Mike Mossey's guide to studying computer science

web: https://learnwithmikemossey.com
email: mike@learnwithmikemossey.com

This is a guide for students studying computer science. It covers some of the ideas I teach to my students every day. Here, I'm giving you these ideas for free. It's my sincere hope that you benefit whether you choose to hire me or not.

The main problem I address in this guide is: You're faced with an AP question or exam question about writing an algorithm. You have no idea what to do or what specific techniques are needed.

You're faced with a difficult question and you have no idea what to do

If you're stuck, you know you have to take at least the first step. But how do you find it?

It seems difficult, because all the pieces of your code need to work together. How do you choose them, step-by-step, starting from the beginning, so that all of them work together in the end?

We'll look at two strategies. (1) Writing the test data first, and asking how you would solve the problem as a human. (2) Writing the final algorithm in any order you need to.

Writing the test data first

I recommend you start by writing test data and asking how *you* would solve the problem, as a human. I don't mean how you would program it, but how you would do it if you were given the same job as your program.

Say your program needs to find the maximum number in a list. First you'll create some test data. So write a list of numbers.

As an aside, this technique of creating the test data first is useful in almost all exam and homework questions about algorithms.

Asking yourself how *you* would solve the problem

So write a list of numbers and look for the biggest one. And notice what you're doing with your eyes and your brain.

Most students tell me something like “I’m just looking at the page and I see the biggest one.” That is how humans often feel when we tackle a problem. We feel like we’re able to process many details at once and just “see” the answer. That’s fine, but then we need to consider *how a program does that.*

Move your eyes in slow motion to discover the steps

A program works step-by-step. So you need to slow down your process: again look at a list of numbers and move your eyes in slow motion. It will soon be clear that your eye has to move across each number, one by one.

Think about how you find the maximum number. If you think about it a little bit, you’ll probably realize that you keep mental track of the largest number you’ve seen so far. Say the first number is 10. As you scan your eyes to the right, you might spot a number bigger than 10, like 20. So the biggest number so far is 20. Keep scanning. You see numbers like -1 or 17, which you know you can ignore because they are less than 20. You keep scanning and replacing the biggest number so far until you reach the end of the list.

What you discover about yourself becomes the algorithm

So you need to scan the numbers one by one. *That’s* something a program can do easily. This is a step by step algorithm that we can implement in Java.

Our code might look something like this. Take a brief look at this, and we’ll discuss it next.

```
Java:

int[] numbers = {1, 8, 56, -100, 23, 40};
int max = numbers[0];
for (int i = 1; i < numbers.length; i++)
{
   if (numbers[i] > max)
   {
      max = numbers[i];
   }
}
System.out.println(“The maximum number is: “ + max);
```

So we need code that works step-by-step. To keep your code simple, try to think of similar problems you’ve seen before… do that before you try a complicated solution.

Write the code in any order

Okay, let's assume that you couldn't quickly come up with the code above. How can you approach it?

The main important step I advocate is allowing yourself to write the code in any order, rather than always working in the same order as the final code.

Write the test data first

In this case, you would write the test data first. It won't be too hard to translate that into Java. It will look something like the first line above:

```
int[] numbers = {1, 8, 56, -100, 23, 40};
```

Try simple solutions you've seen before, like a loop

If you're midway through a computer science class, you know that a loop can work here... it can step through a list of numbers. No need to invent that yourself.

So now, write the outline of the loop without worring about the details.

```
for (int i = 0; i < numbers.length; i++)
{
    … something to be determined goes here…
}
```

Note that I'm starting this suggestion by writing the outline of the loop first. This is just one possible order. You can experiment in *any* order you like. I'm a believer in the fact that however you write it, you'll eventually come around to the solution, even if you have a make a few corrections along the way.

Note that I'm starting the loop at index 0. This fact will come back later.

Fill the inside of the loop

Next you need a way to keep track of the maximum number. The variable 'max' does that in the code above.

So next you need to fill in the inside of the loop. I realize I'm going fast here, but if you're in a computer science class, the teacher will have given you many opportunities to learn these things and the overall pace will be slower. The important thing is to have some way of piecing together the information you're learning in class, and I hope my ideas here can help.

```
for (int i = 0; i < numbers.length; i++)
{
```

```
    if (numbers[i] > max)
    {
        max = numbers[i];
    }
}
```

You know the middle of the loop needs to address this idea of tracking the max number seen so far. You have probably seen something like the above.

Notice that I haven't declared or initialized 'max' yet. At this point, I'm just thinking that I need something like it. In fact, I'm not sure yet if it's the right solution. I just put that in there to see what ideas I come up next.

Declare and initialize variables

Now I'll think above how to declare and initialize 'max'. Notice how this is one of the first lines of code in the order of the program, but I'm getting to it *last.*

```
int[] numbers = {1, 8, 56, -100, 23, 40};
int max = numbers[0];
for (int i = 1; i < numbers.length; i++)
{
    if (numbers[i] > max)
    {
        max = numbers[i];
    }
}
```

Since I don't want this initial guide to get too long, I went ahead and took the liberty of introducing another idea here. Remember that when you use your eyes to do this, you start with the first number and scan to the right? That's why I initialized 'max' to the first number, 'numbers[0]'.

And then I modified the loop. Because I'm already starting with the first number, I can skip it in the loop; that is, I start the index of the loop at '1' (which is the *second* number in a zero-origin language like Java).

Things to remember

What you can take away from this guide is

- Write the test data first.
- Experiment with your test data as if *you* were doing the same job as the computer.
- Try filling in the code in a non-linear order; that is, you don't have to write it in order from the beginning to the end.

Enjoy studying computer science!