

# Re: "Sorting" assignment

---

*Source:* <http://coding.derkeiler.com/Archive/General/comp.programming/2008-02/msg00224.html>

---

- *From:* spinoza1111 <spinoza1111@xxxxxxxx>
  - *Date:* Thu, 7 Feb 2008 20:54:43 -0800 (PST)
- 

On Feb 7, 10:40 pm, Julienne Walker <happyfro...@xxxxxxxx> wrote:

On Feb 6, 10:52 pm, spinoza1111 <spinoza1...@xxxxxxxx> wrote:

On Feb 6, 12:20 am, Julienne Walker <happyfro...@xxxxxxxx> wrote:

On Feb 5, 10:56 am, spinoza1111 <spinoza1...@xxxxxxxx> wrote:

What's interesting is how excited people have gotten about truly de minimis performance issues while claiming, in practice, that an  $N^2$  algorithm such as the bubble sort should be given a free pass because it seems, oh, so very simple. Sure, it's simple, so simple that in implementing it you learn nothing of value.

Learning bubble sort can teach you a lot about algorithms.  
For

Cf. Thomas H Cormen et al. Introduction to Algorithms MIT 2001. Cormen starts with insertion sort because it doesn't quite square the array length, and his first full chapter on sorting describes heap and quick sorting.

Note that I said bubble sort can teach a lot about algorithms, not that it *should* be taught as a first sorting algorithm or that everyone teaches it as a first sorting algorithm. I even explained how I would tailor which algorithm to start with by how the student thinks

## Re: "Sorting" assignment

and that this will almost always result in insertion sort or selection sort. ;-)

example, you can use bubble sort as a simple implementation for adding step-based execution and performance analysis. It's a simple algorithm for introducing asymptotic notation and time complexity. It's also an immediately useful exercise that covers non-trivial loop and array logic, modularization, and comparison based sorting. I'd hardly call that "nothing of value".

Nothing of beauty.

The reality of programming is that it's not always elegant or beautiful. If you only learn the beautiful side of programming, you won't be very useful in the real world.

...and we must, mustn't we, be useful in the real world.

Some poor soul in San Francisco, a sixty year old hippie named Da Vid, lost a ballot initiative yesterday to have Alcatraz turned into a global peace park.

Ha ha ha.

Note our reaction, one which doesn't come from within, but is owned curiously in the way people say things, self-protectively, in the corporation.

It's bwah, ha ha.

Despite the fact that Alcatraz is a fucking eyesore and a memorial to a generation of Depression kids who were railroaded into lives of crime, who were indeed "victims of society", and despite the fact that a global peace park might be a better and more inspiring place to bring the kids.

News fa lash. There is an unnoticed distinction here between two educational styles. In the one, we learn things are beautiful because they are there, and cannot be jiggered by the teacher, but can be understood, as if they had after all an independent reality...

## Re: "Sorting" assignment

and education in which the difficulty and essence of what is learned is always being negotiated offline.

Let's just be aware for starters that I've learned Buddhist right livelihood and am almost as old as Da Vid, and have no interest in being useful in creating any new Alcatraz.

I'd say that the students have the right to start with something of reasonable elegance and beauty which derives that elegance and beauty from the fact that we developed this algorithm before computers. The insertion sort can be illustrated and is illustrated by Cormen using a poker hand.

I'd say that the students have the right to start with whatever is easiest for them to understand. That way they can progressively learn

Why? Why not start with the hard stuff, get it over with?

That's what we did in high school Latin. That's what Chinese kids do in learning ideograms. And my first CS professor started us out in machine language.

the gamut of algorithms and build on existing knowledge. Once more I'll offer myself as an example. I didn't understand the "poker hand" explanation of insertion sort at first. I had a much easier time of things once I had a couple of sorting algorithms under my belt.

It doesn't matter how elegant an algorithm is if the student can't wrap her mind around it.

Who is "the student" here? The solecism is similar to the solecism noticed by Dijkstra in a text in Dutch in which the English word for the "user" was untranslated because to the translator, who knew more English grammar, apparently, than many English speakers, a noun phrase that starts with a definite article needs to have a preintroduced referent.

You don't tell children a story by saying the monster came out of the closet. You introduce the monster using the indefinite article and appropriate adjectives.

Re: "Sorting" assignment

## Re: "Sorting" assignment

Likewise, Dijkstra didn't know what SORT of user was being spoken of. Here, all I know is that the student is female (hint: there is no solution to the absence of a non-neuter pronoun in English that means male or female person, but in your passage you could replace "her" with "the student's").

I for one am sick to death of being treated as an abstraction by educationist theoretical types who reason in terms of signifiers with no referent.

Keep in mind that those learning bubble sort are likely to be very new to programming, so while implementing the algorithm won't teach \*you\* anything, it's very instructive to a less experienced programmer.

<snip> However, he did see a contradiction between the tentative way in which people talk about other people in sets and classes (such as "less experienced programmer") in such a way that the rule governing the set replaces thought.

I get the impression you're basically saying "exceptions exist". That's nice, but exceptions are just that: exceptions. You're welcome to show me a study or two that proves students learn nothing from bubble sort, but until then I'll maintain that lessons can be learned even from an "ugly" algorithm.

The "studies" are studies that established in the 1990s that a striking plurality of employed programmers DO NOT CODE.

They get coffee at Starbuck's, they bring up SQL Server, they make changes to a data base, they look at code and refuse to read it, they run down their coworkers behind their coworkers' back especially if their coworkers aren't young white males, they get into this ng, and they write "white papers" in bonehead English.

They take classes in Rational development methodologies. They play computer games.

The fact is that sometime in the 1990s, "how to debug a C program"

Re: "Sorting" assignment

Re: "Sorting" assignment

became "change your major".

The fact is that an older person can indeed have learned something like C in 1992 in order to help John Nash and make bonehead mistakes ten years later. This is why air traffic controllers aren't allowed at first to separate tin after a season at Betty Ford. But in education, older people with older certifications falsely claim that they are thinking of the students when they oversimplify the material. In many cases they are trying to avoid embarrassment.

It is true that things can be simplified. My own book, "Build Your Own .Net Language and Compiler" (I wanted to call it "Build Your Own Goddamn .Net Language and Compiler" but Apress would not let me) is in a sense a prologomena to the Dragon book in this regard.

But the problem that confronts the CS instructor in a diverse environment is that some kids will relate to the bubble sort, and they should not so cathect.

-Jul