Subject: Astronomys` Sixth Neighbour Needs Help
Posted by touser2001 on Thu, 24 Jul 2003 20:40:40 GMT
View Forum Message <> Reply to Message

I have to first say hello to everyone!!   I am new to this (and all) newsgroup(s).
I had a small IDL class and several months of on-off IDL work in making astronomy-related programs.
My latest program needs some expert help and thus I am writing this post in hope of a small discussion to enlighten me. I sincerely apologize if the post is too long!!

Basically, the program reads a 2-column n-line file (n is number of stars in field, the columns are the position of the stars, Right Ascencion and Declination which are similar to latitude and longitude). My objective is to obtain, for each star, the distance to its 6th neighbour. Simply, Right ascencsion is x and Declination is y on a xy grid.

What I am doing is:

```
k=0
while k lt u do begin                  ; I know the number of stars
(lines) which is equal to "u"
 i=0              ;
 while i lt u do begin  ;
  dxki = (x(k) - x(i))   ;this variable is the RA distance between
point i
                  ;and the kth point in the array

  dyki = (y(k) - y(i))   ;this variable is the Dec distance between
point i
                  ;and the kth point in the array

  dki = sqrt (dxki^2 + dyki^2) ;Now I define the distance, in 2-d
space, between the two objects i and k

  D2(k,i) = dki  ;This array contains all the distance data for all
the stars
     ;The reference star is the column number (k)
  i = i+1   ;for all the values of i
 endwhile   ;
k=k+1
endwhile
```

The above loop calculates the distance from star 1 to 2, 1 to 3 etc etc and then 2 to 1, 2 to 2. I note that distance 1 to 2 is the same as 2 to 1 so a first improvement would be to say that once 1 to 2 is done it need to do 2 to 1 (and this for 3 to 1 etc) ... can I implement that in the loop? Or any ideas as to improve this distance calculation?

The loop gives me an array whose dimension is u x u. This is a huge array when the star file is 10,000 stars... Which brings me to the second critical stage: calculating the sixth neighbour by first sorting, in ascending order, the distances in each column.
I did:

```
n=0
while n lt u do begin
D2n = D2(n,*)           ; this is so that I am sorting column by
column

 for j=0, n_elements(D2n)-2 do begin  ;
  for l=j+1, n_elements(D2n)-1 do begin ;
   if D2n[l] lt D2n[j] then begin ;This just arranges distances in
increasing order.
   temp=D2n[j]   ;
   D2n[j]=D2n[l]   ;
   D2n[l]=temp   ;So the array that I get is now the array of
distances
   endif    ;to the nearest neighbours, in order of the neighbour
number!
  endfor     ;
 endfor
D2(n,*)=D2n
n=n+1       ;
endwhile
```

Now that all columns are sorted I just go to line number six to get the sixth neighbour of each star!

I really don`t know if this is the best (quickest) method for organizing a column in ascending order... the second improvement would be to implement a different sorting algorithm. Note: the program does this for each of the 10,000 columns...

My problem is that for a field with 10,000 stars the program takes around 3 days (!) to run and it slows down all computer programs (unix system) (last time I ran the program it was killed by the system administrator since it took up too much memory).

I wish the department would hire some professional programmers!! Not that I don`t like IDL but I wouldn`t mind doing some astronomy too ;-)

Greatly appreciate and and all help!!!!!!!!!!!!
Bruno
PhD Student

Subject: Re: Astronomys` Sixth Neighbour Needs Help
Posted by Pavel Romashkin on Fri, 25 Jul 2003 22:34:13 GMT
View Forum Message <> Reply to Message

Hi Bruno,
What you're discovering in the meantime is that, for large arrays
(n*10E7 points) memory allocation is slower than loops. This is what's
happening with Rob's code. The program, while great, expands the data
size by at least a factor of 8 more than necessary (counting matrices
and index expansion for SORT), hence the incredible memory consumption.

Try the following:

```
FUNCTION CALCULATE_6TH_NEIGHBOR, x, y
; Calculate all of the distances
n = n_elements(x)
u = 1+bytarr(n)
d6 = fltarr(n)

dx = u#x
dy = u#y
for i = 0L, n-1 do dx[0, i] = dx[*, i] - x
for i = 0L, n-1 do dy[0, i] = dy[*, i] - x
d2 = sqrt(dx^2+dy^2)

for i = 0L, n-1 do begin
 temp = d2[*, i]
 ind = sort(temp)
 d6[i] = temp[ind[6]]
endfor
return, d6
END
```

