

Visualizing Sorts ↓

5 2 7 4 1 6 3 0

"Each of our students is going to be representing an integer. That int is initially going to be in unsorted order, and I claim that using an algorithm, step by step instructions, we can probably sort these folks in at least a couple of different ways."

So they're in wardrobe right now just getting their very own Howard T-shirts (...), which will then represent an element of our array.

How would you go about sorting these eight numbers on the screen?


"the number at the end, the following number."

"It is bigger, then I keep it as it is..."

"If not, then change them..."

So if it's out of order, you would just start to SWAP things. And that seems reasonable.

there's a whole bunch of mistakes to fix here, because things are pretty out of order.

But probably, if you start to solve small problems at a time, you can achieve the end result of getting the whole thing sorted. * 

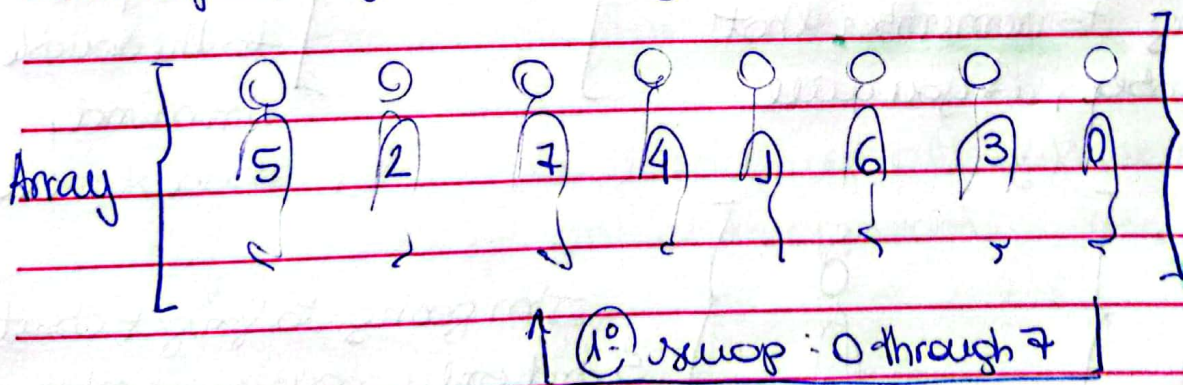
Other instincts, if you were just handed these numbers, how you might go about sorting them?

→ "find the smallest one first and put it at the beginning."

→ do that again and again...

that would seem to give you a couple of different algorithms.

"If you all are attired here, do you want to come on up if you're ready?"



Computer can ONLY look at 1 memory location, at

at one locker at a time, so can a computer only max 1 number at a time — sort of opening a locker, checking what's there, moving it as needed. *

→ So let's try this more methodically based on the 2 audience suggestions...

Eventually, I'll go to translate this to pseudocode and then code.

- 1^o Search for the smallest number;
- 2^o Starting from left to right; → CONVENTION
- 3^o

0
5

5 at this moment is the smallest number I've seen.

I'm going to remember that in a variable, if you will

TAKE 1 MORE STEP

0
2

Ok, I'm going to compare to the variable in mind, obviously <<<

I'm going to update the variable in mind, because 2 is smaller

0
7

ignore that because is > than 2

I'm going to forget about 5 and only now remember 2 as the new smallest element.

* I have to check All values to see if there's something even smaller because 6 isn't, 3 isn't and 0 is.

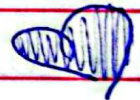
— Where should number 0 go according to this proposed algorithm? —

"So before doing this for real, let's have you pop out in front."

↓
It's not a good idea. It's a lot of work!

So what would be marginally smarter perhaps?
A little more efficient?

S W A P P I N G
replace 2 values.



"Software typically, at least as we've written it thus far, can only do one thing at a time."

So in a bit, we'll add up all of these steps. But for now, let's take one other approach. involve

Selection Sort → PSEUDOCODE →

3-17 numbers.

For i from 0 to $n-1$

Find smallest number between numbers $[i]$ and numbers $[n-1]$

Swap smallest number with numbers $[i]$

That was a lot of steps to be adding up, it's probably more than (n) , because I went through the list again and again.

$$n + (n-1) / 2 \dots$$

No matter what n is — and the bigger it is, the bigger raising it to the power 2 is going to be.

At the end of the day, n gets large (really large), the dominant factor is indeed that 2.

So that is to say

big O

$$O(n^2)$$

Selection
~~Bubble~~ Sort

a computer scientist

would describe bubble sort as taking on the order of n squared steps.

If we want to just be able to generally compare 2 algorithms' performance, I think it's going to suffice if we look at the highest order term to get a sense of what the algorithm feels like, if you will, or what it even looks graphically.

But in the best case, how many steps does selection sort take? What could you imagine meaning the best possible scenario when you're trying to sort a bunch of numbers?

↳ They're already sorted, right? ☺

↳ But does this algorithm leverage that fact in practice? Best I only knew that zero needs to be at the beginning once I've looked at all 8 people.

And then I would have realized, that was a waste of time. But then what ~~if~~ would I have done?



that would seem to imply the omega notation, the best case scenario, even, a lower bound on the running time would be what, then?

$$\Omega(n^2)$$

