Subject: Re: advise for saving a for-loop
Posted by Jean H. on Mon, 14 Sep 2009 16:50:18 GMT
View Forum Message <> Reply to Message

Bernhard Reinhardt wrote:
> Hi,
>
> I don't know if there's a special term for what I'm trying to do:
>
> I have two 2D arrays of the same size (msg_x and msg_y) which contain x-
> and y-values. So msg_y consists of rows which contain mainly the same
> values and msg_y consists of columns which contain mainly the same
> values. But it has to mentioned that values are slightly changing in a
> row or column. That's what make's things nasty.
> For msg_y it means, it may look like:
> 1000 1000 1000 1000 [..] 1001 1001 1001 1001 [..] 1002 1002
> 1001 1001 1001 1001 [..] 1002 1002 1002 1002 [..] 1003 1003
> 1002 1002 1002 [..] 1003 1003 1003 1003 [..] 1004 1004 1004
>
> I also have two linear arrays li_x and li_y of the same size. I now want
> to make a map with the same dimensions of msg_x with a 1 where the
> points in the linear arrays match into the pseudo-grid and 0 elsewhere.
>
> Here's how I do it at the moment:
>
>  for i=0, N_ELEMENTS(li_x)-1 do begin
>  ind=WHERE(msg_x eq li_x[i] AND msg_y eq li_y[i])
>  if ind[0] ne -1 then ligrid[ind] = 1
>  endfor
>
> The 2-D arrays have sizes of 600x600 or 1800x1800 and the linear arrays
> are of size 10000.
>
> This means where has to search 10000 over the two 2D-arrays which takes
> some time.
>
> I guess there must be a smarter way to do. I thought about some
> solutions involving sort and histogram but so far I couldn't come up
> with a solution without for-loops.
>
> I'd be pleased if someone of you could enlighten me.
>
> Regards,
>
> Bernhard

Hi Bernhard,

yes, histogram is the way to go. You will want to 1) intersect msg_x and
li_x, then 2) msg_y and li_y and 3) the ouptut of 1 and 2 (index based)

Have a look at
http://www.dfanning.com/tips/set_operations.html

you can get ri from the 1st histogram and return the following, to get
the index:

```
r = Where((Histogram(a, Min=mina, Max=maxa, reverse_indices=ri) NE 0)
AND (Histogram(b, Min=mina, Max=maxa) NE 0), count)

 IF count eq 0 THEN RETURN, -1
  toReturn = ri[ri[r[0]]:ri[r[0]+1]-1]

  for Rcount = 1, count-1 do begin
   toReturn = [toReturn,ri[ri[r[Rcount]]:ri[r[Rcount]+1]-1]]
  endfor
```

Jean

---

Subject: Re: advise for saving a for-loop
Posted by Bernhard Reinhardt on Tue, 15 Sep 2009 09:21:04 GMT
View Forum Message <> Reply to Message

Jean H. wrote:
> Bernhard Reinhardt wrote:
>> Hi,
>>
>> I don't know if there's a special term for what I'm trying to do:
>>
>> I have two 2D arrays of the same size (msg_x and msg_y) which contain
>> x- and y-values. So msg_y consists of rows which contain mainly the
>> same values and msg_y consists of columns which contain mainly the
>> same values. But it has to mentioned that values are slightly changing
>> in a row or column. That's what make's things nasty.
>> For msg_y it means, it may look like:
>> 1000 1000 1000 1000 [..] 1001 1001 1001 1001 [..] 1002 1002
>> 1001 1001 1001 1001 [..] 1002 1002 1002 1002 [..] 1003 1003
>> 1002 1002 1002 [..] 1003 1003 1003 1003 [..] 1004 1004 1004
>>
>> I also have two linear arrays li_x and li_y of the same size. I now
>> want to make a map with the same dimensions of msg_x with a 1 where
>> the points in the linear arrays match into the pseudo-grid and 0
>> elsewhere.
>>
>> Here's how I do it at the moment:

