

Implementing Selection Sort

Module Learning Outcomes / Resources	10 min
Introduction	1 min
Developing an Algorithm	4 min
Translating to Code	3 min
In Place	9 min
Efficiency	5 min
Summary	1 min
Programming Exercise: Implementing Selection Sort	10 min
Practice Quiz: Implementing Selection Sort	5 questions


Sorting at Scale

Review

Sorting in Place

5617433

- Often want to sort **in place**
 - Reorder input array
 - Do not use separate output array



be sorted without making any other new array list to hold the output.

In Place

Have a question? Discuss this lecture in the week forums.

Interactive Transcript

Search Transcript

English

0:03

You have just written a sort which destroys the input and produces its output in a new array list. However, programmers often want to sort in place, modifying the input array list to [be sorted without making any other new array list to hold the output](#). You can sort in place with a similar principle to what you just did except that you swap the elements inside the array. Let's see more. First, we're going to look through each element from lowest to highest index and find the element that belongs there. We'll use this arrow to indicate that position. The element that goes in this position is the smallest element in the entire input array list. So, we swap them and go to the next position. The element that belongs here is the smallest element in the rest of the input, shown with a dotted box here. In this case it is 17, which is already in the right place. We can just leave it where it is, although if we swap it with itself it will not hurt anything and removes a case from the code.

1:03

The rest of this process proceeds similarly,

1:07

until finally the input has been sorted in place. >> Hi. We're now going to sort the earthquakes by their magnitude. So I have started.

1:19

Our method here which is called sort by magnitude, but we're going to write a helper function first because what we need to do is, we need to find the index location of the smallest magnitude of a part of the array list.

1:36

So I've actually already written that code here, let me just go over it real quick. We have the method, get smallest magnitude, and we are passing in the array list of type quake entry. And we're also passing in a second parameter, from. So this is the index position we want to start at and we want to find the smallest magnitude from that point over in the array list.

2:04

So we're gonna assume the smallest one is the first one to start with cuz that's all we've seen, and we're calling that min index. . And then we're gonna loop over the rest of the quake entries in this four loop here. And then what we'll do is we'll compare.

2:22

The current one we're looking at, which is the ith position, with the current min index position that we found. And whenever we find one smaller, we're going to reset the min index equal to that ith location. And then once we've looked at everything from here over, we would have found the index position that has the smallest magnitude.

2:47

So we're going to return that where you see return min index.

2:52

So get smallest magnitude will return the index location of the smallest magnitude. From over. Okay? So now we want to do the sort. We wanna sort them by their magnitude and we're gonna use this helper function. So the first thing we're gonna do is we have an array list of quake entries and that variable's called n. And so we're gonna iterate over that list, so we'll create a four loop here.

3:29

We're gonna look at everything so from zero to in.size.

3:38

And we'll increment by one each time.

3:52

And then we're going to find them in index by just calling our helper function. So the first time it'll look through the whole array list and tell us the index of the smallest.

4:03

So we'll say int min index.

4:11

We'll just call our method And we'll pass n. And we'll pass i. So i the first time is zero.

4:26

The first time i is going to be zero and it will find the smallest, the index position of the smallest throughout the whole array list. Now, once we know where the smallest is, we need to put the smallest where it belongs. So the first time, the smallest belongs in slot zero. So we find the location of where the smallest is, we're gonna call that min-index. And then we're gonna swap it with the thing in slot zero. So we need to write the code for that. So we're gonna create a quake entry called qi. And this one will store temporarily the quake entry in the ith slot.

5:15

And again the first time we do this, i is zero, so it's grabbing the one there, because we're going to have to swap it with what we're going to create a quake entry for the one where the smallest magnitude is, we'll call that q min.

5:35

And that one is at location min index where we just found out.

5:43

And now that we've temporarily stored them, we can now set the one at location i.

5:51

To q min, And then we'll set the one at location min index to i. So we're just swapping the two in the ith position and the min index position.

6:21

Let's see. Okay.

6:36

All right. And then we'll compile this.

6:41

Okay, so that's good so far. Now we have to test this out. Let me just go over what we just did again. So we have a link that's gonna go all the way through. The first time i is zero. It finds a min index by calling our helper method in slot zero, and then we swap slot zero with min index. The second time through the loop, i is one. So now we're looking for one over, because we know the smallest is in slot zero. And then at that point we're gonna find the smallest from one over. We'll set it to min index, we'll swap that with position one, and so on as we go. Now we need to test this, so I now have a tester method down here that I've started. And we're gonna have the normal thing where we're gonna get our earthquake data here. We're just getting it from a file and we're going to. Right now all we do is just print out all the quake entries, but we wanna sort them first before we print them out. So we're going to call the method we just wrote, which is called sort by magnitude.

7:41

Here so we'll just type this in here. Sort by magnitude, and we're gonna pass it the quake entry, which is in this case called list, and then we'll print them out. So here they should be, our earthquakes should be sorted by magnitude when we test this. So let me compile it.

8:07

It's good.

8:09

Now we wanna go over to blue J, and we wanna create a quake sort, and we're gonna call our test sort method, and let's see what happens.

8:22

All right, so we get lots of data.

8:25

And you can see it stopped, but you can see they're sorted by the magnitude, because the largest one is 7, then 6.5, 5.9, 5.8, and so on. So anyway, it looks like it worked. All right, happy coding.

Downloads

Lecture Video	mp4
Subtitles (English)	WebVTT
Transcript (English)	txt

Would you like to [help us translate](#) the transcript and subtitles into additional languages?