Subject: Re: Strange behaviour of Uniq static method
Posted by Johan Gustafsson on Fri, 08 Jul 2016 11:39:39 GMT
View Forum Message <> Reply to Message

Den torsdag 30 juni 2016 kl. 17:45:27 UTC+2 skrev Markus Schmassmann:
> On 29.06.2016 21:09, Dick Jackson wrote:
>> On Wednesday, 29 June 2016 02:14:02 UTC-7, Johan Gustafsson wrote:
>>> I've encountered a strange behaviour of the static method Uniq (not the old
>>> Uniq function, more about that later). To give a short example:
>>>
>>> IDL> x = [FltArr(5), -FltArr(5), RandomN(seed, 5)]
>>> IDL> Print, x.Uniq()
>>> -1.73792 -1.55209 -0.0861842 0.000000 -0.000000 0.000000
>>> -0.000000 0.000000 -0.000000 0.000000 0.0552376
> 0.835585
>>>
>>> The problem is the repeated zeros in array with supposed unique elements. It
>>> seems like the Uniq method treats 0. and -0. as two different values,
> which I
>>> believe is a bit unlogical. Also, according to the help page x.Uniq()
> should
>>> be equivalent to x[Uniq(x, Sort(x))], but
>>>
>>> IDL> Print,x[Uniq(x, Sort(x))]
>>> -1.73792 -1.55209 -0.0861842 0.000000 0.0552376 0.835585
>>>
>>> which is the result I would expect.
>>>
>>> I don't know if I really have a question, but it would be nice if someone could
>>> confirm that x.Uniq() in the example indeed does not give the expected
> output.
>>> Is this a known bug?
>>
>> That is indeed strange… it seems that -0.0 and 0.0 are considered equal:
>>
>> IDL> -0.0 eq 0.0
>> 1
>>
>> … yet they are distinct IEEE floating point values (showing the conversion to
>> byte values):
>>
>> IDL> byte(0.0, 0, 4)
>> 0 0 0 0
>> IDL> byte(-0.0, 0, 4)
>> 0 0 0 128
>>
>> … and it would depend on the sorting algorithm how the ten "equal but distinct"
>> values get sorted in your array of fifteen values. What you show is that

> the
>> static x.Uniq() method may be using a sorting method, which handles these
>> differently from Sort(). I'd call it a bug, one that comes only with the
> unusual
>> occurrence of -0.0.
>>
>> Of course, you can work around this with an extra step:
>>
>> IDL> x = [FltArr(5), -FltArr(5), RandomN(seed, 5)]
>> IDL> x[Where(x EQ -0.0, /NULL)] = 0.0
>> IDL> Print, x.Uniq()
>> -0.109547 -0.0809556 -0.0519432 0.000000 0.209843 0.807860
>> IDL> Print,x[Uniq(x, Sort(x))]
>> -0.109547 -0.0809556 -0.0519432 0.000000 0.209843 0.807860
>>
>> May I ask, how did you come across this? Most arithmetic operations that result
>> in zero do not give -0.0. If you convert from a string or text read from
> a file
>> that is '-0.0', or if you negate 0.0 explicitly, IDL results in -0.0, but I
>> wonder if there was another tricky case we should be aware of.
> If you use Dick's approach with
>> IDL> x[Where(x EQ -0.0, /NULL)] = 0.0
> you might also have to deal with different binary representations of
> NaN's to be sure to get the expected result:
>> IDL> x[where(finite(x,/nan),/null)]=!values.f_nan
> Might not be necessary in your particular case, but in a bugfix it
> should be considered.

Thank you, both Dick and Markus!

The NaN case was no concern for my case, but I agree that it is for the general situation.

/Johan