Not all loops are bad in IDL.
Why are these loops faster? Memory allocation is much smaller, thus
contiguous chunks are easier available - faster; all loops are aligned
by the row and kept short - this is important. In my test, speed is
higher by a factor of 20 with identical results.
Cheers,
Pavel

astronomer wrote:
>
> AMAZING!!!!
>

> I am extremely amazed and impressed at the increase in processing
> speed that the changes made!!!!!!
> I was not expecting such a change! Amazing how inefficient my program
> was. Where was all the time going to? I mean, how come the for loops
> made the program so slow whereas now...
>
> I can`t thank you enough!
> I managed to run a 3800 star file in under 11 minutes whereas before
> it took over 24 hours. Now I am running a 9300 star file!
> For the larger file I am using a dual processor computer with 1 Gb and
> with 2400GHz each processor. The program sucks 99.8% of the CPU but
> it is running.
>
> I am also very surprised at how much memory is allocated to the IDL
> processes. What I mean is that the whole computer would slow down
> immensely, giving priority to IDL. Strange.
>
> This is indeed a good introduction to this newsgroup! and to IDL.
>
> Thanks Rob!

Subject: Re: Astronomys` Sixth Neighbour Needs Help
Posted by touser2001 on Sat, 26 Jul 2003 02:14:09 GMT

View Forum Message <> Reply to Message

I can begin to imagine how people must have felt when electricity
became widely available... or when the wheel was invented even!!!

I was already content with the first change by Rob, before that I had
actually thought there was some limit and I wouldn`t be capable of
reading a 3800 star file in less than a day... but 40 seconds??!!
Inappropriate, but the words that come to mind are: Shock and Awe!

I have to confess that being a begginner IDLer there is much of the
improvements that I do not understand.
In particular what is the meaning of # in the dx = u#x (so why does
Rob use dx = u#x - x#u and Pavel only dx = u#x?
Also, in the sort, I don`t understand how it works yet but, there is
no mention of it being an increasing distance sort yet it is.
Also, Pavel, you pointed out that in large arrays memory allocation is
slower than loops. Where in Robs code was "memory allocation"
substituting loops?

I have to admit that the science that I can do now is greatly enhanced
but I am most intrigued by the inefficiency of my program... the
number of calculations is the same so how come the huge speed
increase?

This is, undoubtedly, my key question now since it influences all my
future programs!

When I write another program what set of rules should I follow to make
it most efficient?
What is the hierarchy of processes (loops better than ...) (do while
better than ...).

First Commandment: thou shalt not say you made some elses` program
Second Commandment: thou shalt always use loops to minimize memory
allocation
etc

I cannot exagerate how much I appreciate all the help!!!!

Bruno
Astronomy PhD Student
University of Florida

Subject: Re: Astronomys` Sixth Neighbour Needs Help
Posted by Ben Panter on Sat, 26 Jul 2003 07:48:53 GMT

View Forum Message <> Reply to Message

Hi Bruno,

> I can`t thank you enough!
> I managed to run a 3800 star file in under 11 minutes whereas before
> it took over 24 hours. Now I am running a 9300 star file!
> For the larger file I am using a dual processor computer with 1 Gb and
> with 2400GHz each processor. The program sucks 99.8% of the CPU but
> it is running.

Apologies if I'm telling you something you already know...

I'm a bit of an IDL numpty myself, and an astronomy Phd.er - but I tend
to use a lot of CPU time, quite often harvested from other user's
workstations. You can make IDL far more polite to your system using a
command called "nice". Different LINUX set ups do it different ways, but
if you have launch, say, idlde or idl, type

nice idlde

or

nice idl

This assigns a priority to your IDL environment below that of other
things (like X and whatever your desktop is). In this way IDL will be
pushed to the back when another program needs a quick bit of CPU.

The codes which I run here take about 1.5 years - but by "borrowing" CPU
time (running my codes niced on other people's machines) we can run in
parallel on 20 - 25 machines and complete in a few weeks.

HTH,

Ben

--
Ben Panter, Edinburgh
My name (no spaces)@bigfoot which is a com.

Subject: Re: Astronomys` Sixth Neighbour Needs Help
Posted by touser2001 on Sat, 26 Jul 2003 20:48:50 GMT

View Forum Message <> Reply to Message

I talked to the administrator here about that nice value and
understand it now. Thanks Benjamin, it sounds like a good
alternative. So you are also dealing with the beauties of astronomy
programming?! They never told me I would have to program this much to
do some science.

I am also writing because in an attempt to test the program with the
changes suggested by Pavel, I tried a 15000 star field and IDL spat
out, after an hour and a half of processing (so it must have just
finished the calculation of the distances and the sorting of these),
that it could not allocate the memory necessary to make the array...
have I now achieved the limits of the program for this computer or is
there some way of going around this problem?