>>
>>   for i=0, N_ELEMENTS(li_x)-1 do begin
>>     ind=WHERE(msg_x eq li_x[i] AND msg_y eq li_y[i])
>>     if ind[0] ne -1 then ligrid[ind] = 1
>>   endfor
>>
>>  The 2-D arrays have sizes of 600x600 or 1800x1800 and the linear
>>  arrays are of size 10000.
>>
>>  This means where has to search 10000 over the two 2D-arrays which
>>  takes some time.
>>
>>  I guess there must be a smarter way to do. I thought about some
>>  solutions involving sort and histogram but so far I couldn't come up
>>  with a solution without for-loops.
>>
>>  I'd be pleased if someone of you could enlighten me.
>>
>>  Regards,
>>
>>  Bernhard
>
>  Hi Bernhard,
>
>  yes, histogram is the way to go. You will want to 1) intersect msg_x and
>  li_x, then 2) msg_y and li_y and 3) the ouptut of 1 and 2 (index based)
>
>  Have a look at
>  http://www.dfanning.com/tips/set_operations.html
>
>  you can get ri from the 1st histogram and return the following, to get
>  the index:
>
>  r = Where((Histogram(a, Min=mina, Max=maxa, reverse_indices=ri) NE 0)
>  AND (Histogram(b, Min=mina, Max=maxa) NE 0), count)
>
>     IF count eq 0 THEN RETURN, -1
>   toReturn = ri[ri[r[0]]:ri[r[0]+1]-1]
>
>   for Rcount = 1, count-1 do begin
>    toReturn = [toReturn,ri[ri[r[Rcount]]:ri[r[Rcount]+1]-1]]
>   endfor

Hi Jean,

I tamed the beast - well a least it seems it's doing what I expect for
now. I modified your suggested code a bit. Intersecting the resulting
indices is the wrong way to go. It would yield way to much "hits".

I now look at indices of values that exist in both msg_x and li_x and then where the corresponding y-values in both arrays match, too. But I have to do it for one single x-value at a time. But although I have a double for-loop, speed-up is about 200x :)

```
  mina=min(msg_x)
  maxa=max(msg_x)
  r = Where((Histogram(msg_x, Min=mina, Max=maxa,
reverse_indices=rim) $  NE 0) AND (Histogram(li_x, Min=mina, Max=maxa,$
 REVERSE_INDICES=ril) $
 NE 0), count)
  IF count gt 0 THEN begin
   for Rcount = 0, count-1 do begin
    lind=ril[ril[r[Rcount]]:ril[r[Rcount]+1]-1]
    mind=rim[rim[r[Rcount]]:rim[r[Rcount]+1]-1]
    for i=0, N_ELEMENTS(lind)-1 do begin
      ind=where(msg_y[mind] eq li_y[lind[i]])
      if ind[0] ne -1 then ligrid[mind[ind]] = 1
      ;ligrid is the map, same dim as msg_x and msg_y
    endfor
   endfor
  endif
```

Bernhard

---

Subject: Re: advise for saving a for-loop
Posted by Jeremy Bailin on Tue, 15 Sep 2009 14:12:38 GMT
View Forum Message <> Reply to Message

On Sep 15, 5:21 am, Bernhard Reinhardt
<wirdseltengele...@freisingnet.de> wrote:
> Jean H. wrote:
>> Bernhard Reinhardt wrote:
>>> Hi,
>
>>> I don't know if there's a special term for what I'm trying to do:
>
>>> I have two 2D arrays of the same size (msg_x and msg_y) which contain
>>> x- and y-values. So msg_y consists of rows which contain mainly the
>>> same values and msg_y consists of columns which contain mainly the
>>> same values. But it has to mentioned that values are slightly changing
>>> in a row or column. That's what make's things nasty.
>>> For msg_y it means, it may look like:
>>> 1000 1000 1000 1000 [..] 1001 1001 1001 1001 [..] 1002 1002
>>> 1001 1001 1001 1001 [..] 1002 1002 1002 1002 [..] 1003 1003
>>> 1002 1002 1002 [..] 1003 1003 1003 1003 [..] 1004 1004 1004

