Subject: Re: Ordered index array
Posted by David Fanning on Wed, 07 Sep 2005 17:01:57 GMT
View Forum Message <> Reply to Message

Emmanuel Christophe writes:

> I would like to change the values on an array according to the order
> they appear. Following this example:
>
> starting with the array
> [3,6,2,1,2,8,1,1]
> I would like to get
> [1,2,3,4,3,5,4,4]
>
> 3 being the first value seen it is affected to 1
> 6 being the second, affected to 2
> ...
>
> I guess this process is similar to lookup tables, or color tables. As I
> would like to process quite big arrays (>60000 long elements), i'm
> looking for an efficient function. I don't really know what to look for
> in IDL help to find something which could help.

How about this:

```
PRO TEST
a = [3,6,2,1,2,7,1,1]
h = Histogram(a, Reverse_Indices=ri, Min=0)
b = Indgen(N_Elements(h)) + 1
c = Intarr(N_Elements(h))
FOR j=0,N_Elements(h)-1 DO BEGIN
  IF ri[j+1] NE ri[j] THEN $
    c[ri[ri[j]:ri[j+1]-1]] = Min(b[ri[ri[j]:ri[j+1]-1]])
ENDFOR
Print, c
END
```

```
      1       2       3       4       3       6       4       4
```

Note that your original example is wrong. :-)

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.

Subject: Re: Ordered index array
Posted by David Fanning on Wed, 07 Sep 2005 17:17:00 GMT
View Forum Message <> Reply to Message

David Fanning writes:

```
>  How about this:
>
>  PRO TEST
>  a = [3,6,2,1,2,7,1,1]
>  h = Histogram(a, Reverse_Indices=ri, Min=0)
>  b = Indgen(N_Elements(h)) + 1
>  c = Intarr(N_Elements(h))
>  FOR j=0,N_Elements(h)-1 DO BEGIN
>     IF ri[j+1] NE ri[j] THEN $
>        c[ri[ri[j]:ri[j+1]-1]] = Min(b[ri[ri[j]:ri[j+1]-1]])
>  ENDFOR
>  Print, c
>  END
>
>        1       2       3       4       3       6       4       4
>
>  Note that your original example is wrong. :-)
```

Whoops! *I* didn't use the original example either (although
yours is still wrong!).

Here is a more complete solution, using the original data:

```
PRO TEST
a = [3,6,2,1,2,8,1,1]
h = Histogram(a, Reverse_Indices=ri, Min=0)
b = Indgen(N_Elements(h)) + 1
c = Intarr(N_Elements(h))
FOR j=0,N_Elements(h)-1 DO BEGIN
   IF ri[j+1] NE ri[j] THEN $
      c[ri[ri[j]:ri[j+1]-1]] = Min(b[ri[ri[j]:ri[j+1]-1]])
ENDFOR
Print, c[0:N_Elements(a)-1]
END
```

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: Ordered index array
Posted by Paolo Grigis on Thu, 08 Sep 2005 08:22:30 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
> David Fanning writes:
>
>
>> How about this:
>>
>> PRO TEST
>> a = [3,6,2,1,2,7,1,1]
>> h = Histogram(a, Reverse_Indices=ri, Min=0)
>> b = Indgen(N_Elements(h)) + 1
>> c = Intarr(N_Elements(h))
>> FOR j=0,N_Elements(h)-1 DO BEGIN
>> IF ri[j+1] NE ri[j] THEN $
>> c[ri[ri[j]:ri[j+1]-1]] = Min(b[ri[ri[j]:ri[j+1]-1]])
>> ENDFOR
>> Print, c
>> END
>>
>> 1 2 3 4 3 6 4 4
>>
>> Note that your original example is wrong. :-)
>
>
> Whoops! *I* didn't use the original example either (although
> yours is still wrong!).
>
> Here is a more complete solution, using the original data:
>
> PRO TEST
> a = [3,6,2,1,2,8,1,1]
> h = Histogram(a, Reverse_Indices=ri, Min=0)
> b = Indgen(N_Elements(h)) + 1
> c = Intarr(N_Elements(h))
> FOR j=0,N_Elements(h)-1 DO BEGIN
> IF ri[j+1] NE ri[j] THEN $
> c[ri[ri[j]:ri[j+1]-1]] = Min(b[ri[ri[j]:ri[j+1]-1]])
> ENDFOR

> Print, c[0:N_Elements(a)-1]
> END
But this will fail if 'a' has more elements than the number of
its different values, for instance a=[3,3,1,2,3,2,2,1,2,3].