Bruno
Astronomy PhD Student
University of Florida

Subject: Re: Astronomys` Sixth Neighbour Needs Help
Posted by JD Smith on Mon, 28 Jul 2003 06:22:48 GMT

View Forum Message <> Reply to Message

On Fri, 25 Jul 2003 19:14:09 -0700, astronomer wrote:

> I can begin to imagine how people must have felt when electricity became

> widely available... or when the wheel was invented even!!!

Bruno,

This is an excellent test problem that demonstrates well the variety
of tradeoffs one must make to get good performance out of IDL.
Pavel's method demonstrates the vectorized version of a brute force
technique, cast in terms of (potentially rather large) arrays. It is
essentially the exact analog of your method, but designed to use array
operations, which IDL is very good at (especially large arrays -- it's
what it was designed for).

The reason why your method performed so poorly is that IDL "for" loops
impose a fairly large (compared to other languages) overhead on each
cycle. For the curious, this is likely because it must travel around
the full interpreter loop on each iteration, possibly compiling
things, processing background widget events, looking for keyboard
input, processing signals, etc. -- very different from, e.g., a loop
in C, which runs quite close to the hardware. As Pavel points out, if
you can do a substantial amount of work in each loop iteration -- more
than the large constant overhead imposed by that cycle -- "for" loops
can give perfectly good performance (despite the quasi-fanatical
facetious rantings to the contrary you may find on certain respected
IDL resource sites).

So one way to get good speed to is to re-cast your problem in terms of
the largest array operation you can, and then let IDL do what it does
best: chew on those arrays. Let's see how this applies to your
problem. Pavel had a few bugs in his code, and I like to build big
arrays another way, so here's my version of the vectorized brute force
search:

```
function nth_neighbor, x, y, k
  n = n_elements(x)
  dn = fltarr(n,/NOZERO)
  d=(rebin(transpose(x),n,n,/SAMPLE)-rebin(x,n,n,/SAMPLE))^2 + $
    (rebin(transpose(y),n,n,/SAMPLE)-rebin(y,n,n,/SAMPLE))^2
  for i=0L,n-1 do dn[i] = sqrt(d[(sort(d[*,i]))[k],i])
  return, dn
end
```

What am I doing here? I just make a huge array to compare every point
with every other point, all at once. I do this by making pairs of
arrays which are the tranpose of each other and subtracting them.
E.g. for 4 points, computing dx looks like:

```
    x0 x0 x0 x0     x0 x1 x2 x3
```

```
     x1 x1 x1 x1     x0 x1 x2 x3
dx = x2 x2 x2 x2  -  x0 x1 x2 x3
     x3 x3 x3 x3     x0 x1 x2 x3
```

So, you see, this 16 element array contains all of the x deltas from each point to each other point (in fact, it wastefully contains them all twice, since for distance purposes x1-x2 and x2-x1 are equivalent, plus it contains distances from each point to itself). If I do something similar for the y values, subtract and add each squared, I get a large array which contains the distances from each point to each other point (actually the square of the distances... I save time by skipping the square-root until it's needed at the end, since sorting on d^2 and d gives the same result). I sort them row by row, and pick out the nth member of the sorted list.

This method works well, and is fast compared to your method, but it has a deep flaw, which you've already run up against. It requires the use of arrays of size n^2 (16 in this small example). This gets big fast! E.g., for 10,000 points, I might need 4 such arrays of size 10,000 x 10,000, which is around 1e8 x 4 x 4=1.6 GB of memory! You're entirely limited by how much memory you have, or, more accurately, how fast your disks and virtual memory sub-system are. While very fast for small arrays which can fit in memory, it just doesn't scale up to large data sets. Now you understand why the code choked on 15,000 points.

The reason brute force fails, cursed by slowness (in your method), or huge storage needs (in Pavel's) is all those comparisons. If you have to compare every single point to every single other point, that's n*(n-1)/2 comparisons, at best.... e.g. n^2. But intuitively, why should we compare all the points? We know the points way over on the other side aren't going to be the closest, but we compute their distances and sort on them as if they ever had a shot anyway. But how can we decide if things are close or not without comparing all their distances? A seeming catch-22.

