. Bauer on Sat, 26 Sep 1998 07:00:00 GMT
View Forum Message <> Reply to Message

Kenneth P. Bowman wrote:

> In article <3602D89B.15BF8C2D@ssec.wisc.edu>, Liam Gumley
> <Liam.Gumley@ssec.wisc.edu> wrote:
>
>> Kenneth P. Bowman wrote:
>>> IDL has a pretty good SORT routine, but no SEARCH routine that I have been
>>> able to find (that is, a procedure to find the index of the closest/first
>>> match in an ordered list). Once again, this can be done with loops, but
>>> such an implementation would almost certainly be much slower than a
>>> built-in function. Since searching and sorting are such basic operations,
>>> does anyone know why there is no SEARCH in IDL?
>>
>> How about the MIN function, e.g.
>>
>> array = findgen(100)
>> value = 37.2
>> result = min(abs(value - array), location)
>> help, location
>
> Again, I'm sure this is an order-N operation, as MIN has to check every
> element, just like WHERE. It has no knowledge that the list is ordered.
>
> Ken
>

Hi Ken,

look at this routine.

The name should mean find_middling_indices. It was written in the past where we
have not the possibilty to use long names.
The help part could be added if you are interested.
In addition to a normal search this routine is able to use a window which is
usefull for some algorithm we have written.

We are thinking that's this routine is very fast, but any ideas to get it faster
are welcome.

Reimar

--
R.Bauer

Institut fuer Stratosphaerische Chemie (ICG-1)
Forschungszentrum Juelich
email: R.Bauer@fz-juelich.de

```
;
; Copyright (c) 1996, Forschungszentrum Juelich GmbH ICG-1
; All rights reserved.
; Unauthorized reproduction prohibited.
; This software may be used, copied, or redistributed as long as it is not
; sold and this copyright notice is reproduced on each copy made.  This
; routine is provided as is without any express or implied warranties
; whatsoever.
;+
; NAME:
;       fi_mi_in
;
; PURPOSE:
;   The result of this function is a two dimensional indexfield.
;   It is used to find the indices which overlaps in client time depending on
master time and time_window
;   The first index is the start and the second the end index of the overlapping
values from client_time.
;
; CATEGORY:
;   MATH
;
; CALLING SEQUENCE:
;   Result=fi_mi_in(client,master,master_time_window,[/help])
;
; INPUTS:
;       client:            The client time
;       master:              The master time
;   master_time_window: The time_window must have same size as master
;
; KEYWORD PARAMETERS:
;  help: gives a short description
;
; OUTPUTS:
;   This function returns a two dimensional indexfield
;   The first index is the start and the second the end index of the used time
window
;   -1 at an index means there were no results
;
; EXAMPLE:
```

```
;      client=[1,2,3,4,5]
;      master=[2,4]
;      time_window=[0,0]
;      Result=fi_mi_in(client,master,time_window)
;      help,result
;      RESULT          LONG      = Array[2, 2]
;      print,result
;           1           1
;           3           3
;
;
; MODIFICATION HISTORY:
;   Written by: R.Bauer (ICG-1),  1996-May-06
;   1998-Mar-02 much more efficiency by combining with suche.pro (F.Rohrer)
;
;-

FUNCTION fi_mi_in,client,master,time_window,help=help

 if keyword_set(help) then begin
     help,call=call
     help_of_interest=within_brackets(call[0],brackets=['<','('])
     message,help_calling_sequence(help_of_interest),/cont
     return,-1
   endif


   laenge_master=(size(master))(1)
   lstx=(size(client))(1)-1


   remember=0
   index=MAKE_ARRAY(2,laenge_master,type=3,value=-1)



   FOR i=0L, laenge_master-1 DO BEGIN
      k = LONG(remember)
      WHILE client(k) LT master(i) -time_window(i)  AND k LT lstx DO k=k+1
      IF client(k) GE master(i) -time_window(i)  and client(k) LE
master(i)+time_window(i) THEN BEGIN
         index(0,i)=LONG(k)
         remember=(index(0,i))(0)
      ENDIF

      if index(0,i) ne -1 then begin
         mx = remember
         WHILE  client(mx) LE master(i)+time_window(i)  and mx LT lstx DO mx=mx+1
```

```
        if client(mx) LE master(i)+time_window(i) then index(1,i)=LONG(mx) else
index(1,i)=LONG(mx)-1
      endif

   ENDFOR

   RETURN,index
END
```

Subject: Re: search routine
Posted by Paolo Grigis on Fri, 01 Jun 2007 13:11:38 GMT
View Forum Message <> Reply to Message

Laurens wrote:
> Hi folks,
>
> From Martin Schultz (posted in 1999) I found the following array-search
> algorithm which seems to do a fine job.
> Except that i'm not able to catch the first element in the array.
>
> Example:
>
> Array = [0,80,100,120,180,300]
> result = search, Array, 4.53
>
> It should return index 0, if I understand it correctly, but it returns 1
> instead. Now I don't quite follow the logic of the function, so maybe
> someone for which it's easy to see can help me in the right direction?

