And the Teacher Shall Be Taught
Timothy Rolfe
Professor of Computer Science Emeritus
Eastern Washington University
319F CEB
Cheney, Washington 99004-2493 USA
http://penguin.ewu.edu/~trolfe/
mailto:Timothy.Rolfe@mail.ewu.edu

Herb Holden, a colleague of mine at Eastern Washington University, had a pet peeve: implementations of binary search that test first for equality (and exit). Thus the least likely condition is checked first, guaranteeing that the search will always use two comparisons until it exits. I find I was guilty of a comparable inefficiency in a programming assignment I posed for my graduate algorithms class, ending up being corrected by a pair of my students.

Berman and Paul, in their Fundamentals of Sequential and Parallel Algorithms [1], pose as an exercise the development of an algorithm to determine the largest two elements in an array. In working up a specimen sequential algorithm, I first checked for new maximum, and then checked for new second largest; thus I begin with, on average, checking for an unlikely event and so end up with nearly two comparisons for each loop iteration.

```java
for ( k = 2; k < n; k++ )
{  if ( x[k].compareTo(maxValue1) > 0 )
{  maxValue2.set(maxValue1);
maxValue1.set(x[k]);
  }
else if ( x[k].compareTo(maxValue2) > 0 )
maxValue2.set(x[k]);
}
```

Ryan Baldwin and Kristopher Smith, however, did the intelligent thing: they first checked whether one of the two values needed changing, so that, on average, each loop iteration required a single comparison.

```java
for ( k = 2; k < n; k++ )
{  if ( x[k].compareTo(maxValue2) > 0 )
if ( x[k].compareTo(maxValue1) > 0 )
{  maxValue2.set(maxValue1);
maxValue1.set(x[k]);
  }
else
maxValue2.set(x[k]);
}
```

As established by numerical experiment with random data, their simple sequential implementation comes very close to the lower bound complexity that Berman and Paul establish for this problem: \( n + \lceil \log_2 n \rceil - 2 \). [2] (It does, of course, have worst-case complexity of \( 2n - 3 \), as does the first algorithm described.)

The lesson: sometimes a simple reordering in your program can gain a doubling in speed.