One method is triangulation. IDL has a lovely function called TRIANGULATE which can compute a so-called Delaunay triangulation (or DT -- do read about it) which has some nice properties. One of these properties is that the nearest neighbors of a point are either connected to it directly by the DT, or connected through some closer point which is (mathematicians would say the nearest neighbors graph is a "sub-graph" of the Delaunay). Here's a function based on this property:

```
function nearest_neighbors,x,y,DISTANCES=nearest_d,NUMBER=k
  if n_elements(k) eq 0 then k=6
  ;; Compute Delaunay triangulation
  triangulate,x,y,CONNECTIVITY=c
  n=n_elements(x)
  nearest=lonarr(k,n,/NOZERO)
  nearest_d=fltarr(k,n,/NOZERO)

  for point=0L,n-1 do begin
     p=c[c[point]:c[point+1]-1] ;start with this point's DT neighbors
     d=(x[p]-x[point])^2+(y[p]-y[point])^2
     for i=1,k do begin
        s=sort(d)              ; A wasteful priority queue, but anyway
        p=p[s] & d=d[s]
        nearest[i-1,point]=p[0] ; Happy Day: the closest candidate is a NN
        nearest_d[i-1,point]=d[0]
        if i eq k then continue

        ;; Add all its neighbors not yet seen
        new=c[c[p[0]]:c[p[0]+1]-1]
        nnew=n_elements(new)
        already_got=[p,nearest[0:i-1,point],point]
        ngot=n_elements(already_got)
        wh=where(long(total(rebin(already_got,ngot,nnew,/SAMPLE) ne $
                          rebin(transpose(new),ngot,nnew,/SAMPLE),1)) $
              eq ngot, cnt)
        if cnt gt 0 then begin
           new=new[wh]
           p=[p[1:*],new]
           d=[d[1:*],(x[new]-x[point])^2+(y[new]-y[point])^2]
        endif else begin
           p=p[1:*]
           d=d[1:*]
        endelse
     endfor
  endfor
  if arg_present(nearest_d) then nearest_d=sqrt(nearest_d)
  return,nearest
end
```

I won't go through all the details. Roughly, it just starts searching
out to nearby points on the DT, expanding outwards until it has
enough. For each of the n points, instead of computing distances to
and sorting n points, it operates on a much smaller number. It gives
you back *all* of the indices (and, optionally, distances) of the k
nearest points, and has one quirk: for points on the boundary, the
point itself is included first on the list. You could obviously test
for this and reject them (since, with half their neighbors "missing",
they're suspect anyway). One other nice feature: once you build the

DT, you can cache it and use it again and again, if, e.g., you're
finding the nearest neighbors interactively.

Here are some timings comparing the two, for finding the 6th/1st-6th
neighbors of a random set of points:

```
    100 Points, 6 Nearest Neighbors
nxn:    0.0082000494
 DT:     0.030847073
```

Hmm... nxn does quite well there. But 100x100 is pretty small ;)

```
   1000 Points, 6 Nearest Neighbors
nxn:      0.79809201
 DT:      0.31871009
```

Starting to suffer, but we've still got a few MB of memory to spare.

```
   5000 Points, 6 Nearest Neighbors
nxn:       23.420803
 DT:       1.5785370
```

Here comes trouble. The relative timings will depend strongly on how
much memory you have. He're we're just managing to fit about 400MB in
memory, but danger looms ahead.

```
   7000 Points, 6 Nearest Neighbors
nxn:       152.13264
 DT:       2.3276160
```

With 800MB or so needed for that big nxn calculation on a 500MB
laptop, we really hit the disk this time. Notice how the DT method
doesn't miss a beat, just scaling up roughly as n.

Keep in mind that as the number of nearest neighbors you want gets
bigger, the brute force method starts looking better and better
(i.e. the problem gets harder and harder), but it still won't scale:

```
   1000 Points, 50 Nearest Neighbors
nxn:      0.81207108
 DT:       3.1622850
```

And now, just to show it can be done:

```
     100000 Points, 6 Nearest Neighbors
 DT:       32.612031
```

I left out nxn since I didn't have a spare 200GB of memory to throw around ;).

So, should we congratulate ourselves for having found a truly fast record-setting algorithm? No, probably not. My code is actually fairly sloppy in the inner loop, re-sorting things that are already mostly sorted, finding and comparing more DT-connected points than strictly necessary to get to the nearest set, concatenating and re-allocating small arrays like mad. The point is, I could afford to be sloppy, since the competition just couldn't scale. By working on the inner loop, I could probably squeeze another factor of 3 out of it in IDL. Carefully re-coded in C, you could easily see a factor of 50-100 speedup. But try something like http://www.cs.umd.edu/~mount/ANN/ before you resort to that. The take home lesson? IDL probably isn't the best tool if you need raw speed on problems it isn't optimized for (adding arrays and the like), but with a nice collection of tricks in your toolkit, you can take it places it wasn't really designed to go.

JD

Subject: Re: Astronomys` Sixth Neighbour Needs Help
Posted by Ben Panter on Mon, 28 Jul 2003 12:40:57 GMT
View Forum Message <> Reply to Message

Bruno,

> I talked to the administrator here about that nice value and
> understand it now. Thanks Benjamin, it sounds like a good
> alternative. So you are also dealing with the beauties of astronomy
> programming?! They never told me I would have to program this much to
> do some science.

<heavy sigh>

Yeup.

