Subject: Re: Array searching efficiency
Posted by penteado on Fri, 11 Feb 2011 00:32:10 GMT
View Forum Message <> Reply to Message

On Feb 10, 9:47 pm, Matt Francis <mattjamesfran...@gmail.com> wrote:
> The first step is to establish, for a given time, which two maps I
> need to interpolate between. I have the times for each map stored in a
> single array, in time order. If I can work out the indices in that
> array of the two map times then I'm done. This is a simple problem to
> solve in any number of ways, but the question is which is the fastest?
>
> I've come up with this:
>
> iup = (WHERE(times-time_now GT 0))[0]
> ilow = iup-1
>
> I'm not sure how WHERE works 'under the hood', but assuming that at
> some level it loops over the given array, then optimally once [times -
> time_now] becomes positive you would stop searching. Implementing that
> kind of algorithm in a WHILE loop is probably slower than using WHERE
> though.

where() is certainly *not* optimal for this. It has no reason to stop
searching on the first occurrence. It will keep searching to the end,
as its job is to return every occurrence, and it cannot assume there
will be only one. Besides, you would be doing one call of where() for
each time you are searching for. A very wasteful repeat of searches.

A single call to value_locate does this. Something like

ind=value_locate(times,times_to_search)

will return an array with the index for each value in times_to_search.
Whether the returned index is below or above the value you search for
depends on the ordering of times. See the help on value_locate.