Subject: Re: Efficient comparison of arrays
Posted by David Foster on Thu, 14 Aug 1997 07:00:00 GMT
View Forum Message <> Reply to Message

J.D. Smith wrote:
> David Foster wrote:
>>
>> 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).
>>
>> B = BINDGEN(50)
>>
>> Here are the results:
>>
>> FIND_ELEMENTS() : 2.3154050
>>
>> CONTAIN() : 132.54824
>
> 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.

Your point is well taken about hardware subtleties. On a Sparc 2 running
Solaris 2.5 here are the timings on the same data you uses above:

contain() (sec) : 10.12
find_elements() (sec) : 4.40

God I hate working on a Sparc 2!

> 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.

All very true. With any method there are going to be some tradeoffs.
But I am skeptical of a method that relies on the sorting of the
arrays in question. In my timing above for CONTAIN(), of the 10.10
seconds, 9.40 are spent sorting the array! On some hardware and with
some data this may not be a problem; on my hardware and with my
data it most definitely is.

I'm just saying that people should check first. When WHERE_ARRAY()
was first posted, it was touted as a superior algorithm simply because
it was vectorized. But in fact it is slow and requires a hell of a
lot of memory. Your method is quite clever and probably works well
in many situations, but people shouldn't rely on posted timings to
compare methods; they should time the methods themselves, on their
machines and with their data.

I have always thought that IDL should provide this functionality, as
well as the removal of the intersection of two arrays from one of
the arrays.

Dave
--

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ ~~~~~~~
David S. Foster Univ. of California, San Diego
Programmer/Analyst Brain Image Analysis Laboratory
foster@bial1.ucsd.edu Department of Psychiatry
(619) 622-5892 8950 Via La Jolla Drive, Suite 2240
La Jolla, CA 92037
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ ~~~~~~~

k