I use IDL to drive something called "MOPED" - basically a very good compression algorithm. It allows you to work out model parameters very quickly - we get star formation history information out of sloan galaxy spectra. In my case, each spectra takes about 3mins - which is great compared to the brute force approach which is more like 10+ hours. 3 minutes sounds pretty short - but then I do have 135 000 galaxies to look at (so far!). If you're very bored one day, check out astro-ph and look for "Panter"

On the plus side though - all this computing means that you

a) Don't have to get cold at telescopes
b) Don't have to reduce your own data
c) Don't have to use blink tables
d) Have unlimited access to Google

I think we get a better deal...

> I am also writing because in an attempt to test the program with the
> changes suggested by Pavel, I tried a 15000 star field and IDL spat
> out, after an hour and a half of processing (so it must have just
> finished the calculation of the distances and the sorting of these),
> that it could not allocate the memory necessary to make the array...
> have I now achieved the limits of the program for this computer or is
> there some way of going around this problem?

You've heard of GIYF? I think here it might be JDSIYF!

Ben

--
Ben Panter, Edinburgh
My name (no spaces)@bigfoot which is a com.

Subject: Re: Astronomys` Sixth Neighbour Needs Help
Posted by David Fanning on Mon, 28 Jul 2003 13:45:35 GMT

View Forum Message <> Reply to Message

JD Smith writes:

> As Pavel points out, if
> you can do a substantial amount of work in each loop iteration -- more
> than the large constant overhead imposed by that cycle -- "for" loops
> can give perfectly good performance (despite the quasi-fanatical
> facetious rantings to the contrary you may find on certain respected
> IDL resource sites).

Every time I go away for a few days I am amazed
at what goes on in this newsgroup. And my wife
wonders why the first thing I do when I return
home is rush to see what's happened on the
IDL newsgroup (even before I give her a kiss)! (Sigh...)

While poor Bruno is sifting through the runes,
trying to ferret out the amazing abilities of IDL
from the tea leaves scattered in front of him,
I should just point out that some of the basics
are available for his perusal and illumination.

(Mostly written by JD, or course, serving this
year as Exalted Grand Illuminator of the Secrets
and Treasures by unanimous vote of the IDL Expert
Programmers Association members at the annual
big hoobah.)

I refer, of course, to the Dimensional Juggling
Tutorial: Working array manipulation magic with
REBIG and REFORM:

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

The Array Concatenation Tutorial: Achieving harmonious
spiritual balance with nested brackets:

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

And (if your brain hasn't exploded yet) the infamous
Histogram: The breathless horror and disgust:

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

And perhaps most germane to this topic, these three don't-
miss articles:

   Are FOR Loops the Embodiment of Pure Evil?
      http://www.dfanning.com/tips/forloops.html

   Wow! Are FOR loops really as bad as JD says they are?
      http://www.dfanning.com/tips/forloops2.html

   Whoa! Does it take, like, a *ton* of memory to subscript arrays?
      http://www.dfanning.com/misc_tips.submemory.html

I have a feeling this thread may end up there, too.
Anytime you speed IDL up from 24 days to 24 seconds
you really ought to write the steps down so you can
remember it. :-)

Cheers,

David
--
David W. Fanning, Ph.D.
Fanning Software Consulting, Inc.
Phone: 970-221-0438, E-mail: david@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: Astronomys` Sixth Neighbour Needs Help
Posted by David Fanning on Mon, 28 Jul 2003 15:32:34 GMT
View Forum Message <> Reply to Message

David Fanning writes:

> I refer, of course, to the Dimensional Juggling
> Tutorial: Working array manipulation magic with
> REBIG and REFORM:

Whoops! Sorry. REBIG is something I wrote in 1978
to compete with CONGRID. It never was popular. :-(

REBIN is what you want. :-)

Cheers,

David

--
David W. Fanning, Ph.D.
Fanning Software Consulting, Inc.
Phone: 970-221-0438, E-mail: david@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: Astronomys` Sixth Neighbour Needs Help
Posted by pford on Mon, 28 Jul 2003 18:30:03 GMT
View Forum Message <> Reply to Message

To go a bit off-topic (because I really don't know the answer):

To get the distances on a spherical surface (RA, Dec), shouldn't you
use great circle navigation equations instead of Pythagoras' theorem?
I would guess for what you are doing, the density of stars is so great
that a linear aproximation would work, but for much larger distances
the error may too large to ignore.

Regards

Patrick Ford, MD
pford@*nospam*bcm.tmc.edu

Subject: Re: Astronomys` Sixth Neighbour Needs Help
Posted by touser2001 on Mon, 28 Jul 2003 21:40:21 GMT

View Forum Message <> Reply to Message

> To get the distances on a spherical surface (RA, Dec), shouldn't you
> use great circle navigation equations instead of Pythagoras' theorem?
> I would guess for what you are doing, the density of stars is so great
> that a linear aproximation would work, but for much larger distances
> the error may too large to ignore.