One could try this:

```
PRO test

a=[3,3,1,2,3,2,2,1,2,3]

b=a
c=intarr(n_elements(a))

h=histogram(a,min=1,reverse_ind=ri)

done=0
rank=1
WHILE NOT done DO BEGIN
    actual_value=b[0]
    ind_actual_value=ri[ri[actual_value-1]:ri[actual_value]-1]
    c[ind_actual_value]=rank
    rank=rank+1
    indremove=where(b NE actual_value,count)
    IF count GT 0 THEN b=b[indremove] ELSE done=1
ENDWHILE

print,a
print,c

END
```

but it will get inefficient as the numbers of different values
in 'a' grows, as the code in the loop get called more and more
times...

Ciao,
Paolo

>
> Cheers,
>
> David
>
>

Subject: Re: Ordered index array
Posted by Emmanuel Christophe on Thu, 08 Sep 2005 08:48:08 GMT
View Forum Message <> Reply to Message

Sorry, my example was confusing:

if the input is
[2,2,2,2,2,2,2,2,2,5,8]
I would like an output like
[1,1,1,1,1,1,1,1,1,2,3]

The property that should be verified is the n-th symbol of the output is
either
- the same as one between 0 and (n-1)th position
- equal to max(output[0:n-1])+1

and output[0]=1

I thought also about using the reverse_indices of the histogram
function, but as it doesn't preserve the order, I'm not sure it would work.

Thanks,
Emmanuel

David Fanning a écrit :
> David Fanning writes:
>
>
>> How about this:
>>
>> PRO TEST
>> a = [3,6,2,1,2,7,1,1]
>> h = Histogram(a, Reverse_Indices=ri, Min=0)
>> b = Indgen(N_Elements(h)) + 1
>> c = Intarr(N_Elements(h))
>> FOR j=0,N_Elements(h)-1 DO BEGIN
>> IF ri[j+1] NE ri[j] THEN $
>> c[ri[ri[j]:ri[j+1]-1]] = Min(b[ri[ri[j]:ri[j+1]-1]])
>> ENDFOR
>> Print, c
>> END
>>
>> 1 2 3 4 3 6 4 4
>>
>> Note that your original example is wrong. :-)
>
>
> Whoops! *I* didn't use the original example either (although

> yours is still wrong!).
>
> Here is a more complete solution, using the original data:
>
> PRO TEST
> a = [3,6,2,1,2,8,1,1]
> h = Histogram(a, Reverse_Indices=ri, Min=0)
> b = Indgen(N_Elements(h)) + 1
> c = Intarr(N_Elements(h))
> FOR j=0,N_Elements(h)-1 DO BEGIN
> IF ri[j+1] NE ri[j] THEN $
> c[ri[ri[j]:ri[j+1]-1]] = Min(b[ri[ri[j]:ri[j+1]-1]])
> ENDFOR
> Print, c[0:N_Elements(a)-1]
> END
>
> Cheers,
>
> David
>
>

Subject: Re: Ordered index array
Posted by Emmanuel Christophe on Thu, 08 Sep 2005 08:58:54 GMT
View Forum Message <> Reply to Message

Thanks Paolo,
Your function does exactly what I want, but my problem is precisely that
I need to do it on huge array (300 000 elements at least), that why i'm
looking for a more 'IDL' way to do it :)

Emmanuel

>
> But this will fail if 'a' has more elements than the number of
> its different values, for instance a=[3,3,1,2,3,2,2,1,2,3].
>
> One could try this:
>
> PRO test
>
> a=[3,3,1,2,3,2,2,1,2,3]
>
> b=a
> c=intarr(n_elements(a))

```
>
>  h=histogram(a,min=1,reverse_ind=ri)
>
>  done=0
>  rank=1
>  WHILE NOT done DO BEGIN
>     actual_value=b[0]
>     ind_actual_value=ri[ri[actual_value-1]:ri[actual_value]-1]
>     c[ind_actual_value]=rank
>     rank=rank+1
>     indremove=where(b NE actual_value,count)
>     IF count GT 0 THEN b=b[indremove] ELSE done=1
>  ENDWHILE
>
>  print,a
>  print,c
>
>  END
>
>
>  but it will get inefficient as the numbers of different values
>  in 'a' grows, as the code in the loop get called more and more
>  times...
>
>  Ciao,
>  Paolo
>
>>
>>  Cheers,
>>
>>  David
>>
>>
```

Subject: Re: Ordered index array
Posted by Paolo Grigis on Thu, 08 Sep 2005 10:03:07 GMT
View Forum Message <> Reply to Message

Emmanuel Christophe wrote:
> Thanks Paolo,
> Your function does exactly what I want, but my problem is precisely that
> I need to do it on huge array (300 000 elements at least), that why i'm
> looking for a more 'IDL' way to do it :)
The problem is not in the number of *elements* of a, but
in the number of *different values* that a can take...
but yes, if this is large, you should not use that routine.

