Subject: Re: Efficient comparison of arrays
Posted by J.D. Smith on Wed, 13 Aug 1997 07:00:00 GMT
View Forum Message <> Reply to Message

David Foster wrote:
>
> J.D. Smith wrote:
>>
>> Just to keep the Astronomy department here from being one-upped....
>>
>> <SNIP>
>>
>> Here is an implementation I just made up:
>>
>> function contain,a,b
>> flag=[replicate(0b,n_elements(a)),replicate(1b,n_elements(b))]
>> s=[a,b]
>> srt=sort(s)
>> s=s(srt) & flag=flag(srt)
>> wh=where(s eq shift(s,-1) and flag ne shift(flag, -1),cnt)
>> if cnt ne 0 then return, s[wh]
>> return,-1
>> end
>>
>> I ran some time tests on the two implementations. While a_in_b is
>> adequate for small vectors, it is prohibitively slow for large ones. An
>> example averages the result in seconds for two 10000 element random
>> integer vectors on the range [0,20000].
>>
>> Results for a_in_b:
>> Average Time: 19.669667
>>
>> Results for contain:
>> Average Time: 0.19233332
>>
>> Ratio: 102.269
>>
>> And for 100 element vectors in the range [0,200]:
>>
>> Results for a_in_b:
>> Average Time: 0.010666664
>>
>> Results for contain:
>> Average Time: 0.0015666644
>>
>
> When you choose a method make sure you test the solutions on
> data that is typical to your operations; don't rely on time

> postings based on artificial situations. Below are results
> comparing FIND_ELEMENTS.PRO (my routine that I've posted already)
> and JD Smith's CONTAIN.PRO function listed above.
>
> The test data are:
>
> A = BYTARR(65536)
> A 256x256 image which is a section of the brain that
> has been coded into discrete values to represent the
> different structures in the brain. Roughly in the
> range 0-128, many repeated values (compresses well).
> Very typical for my situation.
>
> B = BINDGEN(50)
>
> Here are the results:
>
> IDL> t1=systime(1) & c = FIND_ELEMENTS(a,b) & t2=systime(1) & $
> print, t2-t1
> 2.3154050
> IDL> t1=systime(1) & d = CONTAIN(a,b) & t2=systime(1) & $
> print, t2-t1
> 132.54824
>
> In some situations the more primitive approach may be better
> (JD Smith's solution is certainly much more elegant and clever).
> Also be aware that some solutions like FIND_ELEMENTS() and
> WHERE_ARRAY() return *all* subscripts for items found, including
> repeats, whereas CONTAIN() does not.
>
> Dave

This is an important point. However, I don't quite understand you're timings. I ran your code with a=round(randomu(sd,65536)*128) and b=bindgen(50). Now granted that I didn't have your brain data, but the values I got were:

contain() time (sec):
 Average Time: 0.73649999

find_elements() time (sec):
 Average Time: 1.2958000

Not an extreme advantage (and one which would fail for smaller b's), but

clearly different from the values you indicate. I am running on a Pentium 166 Linux machine. Perhaps just another indication of the hardware subtleties we've all grown accustomed to.

On another point, it is important to note that the problem of finding where b's values exist in a (find_elements()) is really quite different from the problem that contain() attempts to address: finding those values which are in the intersection of the vectors a and b (which may be of similar sizes, or quite different). The former is a more difficult problem, in general, which nonetheless can be solved quite rapidly as long as one vector is quite short. But the time taken scales as the number of elements in b, as opposed to the comparative size of b (to the total elements in a and b) -- i.e. nearly constant with increasing length of b. Anyway, it is important to understand the various scales, sizes and efficiencies in the problem you are trying to solve if you hope to come up with an effective solution.

JD