You are correct but I am in the case where the distances are small and
the number of stars are many (very dense regions).

Thanks for telling me that.

Bruno
Astronomy PhD Student
University of Florida

Subject: Re: Astronomys` Sixth Neighbour Needs Help
Posted by touser2001 on Mon, 28 Jul 2003 21:53:59 GMT
View Forum Message <> Reply to Message

Overload of data here! :)

> "While poor Bruno is sifting through the runes,
> trying to ferret out the amazing abilities of IDL
> from the tea leaves scattered in front of him,"

David is right, I was meticulously sifting through JD`s message (thank
you for the detailed message!) in an attempt to make sense of as much
as possible. Now I have several links (the link:
http://www.dfanning.com/misc_tips.submemory.html is not working) to
help me in this quest!!
I am sure they will come in handy since the other procedures I have to
do are all procedures on arrays.
I am at the same time updating old versions of the program by
implementing the changes, trying to understand the changes, and trying
to continue the program to get some science done (but now everytime I
add something to the program I can`t help feeling that it must be
soooo inefficient! someday I`ll laugh (or cry) upon reviewing my
code).

Thanks everyone!!!

Bruno
Astronomy PhD Student
University of Florida

Subject: Re: Astronomys` Sixth Neighbour Needs Help
Posted by David Fanning on Mon, 28 Jul 2003 22:27:55 GMT
View Forum Message <> Reply to Message

astronomer writes:

> Now I have several links (the link:
> http://www.dfanning.com/misc_tips.submemory.html is not working) to
> help me in this quest!!

Right. This should be this:

   http://www.dfanning.com/misc_tips/submemory.html

> I am at the same time updating old versions of the program by
> implementing the changes, trying to understand the changes, and trying
> to continue the program to get some science done (but now everytime I
> add something to the program I can`t help feeling that it must be
> soooo inefficient! someday I`ll laugh (or cry) upon reviewing my
> code).

Don't worry about it. Everyone I've ever known who
leaves an IDL programming class (including the
instructor!) laments about how they are going to
have to spend the next two weeks re-writing their
programs. It's an on-going and never-ending process.

Do what you can, but don't dwell on it. Just
try to write better programs in the future.
At the end of a career no one wishes they had
spent more time revising their programs. :-)

Cheers,

David

P.S. Let's just say I'm looking at some code today
that has me shaking my head and wondering just what
(if anything) was going on in the mind of the author
when he wrote it. :-)

--
David W. Fanning, Ph.D.
Fanning Software Consulting, Inc.
Phone: 970-221-0438, E-mail: david@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: Astronomys` Sixth Neighbour Needs Help
Posted by jjbezair on Tue, 29 Jul 2003 13:34:22 GMT
View Forum Message <> Reply to Message

In article <c2959ed8.0307281353.1ea32426@posting.google.com>,
astronomer <touser2001@yahoo.com> wrote:

[snip]

> I am sure they will come in handy since the other procedures I have to
> do are all procedures on arrays.
> I am at the same time updating old versions of the program by
> implementing the changes, trying to understand the changes, and trying
> to continue the program to get some science done (but now everytime I
> add something to the program I can`t help feeling that it must be
> soooo inefficient! someday I`ll laugh (or cry) upon reviewing my
> code).
>

Keep in mind that if you are writing software that you will be using in
your own analysis (as opposed to software that will be distributed and
used by other people too) that the goal is always to minimize the total
amount of time the analysis takes you (including writing the software!). I
always have to catch myself from trying to optimize code just for the sake
of optimizing code (I have a hidden computer programmer geek side in
addition to my physicist geek side). A very wise collaborator gave me an
excellent guideline that I'm sure shaved months off of my thesis
time. Only work on making code run faster while you are waiting for it to
finish running and have nothing else to work on. This time has been
specially set aside by the computer gods for you to work on optimization.

regards,
Jeff Bezaire

>
> Thanks everyone!!!
>
> Bruno
> Astronomy PhD Student
> University of Florida

--

Subject: Re: Astronomys` Sixth Neighbour Needs Help
Posted by Pavel Romashkin on Tue, 29 Jul 2003 15:40:47 GMT
View Forum Message <> Reply to Message

I thought for a minute that REBIG has something to do with one of those
"enlargement" topics that flood my public mailbox every day :-)

Cheers,
Pavel

David Fanning wrote:
>
> Whoops! Sorry. REBIG is something I wrote in 1978
> to compete with CONGRID.

Subject: Re: Astronomys` Sixth Neighbour Needs Help
Posted by David Fanning on Tue, 29 Jul 2003 17:16:09 GMT
View Forum Message <> Reply to Message

astronomer writes:

> My latest program needs some expert help and thus I am writing this
> post in hope of a small discussion to enlighten me.