>
>>> I also have two linear arrays li_x and li_y of the same size. I now
>>> want to make a map with the same dimensions of msg_x with a 1 where
>>> the points in the linear arrays match into the pseudo-grid and 0
>>> elsewhere.
>
>>> Here's how I do it at the moment:
>
>>>  for i=0, N_ELEMENTS(li_x)-1 do begin
>>>  ind=WHERE(msg_x eq li_x[i] AND msg_y eq li_y[i])
>>>  if ind[0] ne -1 then ligrid[ind] = 1
>>>  endfor
>
>>> The 2-D arrays have sizes of 600x600 or 1800x1800 and the linear
>>> arrays are of size 10000.
>
>>> This means where has to search 10000 over the two 2D-arrays which
>>> takes some time.
>
>>> I guess there must be a smarter way to do. I thought about some
>>> solutions involving sort and histogram but so far I couldn't come up
>>> with a solution without for-loops.
>
>>> I'd be pleased if someone of you could enlighten me.
>
>>> Regards,
>
>>> Bernhard
>
>> Hi Bernhard,
>
>> yes, histogram is the way to go. You will want to 1) intersect msg_x and
>> li_x, then 2) msg_y and li_y and 3) the ouptut of 1 and 2 (index based)
>
>> Have a look at
>> http://www.dfanning.com/tips/set_operations.html
>
>> you can get ri from the 1st histogram and return the following, to get
>> the index:
>
>> r = Where((Histogram(a, Min=mina, Max=maxa, reverse_indices=ri) NE 0)
>> AND (Histogram(b, Min=mina, Max=maxa) NE 0), count)
>
>>  IF count eq 0 THEN RETURN, -1
>> toReturn = ri[ri[r[0]]:ri[r[0]+1]-1]
>
>> for Rcount = 1, count-1 do begin
>> toReturn = [toReturn,ri[ri[r[Rcount]]:ri[r[Rcount]+1]-1]]

```
>>   endfor
>
>  Hi Jean,
>
>  I tamed the beast - well a least it seems it's doing what I expect for
>  now. I modified your suggested code a bit. Intersecting the resulting
>  indices is the wrong way to go. It would yield way to much "hits".
>
>  I now look at indices of values that exist in both msg_x and li_x and
>  then where the corresponding y-values in both arrays match, too. But I
>  have to do it for one single x-value at a time. But although I have a
>  double for-loop, speed-up is about 200x :)
>
>     mina=min(msg_x)
>     maxa=max(msg_x)
>     r = Where((Histogram(msg_x, Min=mina, Max=maxa,
>  reverse_indices=rim) $  NE 0) AND (Histogram(li_x, Min=mina, Max=maxa,$
>           REVERSE_INDICES=ril) $
>           NE 0), count)
>     IF count gt 0 THEN begin
>      for Rcount = 0, count-1 do begin
>       lind=ril[ril[r[Rcount]]:ril[r[Rcount]+1]-1]
>       mind=rim[rim[r[Rcount]]:rim[r[Rcount]+1]-1]
>       for i=0, N_ELEMENTS(lind)-1 do begin
>         ind=where(msg_y[mind] eq li_y[lind[i]])
>         if ind[0] ne -1 then ligrid[mind[ind]] = 1
>                 ;ligrid is the map, same dim as msg_x and msg_y
>       endfor
>      endfor
>     endif
>
>  Bernhard
```

Why don't you generate one for msg_x and li_x using the technique you have, another for msg_y and li_y, and then multiply them together? That'll save the for loop, if you can afford the memory of having an extra 2D array sitting around.

-Jeremy.