Sort arrays

Selecting transcript lines in this section will navigate to timestamp in the video

- Sometimes we'll want to search for multiple elements in an array within a short period of time. If we run a linear search every single time we want to search, this can take quite a while. If we know what we're going to search for multiple items, it might be worth sorting the array first in order to make each search faster. Now, how do we sort items? Well, there's usually a natural order to our data. For example, if we had an array of numbers, a sorted version of this array could have the numbers in increasing or decreasing order. With characters and strings, we can sort in alphabetical order. A should come before b and b before c and so forth. To sort items, we can implement a sorting function ourselves or usually this behavior is built into the programming language through a sort function. Sometimes you'll have to call this method on your collection of objects in an object-oriented programming language, or pass it as a parameter to a sorting function. Now, there are times where there is not a natural order to your data. So far, we've talked about non-custom datatypes and data structures. We haven't talked about classes, structs, or objects. What if we wanted to store data about a given person. We might hold data about their first name, last name, and age, in some kind of structure. Imagine having many of these people in an array. Say we have three people and they each have their own firstName, lastName, and age. If we wanted to sort the array, which person data should come first? Do we sort by firstName, lastName or age? Well, since there is not a natural order to our data, that's something we get to define, and we usually do it with something called a comparator. We say given two of these people data, if one age is smaller than the other's age or if one's lastName is alphabetically first, then that person should come first. What we actually compare by is something we define as programmers. Again, we don't get too deep into this, but the main idea here is sometimes you will need to sort custom objects, and to do that, we have to define how we compare each object to each other. We define how we should order the objects. Now, sorting, like searching, is computationally expensive, because you have to do a lot of comparisons and checks. How many comparisons you do depends on which sorting algorithm you use and how much information you have about your data. Either way, it takes a lot of time and you'll want to minimize it in your code.