You could use the built-in function value_locate instead:

result=value_locate(array,4.53)

which returns 0.

Ciao,
Paolo

Subject: Re: search routine
Posted by cmancone on Fri, 01 Jun 2007 13:32:58 GMT
View Forum Message <> Reply to Message

On Jun 1, 9:11 am, Paolo Grigis <pgri...@astro.phys.ethz.ch> wrote:
> Laurens wrote:

>> Hi folks,
>
>> From Martin Schultz (posted in 1999) I found the following array-search
>> algorithm which seems to do a fine job.
>> Except that i'm not able to catch the first element in the array.
>
>> Example:
>
>> Array = [0,80,100,120,180,300]
>> result = search, Array, 4.53
>
>> It should return index 0, if I understand it correctly, but it returns 1
>> instead. Now I don't quite follow the logic of the function, so maybe
>> someone for which it's easy to see can help me in the right direction?
>
> You could use the built-in function value_locate instead:
>
> result=value_locate(array,4.53)
>
> which returns 0.
>
> Ciao,
> Paolo

If you wanted to program it up, you'd be better off with array
operations anyway, something like this:

function search_array, arr, val
w = where(arr - val le 0 AND shift(arr,-1) - val ge 0)
return,w

Subject: Re: search routine
Posted by cmancone on Fri, 01 Jun 2007 13:34:47 GMT
View Forum Message <> Reply to Message

On Jun 1, 9:11 am, Paolo Grigis <pgri...@astro.phys.ethz.ch> wrote:
> Laurens wrote:
>> Hi folks,
>
>> From Martin Schultz (posted in 1999) I found the following array-search
>> algorithm which seems to do a fine job.
>> Except that i'm not able to catch the first element in the array.
>
>> Example:
>
>> Array = [0,80,100,120,180,300]

>> result = search, Array, 4.53
>
>> It should return index 0, if I understand it correctly, but it returns 1
>> instead. Now I don't quite follow the logic of the function, so maybe
>> someone for which it's easy to see can help me in the right direction?
>
> You could use the built-in function value_locate instead:
>
> result=value_locate(array,4.53)
>
> which returns 0.
>
> Ciao,
> Paolo

Actually, that's not so good. It assumes your array is always
increasing, and positive. If you used positive values, you would find
that my above example would report the index AFTER where the value
appears...

Subject: Re: search routine
Posted by Laurens on Fri, 01 Jun 2007 13:38:26 GMT
View Forum Message <> Reply to Message

Paolo Grigis wrote:
>
>
> Laurens wrote:
>> Hi folks,
>>
>> From Martin Schultz (posted in 1999) I found the following
>> array-search algorithm which seems to do a fine job.
>> Except that i'm not able to catch the first element in the array.
>>
>> Example:
>>
>> Array = [0,80,100,120,180,300]
>> result = search, Array, 4.53
>>
>> It should return index 0, if I understand it correctly, but it returns
>> 1 instead. Now I don't quite follow the logic of the function, so
>> maybe someone for which it's easy to see can help me in the right
>> direction?
>
> You could use the built-in function value_locate instead:
>
> result=value_locate(array,4.53)

>
> which returns 0.
>
> Ciao,
> Paolo
>
hehe, thanks a lot~! That's the one I was looking for...