If anyone wants to read a fictionalized account of this thread
(remember, I wasn't here to read it in person), here it is:

   http://www.dfanning.com/code_tips/slowloops.html

Cheers,

David

--
David W. Fanning, Ph.D.
Fanning Software Consulting, Inc.
Phone: 970-221-0438, E-mail: david@dfanning.com
Coyote's Guide to IDL Programming: http://www.dfanning.com/
Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: Astronomys` Sixth Neighbour Needs Help
Posted by touser2001 on Tue, 29 Jul 2003 22:21:55 GMT
View Forum Message <> Reply to Message

David Fanning <david@dfanning.com> wrote in message
news:<MPG.199059c22bb394af9896bb@news.frii.com>...
> astronomer writes:
>
>> My latest program needs some expert help and thus I am writing this

>> post in hope of a small discussion to enlighten me.
>
> If anyone wants to read a fictionalized account of this thread
> (remember, I wasn't here to read it in person), here it is:
>
>   http://www.dfanning.com/code_tips/slowloops.html
>
> Cheers,
>
> David

Hehe, very funny!!!
DON`T YOU GUYS HAVE ANYTHING BETTER TO DO???? (Luckily not!)
Except for the second name and the university you nearly got it all
right. I guess it isn`t too hard to notice I had a little Fortran as
an undergrad... and yup you guys were my last shot.

Bruno Ferreira

Subject: Re: Astronomys` Sixth Neighbour Needs Help
Posted by google_forums on Mon, 11 Aug 2003 14:53:50 GMT
View Forum Message <> Reply to Message

A slight twist to the Astronomer's problem...

I am running a similar loop to Astronomer, the exception is that
rather than always needing the distance to the sixth closest point,
the point that I need depends on another variable. What I am using
this for is to spread points out across a map so that they are evenly
distributed across the map while still maintaining the "best" points.
For example, if I have a list of cities with Latitude and Longitude
and population and I want to mostly display the largest cities, but
where there are clusters of large cities (I.E. around Chicago) that
are near eachother, I would only want to display the biggest, while if
there was a smaller city in the middle of nowhere, I would want it to
show. To do this I take a score factor (in this case population) and
from each point, I calculate the distance to the nearest point that
has a higher score... I then either use this as a weight factor along
with the score, or I use it alone depending on how much I want the
score to come through vs. how evenly I want the points spread out. My
code below works fine, but as in Astronomer's case it is very slow --
I typically need to run this code for around 12,000 points but in some
cases for up to 40,000 cases -- the latter, I have yet to have the
patience for... in any case, here's the code:

```
pro closestpoint, data

; data is a preexisting structure that has fields for score (the
ranking
; field), latitude, longitude, id, and output fields to hold the
closest
; point with a higher field, the distance to that point, and a
"combined
; score" which weights the distance and the score


distance=data.distance
closest=data.closest
tmp=distance
k=0
for i=0L,n_elements(data)-1 do begin
   if k eq 0 then print, 'counter..............................',i, '
    ',data[i].id

   for j=0L,n_elements(data)-1 do begin
   lon1=data[j].longitude
   lat1=data[j].latitude
   lon2=data[i].longitude
   lat2=data[i].latitude


     if ((data[i].longitude eq data[j].longitude) and
(data[i].latitude eq data[j].latitude) $
         and (data[i].id ne data[j].id) and (data[i].score le
data[j].score)) then begin
     distance[i]=0.01
     closest[i]=data[j].id
     print, 'Station Colocation Found: ' + data[i].id + ' ' +
data[j].id + '     distance: ' + string(distance[i])
     endif else begin
           if ((data[i].score le data[j].score) and (data[i].id ne
data[j].id)) then begin

           tstdistance=map_2points(lon1,lat1,lon2,lat2, /miles)
           ;if k eq 0 then print, 'calculated
distance....',i,data[i].score, j,
data[j].score,tstdistance,distance[i]

               if ((tstdistance lt distance[i]) or (distance[i] eq 0))
then begin
               distance[i]=tstdistance
               closest[i]=data[j].id
                    if (k eq 0) then print, 'distance to ' +
data[j].id + ' = ' + string(tstdistance) +' closest=' + closest[i]
              endif
          endif
     endelse



   endfor

data[i].distance=distance[i]


k=k+1
if k eq 500 then k=0
endfor
data.combined=(data.distance * mean(data.score) / mean(data.distance)
* 4)+data.score
data.closest=closest
END
```

Subject: Re: Astronomys` Sixth Neighbour Needs Help
Posted by wmconnolley on Mon, 11 Aug 2003 15:28:32 GMT
View Forum Message <> Reply to Message

I come across this sometimes. The basis of the solution is usually
something like:

```
>    lon1=data.longitude & lon1(i)=-999
>    lat1=data.latitude  & lat1(i)=-999

>    mindist=min(sqrt((lon1-lon1(i))^2+(lat1-lat1(i)^2),j)
```

That finds you the closest city (j), without using the inner loop, and
so is much faster. OK, it uses distance in lat-lon space: if you care about
the exact coordinates you can use convert_coord to get it in whatever
map projection you are using.

