Subject: Re: IDL sorting
Posted by Loren Anderson on Wed, 17 Oct 2007 15:37:33 GMT
View Forum Message <> Reply to Message

On Oct 17, 11:25 am, Wox <nom...@hotmail.com> wrote:
> Hi,
>
> I'm have that IDL sorting problem again: original subscript order is
> not maintained when values are equal. Is there a more elegant way then
> resorting each subarray of equal values on there original subscript?
>
> Thanks.

Try bsort: http://idlastro.gsfc.nasa.gov/ftp/pro/misc/bsort.pro

-Loren

Subject: Re: IDL sorting
Posted by David Fanning on Wed, 17 Oct 2007 15:38:12 GMT
View Forum Message <> Reply to Message

Wox writes:

> 'm have that IDL sorting problem again: original subscript order is
> not maintained when values are equal. Is there a more elegant way then
> resorting each subarray of equal values on there original subscript?

I'm not sure it's more elegant, but the NASA BSORT
routine certainly takes a lot of the work out of it. :-)

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

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting, Inc.
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Sepore ma de ni thui. ("Perhaps thou speakest truth.")

Subject: Re: IDL sorting
Posted by Loren Anderson on Wed, 17 Oct 2007 21:15:31 GMT
View Forum Message <> Reply to Message

On Oct 17, 11:25 am, Wox <nom...@hotmail.com> wrote:
> Hi,
>
> I'm have that IDL sorting problem again: original subscript order is
> not maintained when values are equal. Is there a more elegant way then
> resorting each subarray of equal values on there original subscript?
>
> Thanks.

Try bsort: http://idlastro.gsfc.nasa.gov/ftp/pro/misc/bsort.pro

-Loren

Subject: Re: IDL sorting
Posted by Wox on Thu, 18 Oct 2007 12:20:42 GMT
View Forum Message <> Reply to Message

On Wed, 17 Oct 2007 09:38:12 -0600, David Fanning <news@dfanning.com>
wrote:

> I'm not sure it's more elegant, but the NASA BSORT
> routine certainly takes a lot of the work out of it. :-)

It does the same I suscribed: resorting each subarray of equal values
by its original subscript. I was hoping that there would be a better
solution.

If I had to sort 32-bit longs, I could e.g. make 64-bit longs with HO
DWORD equal to the value and the LO DWORD equal to the subscript.
Sorting the resulting 64-bit longs would do the trick, wouldn't it?

```
function sort,array
array64=ishft(long64(array),32)+lindgen(n_elements(array))
return,sort(array64)
end
```

However, I want to sort 64-bit values. Maybe there is another way, so
that sort will do the right thing. Btw, I didn't want to ask this, but
why is IDL's sort doing this? Is there any situation where mixing up
the order of equal values has a benefit?

Subject: Re: IDL sorting
Posted by wlandsman on Thu, 18 Oct 2007 15:50:56 GMT

On Oct 18, 8:20 am, Wox <nom...@hotmail.com> wrote:

> Sorting the resulting 64-bit longs would do the trick, wouldn't it?
>
> function sort,array
> array64=ishft(long64(array),32)+lindgen(n_elements(array))
> return,sort(array64)
> end

That is clever. My tests on my V6.4 Linux box find that it is usually faster than bsort.pro. It is somewhat slower when there are only a few duplicate values

> Btw, I didn't want to ask this, but
> why is IDL's sort doing this?

IDL just uses the sort algorithm of the underlying OS. As far as I am aware, the SORT function on Linux boxes *does* preserve the order of equal values, but that on Mac and Windows machines does not. I would be interested to hear if anyone finds any exceptions to this rule.

> Is there any situation where mixing up
> the order of equal values has a benefit?

None that I can think of. But if you just want the fastest SORT possible, you might not care what happens to the equal values.

Actually, I think a good suggestion to ITTVIS would be to add a /preserve_equal keyword or something similar to SORT(). This topic comes up repeatedly.

Subject: Re: IDL sorting
Posted by Karl Schultz on Thu, 18 Oct 2007 18:30:14 GMT

wlandsman <wlandsman@gmail.com> wrote:
> On Oct 18, 8:20 am, Wox <nom...@hotmail.com> wrote:
>
>> Sorting the resulting 64-bit longs would do the trick, wouldn't it?
>>
>> function sort,array
>> array64=ishft(long64(array),32)+lindgen(n_elements(array))
>> return,sort(array64)

>> end
>
> That is clever.   My tests on my V6.4 Linux box find that it is
> usually faster than bsort.pro. It is somewhat slower when there are
> only a few duplicate values
>
>
>> Btw, I didn't want to ask this, but
>> why is IDL's sort doing this?

Because it is not a *stable* sort. Stable sorting algorithms preserve
the order of equal keys.

> IDL just uses the sort algorithm of the underlying OS.   As far as I
> am aware, the SORT function on Linux boxes *does* preserve the order
> of equal values, but that on Mac and Windows machines does not.   I
> would be interested to hear if anyone finds any exceptions to this
> rule.

Are you using this SORT function from the command line? If so, you
are using a shell function or a sort program in your PATH. Someone
probably decided that a stable sort made more sense for people sorting
things from the command line or from shell scripts. Reasonable.

