Subject: Strange behaviour of Uniq static method
Posted by Johan Gustafsson on Wed, 29 Jun 2016 09:14:00 GMT
View Forum Message <> Reply to Message

This is my first post here, so hello everybody.

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?

I use IDL 8.5.1 under Windows 10

/Johan

Subject: Re: Strange behaviour of Uniq static method
Posted by Dick Jackson on Wed, 29 Jun 2016 19:09:40 GMT
View Forum Message <> Reply to Message

On Wednesday, 29 June 2016 02:14:02 UTC-7, Johan Gustafsson wrote:
> This is my first post here, so hello everybody.
>
> 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?
>
> I use IDL 8.5.1 under Windows 10
>
> /Johan

Welcome aboard, Johan!

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.

Cheers,
-Dick

Subject: Re: Strange behaviour of Uniq static method
Posted by Johan Gustafsson on Thu, 30 Jun 2016 09:39:33 GMT
View Forum Message <> Reply to Message

Den onsdag 29 juni 2016 kl. 21:09:42 UTC+2 skrev Dick Jackson:
> On Wednesday, 29 June 2016 02:14:02 UTC-7, Johan Gustafsson wrote:
>> This is my first post here, so hello everybody.
>>
>> 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?
>>
>> I use IDL 8.5.1 under Windows 10
>>
>> /Johan
>
> Welcome aboard, Johan!
>
> 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.
>
> Cheers,
> -Dick
>
> Dick Jackson Software Consulting Inc.
> Victoria, BC, Canada --- http://www.d-jackson.com

Hi!

The background to how I encountered this problem is a bit complicated (as it always tends to be, I guess). The array I originally was examining was the result of a process that involved repeated convolution with a Gaussian kernel via the FFT function. It seems like the use of FFT caused some small negative values to be introduced in intermediate results. To make a long story short, I think that the negative zeros were a result of underflow, like in

```
IDL> x = exp(-75.)
IDL> y = -exp(-75.)
IDL> Print, x
 2.67864e-033
IDL> Print, y
-2.67864e-033
IDL> Print, x*y
   -0.000000
% Program caused arithmetic error: Floating underflow
```

/Johan

Subject: Re: Strange behaviour of Uniq static method
Posted by Markus Schmassmann on Thu, 30 Jun 2016 15:45:25 GMT
View Forum Message <> Reply to Message

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.

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
