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

> <snip>
>
> 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.
>

It all boils down to this... the bulk of the time taken by contain() is in sorting. This is obvious. Let a and b be the two vectors in question. Let a have n elements and b have m elements. The approximate number of operations to do the sorting is then (n+m)log(n+m) for an efficient sorting algorithm, on average. On the other hand, find_elements() necessarily takes on order (n x m) operations (for each of the m elements in b, compare it with all n elements in a). If n>>m then the sorting term is approximately nlog(n). Which method takes more operations? The ratio of the two operation counts is r=log(n)/m. When this is unity, the two methods will be roughly on equal footing. If r is much greater than 1, find_elements() will be faster. For r much less than one, contain() with it's sorting will be faster. In the case of n=m, r=2log(2n)/n << 1 for any sizeable n (> 10, say).

So, truly, it does depend critically on your data. I found, on my machine, an equality at approximately m=25 for n=65536. Log(n)=16 in this case, so it's not too far off. For larger, n, the test gets more accurate (until memory becomes an issue).

JD