

Re: Help needed for a sorting code in the literature

Source: <http://www.derkeiler.com/Newsgroups/sci.crypt/2004-12/1941.html>

From: Gregory G Rose (ggr_at_qualcomm.com)

Date: 12/30/04

Date: 30 Dec 2004 12:12:33 -0800

In article <cr1e53\$N8s\$03\$1@news.t-online.com>, Mok-Kong Shen <mok-kong.shen@t-online.de> wrote:

>

>The code of the heapsort algorithm as given in books by
>e.g. Sedgewick deals with arrays with indices of the kind
>a[1..n]. For PLs like C, one would like to sort arrays with
>indices of the kind a[0..n-1]. I have found sofar only one
>book that gives code for the latter case, namely "Numerical
>Recipes in C++" by W. H. Press et al. (which itself refers
>to Sedgewick). I adapted the code there (which is in C++
>and is capable of dealing with a variety of data types being
>sorted) to C for sorting arrays of 'int'. However, the
>resulting code does not always function correctly. As concrete
>examples, of 6 random sample cases tested for an 'int' array
>of size 5 consisting of [0..4] the following input

>

> 2 1 3 0 4

>

>was wrongly sorted to

>

> 0 1 2 4 3

>

>while the following inputs

>

> 3 0 4 2 1

> 4 1 0 2 3

> 3 0 1 4 2

> 2 3 1 4 0

>

>were all processed correctly.

>

>Since I only have the book but not the corresponding (original)
>software, I couldn't be sure whether this is due to my coding
>error in adapting the code from the text of the book or whether
>this is due to an error of the authors of the book (though I
>have carefully checked my coding several times to be sure that
>it corresponds to the code given in the book). If there is
>some person of our group who happens to possess the original

sci.crypt: Re: Help needed for a sorting code in the literature

>software of the book, I should appreciate very much to know
>a confirmation/refutation of my computing result above.
>
>M. K. Shen
>

I can't help you with the book, but I might be able to help you with a possible problem in their/your translation.

Heapsort works by forming a "heap", which has a certain invariant: that all elements twice as far down the array as the current element are {greater, or less, than} (depending on just how you implement) it.

Now, when your array origin is 1, "twice as far down" is expressed as:

$\{a[i] > a[2*i] \ \&\& \ a[i] > a[2*i + 1]\}$.

But if your array is origin 0, it *should* be expressed as:

$\{a[i] > a[2*i + 1] \ \&\& \ a[i] > a[2*i + 2]\}$.

(All this ignoring that you need to check whether you've run off the end of the array, of course. :-) Somewhat surprisingly, getting this wrong will still form something very much like a heap, and hence the heapsort algorithm will *almost* sort your input, and often will get it right by luck. (I know -- I've been there.)

I think that's what's happening to you.

I like heapsort -- it suffers no worst-case behaviours like quicksort; it always runs in $n \log n$ time.

Greg.

--

Greg Rose
232B EC8F 44C6 C853 D68F E107 E6BF CD2F 1081 A37C
Qualcomm Australia: <http://www.qualcomm.com.au>