IDL uses the C lib function qsort() which is usually an implmentation of
QuickSort, a good overall sort function for general purpose sorting.
Since IDL has no idea what you are sorting, it is actually a pretty good
choice. However, it is not stable. Speed may be more important to some
people than stability.

>
>> Is there any situation where mixing up
>> the order of equal values has a benefit?
>
> None that I can think of.   But if you just want the fastest SORT
> possible, you might not care what happens to the equal values.

Exactly. Or your application may not care about equal values, regardless
of speed issues.

> Actually, I think a good suggestion to ITTVIS would be to add a /
> preserve_equal keyword or something similar to SORT().   This topic
> comes up repeatedly.

Yep, perhaps /STABLE

--
Karl Schultz   kws@frii.com

There are 844,739 ways to enjoy a Waffle House hamburger.

Subject: Re: IDL sorting
Posted by wlandsman on Thu, 18 Oct 2007 20:43:19 GMT
View Forum Message <> Reply to Message

On Oct 18, 2:30 pm, Karl Schultz <k...@io.frii.com> wrote:
> Because it is not a *stable* sort. Stable sorting algorithms preserve
> the order of equal keys.
>
>> IDL just uses the sort algorithm of the underlying OS. As far as I
>> am aware, the SORT function on Linux boxes *does* preserve the order
>> of equal values, but that on Mac and Windows machines does not. I
>> would be interested to hear if anyone finds any exceptions to this
>> rule.
>
> Are you using this SORT function from the command line? If so, you
> are using a shell function or a sort program in your PATH. Someone
> probably decided that a stable sort made more sense for people sorting
> things from the command line or from shell scripts. Reasonable.
>

I don't understand this paragraph. I am just using the IDL intrinisc
SORT command. On every Linux box I have ever been on, it appears
that the C lib sort algorithm used by IDL SORT() *is* stable, whereas
it is *not* stable on Windows or MacOS.

But I like the idea of adding a /STABLE keyword to SORT. Thanks, --
Wayne

Subject: Re: IDL sorting
Posted by Karl Schultz on Thu, 18 Oct 2007 21:58:15 GMT
View Forum Message <> Reply to Message

wlandsman <wlandsman@gmail.com> wrote:
> On Oct 18, 2:30 pm, Karl Schultz <k...@io.frii.com> wrote:
>> Because it is not a *stable* sort. Stable sorting algorithms preserve
>> the order of equal keys.
>>
>>> IDL just uses the sort algorithm of the underlying OS. As far as I
>>> am aware, the SORT function on Linux boxes *does* preserve the order
>>> of equal values, but that on Mac and Windows machines does not. I
>>> would be interested to hear if anyone finds any exceptions to this
>>> rule.
>>

>> Are you using this SORT function from the command line? If so, you
>> are using a shell function or a sort program in your PATH. Someone
>> probably decided that a stable sort made more sense for people sorting
>> things from the command line or from shell scripts. Reasonable.
>>
>
> I don't understand this paragraph. I am just using the IDL intrinisc
> SORT command. On every Linux box I have ever been on, it appears
> that the C lib sort algorithm used by IDL SORT() *is* stable, whereas
> it is *not* stable on Windows or MacOS.
>

When you said "SORT function on Linux boxes", I thought you meant from the Linux command line. My bad.

So it looks like the qsort() implementation on the Linux distros you tried happens to be stable. That's all.

--
Karl Schultz kws@frii.com
There are 844,739 ways to enjoy a Waffle House hamburger.

Subject: Re: IDL sorting
Posted by JD Smith on Fri, 02 Nov 2007 00:02:42 GMT
View Forum Message <> Reply to Message

On Thu, 18 Oct 2007 21:58:15 +0000, Karl Schultz wrote:

> wlandsman <wlandsman@gmail.com> wrote:
>> On Oct 18, 2:30 pm, Karl Schultz <k...@io.frii.com> wrote:
>>> Because it is not a *stable* sort. Stable sorting algorithms preserve
>>> the order of equal keys.
>>>
>>>> IDL just uses the sort algorithm of the underlying OS. As far as I
>>>> am aware, the SORT function on Linux boxes *does* preserve the order
>>>> of equal values, but that on Mac and Windows machines does not. I
>>>> would be interested to hear if anyone finds any exceptions to this
>>>> rule.
>>>
>>> Are you using this SORT function from the command line? If so, you
>>> are using a shell function or a sort program in your PATH. Someone
>>> probably decided that a stable sort made more sense for people sorting
>>> things from the command line or from shell scripts. Reasonable.
>>>
>>
>> I don't understand this paragraph. I am just using the IDL intrinisc
>> SORT command. On every Linux box I have ever been on, it appears

>> that the C lib sort algorithm used by IDL SORT() *is* stable, whereas
>> it is *not* stable on Windows or MacOS.
>>
>
> When you said "SORT function on Linux boxes", I thought you meant from the
> Linux command line. My bad.
>
> So it looks like the qsort() implementation on the Linux distros you tried
> happens to be stable. That's all.

I side with Wayne: this platform difference has a real impact on many SORT-based algorithms. I understand the goal of re-using a tuned system QSORT, but going the extra step to get it to function the same on all IDL-supported systems would seem a no-brainer.

JD
