Subject: Re: An algorithm puzzle
Posted by Jonathan Dursi on Sat, 14 Jun 2008 06:09:49 GMT
View Forum Message <> Reply to Message

On Jun 14, 12:09 am, David Fanning <n...@dfanning.com> wrote:
> Y.T. writes:
>> I'm currently brute-forcing it with two for-loops where I calculate
>> the distance between every single element and every single "other"
>> element and then finding the minimum. Needless to say this takes about
>> a metric forever and I figured you folks usually have really clever
>> ideas so I'm throwing this out here to see whether there isn't some
>> obscure usage of histogram that does exactly what I want...
>
> And I'll steal an IDL T-shirt from the IDL Workbench Seminar
> next Tuesday for the first person who's solution runs in
> less than, say, 10 seconds! Be sure to specify your size. :-)

Ok, I won't swear this is 100% yet because it's late, but it's a fun problem and I wanted to give it a go.

These sorts of shortest-path problems immediately call to mind something
like Dijkstra's algorithm:
http://en.wikipedia.org/wiki/Dijkstra%27s_algorithm
which is a remarkably simple algorithm for finding the shortest paths between points in a graph. For these sorts of problems, `greedy' methods
work really well.

So this is my attempt at an implementation -- the trick here being to pretend
that all zeros are really just one vertex with lots of neighbors, and to
proceed from there. This is a really hacky attempt, but seems to work
at least for the simple cases:

```
IDL> print, byte(barr)
  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  1  1  1  1  1  1  1  1  0  0
  0  0  1  1  1  1  1  1  1  1  0  0
  0  0  1  1  1  1  1  1  1  1  0  0
  0  0  1  1  1  1  1  1  1  1  0  0
  0  0  1  1  1  1  1  1  1  1  0  0
  0  0  1  1  1  1  1  1  1  1  0  0
  0  0  1  1  1  1  1  1  1  1  0  0
  0  0  1  1  1  1  1  1  1  1  0  0
  0  0  1  1  1  1  1  1  1  1  0  0
```

```
   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0

IDL> dijkstra, barr, rarr
in iter:       1:          36 active cells
at end of iter:       1:          16 active cells
in iter:       1:          16 active cells
at end of iter:       1:           4 active cells
in iter:       1:           4 active cells
at end of iter:       1:           0 active cells

IDL> print, byte(rarr)
   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   1   1   1   1   1   1   1   1   0   0
   0   0   1   2   2   2   2   2   2   1   0   0
   0   0   1   2   3   3   3   3   2   1   0   0
   0   0   1   2   3   4   4   3   2   1   0   0
   0   0   1   2   3   4   4   3   2   1   0   0
   0   0   1   2   3   3   3   3   2   1   0   0
   0   0   1   2   2   2   2   2   2   1   0   0
   0   0   1   1   1   1   1   1   1   1   0   0
   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0


pro dijkstra, barr, rarr

        infinity = 1e14

        s = size(barr,/dimensions)
        rarr = fltarr(s[0]+2,s[1]+2)+1
        rarr[1:s[0], 1:s[1]] = barr

        rarr[where(rarr gt 0)] = infinity

        min4neigh = min([[[shift(rarr,1,0)]],[[shift(rarr,-1,0)]],
[[shift(rarr,0,1)]],[[shift(rarr,0,-1)]]],dimension=3)
        rarr[where((rarr ge infinity) and (min4neigh eq 0))] = 1.
        min8neigh = min([[[shift(rarr,1,1)]],[[shift(rarr,-1,1)]],
[[shift(rarr,-1,-1)]],[[shift(rarr,1,-1)]]],dimension=3)
        rarr[where((rarr ge infinity) and (min8neigh eq 0))] =
sqrt(2.)

        iter = 1
        active = where(rarr ge infinity, nactive)
        while (nactive gt 0) do begin
                print, 'in iter: ', iter, ': ', nactive,' active
```

```
cells'
            min4neigh = min([[[shift(rarr,1,0)]],
[[shift(rarr,-1,0)]],[[shift(rarr,0,1)]],[[shift(rarr,
0,-1)]]],dimension=3)
            min8neigh = min([[[shift(rarr,1,1)]],
[[shift(rarr,-1,1)]],[[shift(rarr,-1,-1)]],[[shift(rarr,
1,-1)]]],dimension=3)

            newdist = min([[min4neigh[active]+1.],
[min8neigh[active] + sqrt(2.)]],dimension=2)
            better = where(newdist lt infinity,nbetter)
            if (nbetter eq 0) then begin
                  print, 'Something is horribly wrong --
iteration did nothing'
            end else begin
                  rarr[active] = newdist
            end

            active = where(rarr ge infinity, nactive)
            print, 'at end of iter: ', iter, ': ', nactive,'
active cells'
      endwhile

      rarr = rarr[1:s[0],1:s[1]]
return
end


  Jonathan
--
Jonathan Dursi
ljdursi@cita.utoronto.ca
http://www.cita.utoronto.ca
```