Subject: Re: search routine
Posted by Laurens on Fri, 01 Jun 2007 13:42:39 GMT
View Forum Message <> Reply to Message

Paolo Grigis wrote:
>
>
> Laurens wrote:
>> Hi folks,
>>
>> From Martin Schultz (posted in 1999) I found the following
>> array-search algorithm which seems to do a fine job.
>> Except that i'm not able to catch the first element in the array.
>>
>> Example:
>>
>> Array = [0,80,100,120,180,300]
>> result = search, Array, 4.53
>>
>> It should return index 0, if I understand it correctly, but it returns
>> 1 instead. Now I don't quite follow the logic of the function, so
>> maybe someone for which it's easy to see can help me in the right
>> direction?
>
> You could use the built-in function value_locate instead:
>
> result=value_locate(array,4.53)
>
> which returns 0.
>
> Ciao,
> Paolo
>
though i'm noticing that it takes its ground number to be returned.
If my array has [0,10,20,30] and i'm searching for 18, it will return
10. Now its just that would want to get 20 instead of 10 :)

regards, Laurens

Subject: Re: search routine
Posted by Paolo Grigis on Fri, 01 Jun 2007 14:20:04 GMT
View Forum Message <> Reply to Message

cmancone@ufl.edu wrote:
> On Jun 1, 9:11 am, Paolo Grigis <pgri...@astro.phys.ethz.ch> wrote:
>> Laurens wrote:
>>> Hi folks,
>>> From Martin Schultz (posted in 1999) I found the following array-search
>>> algorithm which seems to do a fine job.
>>> Except that i'm not able to catch the first element in the array.
>>> Example:
>>> Array = [0,80,100,120,180,300]
>>> result = search, Array, 4.53
>>> It should return index 0, if I understand it correctly, but it returns 1
>>> instead. Now I don't quite follow the logic of the function, so maybe
>>> someone for which it's easy to see can help me in the right direction?
>> You could use the built-in function value_locate instead:
>>
>> result=value_locate(array,4.53)
>>
>> which returns 0.
>>
>> Ciao,
>> Paolo
>
>
> If you wanted to program it up, you'd be better off with array
> operations anyway, something like this:
>
> function search_array, arr, val
> w = where(arr - val le 0 AND shift(arr,-1) - val ge 0)
> return,w

"where" is much slower, so I would not recommend it.

Ciao,
Paolo

>

Subject: Re: search routine
Posted by Paolo Grigis on Fri, 01 Jun 2007 14:23:06 GMT
View Forum Message <> Reply to Message

cmancone@ufl.edu wrote:
> On Jun 1, 9:11 am, Paolo Grigis <pgri...@astro.phys.ethz.ch> wrote:

>> Laurens wrote:
>>> Hi folks,
>>> From Martin Schultz (posted in 1999) I found the following array-search
>>> algorithm which seems to do a fine job.
>>> Except that i'm not able to catch the first element in the array.
>>> Example:
>>> Array = [0,80,100,120,180,300]
>>> result = search, Array, 4.53
>>> It should return index 0, if I understand it correctly, but it returns 1
>>> instead. Now I don't quite follow the logic of the function, so maybe
>>> someone for which it's easy to see can help me in the right direction?
>> You could use the built-in function value_locate instead:
>>
>> result=value_locate(array,4.53)
>>
>> which returns 0.
>>
>> Ciao,
>> Paolo
>
> Actually, that's not so good. It assumes your array is always
> increasing, and positive.

Yes, the array has to be sorted.
No, it doesn't have to be positive.

Ciao,
Paolo

If you used positive values, you would find
> that my above example would report the index AFTER where the value
> appears...
>

Subject: Re: search routine
Posted by Paolo Grigis on Fri, 01 Jun 2007 14:27:24 GMT
View Forum Message <> Reply to Message

Laurens wrote:
> Paolo Grigis wrote:
>>
>>
>> Laurens wrote:
>>> Hi folks,
>>>
>>> From Martin Schultz (posted in 1999) I found the following
>>> array-search algorithm which seems to do a fine job.

>>> Except that i'm not able to catch the first element in the array.
>>>
>>> Example:
>>>
>>> Array = [0,80,100,120,180,300]
>>> result = search, Array, 4.53
>>>
>>> It should return index 0, if I understand it correctly, but it
>>> returns 1 instead. Now I don't quite follow the logic of the
>>> function, so maybe someone for which it's easy to see can help me in
>>> the right direction?
>>
>> You could use the built-in function value_locate instead:
>>
>> result=value_locate(array,4.53)
>>
>> which returns 0.
>>
>> Ciao,
>> Paolo
>>
> though i'm noticing that it takes its ground number to be returned.
> If my array has [0,10,20,30] and i'm searching for 18, it will return
> 10. Now its just that would want to get 20 instead of 10 :)

well, in that case, just increase the index of the returned
value by one (though you'd better check that it wasn't the
last element...).

Ciao,
Paolo

>
> regards, Laurens