Paolo

>
> Emmanuel
>
>
>
>
>> But this will fail if 'a' has more elements than the number of
>> its different values, for instance a=[3,3,1,2,3,2,2,1,2,3].
>>
>> One could try this:
>>
>> PRO test
>>
>> a=[3,3,1,2,3,2,2,1,2,3]
>>
>> b=a
>> c=intarr(n_elements(a))
>>
>> h=histogram(a,min=1,reverse_ind=ri)
>>
>> done=0
>> rank=1
>> WHILE NOT done DO BEGIN
>> actual_value=b[0]
>> ind_actual_value=ri[ri[actual_value-1]:ri[actual_value]-1]
>> c[ind_actual_value]=rank
>> rank=rank+1
>> indremove=where(b NE actual_value,count)
>> IF count GT 0 THEN b=b[indremove] ELSE done=1
>> ENDWHILE
>>
>> print,a
>> print,c
>>
>> END
>>
>>
>> but it will get inefficient as the numbers of different values
>> in 'a' grows, as the code in the loop get called more and more
>> times...
>>
>> Ciao,
>> Paolo
>>
>>

>>> Cheers,
>>>
>>> David
>>>
>>>

Subject: Re: Ordered index array
Posted by Paolo Grigis on Thu, 08 Sep 2005 12:30:25 GMT
View Forum Message <> Reply to Message

Emmanuel Christophe wrote:
> Thanks Paolo,
> Your function does exactly what I want, but my problem is precisely that
> I need to do it on huge array (300 000 elements at least), that why i'm
> looking for a more 'IDL' way to do it :)
>
> Emmanuel
Ok, let's do it. We'll use a loop, but this should not be too
bad as long as you don't have more than a few million elements,
provided that we just do simple operations in the loop.

```
PRO test2

a=[1,3,3,1,2,3,2,2,1,2,3]

c=lonarr(n_elements(a))

h=histogram(a,min=1L,reverse_ind=ri)

listlowestind=replicate(-1L,n_elements(h))

FOR i=0L,n_elements(h)-1 DO BEGIN
    IF ri[i] NE ri[i+1] THEN listlowestind[i]=ri[ri[i]]
ENDFOR

indok=where(listlowestind GT -1L,countindok)
indoffset=n_elements(listlowestind)-countindok
indsorted=(sort(listlowestind))

FOR i=0L,countindok-1 DO BEGIN
    this_element=a[listlowestind[indsorted[i+indoffset]]]
    c[ri[ri[this_element-1]:ri[this_element]-1]]=i+1
ENDFOR

print,c

end
```

The idea here is to use histogram to scout for the first occurrence
of each integer number and store it in listlowestind. We then
sort the list, such that we know which element occurs first,
and then with a second loop we fill the output array c with
the occurrences of each element, and since we have sorted the
array first, we know which rank to assign to each element.
This should be moderately efficient. I assumed that the array
a has integer elements starting from 1, if not the case, just
sum 1-min(a) to the array before performing the operation.

Ciao,
Paolo

>
>
>
>
>> But this will fail if 'a' has more elements than the number of
>> its different values, for instance a=[3,3,1,2,3,2,2,1,2,3].
>>
>> One could try this:
>>
>> PRO test
>>
>> a=[3,3,1,2,3,2,2,1,2,3]
>>
>> b=a
>> c=intarr(n_elements(a))
>>
>> h=histogram(a,min=1,reverse_ind=ri)
>>
>> done=0
>> rank=1
>> WHILE NOT done DO BEGIN
>>   actual_value=b[0]
>>   ind_actual_value=ri[ri[actual_value-1]:ri[actual_value]-1]
>>   c[ind_actual_value]=rank
>>   rank=rank+1
>>   indremove=where(b NE actual_value,count)
>>   IF count GT 0 THEN b=b[indremove] ELSE done=1
>> ENDWHILE
>>
>> print,a
>> print,c
>>
>> END

>>
>>
>> but it will get inefficient as the numbers of different values
>> in 'a' grows, as the code in the loop get called more and more
>> times...
>>
>> Ciao,
>> Paolo
>>
>>
>>> Cheers,
>>>
>>> David
>>>
>>>

Subject: Re: Ordered index array
Posted by David Fanning on Thu, 08 Sep 2005 13:02:36 GMT
View Forum Message <> Reply to Message