-W.

--
William M Connolley | wmc@bas.ac.uk | http://www.antarctica.ac.uk/met/wmc/
Climate Modeller, British Antarctic Survey | Disclaimer: I speak for myself
I'm a .signature virus! copy me into your .signature file & help me spread!

Subject: Re: Astronomys` Sixth Neighbour Needs Help

Posted by google_forums on Mon, 11 Aug 2003 21:20:57 GMT
View Forum Message <> Reply to Message

The trick is that I need the closest point that satisfies my condition
(the closest point that has a score -- population or whatnot --
greater than the test station)....

-David

wmc@bas.ac.uk wrote in message news:<3f37b61f@news.nwl.ac.uk>...
```
>  I come across this sometimes. The basis of the solution is usually
>  something like:
>
>>    lon1=data.longitude & lon1(i)=-999
>>    lat1=data.latitude  & lat1(i)=-999
>
>>    mindist=min(sqrt((lon1-lon1(i))^2+(lat1-lat1(i)^2),j)
>
>  That finds you the closest city (j), without using the inner loop, and
>  so is much faster. OK, it uses distance in lat-lon space: if you care about
>  the exact coordinates you can use convert_coord to get it in whatever
>  map projection you are using.
>
>  -W.
```

Subject: Re: Astronomys` Sixth Neighbour Needs Help
Posted by wmconnolley on Tue, 12 Aug 2003 08:40:30 GMT
View Forum Message <> Reply to Message

Bitner <google_forums@gyttja.org> wrote:
```
> The trick is that I need the closest point that satisfies my condition
> (the closest point that has a score -- population or whatnot --
> greater than the test station)....

> wmc@bas.ac.uk wrote in message news:<3f37b61f@news.nwl.ac.uk>...
>>  I come across this sometimes. The basis of the solution is usually
>>  something like:
>>
>>>    lon1=data.longitude & lon1(i)=-999
>>>    lat1=data.latitude  & lat1(i)=-999
>>
```

That should have been:

```
mindist=min(sqrt((lon1-lon(i))^2+(lat1-lat(i)^2),j)
```

If you need other conditions then:

```
k=where(score gt whatnot)
mindist=min(sqrt((lon1(k)-lon(i))^2+(lat1(k)-lat(i)^2),jj)
j=k(jj)
```

-W.

--
William M Connolley | wmc@bas.ac.uk | http://www.antarctica.ac.uk/met/wmc/
Climate Modeller, British Antarctic Survey | Disclaimer: I speak for myself
I'm a .signature virus! copy me into your .signature file & help me spread!

Subject: Re: Astronomys` Sixth Neighbour Needs Help
Posted by google_forums on Tue, 12 Aug 2003 16:11:46 GMT
View Forum Message <> Reply to Message

```
>>>
>>>>    lon1=data.longitude & lon1(i)=-999
>>>>    lat1=data.latitude  & lat1(i)=-999
>>>
```

Why the lon1(i)=-999 ?

```
>
>  That should have been:
>
>  mindist=min(sqrt((lon1-lon(i))^2+(lat1-lat(i)^2),j)
>
```

Should this be :
mindist=min(sqrt((lon1-lon1(i))^2 + (lat1-lat1(i))^2),j)

Thanks,
David

Subject: Re: Astronomys` Sixth Neighbour Needs Help
Posted by wmconnolley on Tue, 12 Aug 2003 16:57:06 GMT
View Forum Message <> Reply to Message

Bitner <google_forums@gyttja.org> wrote:

```
>>>>
>>>>  >   lon1=data.longitude & lon1(i)=-999
>>>>  >   lat1=data.latitude  & lat1(i)=-999
```

> Why the lon1(i)=-999 ?

So it doesn't match itself as the min.

>> That should have been:
>>
>> mindist=min(sqrt((lon1-lon(i))^2+(lat1-lat(i)^2),j)

> Should this be :
> mindist=min(sqrt((lon1-lon1(i))^2 + (lat1-lat1(i))^2),j)

No, because lon1(i) has been set to junk, so use lon(i) which has the
true value.

-W.

--
William M Connolley | wmc@bas.ac.uk | http://www.antarctica.ac.uk/met/wmc/
Climate Modeller, British Antarctic Survey | Disclaimer: I speak for myself
I'm a .signature virus! copy me into your .signature file & help me spread!