Subject: Philosophy of for loops
Posted by tclement on Mon, 28 Aug 2000 07:00:00 GMT
View Forum Message <> Reply to Message

Hello, again, everyone!

I was just wondering what the general concensus of the "IDL Expert Programmers" was on the use of for loops. When I first learned IDL, I remember getting from someone or somewhere the mantra "for loops are evil" because they take up so much time. Of course, as I learn more and watch what goes on in this group, it seems like "for loops are sometimes evil" would be a better mantra. The question then becomes, when do they become evil?

In response to my thread on summing diagonal elements, Craig said that for loops aren't always bad if you can do a lot at once, and his code proves that you can have some fast loops.

So what defines a slow loop? Is it having a bunch of accesses to sub-elements of arrays? Is it just having a bunch of statments? I suppose I could do some tests of my own, and I have a little, but it's much more fun to hear what you all have to say on the subject. I wouldn't have seen any IDL-ku if I just kept my thoughts to myself!

Todd

Subject: Re: Philosophy of for loops
Posted by Craig Markwardt on Wed, 30 Aug 2000 07:00:00 GMT
View Forum Message <> Reply to Message

landsman@my-deja.com writes:
> (1) for j=0,2047 do for i=0,2047 do outarr[i,j] = median(inarr[*,i,j])
>
> (2) for j=0,2047 do begin
> for i=0,2047 do begin
> outarr[i,j] = median(inarr[*,i,j])
> endfor
> endfor
>
> Form (1) is slightly faster, but the calculation cannot be interrupted
> with a Control-C. Also, it is my impression that the speed difference
> is less than it used to be, and that form (2) is now better optimized.

I hadn't realized these were different! My choice between the two forms usually revolves around stylistic concerns, i.e., does the thing fit on the line. My guess is that form (2) is a little slower *because* it is doing the keyboard checking. I have some roundabout

evidence that this is true.

Craig

--
---------- ----------
Craig B. Markwardt, Ph.D. EMAIL: craigmnet@cow.physics.wisc.edu
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response
---------- ----------

Subject: Re: Philosophy of for loops
Posted by Struan Gray on Wed, 30 Aug 2000 07:00:00 GMT
View Forum Message <> Reply to Message

Wayne, landsman@my-deja.com writes:

> inarr= randomn(seed, 3, 2048,2048)
> outarr = fltarr(2048,2048,/nozero)
>
> (1) for j=0,2047 do for i=0,2047 do outarr[i,j] = median(inarr[*,i,j])
>
> (2) for j=0,2047 do begin
> for i=0,2047 do begin
> outarr[i,j] = median(inarr[*,i,j])
> endfor
> endfor
>
> Form (1) is slightly faster, but the calculation
> cannot be interrupted with a Control-C. Also,
> it is my impression that the speed difference
> is less than it used to be, and that form (2)
> is now better optimized.

On my machine (a Mac G3 powerbook, IDL 5.3) (1) is slightly faster for small arrays but the difference is insignificant by the time you are up to 2048x2048.

> (I also assume that the two FOR loops are unavoidable
> here, but I would be delighted to be proved wrong.)

I often take data which consists of 1D spectra on a 2D spatial grid, ending up with arrays which are (Nspecpoints, Ngridx, Ngridy) in dimension. Often I want to do some processsing operation on all the individual spectra, and it helps a lot to 'unwrap' the array so that you do one loop instead of two nested ones. In your case the code would look like this:

```
npoints = 2048
inarr = reform(inarr, 3, npoints*npoints, /overwrite)
outarr = fltarr(npoints*npoints, /nozero)
for i=0L, long(npoints)*npoints - 1 do $
  outarr[i] = median(inarr[*,i])
inarr = reform(inarr, 3, npoints, npoints, /overwrite)
outarr = reform(outarr, npoints, npoints, /overwrite)
```

For the sorts of arrays I use (100, 200, 200) this is quite a bit faster, but interestingly enough by the time you get up to arrays like yours the speed advantage has gone. Watch out for overflow on your loop indices.

In this particular case you can actually work without loops altogether:

```
inarr = reform(inarr, 3*npoints*npoints, /overwrite)
outarr = reform(median(inarr, 3), 3, npoints, npoints, /overwrite)
outarr = reform((temporary(outarr))[1,*,*])
inarr = reform(inarr, 3, npoints, npoints, /overwrite)
```

The median filter creates a whole load of 'wrong' elements, but we can ignore them and eliminating the loop speeds the whole thing up so much that it's worth the overhead to calculate them. This version was substantially faster than the other three at all array sizes.

Both of these techniques require the 'interesting' dimension to be first, but sandwiching the code with a ROTATE or TRANSPOSE will do that without a punitive overhead so they can be made quite general.

I apologise to J.D. for not fitting HISTOGRAM in there somewhere.

Struan

Subject: Re: Philosophy of for loops
Posted by edward.s.meinel on Thu, 31 Aug 2000 13:40:27 GMT
View Forum Message <> Reply to Message

In article <onn1huybd3.fsf@cow.physics.wisc.edu>,
craigmnet@cow.physics.wisc.edu wrote:
>
> landsman@my-deja.com writes:
>> (1) for j=0,2047 do for i=0,2047 do outarr[i,j]=median(inarr[*,i,j])
>>
>> (2) for j=0,2047 do begin

>> for i=0,2047 do begin
>> outarr[i,j] = median(inarr[*,i,j])
>> endfor
>> endfor
>>
>> Form (1) is slightly faster, but the calculation cannot be interrupted
>> with a Control-C. Also, it is my impression that the speed
>> difference is less than it used to be, and that form (2) is now
>> better optimized.
>
> I hadn't realized these were different! My choice between the two
> forms usually revolves around stylistic concerns, i.e., does the thing
> fit on the line. My guess is that form (2) is a little slower
> *because* it is doing the keyboard checking. I have some roundabout
> evidence that this is true.
>

So does that mean that form (3) is slower than form (1)?

(3) for j=0,2047 do $
  for i=0,2047 do $
  outarr[i,j]=median(inarr[*,i,j])

I guess that would mean that readable code is slower than unreadable
code...

Ed Meinel

Subject: Re: Philosophy of for loops
Posted by promashkin on Thu, 31 Aug 2000 15:46:40 GMT
View Forum Message <> Reply to Message

> So does that mean that form (3) is slower than form (1)?

I am sure the speed difference would be much less than the time required
to find out by how much exactly :-(
Cheers,
Pavel

Subject: Re: Philosophy of for loops
Posted by Craig Markwardt on Thu, 31 Aug 2000 15:51:06 GMT

edward.s.meinel@aero.org writes:
>
> So does that mean that form (3) is slower than form (1)?
>
> (3) for j=0,2047 do $
> for i=0,2047 do $
> outarr[i,j]=median(inarr[*,i,j])
>

No, I think your form (3) would be interpretted as a single line, so it would be equivalent to form (2). The following form (4) would be slightly slower than (1), but keep in mind that small amounts of processing in the outer loop are pretty much negligible.

```
for j = 0, 2047 do begin
  for i = 0, 2047 do begin
      outarr[i,j]=median(inarr[*,i,j])
  endfor
endfor
```

Craig

--

Craig B. Markwardt, Ph.D. EMAIL: craigmnet@cow.physics.wisc.edu
Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response