Paolo Grigis writes:

> But this will fail if 'a' has more elements than the number of
> its different values, for instance a=[3,3,1,2,3,2,2,1,2,3].
>
> One could try this:
>
> PRO test
>
> a=[3,3,1,2,3,2,2,1,2,3]
>
> b=a
> c=intarr(n_elements(a))
>
> h=histogram(a,min=1,reverse_ind=ri)
>
> done=0
> rank=1
> WHILE NOT done DO BEGIN
> actual_value=b[0]
> ind_actual_value=ri[ri[actual_value-1]:ri[actual_value]-1]
> c[ind_actual_value]=rank
> rank=rank+1
> indremove=where(b NE actual_value,count)
> IF count GT 0 THEN b=b[indremove] ELSE done=1
> ENDWHILE
>

> print,a
> print,c
>
> END
>
>
> but it will get inefficient as the numbers of different values
> in 'a' grows, as the code in the loop get called more and more
> times...

Humm. Maybe I'm missing something (I haven't had my coffee yet),
but couldn't this be solved with the original code I wrote by
changing these two lines:

```
; Create index array and output array.
b = Indgen(N_Elements(a))+ 1
c = Intarr(N_Elements(a))
```

To these:

```
; Create index array and output array.
b = Indgen(N_Elements(a) > N_Elements(h)) + 1
c = Intarr(N_Elements(a) > N_Elements(h))
```

Then,

```
IDL> a=[3,3,1,2,3,2,2,1,2,3]
IDL> Print, a
   1  1  3  4  1  4  4  3  4  1
```

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: Ordered index array
Posted by David Fanning on Thu, 08 Sep 2005 13:08:46 GMT
View Forum Message <> Reply to Message

David Fanning writes:

> Then,

>
>   IDL> a=[3,3,1,2,3,2,2,1,2,3]
>   IDL> Print, a
>   1 1 3 4 1 4 4 3 4 1

Well, I'll go get my coffee now, but I think
you know this should be:

   IDL> a=[3,3,1,2,3,2,2,1,2,3]
   IDL> Print, Test(a)
      1 1 3 4 1 4 4 3 4 1

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: Ordered index array
Posted by Paolo Grigis on Thu, 08 Sep 2005 13:51:41 GMT
View Forum Message <> Reply to Message

David Fanning wrote:
> Paolo Grigis writes:
>
> [...]
>
> Humm. Maybe I'm missing something (I haven't had my coffee yet),
> but couldn't this be solved with the original code I wrote by
> changing these two lines:
>
>   ; Create index array and output array.
>   b = Indgen(N_Elements(a))+ 1
>   c = Intarr(N_Elements(a))
>
> To these:
>
>   ; Create index array and output array.
>   b = Indgen(N_Elements(a) > N_Elements(h)) + 1
>   c = Intarr(N_Elements(a) > N_Elements(h))
>
> Then,
>
>   IDL> a=[3,3,1,2,3,2,2,1,2,3]

> IDL> Print, a
> 1 1 3 4 1 4 4 3 4 1

Well, I guess this depends on whether the original poster minds having
gaps in the "rankings" (no element with "rank" 2 in your example)...

Ciao,
Paolo

>
> Cheers,
>
> David
>
>

Subject: Re: Ordered index array
Posted by David Fanning on Thu, 08 Sep 2005 13:55:19 GMT
View Forum Message <> Reply to Message

Paolo Grigis writes:

>> Then,
>>
>> IDL> a=[3,3,1,2,3,2,2,1,2,3]
>> IDL> Print, a
>> 1 1 3 4 1 4 4 3 4 1
>
> Well, I guess this depends on whether the original poster minds having
> gaps in the "rankings" (no element with "rank" 2 in your example)...

It's like golf: "tied for first and third". :-)

Cheers,

David

--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: Ordered index array
Posted by Emmanuel Christophe on Thu, 08 Sep 2005 14:51:20 GMT

Thanks to you both, the solution proposed by Paolo does exactly what I wanted (I don't want gaps in the "rankings") and this is fast enough for my case.

Thanks again,
Emmanuel

David Fanning a écrit :
> Paolo Grigis writes:
>
>
>>> Then,
>>>
>>> IDL> a=[3,3,1,2,3,2,2,1,2,3]
>>> IDL> Print, a
>>> 1 1 3 4 1 4 4 3 4 1
>>
>> Well, I guess this depends on whether the original poster minds having
>> gaps in the "rankings" (no element with "rank" 2 in your example)...
>
>
> It's like golf: "tied for first and third". :-)
>
> Cheers,
>
> David
>